Learning Graph Representation via Graph Entropy Maximization

Ziheng Sun [1 2] **Xudong Wang** [1] **Chris Ding** [1] **Jicong Fan** [1 2]

Abstract

Graph representation learning aims to represent graphs as vectors that can be utilized in downstream tasks such as graph classification. In this work, we focus on learning diverse representations that can capture the graph information as much as possible. We propose quantifying graph information using graph entropy, where we define a probability distribution of a graph based on its nodes' representations and global-graph representation. However, the computation of graph entropy is NP-hard due to the complex vertex-packing polytope involved in its definition. To address this challenge, we provide an approximation method leveraging orthonormal representations for graph entropy maximization. The proposed method is implemented via graph neural networks, resulting in informative node-level and graph-level representations. Experimental results demonstrate the effectiveness of our method in comparison to many baselines in unsupervised learning and semi-supervised learning tasks. The code of our method is available at `https://github.com/MathAdventurer/GeMax`.

1. Introduction

Graphs, representing entities and their relationships, are crucial in diverse fields like chemistry (Debnath et al., 1991; Kriege & Mutzel, 2012), biology (Borgwardt et al., 2005), and social sciences (Yanardag & Vishwanathan, 2015). Graph representation learning, transforming graphs into vectors for tasks such as classification, is a challenging problem, due to the non-Euclidean nature of graph data. Numerous studies have focused on unsupervised graph-level representation learning using graph neural networks (GNNs), a notable approach in this domain. Key methodologies include InfoGraph (Sun et al., 2019), which maximizes mutual information between graph-level and node-level representations, and Graph Contrastive Learning techniques like GraphCL (You et al., 2020), AD-GCL (Suresh et al., 2021), and JOAO (You et al., 2021), which enhance graph representations through various augmentation strategies. AutoGCL (Yin et al., 2022) innovates with learnable graph view generators, while GraphACL (Luo et al., 2023a) introduces a novel self-supervised approach. InfoGCL (Xu et al., 2021) and SFA (Zhang et al., 2023) focus on information transfer and feature augmentation in contrastive learning, respectively. GCS (Wei et al., 2023), NCLA (Shen et al., 2023), S^3-CL (Ding et al., 2023), and ImGCL (Zeng et al., 2023) further refine graph augmentation and learning techniques. GRADATE (Duan et al., 2023) integrates subgraph contrast into multi-scale learning networks. These methods are commonly rooted in the InfoMax principle (Linsker, 1988), which will be detailed in Section 4.1. Other types of methods for graph representation learning include VGAE (Kipf & Welling, 2016; Hamilton et al., 2017; Cui et al., 2020), graph embedding (Wu et al., 2020; Yu et al., 2021; Bai et al., 2019; Verma & Zhang, 2019), self-supervised learning (Liu et al., 2022b; Hou et al., 2022; Lee et al., 2022; Xie et al., 2022; Wu et al., 2021; Rong et al., 2020; Zhang et al., 2021b;a; Xiao et al., 2022), and various contrastive learning methods (Le-Khac et al., 2020; Qiu et al., 2020; Ding et al., 2022; Xia et al., 2022; Fang et al., 2022; Trivedi et al., 2022; Han et al., 2022; Mo et al., 2022; Yin et al., 2022; Xu et al., 2021; Zhao et al., 2021; Zeng & Xie, 2021; Li et al., 2022a;b; Wei et al., 2022). More recently, Sun et al. (2023) presented a Lovász principle for graph representation learning, which is based on the graph Lovász number (Lovász, 1979) and uses the handle vector learned by a GNN as graph representation.

For unsupervised graph representation learning, it is crucial to ensure that the representations contain sufficient information useful for downstream tasks. One may recall that the representations given by an autoencoder are often very useful for downstream tasks. The reason is that the representations have preserved the major information of the input data—they can well reconstruct the input data. For graph data, it is impossible to use the vector representation of each graph to reconstruct the graph itself, but it is possible to make the vector preserve sufficient information from the

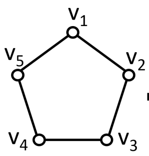
[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), China [2]Shenzhen International Center for Industrial and Applied Mathematics, Shenzhen Research Institute of Big Data, Shenzhen, China. Correspondence to: Jicong Fan <fanjicong@cuhk.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

graph. To quantify information, we can use entropy. Entropy is a fundamental concept in information theory and is very useful for many machine learning problems such as graph learning. Various entropy concepts have evolved to quantify the complexity and information of graphs (Dehmer & Mowshowitz, 2011; Dehmer, 2008). The most representative ones are the structural entropy (Mowshowitz & Dehmer, 2012) for node-level analysis and the edge entropy (Jiang et al., 2020; Wang et al., 2021; Grebenkina et al., 2018; Aziz et al., 2020; Luo et al., 2023b) for evaluating edge connectivity. Additionally, von Neumann entropy (Liu et al., 2021; 2022a; Passerini & Severini, 2008; Minello et al., 2019; Ye et al., 2014; Dong et al., 2019) and Rényi entropy (Pál et al., 2010; Oggier & Datta, 2021) address spectral complexity and graph clustering, respectively, while M-ILBO (Ma et al., 2023) focuses on dataset entropy. It is important to distinguish these from János Körner's original graph entropy concept (Körner, 1973), grounded in information theory and combinatorics. This concept, dating back to Shannon's work (Shannon, 1948) and further developed by Körner (Körner, 1973) and Lovász (Lovász, 1979), involves complex computational challenges like the vertex-packing polytope, leading to NP-hard computation. Although graph entropy is theoretically important and useful in the domains of combinatorics and information theory, it remains unexplored in the field of graph learning. Crucially, our findings indicate that graph entropy is superior in leveraging the inherent structure of graphs compared to other entropy approaches such as Shannon entropy and Rényi entropy.

This work introduces a novel approach called **G**raph **E**ntropy **Max**imization (GeMax) to graph representation learning, marking the first instance of graph entropy's application in this context. Our approach establishes a probability distribution for a graph by incorporating its nodes' representations and a global graph representation learned through two graph neural networks respectively. The computation of graph entropy, however, presents a significant challenge, as it is NP-hard due to the complexity associated with the vertex-packing polytope in its definition. To tackle this challenge, we introduce a method of maximizing the approximation of graph entropy by utilizing Lovász's orthonormal representations. Our contributions are as follows.

- We introduce GeMax, a novel method for graph representation learning, marking the inaugural exploration of Körner's graph entropy within the graph learning community.
- Recognizing the NP-hard computation of graph entropy, we propose a tractable approximation for GeMax via leveraging orthonormal representations and present an alternating updating method for its optimization.
- We conduct extensive experiments to evaluate the performance of our GeMax method in comparison to InfoMax Principle (Linsker, 1988; Sun et al., 2019), Lovász Principle (Sun et al., 2023), as well as Shannon entropy and Rényi entropy maximization, in unsupervised and semi-supervised graph representation learning tasks.



$$B = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 & \boldsymbol{b}_4 & \boldsymbol{b}_5 & \boldsymbol{b}_6 & \boldsymbol{b}_7 & \boldsymbol{b}_8 & \boldsymbol{b}_9 & \boldsymbol{b}_{10} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Figure 1: Indicator matrix of independent sets of a pentagon

2. Preliminary

In this section, we present the definition of graph entropy (Körner, 1973) and Lovász's orthonormal representations (Lovász, 1979). Graph entropy, a crucial concept in probabilistic graph theory, was first introduced by János Körner (Körner, 1973). We focus on its combinatorial definition, which revolves around the vertex-packing polytope. An independent set in a graph G is a group of vertices where no two are adjacent. Let $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_{N_b}] \in \{0,1\}^{|V|\times N_b}$ be the indicator matrix of the independent sets of G, with N_b being the number of such sets, and each column $\boldsymbol{b}_i$ indicating a specific independent set. For instance, on the pentagon graph shown by Figure 1, where $N_b = 10$, the vector $\boldsymbol{b}_6 = [1,0,1,0,0]^\top$ highlighted by the blue rectangle is the independent set comprising $\{v_1, v_3\}$. The vertex-packing polytope VP(G) of graph G is defined as follows.

Definition 2.1 (vertex-packing polytope)**.** For a graph G with vertices set V, VP(G), the vertex-packing polytope of G, is defined as the convex corner of its independent sets' indicator vectors. Specifically, with $\boldsymbol{B} \in \{0,1\}^{|V|\times N_b}$ as the indicator matrix and $\boldsymbol{\lambda} \in \mathbb{R}_+^{N_b}$, we define VP$(G)$ as

$$\mathrm{VP}(G) := \Big\{\boldsymbol{a} \in \mathbb{R}^{|V|} : \boldsymbol{a} = \boldsymbol{B}\boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq 0,\ \sum\nolimits_i \lambda_i = 1\Big\}.$$

In a probabilistic graph (G,P), where P represents the probability distribution across its vertices, defined as $P = \{P_1, P_2, ..., P_n\}$ with P_i being the probability density of vertex i, graph entropy is denoted as $H_k(G,P)$. Based on VP(G), the definition of $H_k(G,P)$ is as follows.

Definition 2.2 (Graph Entropy (Körner, 1973))**.** For a probabilistic graph (G,P), its entropy with respect to P is

$$H_k(G,P) := \min_{\boldsymbol{a}\in \mathrm{VP}(G)} \sum_{i=1}^{|G|} -P_i \log(a_i).$$

László Lovász established the concept of orthonormal representations for graphs, which can be formally defined as:

Definition 2.3 (Set of Lovász's Orthonormal Representations (Lovász, 1979))**.** Consider a graph $G = (V,E)$. Each vertex i in G is represented by a unit vector $\boldsymbol{z}_i \in \mathbb{R}^d$, indicating its d-dimensional representation. The set of orthonormal representations for G, denoted as $\mathcal{T}(G)$, is:

$$\mathcal{T}(G) := \{\boldsymbol{Z} \in \mathbb{R}^{n\times d} : \ \|\boldsymbol{z}_i\|_2 = 1 \text{ for } i = 1, 2, ..., n; \quad \boldsymbol{z}_i^\top \boldsymbol{z}_j = 0, \ \forall (i, j) \notin E\}.$$

3. Proposed Methods

3.1. Graph Entropy Maximization

Given a set of N graphs $\mathcal{G} = \{G_1, G_2, \dots, G_N\}$ drawn from some unknown distribution $\mathcal{D}$ in $\mathbb{G}$, we want to learn a model $F : \mathbb{G} \to \mathbb{R}^d \times \mathbb{R}^{n\times d}$ to represent each graph as a vector and represent its vertices as vectors, i.e., $(\mathbf{g}_j, \boldsymbol{Z}_j) = F(G_j)$, where $\mathbf{g}_j \in \mathbb{R}^d$ and $\mathbf{Z}_j \in \mathbb{R}^{n_j\times d}$ denote the graph-level and node-level representations of G_j respectively, and n_j is the number of nodes of G_j. F should capture the important information of the underlying distribution $\mathcal{D}$ and $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_N$ should be useful in downstream tasks such as graph classification. A fundamental question is

How to quantify the goodness of $(\mathbf{g}_j, \boldsymbol{Z}_j)$?

We propose to use graph entropy H_k, defined by Definition 2.2, as a measure of the goodness of graph representations. We have the following observation.

- Graph entropy serves as an indicator of the inherent uncertainty present in a probabilistic graph (G, P), according to the minimal possible code rate definition in graph theory (Körner, 1973).
- By Definition 2.2, we can regard $H_k(G, P)$ as the minimum "cross-entropy" between P and all possible $\boldsymbol{a}$ in VP(G), though $\sum a_i$ may not be 1. Thus, $H_k(G, P)$ measures the minimum inconsistency between P and the combination of independent sets of G.

Graph entropy, if computationally feasible, is particularly valuable for graph representation learning, as it directly correlates the entropy measure with the graph's structure through VP(G). In contrast, other commonly used entropy measures such as Shannon entropy are primarily defined upon distribution, without direct consideration of the graph's structural aspects. Consequently, these traditional entropy measures may not be effective in producing graph representations with rich structural information.

In Definition 2.2, H_k is based on the graph structure G and vertex distribution P, where the former is given and fixed but the latter is unknown and should be determined if possible. We define P using $(\mathbf{g}, \mathbf{Z})$, i.e., P is a function of $F(G)$, formulated as $P_{F(G)}$, connecting graph entropy and graph representations. Then we propose to find an F that yields the representations with maximum graph entropy:

$$\max_{F\in\mathcal{F}} \ \mathbb{E}_{G\sim\mathcal{D}} \left[H_k \left(G, P_{F(G)}\right)\right], \tag{1}$$

where $\mathcal{F}$ denotes some hypothesis space. Based on the previous discussion, maximizing graph entropy implies that $(\mathbf{g}, \mathbf{Z})$ makes P inconsistent with the combination of independent sets of G, thereby preserving the connection information on G. A common model $F(G)$ on all graphs may make the representations of different graphs similar, but maximizing graph entropy preserves discriminative information of the graphs useful for downstream tasks such as graph classification.

In this work, we let F be a GNN. Specifically, denote $\mathbb{A}$ as the space of the adjacency matrix $\boldsymbol{A}$ and $\mathbb{X}$ as the space of node feature matrix $\boldsymbol{X}$. Denote $F = (F_g, F_Z)$ and let $F_g(\cdot, \cdot; \theta) : \mathbb{A} \times \mathbb{X} \to \mathbb{R}^d$ be a GNN with parameter θ for graph-level representation learning and $F_Z(\cdot, \cdot; \phi) : \mathbb{A} \times \mathbb{X} \to \mathbb{R}^{n\times d}$ be another GNN with parameters ϕ for node-level representation learning. For $G \in \mathcal{G}$ with adjacency matrix $\boldsymbol{A}$ and feature matrix $\boldsymbol{X}$, we obtain

$$\mathbf{g}^\theta = F_g(\boldsymbol{A}, \boldsymbol{X}; \theta), \ \text{ and } \ \boldsymbol{Z}^\phi = F_Z(\boldsymbol{A}, \boldsymbol{X}; \phi). \tag{2}$$

Then we denote the probability of nodes in G as

$$P_{F(G)} = [P_1(\mathbf{g}^\theta, \boldsymbol{Z}^\phi), ..., P_n(\mathbf{g}^\theta, \boldsymbol{Z}^\phi)] \triangleq P(\boldsymbol{A}, \boldsymbol{X}; \theta, \phi),$$

where $0 \le P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi) \le 1$, $\sum P_i(\mathbf{g}, \boldsymbol{Z}) = 1$, and $P_i(\boldsymbol{A}, \boldsymbol{X}; \theta, \phi) \equiv P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi)$. Here we can define $P_{F(G)}$ as a Boltzmann distribution as follows:

$$P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi) := \frac{\exp(-\|\boldsymbol{z}_i^\phi - \mathbf{g}^\theta\|_2^2)}{\sum_{l\in V} \exp(-\|\boldsymbol{z}_l^\phi - \mathbf{g}^\theta\|_2^2)}, \ \forall\, i \in V. \tag{3}$$

Note that instead of the Boltzmann distribution, one may use other distributions such as

$$P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi) = \frac{(1 + \|\boldsymbol{z}_i^\phi - \mathbf{g}^\theta\|^2)^{-1}}{\sum_{l\in V}(1 + \|\boldsymbol{z}_l^\phi - \mathbf{g}^\theta\|^2)^{-1}}.$$

We empirically find that our method is not sensitive to the definition of $P_{F(G)}$, possibly due to the high expressive ability of neural networks.

Now, invoking the definitions of F and P into (1), we solve

$$\max_{\theta,\phi} \ \sum_{j=1}^{N} H_k\left(G_j, P(\boldsymbol{A}_j, \boldsymbol{X}_j; \theta, \phi)\right) \tag{4}$$

or equivalently

$$\max_{\theta,\phi} \ \sum_{j=1}^{N} \min_{\boldsymbol{a}\in\text{VP}(G_j)} \sum_{i=1}^{n_j} -P_i(\boldsymbol{A}_j, \boldsymbol{X}_j; \theta, \phi) \log(a_i), \tag{5}$$

which is our Graph Entropy Maximization (GeMax) method for graph representation learning. The computation of graph entropy and the procedure of the proposed GeMax method are illustrated in Figure 2.

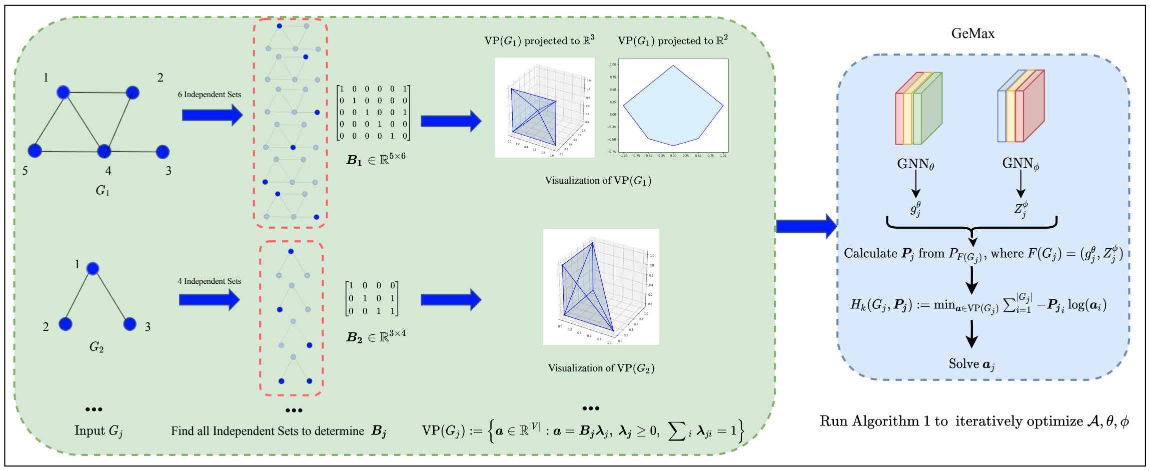

1
2
5
4
3
G_1
6 Independent Sets
$\begin{bmatrix}1&0&0&0&0&1\\0&1&0&0&0&0\\0&0&1&0&0&1\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}$
$B_1 \in \mathbb{R}^{5\times 6}$
VP(G_1) projected to $\mathbb{R}^3$
VP(G_1) projected to $\mathbb{R}^2$
Visualization of VP(G_1)
1
2
3
G_2
4 Independent Sets
$\begin{bmatrix}1&0&0&0\\0&1&0&1\\0&0&1&1\end{bmatrix}$
$B_2 \in \mathbb{R}^{3\times 4}$
Visualization of VP(G_2)
Input G_j
Find all Independent Sets to determine B_j
VP$(G_j) := \left\{ a \in \mathbb{R}^{|V|} : a = B_j \lambda_j,\ \lambda_j \geq 0,\ \sum_i \lambda_{ji} = 1 \right\}$
GeMax
GNN$_\theta$
GNN$_\phi$
g_j^θ
Z_j^ϕ
Calculate P_j from $P_{F(G_j)}$, where $F(G_j) = (g_j^\theta, Z_j^\phi)$
$H_k(G_j, P_j) := \min_{a \in \mathrm{VP}(G_j)} \sum_{i=1}^{|G_j|} -P_{j_i} \log(a_i)$
Solve a_j
Run Algorithm 1 to iteratively optimize $\mathcal{A}, \theta, \phi$

Figure 2: The computation of graph entropy using toy graphs and the illustration of proposed GeMax method

However, directly solving GeMax is computationally challenging for large graphs because computing the vertex-packing polytope VP(G) involves computing the indicator matrix $\boldsymbol{B}$ of independent sets, which is equivalent to solving the NP-hard graph coloring problem (Jensen & Toft, 2011). Previous studies (Rezaei, 2013) typically utilized graph entropy H_k for theoretical analysis and provided its upper and lower bounds, instead of computing it exactly. The best-known lower bound of H_k (Boreland, 2018) is as

$$H(P) - \log \alpha(G) \leq H_k(G, P) \tag{6}$$

where $\alpha(G)$ is the independent number and $H(P)$ is the Shannon entropy. Since $\alpha(G)$ remains constant for a given G, maximizing this lower bound is equivalent to maximizing Shannon entropy over the vertex set. This approach does not directly capture any topological information of the graph's structure. The experiments in Section 5.2 will show the unsatisfactory performance of this approach. To overcome this limitation, we introduce an approximation method for graph entropy maximization in the following two sections.

3.2. Approximation Method for GeMax

To leverage the information from the vertex packing polytope, while avoiding NP-hard calculations, we optimize objective (4) over its subset, denoted as VP$_{\text{Sub}}(G)$.

Definition 3.1 (Subset of VP(G))**.** Let $\mathbb{1}(\cdot)$ be an element-wise indicator function such that $[\mathbb{1}(\boldsymbol{a})]_i = 1$ if $a_i > 0$ and $[\mathbb{1}(\boldsymbol{a})]_i = 0$ for $a_i = 0$. We define a subset of VP(G) as

$$\text{VP}_{\text{Sub}}(G) := \left\{ \boldsymbol{a} \in \mathbb{R}^{|V|} : \mathbb{1}(\boldsymbol{a}) \in \text{VP}(G), 0 \leq a_i \leq 1 \right\}.$$

The following proposition indicates that VP$_{\text{Sub}}(G)$ maintains the two important properties of VP(G).

Proposition 3.2. *1) VP$_{Sub}(G)$ is a convex corner; 2) All the indicator vectors of independent sets of G are contained in VP$_{Sub}(G)$, i.e., $\boldsymbol{b}_i \in$ VP$_{Sub}(G)$, $\forall i = 1, 2, ..., N_b$.*

Thus, solving GeMax, namely (5), over VP$_{\text{Sub}}(G)$ instead of VP(G), can also leverage the information of independent sets of G. Nevertheless, this subset still necessitates calculating the independent set indicator matrix $\boldsymbol{B}$, which remains NP-hard. To address this challenge, we define $P_{F(G)}$ on the set of orthonormal representations, presented by Definition 2.3, rather than the entire real space. The following theorem provides the connection between orthonormal representations and our VP$_{\text{Sub}}(G)$.

Theorem 3.3. *Let $\boldsymbol{D}_{\boldsymbol{a}} = diag(\boldsymbol{a}) = diag(a_1, a_2, \ldots, a_n)$ with $0 \leq a_i \leq 1\ \forall i \in [n]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n]^\top \in \mathbb{R}^{n\times d}$. If $\boldsymbol{Z} \in \mathcal{T}(G)$ and $\boldsymbol{z}_i^\top \boldsymbol{z}_j \neq 0\ \forall (i,j) \in E$, then $\boldsymbol{D}_{\boldsymbol{a}}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D}_{\boldsymbol{a}} = \boldsymbol{D}_{\boldsymbol{a}}^2$ if and only if $\boldsymbol{a} \in$ VP$_{Sub}(G)$*

According to Theorem 3.3, solving GeMax (5) over VP$_{\text{Sub}}(G)$ is equivalent to solving the following problem

$$\begin{aligned}
&\max_{\theta,\phi} \sum_{j=1}^{N} \min_{\boldsymbol{a}_j} \sum_{i=1}^{n_j} -P_i(\mathbf{g}_j^\theta, \boldsymbol{Z}_j^\phi) \log(a_{j(i)}), \\
&\quad \text{s.t. } \boldsymbol{Z}_j^\phi \in \mathcal{T}(G_j),\ \boldsymbol{D}_{\boldsymbol{a}_j}(\boldsymbol{Z}_j^\phi {\boldsymbol{Z}_j^\phi}^\top)\boldsymbol{D}_{\boldsymbol{a}_j} = \boldsymbol{D}_{\boldsymbol{a}_j}^2, \\
&\qquad 0 \leq a_{ij} \leq 1,\ \forall\, i \in [n_j],\ j \in [N].
\end{aligned} \tag{7}$$

Note that here we do not need to consider the constraints $\boldsymbol{z}_i^\top \boldsymbol{z}_j \neq 0\ \forall (i,j) \in E$ because in GNN, the connected nodes always share some information, which means $\boldsymbol{z}_i^\top \boldsymbol{z}_j \neq 0$ always holds.

Objective (7) serves as an effective approximation to objective (5), surpassing the approach of merely maximizing the lower bound of graph entropy in graph representation learning, as detailed in (6). The employment of orthonormal representations enables the preservation of non-adjacency characteristics and hence pursues diverse representations, consistent with the information maximization goal of GeMax. Actually, (7) is graph entropy maximization under the condition that $P_{F(G)}$ is defined over orthonormal representations.

We can use the Lagrange multiplier method or exact penalty method to solve the constrained optimization problem (7). In this work, we propose to relax (7) to a regularized optimization problem, for which the optimization is much easier than the constrained optimization. Specifically, first, for the orthonormality constraint $\boldsymbol{Z}_j^\phi \in \mathcal{T}(G_j)$, we define the following regularization

$$\mathcal{L}_{\text{orth}}(\mathcal{G};\phi) := \sum_{j=1}^{N} \left\| \boldsymbol{M}_j \odot \left(\boldsymbol{Z}_j^\phi (\boldsymbol{Z}_j^\phi)^\top - \boldsymbol{I}_n \right) \right\|_F^2, \tag{8}$$

where $\boldsymbol{M}_j = \mathbf{1}_{n_j \times n_j} - \boldsymbol{A}_j$ is a binary mask matrix, and $\boldsymbol{I}_{n_j}$ is an identity matrix of size $n_j \times n_j$, and the operator $\odot$ denotes the Hadamard product.

For the constraints $\boldsymbol{D}_{\boldsymbol{a}_j}(\boldsymbol{Z}_j^\phi \boldsymbol{Z}_j^{\phi\top})\boldsymbol{D}_{\boldsymbol{a}_j} = \boldsymbol{D}_{\boldsymbol{a}_j}^2$ and $\boldsymbol{a}_{ij} \leq 1$, we define the following regularization, termed as sub-vertex-packing polytope loss:

$$\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A}) := \sum_{j=1}^{N} \left\| \boldsymbol{D}_{a_j} \left(\boldsymbol{Z}_j^\phi (\boldsymbol{Z}_j^\phi)^\top \right) \boldsymbol{D}_{a_j} - \boldsymbol{D}_{a_j}^2 \right\|_F^2$$
$$\text{s.t. } 0 \leq a_{ij} \leq 1,\ \forall i \in [n_j],\ \boldsymbol{a}_j \in \mathcal{A} \tag{9}$$

where $\boldsymbol{D}_{a_j} = \text{diag}(\boldsymbol{a}_j)$ and $\mathcal{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$. Note that the constraints $0 \leq a_{ij} \leq 1$ can be easily handled through the projection method.

For convenience, we denote the graph entropy objective in (7) as follows:

$$\mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) = \sum_{j=1}^{N}\sum_{i=1}^{n_j} -P_i(\mathbf{g}_j^\theta, \boldsymbol{Z}_j^\phi)\log(a_{j(i)}). \tag{10}$$

Then the objective for θ and ϕ is

$$\begin{aligned}\mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A}) :=& \mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) - \mu \cdot \mathcal{L}_{\text{orth}}(\mathcal{G};\phi) \\ &- \gamma \cdot \mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})\end{aligned} \tag{11}$$

while the objective for $\mathcal{A}$ is

$$\mathcal{J}_2(\mathcal{G};\theta,\phi,\mathcal{A}) := \mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) + \gamma \cdot \mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})$$
$$\text{s.t. } 0 \leq a_{ij} \leq 1,\ \forall i \in [n_j],\ \boldsymbol{a}_j \in \mathcal{A}, \tag{12}$$

where $\mu > 0$ and $\gamma > 0$ are regularization hyperparameters.

3.3. Optimization Algorithm

We propose to use alternating updating to solve (11) and (12), namely, alternately optimize $\mathcal{A}$ and (θ, ϕ). Suppose at iteration t we have $\mathcal{A}^{(t)} = \{\boldsymbol{a}_1^{(t)}, \ldots, \boldsymbol{a}_N^{(t)}\}$ where $\boldsymbol{a}_j^{(t)} \in$ VP$_{\text{Sub}}(G_j)$ $\forall j \in [N]$. At iteration $t+1$, we fix $\mathcal{A}^{(t)}$ and solve the following sub-problem:

Algorithm 1 Optimization for GeMax (11) and (12)

```
Input: $\mathcal{G}$, $\mu$, $\gamma$, $t = 0$.
1: Random initialization of parameters: $\theta^{(0)}$, $\phi^{(0)}$.
2: Let $\mathcal{A}^{(0)} = \{\boldsymbol{a}_j^{(0)}\}_{j=1}^N = \{[1/n_j, ..., 1/n_j]\}_{j=1}^N$.
3: repeat
4:     $\theta^{(t+1)}, \phi^{(t+1)} = \text{argmax}_{\theta,\phi}\ \mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A}^{(t)})$.
5:     $\mathcal{A}^{(t+1)} = \text{argmin}_{\mathcal{A}\in\mathcal{C}}\ \mathcal{J}_2(\mathcal{G};\theta^{(t+1)},\phi^{(t+1)},\mathcal{A})$.
6:     $t \leftarrow t + 1$
7: until convergence conditions are met
Output: $\theta^{(t+1)}$, $\phi^{(t+1)}$.
```

$$\theta^{(t+1)}, \phi^{(t+1)} = \underset{\theta,\phi}{\text{argmax}}\ \mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A}^{(t)}). \tag{13}$$

Then, fixing $\theta^{(t+1)}$ and $\phi^{(t+1)}$, we solve

$$\mathcal{A}^{(t+1)} = \underset{\mathcal{A}\in\mathcal{C}}{\text{argmin}}\ \mathcal{J}_2(\mathcal{G};\theta^{(t+1)},\phi^{(t+1)},\mathcal{A}) \tag{14}$$

where $\mathcal{C}$ denotes the constraints $0 \leq a_{ij} \leq 1$ for each vector in $\mathcal{A}$. It can be solved by projected gradient descent. Specifically, at iteration $s+1$ of the inner loop of the subproblem, for each vector in $\mathcal{A}$, let

$$\boldsymbol{a}^{(s+1)} = \text{Proj}_{[0,1]}\left(\boldsymbol{a}^{(s)} - \epsilon\boldsymbol{\delta}^{(s+1)}\right), \tag{15}$$

where $\boldsymbol{\delta}$ is the derivative of $\boldsymbol{a}$, ϵ denotes the step size, and the element-wise projection operator Proj$_{[0,1]}$ is

$$\text{Proj}_{[0,1]}(\bar{a}) = \begin{cases} 0, & \text{if } \bar{a} \leq 0, \\ 1, & \text{if } \bar{a} \geq 1, \\ \bar{a}, & \text{otherwise.} \end{cases} \tag{16}$$

Algorithm 1 summarizes the whole procedures of the optimization. In our experiments, we set $\mu = \gamma = 0.5$.Note that the solution of the regularized optimization cannot exactly satisfy the constraints in (7). To ensure the constraints, one may consider using the inexact penalty method to solve the problem, which is just increasing γ and η gradually in the iterative optimization. We have found that the representations given by the regularized optimization are as good as those given by the constrained optimization. The comparison experiments between regularized and constrained optimizations are in Appendix D.4.

The time and space complexities of Algorithm 1 are shown by the following proposition.

Proposition 3.4. *Without loss of generality, suppose $n_1 = n_2 = \cdots = n$, $|E_1| = |E_2| = \cdots = m$, both F_g and F_Z have $L-1$ hidden layers, the widths of the hidden layers and the input feature dimension of the graphs are all d, the* *numbers of iterations of the subproblems for $\{\theta, \phi\}$ and $\mathcal{A}$ are τ_1 and τ_2 respectively, and the number of the outer loop iterations of Algorithm 1 is T. Then the space complexity of the algorithm is $\mathcal{O}(N(m + Ldn + n^2) + Ld^2)$ and the time complexity of the algorithm is $\mathcal{O}(T(\tau_1 N(L(dm + d^2 n) + dn^2) + \tau_2 N n^2))$.*

Note that for mini-batch optimization, the N in the proposition should be replaced by the bath size B. In general, the space and time complexities of Algorithm 1 are linear with the number of graphs. Therefore, our method GeMax is scalable to large datasets provided that n is not too large.

4. Related Work

4.1. Previous Graph Representation Learning Principles

InfoMax Principle InfoMax (Linsker, 1988) is highly influential in current unsupervised graph representation learning and serve as a foundation of numerous methods, including InfoGraph (Sun et al., 2019), GraphCL (You et al., 2020), AD-GCL (Suresh et al., 2021), JOAO (You et al., 2021), AutoGCL (Yin et al., 2022), GraphACL (Luo et al., 2023a), InfoGCL (Xu et al., 2021), GCS (Wei et al., 2023), and ImGCL (Zeng et al., 2023). Taking InfoGraph (Sun et al., 2019) as an example, it maximizes the mutual information between graph-level and node-level representations:

$$\phi^*, \theta^*, \varphi^* = \arg\max_{\phi,\theta,\varphi} \sum_{j=1}^{|\mathcal{G}|} \frac{1}{|V_j|} \sum_{i \in V_j} I_\varphi(\boldsymbol{g}_j^\theta, \boldsymbol{z}_{ij}^\phi), \tag{17}$$

where I_φ is the Jensen-Shannon MI estimator. More details of InfoMax can be found in Appendix D.1.

Lovász Principle The Lovász principle (Sun et al., 2023), based on the Lovász number $\vartheta(G)$ (Lovász, 1979) for a graph G, uses the *handle* vector $\boldsymbol{c}$ as graph representation. The formulation is

$$\phi^*, \theta^* = \arg\min_{\phi,\theta} \sum_{j=1}^{|\mathcal{G}|} \max_{i \in V_j} \frac{1}{\left((\boldsymbol{z}_i^\phi)^\top \boldsymbol{g}_j^\theta\right)^2} + \eta \ell_{\text{orth}}(\mathcal{G}; \theta, \phi), \tag{18}$$

where $\ell_{\text{orth}}(\mathcal{G}; \theta, \phi)$ is the orthonormal regularization. It was shown by (Sun et al., 2023) that the Lovász principle often outperforms the InfoMax principle in graph representation learning. Both the Lovász principle and InfoMax principle will be compared with our GeMax in Section 5.

Information Bottleneck Principle Graph Information Bottleneck (GIB) (Wu et al., 2020) and Subgraph Information Bottleneck (SIB) (Yu et al., 2021) aim to obtain minimal yet sufficient representations for downstream tasks. However, their effectiveness diminishes in the absence of predefined downstream tasks during the learning stage, limiting their suitability for unsupervised and semi-supervised graph learning. Therefore, we will not include GIB and SIB in the comparison experiments.

4.2. Entropy-Based Graph Learning Methods

Structural Entropy Structural entropy (Mowshowitz & Dehmer, 2012) leverages Shannon entropy and node structural components like node degree, widely used for topological analysis (Luo et al., 2021; Yang et al., 2023; Wang et al., 2023; Wu et al., 2022; Zou et al., 2023; Fang et al., 2021). Another key metric, edge entropy (Jiang et al., 2020; Wang et al., 2021; Grebenkina et al., 2018; Aziz et al., 2020; Luo et al., 2023b), focuses on edge interconnectivity to assess graph structures. For instance, the degree entropy of graph G with vertex set V and node degree d_i is:

$$H_{\text{deg}}(P) = -\sum_{i \in V} P_i \log P_i, \text{ where } P_i = \frac{d_i}{\sum_{j \in V} d_j}. \tag{19}$$

The degree entropy of a graph G, inherently fixed by its degree structure, presents reformulation challenges in graph representation learning. It is notably complex to adapt this entropy into a format suitable for optimization via node and graph representations. Similar challenges extend to other structural entropy forms. Consequently, these entropies are excluded from our comparative experiments due to their inherent limitations.

Non-Structural Entropy Non-structural entropy methods, such as Shannon entropy (Shannon, 1948), Rényi entropy (Pál et al., 2010; Oggier & Datta, 2021) used in graph clustering, and the von Neumann entropy (Liu et al., 2021; 2022a; Passerini & Severini, 2008; Minello et al., 2019; Ye et al., 2014; Dong et al., 2019) for spectral complexity, differ significantly from structural approaches. M-ILBO (Ma et al., 2023) calculates graph dataset entropy by optimizing Information Lower Bound (ILBO).

These entropies are defined on probability distributions, as shown in Eq. (3) for graph G. Thus, they can be reformulated as objective functions for graph representation learning. For instance, Shannon entropy's representation objective is

$$J_{\text{Sh}}(\mathcal{G}; \theta, \phi) = \sum_{j}^{|\mathcal{G}|} \sum_{i \in V_j} -P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi) \log(P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi)) \tag{20}$$

Rényi entropy's representation objective is expressed as

$$J_{\text{Rényi}}(\mathcal{G}; \theta, \phi) = \sum_{j}^{|\mathcal{G}|} \frac{1}{1-\alpha} \log \left(\sum_{i \in V_j} (P_i(\mathbf{g}^\theta, \boldsymbol{Z}^\phi))^\alpha \right) \tag{21}$$

where we set $\alpha = 1$. However, some types, like von Neumann entropy, are less suited for graph representation learning due to their reliance on eigenvalues. Both Shannon and Rényi entropies will be compared with GeMax in Section 5.

5. Experiments

In this section, we evaluate the effectiveness of our GeMax method in graph learning tasks including unsupervised and semi-supervised representation learning, on TUdataset (Morris et al., 2020). The statistics of the considered graph datasets are in Table 1.

Table 1: Statistics of TUdataset (Morris et al., 2020)

Name	# of graphs	# of classes	# of nodes	node labels	node attributes
MUTAG	188	2	17.9	yes	no
PROTEINS	1113	2	39.1	yes	yes
DD	1178	2	284.32	yes	no
NCI1	4110	2	29.9	yes	no
COLLAB	5000	3	74.49	no	no
IMDB-B	1000	2	19.8	no	no
REDDIT-B	2000	2	429.63	no	no
REDDIT-M5K	4999	5	508.52	no	no

5.1. Comparison with InfoMax and Lovász Principles

Our objective is to evaluate three unsupervised graph representation learning principles: InfoMax (17), the Lovász principle (18), and our GeMax (5). To facilitate fair comparisons, we maintain the neural network architectures used in InfoMax methods while substituting their objective with either the Lovász Principle (18) or our GeMax (5), keeping all other structures and parameters unchanged. It is important to note that information bottleneck principles, being task-specific, are not included in this comparison.

Unsupervised Learning Following (Sun et al., 2019; Luo et al., 2023a; Wei et al., 2023; Sun et al., 2023), we train a graph representation model on unlabeled data to obtain graph representations and use these representations and graph labels to train an SVM classifier. Our experimental setup follows GraphACL (Luo et al., 2023a). Actually, all the six InfoMax methods, including GraphCL (You et al., 2020), AD-GCL (Suresh et al., 2021), JOJOv2 (You et al., 2021), AutoGCL (Yin et al., 2022), GraphACL (Luo et al., 2023a) and GCS (Wei et al., 2023) are based on the architecture of InfoGraph (Sun et al., 2019). Specifically, they use a 5-layer GIN (Xu et al., 2018) with hidden size 128 as the representation model. The model is trained with a batch size of 128 and a learning rate of 0.001. For those contrastive learning methods (e.g., JOJOv2 and AutoGCL), they use 30 epochs of contrastive pre-training under the naive strategy. We repeated the experiments 10 times with different random seeds. Each time, we performed 10-fold cross-validation on each dataset. In each fold, we use 90% of the total data as unlabeled data for contrastive pre-training and 10% as labeled testing data. More details are in Appendix D.1 and the sensitivity analysis for hyperparameters is in Appendix D.5. The average classification accuracy (ACC) with standard deviation is reported in Table 2 while the average results across all methods are in Figure 3a. Our GeMax outperformed InfoMax and Lovász principles in almost all cases.

Semi-supervised Learning Following (Sun et al., 2019; Yin et al., 2022; Sun et al., 2023), we compare the three unsupervised graph representation learning principles in semi-supervised learning tasks. The loss function of semi-supervised InfoMax method methods is shown by (33) in Appendix D.1. To ensure fair comparisons, we follow (33) and replace the InfoMax (17) with Lovász Principle (18) or our GeMax (5). Following the settings of AutoGCL (Yin et al., 2022), we employ a 10-fold cross-validation on each dataset. For each fold, we use 80% of the total data as the unlabeled data, 10% as labeled training data, and 10% as labeled testing data. The classifier for labeled data is a ResGCN (Chen et al., 2019) with 5 layers and a hidden size of 128. More details about the semi-supervised learning are in Appendix D.1. We repeat each experiment 10 times and report the average classification accuracy in Table 3. The average results across all methods are shown in Figure 3b. Consistent with the unsupervised learning tasks, our GeMax still outperformed InfoMax and Lovász principles in semi-supervised graph representation learning.

5.2. Comparison of Different Entropy Measures

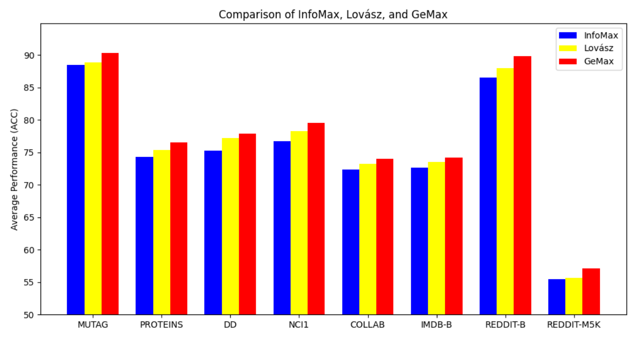
We evaluate the efficacy of graph entropy against other measures in graph representation learning. Despite various entropies being explored in GNNs, a standardized approach for graph representation learning remains elusive. As outlined in the related work, our experiments adapt established supervised and unsupervised frameworks, substituting the GeMax objective (Eq. (5)) with Shannon (Eq. (20)) or Rényi entropy objectives (Eq. (21)). After conducting experiments 10 times, average accuracies are reported in Tables 4 and 6. Our visual analyses in Figures 4a and 4b indicate graph entropy's superiority in capturing topological information, outperforming Shannon and Rényi entropies that primarily focus on vertex set entropy and may miss crucial structural aspects of graphs.

5.3. More Experiment Results

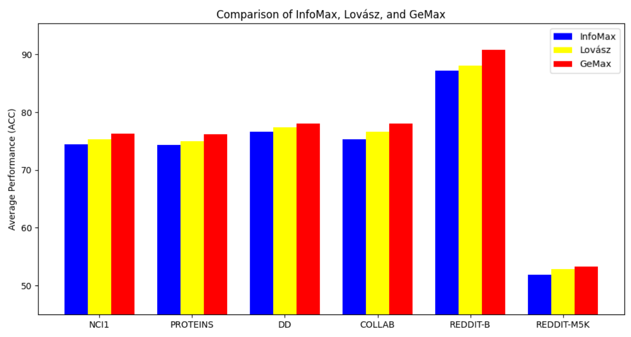
We provide additional experiment results and analyses in the appendix. We compare the performance of graph entropy against other entropy measures in graph representation learning tasks (Appendix D.2) and present a comparison between the exact computation of graph entropy and other graph learning principles for small graphs (Appendix D.3). Furthermore, We explore an alternative optimization approach using the inexact penalty method (Appendix D.4),

Table 2: Classification accuracy (%) of unsupervised learning using different principles. The baseline results are from previous works. The number in **bold** denote the best principle among the three ones. The numbers with * denote he best method for a dataset. This notation is also applied in Table 3, Table 4, and Table 6.

Method	Principle	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	InfoMax	89.01±1.13	74.44±0.31	72.85±1.78	76.20±1.06	70.65±1.13	73.03±0.87	82.50±1.42	53.46±1.03
	Lovász	89.67±1.54	75.26±1.43	74.13±1.49	78.21±1.35	71.46±1.21	**73.87±1.32**	84.76±1.86	54.57±1.38
	GeMax	**92.44±1.23***	**76.87±1.99**	**75.43±1.10**	**80.21±1.83**	**73.25±1.17**	73.31±1.53	**86.45±1.43**	**56.16±2.40**
GraphCL	InfoMax	86.80±1.34	74.39±0.45	78.62±0.40	77.87±0.41	71.36±1.15	71.14±0.44	89.53±0.84	55.99±0.28
	Lovász	87.24±1.96	75.87±1.17	**79.14±1.67**	79.13±1.27	72.52±1.37	72.44±1.46	89.87±2.13	56.12±1.73
	GeMax	**88.83±1.10**	**77.60±1.18***	78.24±1.84	**80.89±1.03**	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
AD-GCL	InfoMax	87.13±1.56	73.59±0.65	74.49±0.52	69.67±0.51	73.32±0.61	71.57±1.01	85.52±0.79	53.00±0.82
	Lovász	87.44±2.13	74.29±2.80	76.25±1.48	75.12±1.13	**73.85±1.05**	73.02±1.35	87.11±1.95	54.61±2.35
	GeMax	**88.37±1.05**	**75.96±2.30**	**77.07±1.02**	**77.15±1.83**	72.02±1.54	**73.62±1.56**	**89.88±1.85**	**56.61±1.38**
JOAOv2	InfoMax	86.91±1.01	71.25±0.85	66.91±1.75	72.99±0.75	70.40±2.21	71.60±0.86	78.35±1.38	55.57±2.86
	Lovász	**87.19±1.92**	73.15±1.46	73.15±2.17	74.15±1.67	72.62±1.43	**72.18±1.72**	84.19±1.67	53.74±1.70
	GeMax	86.47±1.19	**74.91±1.90**	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	71.62±1.18	**87.19±1.89**	**55.24±1.28**
AutoGCL	InfoMax	88.64±1.08	75.80±0.36	77.57±0.60	82.00±0.29	70.12±0.68	73.30±0.40	88.58±1.49	56.75±0.18
	Lovász	89.02±1.47	76.23±1.25	78.95±1.39	**82.63±2.12***	71.31±1.72	73.95±1.36	89.41±1.81	57.28±1.62
	GeMax	**90.63±1.15**	**77.12±1.75**	**80.17±0.87**	81.11±1.85	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	**57.86±1.72***
GraphACL	InfoMax	90.21±0.94	75.47±0.38	79.34±0.42	**81.22±1.31**	74.72±0.55	74.29±0.67	88.58±1.49	56.11±1.33
	Lovász	90.55±0.82	75.88±1.36	80.22±1.01	80.57±1.76	74.48±0.78	74.33±1.88	89.30±1.60	56.17±2.07
	GeMax	**91.86±0.70**	**77.29±1.14**	**81.08±1.09***	81.19±1.86	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
GCS	InfoMax	90.45±0.81	75.02±0.39	77.22±0.30	77.37±0.30	75.56±0.41	73.43±0.38	90.98±0.28	57.04±0.49
	Lovász	90.78±1.28	**76.58±0.59**	78.69±0.53	78.35±1.04	**76.62±0.72***	74.58±1.01	91.23±1.44	56.97±1.51
	GeMax	**91.37±0.95**	76.12±0.76	**78.82±0.76**	**80.08±1.50**	76.15±1.46	**75.19±1.76***	**92.51±1.05***	**57.45±1.79**

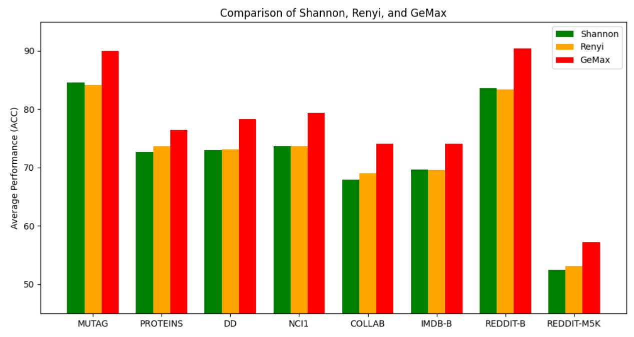


(a) Average ACC of Table 2 (Unsupervised)

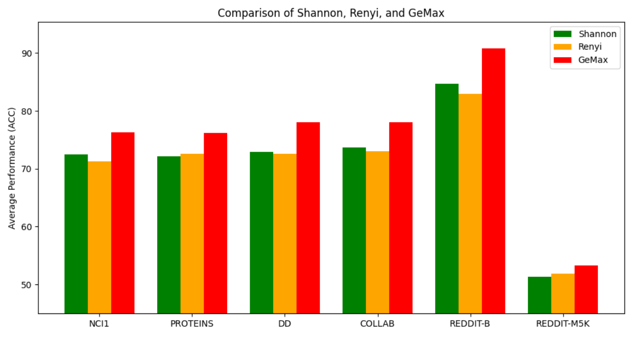


(b) Average ACC of Table 3 (Semi-supervised)

Figure 3: Average results of graph representation learning using different principles



(a) Average ACC of Table 4 (Unsupervised)



(b) Average ACC of Table 6 (Semi-supervised)

Figure 4: Average results of graph representation learning via maximizing different entropy objectives

with results suggesting that regularized optimization yields representations of comparable quality to constrained optimization. Additional investigations include parameter sensitivity analysis (Appendix D.5), ablation studies to elucidate the significance of each component in the GeMax objectives (Appendix D.6), and comparisons of different probability distributions for defining $P_{F(G)}$ (Appendix D.7). We also illustrate the convergence behavior of GeMax (Appendix

Table 3: Classification accuracy (%) of semi-supervised learning using different graph learning principles.

Method	Principle	NCI1	PROTEINS	DD	COLLAB	REDDIT-B	REDDIT-M5K
InfoGraph	InfoMax	73.79±0.44	75.13±0.74	76.99±1.29	73.79±1.25	86.50±1.37	**53.66±1.85**
	Lovász	72.58±1.45	74.56±1.01	**77.05±1.17**	74.82±0.22	88.14±1.71	51.62±0.96
	GeMax	**74.89±1.21**	**75.87±1.94**	75.17±1.95	**76.67±1.39**	**90.67±1.99**	53.08±1.54
GraphCL	InfoMax	74.63±0.25	74.17±0.34	76.17±1.37	74.23±0.21	89.11±0.19	52.55±0.45
	Lovász	75.46±1.53	75.12±1.87	77.46±1.52	76.12±1.15	89.87±1.68	53.69±1.68
	GeMax	**76.59±1.55**	**76.06±1.83**	**78.25±1.23**	**77.73±1.27**	**91.84±1.64***	**54.10±3.22**
AD-GCL	InfoMax	75.18±0.31	73.96±0.47	77.91±0.73	75.82±0.26	90.10±0.15	53.49±0.28
	Lovász	**76.62±1.83**	74.21±1.71	78.27±1.39	76.27±1.74	90.36±1.56	54.06±1.32
	GeMax	76.47±1.82	**76.76±1.94***	**79.20±0.65***	**77.79±1.66**	**91.18±0.82**	**56.64±1.23***
JOAOv2	InfoMax	74.86±0.39	73.31±0.48	75.81±0.72	75.53±0.18	88.79±0.65	52.71±0.28
	Lovász	76.13±1.76	73.73±1.86	76.27±1.48	77.35±1.27	89.31±1.85	53.17±1.76
	GeMax	**77.36±1.48***	**75.18±0.80**	**77.53±1.27**	**78.10±1.75**	**90.38±1.87**	**53.27±2.18**
AutoGCL	InfoMax	73.75±2.25	75.65±1.40	77.50±4.41	77.16±1.48	79.80±1.47	49.91±2.70
	Lovász	75.77±1.48	**76.36±1.57**	78.16±1.61	77.63±1.78	84.64±1.53	51.31±1.81
	GeMax	**76.86±1.98***	76.23±1.47	**79.61±1.21***	**78.21±1.27**	**88.43±1.57**	**51.63±1.97**
GraphACL	InfoMax	74.35±1.17	73.20±1.57	75.71±0.82	75.32±0.13	88.49±0.94	51.70±1.05
	Lovász	75.32±1.38	74.99±1.35	76.33±2.17	77.24±1.77	89.67±1.84	**54.54±1.67***
	GeMax	**76.56±1.67**	**76.89±1.41***	**77.28±0.46**	**79.02±1.20***	**91.38±1.50**	52.98±3.02
GCS	InfoMax	74.79±1.36	75.31±1.21	76.18±1.76	75.06±1.25	87.82±1.74	49.16±4.50
	Lovász	75.37±1.26	76.16±1.28	78.21±1.05	76.86±1.14	84.69±1.56	51.52±0.58
	GeMax	**75.69±1.47**	**76.24±1.04**	**79.13±1.55**	**78.72±1.75***	**92.05±1.45***	**54.49±2.22**

Table 4: Classification accuracy (%) of unsupervised learning via maximizing different entropy objectives

Method	Principle	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	Shannon	84.89±1.84	69.27±1.43	71.77±1.64	72.75±1.44	68.95±1.49	69.62±1.28	81.55±1.49	49.25±1.27
	Rényi	86.50±0.92	71.37±1.93	72.38±1.31	73.17±1.71	67.16±1.78	71.13±1.49	82.92±1.65	52.28±1.46
	GeMax	**92.44±1.23***	**76.87±1.99**	**75.43±1.10**	**80.21±1.83**	**73.95±1.17**	**74.31±1.53**	**86.45±1.43**	**56.16±2.40**
GraphCL	Shannon	85.25±1.47	73.13±1.52	75.43±1.32	76.13±1.28	67.93±1.41	68.95±1.34	84.27±1.34	52.90±1.57
	Rényi	83.19±0.76	74.24±1.75	74.14±1.44	73.11±1.96	71.73±1.29	70.64±1.26	83.91±1.47	53.24±1.21
	GeMax	**88.83±1.10**	**77.60±1.18***	**78.24±1.84**	**80.89±1.03**	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
AD-GCL	Shannon	85.13±0.25	71.31±1.26	73.21±0.74	69.82±1.58	66.14±1.94	69.13±1.21	82.29±1.48	54.86±1.27
	Rényi	82.25±1.32	72.01±0.33	71.12±1.53	72.39±1.89	69.26±1.56	68.86±1.95	81.59±1.57	54.28±1.38
	GeMax	**88.37±1.05**	**75.96±2.30**	**77.07±1.02**	**77.15±1.83**	**73.02±1.54**	**73.62±1.56**	**89.88±1.85**	**56.61±1.38**
JOAOv2	Shannon	84.01±1.69	74.15±1.05	70.81±1.35	70.99±1.28	67.11±1.81	68.22±1.87	80.30±1.82	50.37±1.26
	Rényi	83.39±1.42	76.37±1.57	73.17±1.48	72.18±1.90	68.21±1.52	67.19±1.59	83.29±1.48	51.46±1.73
	GeMax	**89.07±1.19**	**74.91±1.90**	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	**72.62±1.18**	**87.19±1.89**	**55.24±1.28**
AutoGCL	Shannon	81.64±1.68	73.57±1.38	72.48±1.62	74.10±1.96	67.12±1.38	69.92±1.47	82.34±1.92	51.52±1.28
	Rényi	85.02±1.27	72.84±1.59	73.59±1.49	75.43±1.82	65.29±1.27	68.54±1.63	83.17±1.89	53.28±1.27
	GeMax	**90.63±1.15**	**77.12±1.75**	**80.17±0.87**	**81.11±1.85**	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	**57.86±1.72***
GraphACL	Shannon	85.21±1.94	69.59±1.27	74.40±1.62	74.85±1.81	69.16±1.55	70.24±1.68	85.62±1.19	51.14±1.38
	Rényi	86.55±1.33	73.16±1.46	72.82±1.19	73.22±1.26	67.28±1.84	72.13±1.87	83.31±1.69	53.78±2.01
	GeMax	**91.86±0.70**	**77.29±1.14**	**81.08±1.09***	**81.19±1.86***	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
GCS	Shannon	86.12±1.64	74.17±1.43	71.89±1.32	76.13±1.35	69.81±1.52	71.16±2.28	86.80±1.18	54.14±1.29
	Rényi	84.45±1.29	73.23±1.07	73.81±1.14	75.35±1.44	72.16±1.01	70.18±1.52	85.36±1.47	52.63±1.17
	GeMax	**91.37±0.95**	**76.12±0.76**	**78.82±0.76**	**80.08±1.50**	**76.15±1.46***	**75.19±1.76***	**92.51±1.05***	**57.45±1.79**

D.8) and empirically evaluate its time complexity (Appendix D.9). Lastly, we provide the code implementation of key components in GeMax (Appendix D.10).

6. Conclusions

We introduced GeMax, a novel graph representation learning approach based on Körner's graph entropy. GeMax uses local node and global graph representations produced by graph neural networks to efficiently approximate the NP-hard graph entropy computation. Our experiments confirmed GeMax's superiority over its competitors, demonstrated its ability to advance graph representation learning, and showcased the practical use of theoretical concepts of graph theory in machine learning.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No.62376236, the Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001), Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone Project (No.HZQSWS-KCCYB-2024016), and the funding UDF01001770 of The Chinese University of Hong Kong, Shenzhen. The authors appreciate the comments of the reviewers and ACs.

Impact Statement

This work introduces an innovative method for graph representation learning, leveraging graph entropy to generate comprehensive graph-level and node-level representations. By approximating graph entropy through orthonormal representations, our approach enhances the capability of graph neural networks in unsupervised and semi-supervised learning tasks. This advancement possibly has broader impacts for scientific and societal benefits, such as enhancing recommendation systems and advancing biomedical research. Apart from offering possible scientific and societal benefits, to the best of our knowledge, our research works do not involve ethical constraints or religious and political restrictions.

References

Aziz, F., Hancock, E. R., and Wilson, R. C. Network entropy using edge-based information functionals. *Journal of Complex Networks*, 8(3):cnaa015, 2020.

Bai, Y., Ding, H., Qiao, Y., Marinovic, A., Gu, K., Chen, T., Sun, Y., and Wang, W. Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098*, 2019.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Boreland, G. A lower bound on graph entropy. In *Mathematical Proceedings of the Royal Irish Academy*, volume 118, pp. 9–20. Royal Irish Academy, 2018.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Bouchon, B., Saitta, L., and Yager, R. R. *Uncertainty and Intelligent Systems: 2nd International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems IPMU'88. Urbino, Italy, July 4-7, 1988. Proceedings*, volume 313. Springer Science & Business Media, 1988.

Chen, T., Bian, S., and Sun, Y. Are powerful graph neural nets necessary? a dissection on graph classification. *arXiv preprint arXiv:1905.04579*, 2019.

Csiszár, I., Körner, J., Lovász, L., Marton, K., and Simonyi, G. Entropy splitting for antiblocking corners and perfect graphs. *Combinatorica*, 10:27–40, 1990.

Cui, G., Zhou, J., Yang, C., and Liu, Z. Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 976–985, 2020.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 (2):786–797, 1991.

Dehmer, M. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*, 201(1-2):82–94, 2008.

Dehmer, M. and Mowshowitz, A. A history of graph entropy measures. *Information Sciences*, 181(1):57–78, 2011.

Ding, K., Xu, Z., Tong, H., and Liu, H. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

Ding, K., Wang, Y., Yang, Y., and Liu, H. Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7378–7386, 2023.

Dong, M., Chen, H., Wang, Y., and Xu, C. Crafting efficient neural graph of large entropy. In *IJCAI*, pp. 2244–2250, 2019.

Duan, J., Wang, S., Zhang, P., Zhu, E., Hu, J., Jin, H., Liu, Y., and Dong, Z. Graph anomaly detection via multi-scale contrastive learning networks with augmented view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7459–7467, 2023.

Fang, P., Wang, F., Shi, Z., Jiang, H., Feng, D., and Yang, L. Huge: An entropy-driven approach to efficient and scalable graph embeddings. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 2045–2050. IEEE, 2021.

Fang, Y., Zhang, Q., Yang, H., Zhuang, X., Deng, S., Zhang, W., Qin, M., Chen, Z., Fan, X., and Chen, H. Molecular contrastive learning with chemical element knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3968–3976, 2022.

Fuglede, B. and Topsoe, F. Jensen-shannon divergence and hilbert space embedding. In *International symposium onInformation theory, 2004. ISIT 2004. Proceedings.*, pp. 31. IEEE, 2004.

Grebenkina, M., Brachmann, A., Bertamini, M., Kaduhm, A., and Redies, C. Edge-orientation entropy predicts preference for diverse types of man-made images. *Frontiers in Neuroscience*, 12:678, 2018.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

Han, X., Jiang, Z., Liu, N., and Hu, X. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, pp. 8230–8248. PMLR, 2022.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Jensen, T. R. and Toft, B. *Graph coloring problems*. John Wiley & Sons, 2011.

Jiang, L. Y., Shi, J., Cheung, M., Wright, O., and Moura, J. M. Edge entropy as an indicator of the effectiveness of gnns over cnns for node classification. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pp. 746–750. IEEE, 2020.

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

Körner, J. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *6th Prague conference on information theory*, pp. 411–425, 1973.

Kriege, N. and Mutzel, P. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.

Lee, N., Lee, J., and Park, C. Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7372–7380, 2022.

Li, S., Zhou, J., Xu, T., Dou, D., and Xiong, H. Geomgcl: Geometric graph contrastive learning for molecular property prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4541–4549, 2022a.

Li, Y., Chang, H., Ma, B., Shan, S., and Chen, X. Optimal positive generation via latent transformation for contrastive learning. *Advances in Neural Information Processing Systems*, 35:18327–18342, 2022b.

Linsker, R. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Liu, X., Fu, L., and Wang, X. Bridging the gap between von neumann graph entropy and structural information: Theory and applications. In *Proceedings of the Web Conference 2021*, pp. 3699–3710, 2021.

Liu, X., Fu, L., Wang, X., and Zhou, C. On the similarity between von neumann graph entropy and structural information: Interpretation, computation, and applications. *IEEE Transactions on Information Theory*, 68(4): 2182–2202, 2022a.

Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., and Philip, S. Y. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5879–5900, 2022b.

Lovász, L. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

Luo, G., Li, J., Su, J., Peng, H., Yang, C., Sun, L., Yu, P. S., and He, L. Graph entropy guided node embedding dimension selection for graph neural networks. *arXiv preprint arXiv:2105.03178*, 2021.

Luo, X., Ju, W., Gu, Y., Mao, Z., Liu, L., Yuan, Y., and Zhang, M. Self-supervised graph-level representation learning with adversarial contrastive learning. *ACM Transactions on Knowledge Discovery from Data*, 2023a.

Luo, Y., Luo, G., Qin, K., and Chen, A. Graph entropy minimization for semi-supervised node classification, 2023b.

Ma, Y., Zhang, X., Zhang, P., and Zhan, K. Entropy neural estimation for graph contrastive learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 435–443, 2023.

Minello, G., Rossi, L., and Torsello, A. On the von neumann entropy of graphs. *Journal of Complex Networks*, 7(4): 491–514, 2019.

Mo, Y., Peng, L., Xu, J., Shi, X., and Zhu, X. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7797–7805, 2022.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Mowshowitz, A. and Dehmer, M. Entropy and the complexity of graphs revisited. *Entropy*, 14(3):559–570, 2012.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Oggier, F. and Datta, A. Renyi entropy driven hierarchical graph clustering. *PeerJ Computer Science*, 7:e366, 2021.

Pál, D., Póczos, B., and Szepesvári, C. Estimation of rényi entropy and mutual information based on generalized nearest-neighbor graphs. *Advances in Neural Information Processing Systems*, 23, 2010.

Passerini, F. and Severini, S. The von neumann entropy of networks. *arXiv preprint arXiv:0812.2597*, 2008.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1150–1160, 2020.

Rezaei, S. S. C. Entropy and graphs. *arXiv preprint arXiv:1311.5632*, 2013.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Shen, X., Sun, D., Pan, S., Zhou, X., and Yang, L. T. Neighbor contrastive learning on learnable graph augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9782–9791, 2023.

Sun, F.-Y., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Sun, Z., Ding, C., and Fan, J. Lovász principle for unsupervised graph representation learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 15920–15933, 2021.

Trivedi, P., Lubana, E. S., Yan, Y., Yang, Y., and Koutra, D. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proceedings of the ACM Web Conference 2022*, pp. 1538–1549, 2022.

Verma, S. and Zhang, Z.-L. Learning universal graph neural network embeddings with aid of transfer learning. *arXiv preprint arXiv:1909.10086*, 2019.

Wang, J., Wilson, R. C., and Hancock, E. R. Network edge entropy decomposition with spin statistics. *Pattern Recognition*, 118:108040, 2021.

Wang, Y., Wang, Y., Zhang, Z., Yang, S., Zhao, K., and Liu, J. User: Unsupervised structural entropy-based robust graph neural network. *arXiv preprint arXiv:2302.05889*, 2023.

Wei, C., Liang, J., Liu, D., and Wang, F. Contrastive graph structure learning via information bottleneck for recommendation. *Advances in Neural Information Processing Systems*, 35:20407–20420, 2022.

Wei, C., Wang, Y., Bai, B., Ni, K., Brady, D., and Fang, L. Boosting graph contrastive learning via graph contrastive saliency. In *International conference on machine learning*, pp. 36839–36855. PMLR, 2023.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 726–735, 2021.

Wu, J., Chen, X., Xu, K., and Li, S. Structural entropy guided graph hierarchical pooling. In *International conference on machine learning*, pp. 24017–24030. PMLR, 2022.

Wu, T., Ren, H., Li, P., and Leskovec, J. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

Xia, J., Wu, L., Chen, J., Hu, B., and Li, S. Z. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*, pp. 1070–1079, 2022.

Xiao, T., Chen, Z., Guo, Z., Zhuang, Z., and Wang, S. Decoupled self-supervised learning for graphs. *Advances in Neural Information Processing Systems*, 35:620–634, 2022.

Xie, Y., Xu, Z., Zhang, J., Wang, Z., and Ji, S. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

Xu, D., Cheng, W., Luo, D., Chen, H., and Zhang, X. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 30414–30425, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

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 Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 114–122, 2023.

Ye, C., Wilson, R. C., Comin, C. H., Costa, L. d. F., and Hancock, E. R. Approximate von neumann entropy for directed graphs. *Physical Review E*, 89(5):052804, 2014.

Yin, Y., Wang, Q., Huang, S., Xiong, H., and Zhang, X. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8892–8900, 2022.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823, 2020.

You, Y., Chen, T., Shen, Y., and Wang, Z. Graph contrastive learning automated. In *International Conference on Machine Learning*, pp. 12121–12132. PMLR, 2021.

Yu, J., Xu, T., Rong, Y., Bian, Y., Huang, J., and He, R. Recognizing predictive substructures with subgraph information bottleneck. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Zeng, J. and Xie, P. Contrastive self-supervised learning for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10824–10832, 2021.

Zeng, L., Li, L., Gao, Z., Zhao, P., and Li, J. Imgcl: Revisiting graph contrastive learning on imbalanced node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11138–11146, 2023.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021a.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11289–11297, 2023.

Zhang, Z., Liu, Q., Wang, H., Lu, C., and Lee, C.-K. Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, 34:15870–15882, 2021b.

Zhao, H., Yang, X., Wang, Z., Yang, E., and Deng, C. Graph debiased contrastive learning with joint representation clustering. In *IJCAI*, pp. 3434–3440, 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. *arXiv preprint arXiv:2303.09778*, 2023.

A. Notations

The major notations used in this paper are shown in Table 5.

Table 5: Notations

Symbol	Description	Symbol	Description
G	a graph	V	vertex set of graph G
$\mathbf{g}$	graph-level representation of G	n	the number of vertices of G
$\boldsymbol{Z}$	node representations matrix of G	$\boldsymbol{z}_i$	representation of node i
(G, P)	a probabilistic graph	$P(\mathbf{g}, \boldsymbol{Z})$	probability distribution on V
$\text{VP}(G)$	vertex-packing polytope of G	$\text{VP}_{\text{Sub}}(G)$	a subset of $\text{VP}(G)$
$P_i(\mathbf{g}, \boldsymbol{Z})$	probability density of vertex i	$H_k(G, P)$	graph entropy
$H(P)$	Shannon entropy	$\alpha(G)$	the independence number of G
$0 \leq \boldsymbol{a} \leq 1$	$0 \leq a_i \leq 0\ \forall i$		

B. Proof for and Propositions and Theorems

B.1. Proof for Proposition 3.2

Definition B.1 (convex corner (Csiszár et al., 1990; Rezaei, 2013))**.** A subset $\boldsymbol{\mathcal{A}} \subseteq \mathbb{R}^n_+$ is called *convex corner* if it is compact and convex, has non-empty interior, and for every $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, $\boldsymbol{a}' \in \mathbb{R}^n_+$ with $\boldsymbol{a}' \leq \boldsymbol{a}$, then we have $\boldsymbol{a}' \in \boldsymbol{\mathcal{A}}$.

Proof. (1) Based on the Definition B.1 of convex corner, if $\text{VP}_{\text{Sub}}(G)$ is a convex corner, then given $\boldsymbol{a} \in \text{VP}_{\text{Sub}}(G)$, $\boldsymbol{a}' \in \mathbb{R}^n_+$ with $\boldsymbol{a}' \leq \boldsymbol{a}$, we need to imply $\boldsymbol{a}' \in \text{VP}_{\text{Sub}}(G)$.

Since $0 \leq \boldsymbol{a} \leq 1$ and $\boldsymbol{a}' \leq \boldsymbol{a}$, we have $\boldsymbol{a}' \leq 1$. Since $\boldsymbol{a}' \in \mathbb{R}^n_+$, we have $0 \leq \boldsymbol{a}' \leq 1$. Given that $0 \leq \boldsymbol{a} \leq 1$ and $0 \leq \boldsymbol{a}' \leq 1$, then $\boldsymbol{a}' \leq \boldsymbol{a}$ implies $\mathbb{1}(\boldsymbol{a}') \leq \mathbb{1}(\boldsymbol{a})$. Suppose $\boldsymbol{a} \in \text{VP}_{\text{Sub}}(G)$. The definition of $\text{VP}_{\text{Sub}}(G)$ implies that $\mathbb{1}(\boldsymbol{a}) \in \text{VP}(G)$. As $\text{VP}(G)$ is a convex corner and $\mathbb{1}(\boldsymbol{a}') \leq \mathbb{1}(\boldsymbol{a})$, we have $\mathbb{1}(\boldsymbol{a}') \in \text{VP}(G)$. Together with the fact that $0 \leq \boldsymbol{a}' \leq 1$, we conclude that $\boldsymbol{a}'$ is in $\text{VP}_{\text{Sub}}(G)$, meaning that $\text{VP}_{\text{Sub}}(G)$ is a convex corner.

(2) Based on the property of the element-wise indicator function $\mathbb{1}(\cdot)$, we have $\boldsymbol{b}_i = \mathbb{1}(\boldsymbol{b}_i)$. The Definition 2.1 indicates $\boldsymbol{b}_i \in \text{VP}(G),\ \ \forall i \in [N_b]$. Then we have $\mathbb{1}(\boldsymbol{b}_i) \in \text{VP}(G),\ \ \forall i \in [N_b]$. That is, $\boldsymbol{b}_i \in \text{VP}_{\text{Sub}}(G),\ \ \forall i \in [N_b]$. □

B.2. Proof for Theorem 3.3

Proof. For convenience, let $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n]^\top = \boldsymbol{D_a}\boldsymbol{Z}$, where $\boldsymbol{v}_i = a_i\boldsymbol{z}_i$, $i \in [n]$. Then

$$\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \boldsymbol{V}\boldsymbol{V}^\top = \left[a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j\right]_{n\times n} = \begin{bmatrix} a_1a_1\boldsymbol{z}_1^\top\boldsymbol{z}_1 & a_1a_2\boldsymbol{z}_1^\top\boldsymbol{z}_2 & a_1a_3\boldsymbol{z}_1^\top\boldsymbol{z}_3 & \cdots & a_1a_n\boldsymbol{z}_1^\top\boldsymbol{z}_n \\ a_2a_1\boldsymbol{z}_2^\top\boldsymbol{z}_2 & a_2a_2\boldsymbol{z}_2^\top\boldsymbol{z}_2 & a_2a_3\boldsymbol{z}_2^\top\boldsymbol{z}_3 & \cdots & a_2a_n\boldsymbol{z}_2^\top\boldsymbol{z}_n \\ a_3a_1\boldsymbol{z}_3^\top\boldsymbol{z}_3 & a_3a_2\boldsymbol{z}_3^\top\boldsymbol{z}_2 & a_3a_3\boldsymbol{z}_3^\top\boldsymbol{z}_3 & \cdots & a_3a_n\boldsymbol{z}_3^\top\boldsymbol{z}_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_na_1\boldsymbol{z}_n^\top\boldsymbol{z}_n & a_na_2\boldsymbol{z}_n^\top\boldsymbol{z}_2 & a_na_3\boldsymbol{z}_n^\top\boldsymbol{z}_3 & \dots & a_na_n\boldsymbol{z}_n^\top\boldsymbol{z}_n \end{bmatrix}.$$

(1) First, we prove that $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \boldsymbol{D}_{\boldsymbol{a}}^2 \Longrightarrow \boldsymbol{a} \in \text{VP}_{\text{Sub}}(G)$.

$\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \boldsymbol{D}_{\boldsymbol{a}}^2$ indicates that all the off-diagonal elements are zeros. Thus, we have

$$a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j = 0, \qquad \forall (i,j) \in E \tag{22}$$

Under the condition $\boldsymbol{z}_i^\top \boldsymbol{z}_j \neq 0$ for every $(i,j) \in E$, we conclude from (22) that

$$a_i = 0 \ \text{ or } \ a_j = 0, \qquad \forall (i,j) \in E. \tag{23}$$

Let $\boldsymbol{c} = [c_1, ..., c_n]^\top = \mathbb{1}(\boldsymbol{a}) \in \{0,1\}^n$, where $\mathbb{1}(\cdot)$ was defined as before. Using (23), we obtain

$$c_i = 0 \ \text{ or } \ c_j = 0, \quad \forall (i,j) \in E. \tag{24}$$

This means that there are no adjacent vertex pairs in the subset induced by $\boldsymbol{c}$. In other words, $\boldsymbol{c}$ is an indicator vector of an independent set of G. Based on the second property of $\text{VP}_{\text{Sub}}(G)$ in Proposition 3.2, all the indicator vectors of the independent set are contained in $\text{VP}_{\text{Sub}}(G)$, meaning that

$$\mathbb{1}(\boldsymbol{a}) = \boldsymbol{c} \in \text{VP}_{\text{Sub}}(G). \tag{25}$$

According to the fact $\boldsymbol{a} \leq \boldsymbol{c}$ and the convex corner property of $\text{VP}_{\text{Sub}}(G)$ in Proposition 3.2, we concluded that

$$\boldsymbol{a} \in \text{VP}_{\text{Sub}}(G).$$

(2) Now we prove $\boldsymbol{a} \in \text{VP}_{\text{Sub}}(G) \Longrightarrow \boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \boldsymbol{D}_{\boldsymbol{a}}^2$.

Definition 3.1 of $\text{VP}_{\text{Sub}}(G)$ indicates that if $\boldsymbol{a} \in \text{VP}_{\text{Sub}}(G)$, then $\boldsymbol{c} = \mathbb{1}(\boldsymbol{a}) \in \text{VP}(G)$. Following Definition 2.1 of $\text{VP}(G)$, we use $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{N_b}] \in \{0,1\}^{|V| \times N_b}$ to denote the indicator matrix of the independent sets, where $\boldsymbol{b}_i = [\boldsymbol{b}_{i(1)}, \ldots, \boldsymbol{b}_{i(n)}]^\top$ is the indicator vector of the i-th independent set. Then we can find a vector $\boldsymbol{\lambda}$ with $\boldsymbol{\lambda} \geq 0$ and $\sum_i \lambda_i = 1$, such that $\boldsymbol{c}$ is the combination of indicator vectors of independent sets, i.e.,

$$\boldsymbol{c} = \boldsymbol{B}\boldsymbol{\lambda} = \sum_i \lambda_i \boldsymbol{b}_i. \tag{26}$$

For convenience, we define a positive index set as

$$\mathcal{Q} := \{i : \lambda_i > 0,\ i \in [N_b]\}. \tag{27}$$

Let c_j be the j-th element of $\boldsymbol{c}$, we have $c_j = \sum_{i \in \mathcal{Q}} \lambda_i \boldsymbol{b}_{i(j)}$. Since every i in the set $\mathcal{Q}$ is positive, if $\boldsymbol{c}_j = 0$, then the j-th element of all the indicator vectors $\boldsymbol{b}_{i(j)}$ must be zero, namely, $\boldsymbol{b}_{i(j)} = 0$ for every $i \in \mathcal{Q}$.

Since each element of $\boldsymbol{b}_i$ is zero or one, we have

$$c_j = \sum_{i \in \mathcal{Q}} \lambda_i \boldsymbol{b}_{i(j)} \leq \sum_{i \in \mathcal{Q}} \lambda_i = 1. \tag{28}$$

The equality, say $c_j = 1$, can be obtained only when $\boldsymbol{b}_{1(j)} = \boldsymbol{b}_{2(j)} = \cdots = \boldsymbol{b}_{|\mathcal{Q}|(j)} = 1$.

The analysis above indicates that if $c_j = 0$, the j-th element of every $\boldsymbol{b}_i$ must be zero. Meanwhile, if $c_j = 1$, the j-th element of every $\boldsymbol{b}_i$ must be one. These indicate that

$$\boldsymbol{c} = \boldsymbol{b}_i, \quad \forall i \in \mathcal{Q}, \tag{29}$$

which means $|Q| = 1$, i.e., $\mathcal{Q}$ contains a single element. Therefore, $\boldsymbol{c}$ is actually one of the indicator vectors of the independent sets. We then have $c_i = 0$ or $c_j = 0$ for every $(i,j) \in E$, which further means

$$a_i = 0 \ \text{ or } \ a_j = 0, \quad \forall (i,j) \in E. \tag{30}$$

Now we can analyse the elements of $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \left[a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j\right]_{n \times n}$. Equation (30) implies that $a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j = 0$ holds for every $(i,j) \in E$. For every $(i,j) \notin E$, we have $\boldsymbol{z}_i^\top \boldsymbol{z}_j = 0$ under the condition that $\boldsymbol{Z}$ are orthonormal representations, and thus $a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j = 0$. Combining these two cases, we have $a_i a_j \boldsymbol{z}_i^\top \boldsymbol{z}_j = 0$ for every off-diagonal element (i,j) of $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a}$.

Each diagonal element of $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a}$ is denoted as $a_i^2 \boldsymbol{z}_i^\top \boldsymbol{z}_i$. Since $\boldsymbol{Z}$ are orthonormal representations, we have $\boldsymbol{z}_i^\top \boldsymbol{z}_i = 1$. Thus, the i-th element in the diagonal of $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a}$ is a_i^2.

The above analysis means $\boldsymbol{D_a}(\boldsymbol{Z}\boldsymbol{Z}^\top)\boldsymbol{D_a} = \boldsymbol{D}_{\boldsymbol{a}}^2$. We finished the proof. □

B.3. Proof for Proposition 3.4

Proof. In the algorithm, we need to store N graphs and each graph has a sparse adjacency matrix of n nodes and m edges and a feature matrix of size $n \times d$. The number of parameters of F_g and F_Z is $\mathcal{O}(Ld^2)$ because there are $2L$ matrices of size $d \times d$. When applying F_g or F_Z to each graph, the neighbor aggregation and feature transformation operations will yield $2L$ matrices of size $n \times d$. We also need to store P_i, a vector of size n for each graph. For the regularization $\mathcal{L}_{\text{orth}}$, we need to store $\boldsymbol{M}_j$ and $\boldsymbol{Z}_j^{\phi}(\boldsymbol{Z}_j^{\phi})^{\top}$, requiring $\mathcal{O}(Nn^2)$, and the storage for $\mathcal{L}_{\text{s-vp}}$ is similar. Putting these together, the total space complexity of the algorithm is $\mathcal{O}(N(m + Ldn + n^2) + Ld^2)$.

Regarding the time complexity, in each iteration of the subproblem for $\{\theta, \phi\}$ (i.e., line 4 in the algorithm), the neighbor aggregation and feature transformation operations in F_g or F_Z are $\mathcal{O}(L(dm + d^2n))$ on each graph, and computing $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{s-vp}}$ requires $\mathcal{O}(Ndn^2)$. The complexity of the backward propagation is similar to that of the forward propagation. The subproblem for $\mathcal{A}$ (i.e., line 5 in the algorithm) requires $\mathcal{O}(\tau_2 Nn^2)$ plus $\mathcal{O}(Ndn^2)$ because $\boldsymbol{D}_{\boldsymbol{a}_j}$ are diagonal matrices and $\boldsymbol{Z}_j^{\phi}(\boldsymbol{Z}_j^{\phi})^{\top}$ are precomputed in line 4. Putting these together, the total time complexity of the algorithm is $\mathcal{O}(T(\tau_1 N(L(dm + d^2n) + dn^2) + \tau_2 Nn^2))$. □

C. Theoretical Foundations of Graph Entropy

Körner's graph entropy is a fundamental concept in the fields of information theory, graph theory, and combinatorics.

Information Theory Körner's graph entropy, rooted in information theory, expands traditional entropy to graph structures and was originally devised to assess communication channel capacity. Established by Claude E. Shannon in 1948 (Shannon, 1948) and further developed by János Körner in 1973 (Körner, 1973), graph entropy measures information transmission over noisy channels, highlighting the shared information within graph elements (Bouchon et al., 1988). László Lovász's introduction of orthonormal representations in 1979 (Lovász, 1979) further advanced this field, focusing on a graph's Shannon capacity and underscoring its structural properties.

Graph Theory In graph theory, graph entropy pertains to probabilistic graphs, denoted as (G, P), where P represents the probability distribution over the vertex set (Rezaei, 2013). Graph entropy measures the uncertainty or randomness inherent in a probabilistic graph. The interpretation of the probability distribution P varies depending on the context. For instance, P might represent the centrality of a vertex, or in the case where G symbolizes a communication channel, P could denote the probability of various communication symbols.

Combinatorics In combinatorics, graph entropy plays a significant role in analyzing and quantifying the complexity and informational content of graph structures (Bouchon et al., 1988). This concept focuses on assessing the combinatorial properties of graphs, such as the arrangement and interrelation of vertices and edges, as well as various subgraph configurations like cliques and independent sets.

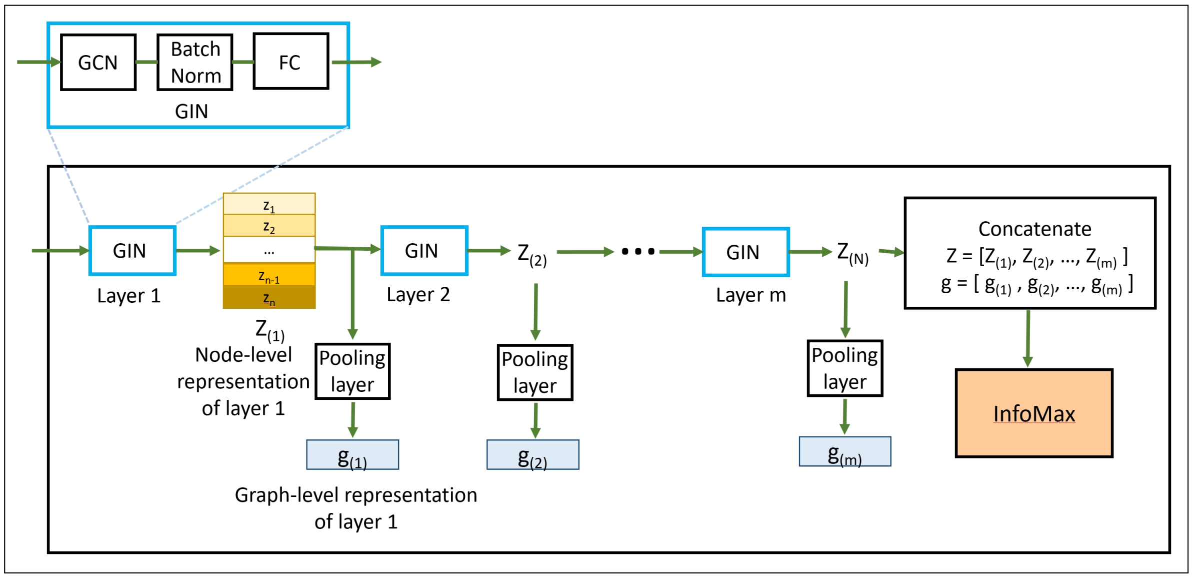


Figure 5: Architecture of InfoGraph with m layers

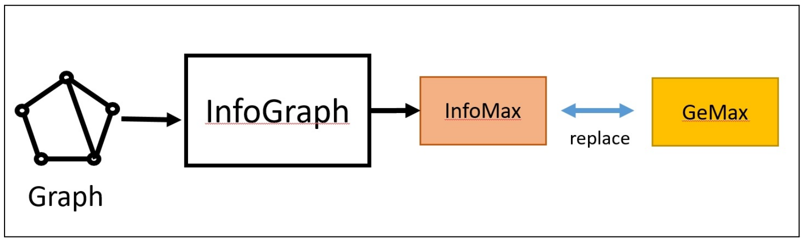


Figure 6: Applying GeMax to InfoGraph network by replacing the InfoMax loss

D. More about the Experiments

D.1. Experiment Baseline: InforMax methods

For unsupervised and semi-supervised graph-level learning, those InfoMax-based methods are the most current and influential methods, each boasting high citations on Google Scholar. In experiments, we replace the InfoMax objective with our GeMax objective while keeping other settings unchanged, as shown in Figure 6. In this work, we compare seven InfoMax-based methods, InfoGraph (Sun et al., 2019) including GraphCL (You et al., 2020), AD-GCL (Suresh et al., 2021), JOJOv2 (You et al., 2021), AutoGCL (Yin et al., 2022), GraphACL (Luo et al., 2023a) and GCS (Wei et al., 2023). All the other six methods share the same graph representation learning architecture with InfoGraph (Sun et al., 2019), as shown in Figure 5.

Unsupervised InfoGraph Following (Nowozin et al., 2016; Sun et al., 2019; Belghazi et al., 2018), suppose the node-level representation $\boldsymbol{z}_p(x)$ and the graph-level representation $\boldsymbol{g}(x)$ are depending on the input x, T_φ is a discriminator parameterized by a neural network with parameters φ, the Jensen-Shannon mutual information (MI) estimator (Fuglede & Topsoe, 2004; Nowozin et al., 2016; Hjelm et al., 2019; Sun et al., 2019) I_φ between $\boldsymbol{z}_v$ and $\boldsymbol{g}$ is defined as

$$I_\varphi(\boldsymbol{z}_p, \boldsymbol{g}) = \mathbb{E}_{\mathbb{P}}[-\text{sp}(-T_\varphi(\boldsymbol{z}_p(x), \boldsymbol{g}(x)))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\text{sp}(T_\varphi(\boldsymbol{z}_p(x'), \boldsymbol{g}(x)))], \tag{31}$$

where x is the input sample from distribution $\mathbb{P}$, x' is the negative sample from distribution $\tilde{\mathbb{P}}$, and $\text{sp}(a) = \log(1 + e^a)$ denotes the softplus function. $\mathbb{P}$ is the empirical probability distribution of the input space and $\tilde{\mathbb{P}}$ is the empirical probability distribution of the negative input space. Many recent graph-level representation learning methods (Sun et al., 2019; You et al., 2020; Yin et al., 2022) are based on the InfoMax principle, i.e., maximizing (31). For example, InfoGraph(Sun et al., 2019) obtains graph-level representations by maximizing the mutual information between the graph-level representation and the node-level representations as follows

$$\phi^*, \theta^*, \varphi^* = \underset{\phi,\theta,\varphi}{\arg\max} \sum_{i=1}^{|\mathcal{G}|} \frac{1}{|V_i|} \sum_{p\in V_i} I_\varphi(\boldsymbol{z}_p^\theta, \boldsymbol{g}_i^\phi), \tag{32}$$

where I_φ is the Jensen-Shannon MI estimator defined by (31).

Semi-supervised InfoGraph For semi-supervised learning, the dataset $\mathcal{G}$ is split into labeled dataset $\mathcal{G}^L$ and unlabeled dataset $\mathcal{G}^U$. They deploy another supervised encoder with parameter ψ and then generate the supervised node-level representations $\boldsymbol{Z}_i^\psi$, graph-level representations $\boldsymbol{g}_i^\psi$ and prediction $\hat{\boldsymbol{y}}_i^\psi$. The loss function of InfoGraph for semi-supervised learning is defined as follows:

$$\mathcal{L}_{\text{info-semi}} = \sum_{l=1}^{|\mathcal{G}^L|} \mathcal{L}_{\text{supervised}}(\hat{\boldsymbol{y}}_l^\psi, \boldsymbol{y}_l) + \sum_{i=1}^{|\mathcal{G}|} \mathcal{L}_{\text{unsupervised}}(\boldsymbol{Z}_i^\theta, \boldsymbol{g}_i^\phi) - \lambda \sum_{i=1}^{|\mathcal{G}|} \frac{1}{|V_i|} I_\varphi(\boldsymbol{g}_i^\phi, \boldsymbol{g}_i^\psi) \tag{33}$$

where $\mathcal{L}_{\text{unsupervised}}$ is derived from (17). The last term encourages the representations learned by the two encoders to have high mutual information.

D.2. Experiment: Entropy Comparison

Table 6 shows the performance of semi-supervised learning via maximizing different entropy objectives: Shannon, Rényi, and our proposed GeMax. Figures 7 and 8 depict the t-SNE visualization of representations learned via these different entropy objectives on the PROTEINS and MUTAG datasets. We see that the representations given by our GeMax are more discriminative than those given by Shannon entropy and Renyi entropy.

Table 6: Performance (ACC) of semi-supervised learning via maximizing different entropy objectives.

methods and entropy		NCI1	PROTEINS	DD	COLLAB	REDDIT-B	REDDIT-M5K
InfoGraph	Shannon	71.10±1.37	71.49±1.20	73.85±1.10	74.02±1.20	83.75±2.18	51.39±1.80
	Rényi	70.23±1.92	72.32±1.51	72.97±1.78	72.71±1.66	82.83±1.20	52.02±1.02
	GeMax	**74.89±1.21**	**75.87±1.94**	**75.17±1.95**	**76.67±1.39**	**90.67±1.99**	**53.08±1.54**
GraphCL	Shannon	72.58±0.57	71.95±1.97	70.88±1.72	71.35±1.45	80.41±1.48	50.68±1.06
	Rényi	71.53±1.34	73.28±1.32	71.40±1.03	73.29±1.60	81.32±1.74	52.63±1.38
	GeMax	**76.59±1.55**	**76.06±1.83**	**78.25±1.23**	**77.73±1.27**	**91.84±1.64**	**54.10±3.22**
AD-GCL	Shannon	72.53±0.61	71.83±1.19	73.07±1.98	75.10±1.14	85.94±0.72	52.35±1.29
	Rényi	70.12±1.61	72.37±1.37	71.29±1.27	70.08±1.86	84.53±1.10	51.37±1.24
	GeMax	**76.47±1.82**	**76.76±1.94**	**79.20±0.65**	**77.79±1.66**	**91.18±0.82**	**56.64±1.23***
JOAOv2	Shannon	73.19±1.36	71.72±1.28	71.18±1.79	74.72±1.80	84.97±2.09	52.29±1.66
	Rényi	72.39±1.77	70.23±1.62	75.84±1.12	75.42±1.71	83.20±1.48	51.34±1.51
	GeMax	**77.36±1.48***	**75.18±0.80**	**77.53±1.27**	**78.10±1.75**	**90.38±1.87**	**53.27±2.18**
AutoGCL	Shannon	73.44±1.44	70.09±1.09	73.37±1.90	73.08±2.07	85.31±1.28	50.53±1.73
	Rényi	70.65±1.89	72.74±1.10	72.51±1.50	72.51±1.04	82.76±1.08	51.14±1.65
	GeMax	**76.86±1.98**	**76.43±1.47**	**79.61±1.21***	**78.21±1.27**	**88.43±1.57**	**51.63±1.97**
GraphACL	Shannon	71.27±1.29	77.07±1.63	73.42±1.33	73.80±1.21	86.82±1.44	50.91±1.21
	Rényi	73.95±1.76	75.09±0.95	72.12±1.92	71.83±1.14	83.61±1.70	53.23±1.66
	GeMax	**76.56±1.67**	**76.89±1.41***	**77.28±0.46**	**79.02±1.20***	**91.38±1.50**	**52.98±3.02**
GCS	Shannon	73.10±1.74	70.80±1.25	74.92±1.59	73.73±1.31	85.66±1.01	50.89±1.21
	Rényi	70.20±1.58	72.51±1.95	72.18±1.88	75.75±1.30	82.76±1.49	51.60±1.70
	GeMax	**75.69±1.47**	**76.24±1.04**	**79.13±1.55**	**78.72±1.75**	**92.05±1.45***	**54.49±2.22**

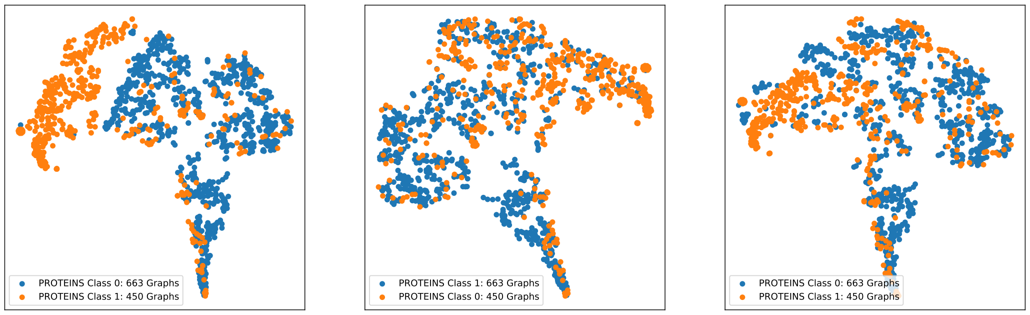


Figure 7: t-SNE visualization of PROTEINS datasets Representations: GeMax(Left), Shannon entropy(Middle), Renyi entropy(Right)

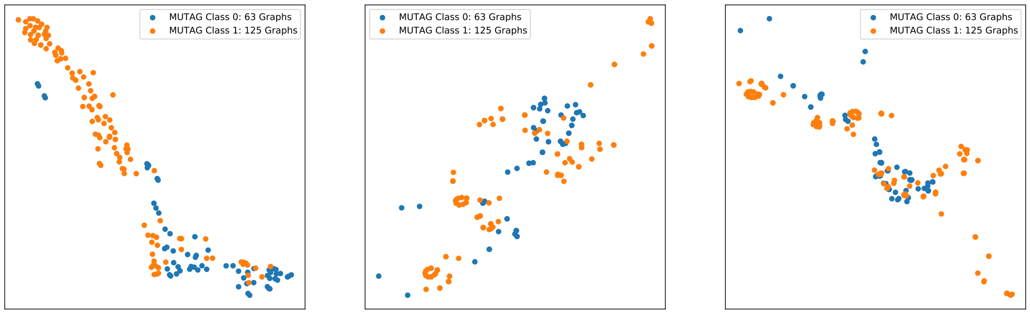


Figure 8: t-SNE visualization of MUTAG datasets Representations: GeMax(Left), Shannon entropy(Middle), Renyi entropy(Right)

D.3. Experiment: Performance Comparison of Exact Computation of Graph Entropy on Small Graphs

For small graphs, we can precisely identify all independent sets via exhaustive search, allowing us to accurately calculate the vertex-packing polytope VP(G) and solve the GeMax problem Eq. (5) exactly, circumventing the need for approximation. We select those graphs with 20 or fewer vertices from the MUTAG and IMDB-B datasets as our toy graph datasets. For the MUTAG dataset, there are 128 graphs (58 for Class 0 and 70 for Class 1) with $n \leq 20$. For the IMDB-B dataset, there are a total of 696 selected graphs, 359 and 337 graphs with $n \leq 20$ for Class 0 and Class 1, respectively. We conduct unsupervised representation learning, followed by classification performance evaluation. Table 7 gives the performance results, which illustrate that the exact computation of GeMax not only surpasses benchmarks such as InfoMax and Lovász but also significantly outperforms the approximate GeMax solution.

Table 7: Performance (ACC%) of unsupervised learning via exact computation of graph entropy and the other principles

Model	Principle	MUTAG$_{n\leq 20}$	IMDB-B$_{n\leq 20}$
InfoGraph	InfoMax	85.12 ± 1.46	71.26 ± 1.75
	Lovász	86.23 ± 1.17	72.57 ± 1.24
	Approx. GeMax	90.04 ± 1.56	73.42 ± 1.60
	Exact. GeMax	$\mathbf{94.36 \pm 1.12}$	$\mathbf{75.35 \pm 1.34}$
AD-GCL	InfoMax	85.22 ± 1.35	71.09 ± 1.18
	Lovász	87.75 ± 1.10	72.24 ± 1.56
	Approx. GeMax	89.68 ± 1.58	73.28 ± 1.71
	Exact. GeMax	$\mathbf{93.25 \pm 1.63}$	$\mathbf{74.89 \pm 1.63}$
JOAOv2	InfoMax	86.29 ± 1.91	72.52 ± 1.65
	Lovász	88.03 ± 1.04	72.83 ± 1.28
	Approx. GeMax	90.37 ± 1.64	73.11 ± 1.85
	Exact. GeMax	$\mathbf{92.52 \pm 1.49}$	$\mathbf{75.69 \pm 1.41}$

D.4. Experiment: Inexact Penalty Method

Algorithm 2 Inexact Penalty Method for GeMax (11) and (12)

```
Input: $\mathcal{G}$, $\mu^{(0)}$, $\gamma^{(0)}$, $t = 0$.
 1: Random initialization of parameters: $\theta^{(0)}$, $\phi^{(0)}$.
 2: Let $\mathcal{A}^{(0)} = \{\boldsymbol{a}_j^{(0)}\}_{j=1}^N = \{[1/n_j, ..., 1/n_j]\}_{j=1}^N$.
 3: Compute $\mathcal{J}^{(0)} := \mathcal{J}(\mathcal{G}; \theta^{(0)}, \phi^{(0)}, \mathcal{A}^{(0)})$.
 4: repeat
 5:     $\theta^{(t+1)}, \phi^{(t+1)} = \text{argmax}_{\theta,\phi}\, \mathcal{J}_1(\mathcal{G}; \theta, \phi, \mathcal{A}^{(t)})$.
 6:     $\bar{\mathcal{A}}^{(t+1)} = \text{argmin}_{\mathcal{A}}\, \mathcal{J}_2(\mathcal{G}; \theta^{(t+1)}, \phi^{(t+1)}, \mathcal{A})$.
 7:     $\boldsymbol{a}_j^{(t+1)} = \text{Proj}_{[0,1]}(\bar{\boldsymbol{a}}_j^{(t+1)}), \ \forall \bar{\boldsymbol{a}}_j^{(t+1)} \in \bar{\mathcal{A}}^{(t+1)}$.
 8:     Compute $\mathcal{J}^{(t+1)}$.
 9:     $\mu^{(t+1)} = \mu^{(t)} \times 1.01, \gamma^{(t+1)} = \gamma^{(t)} \times 1.01$
10: until Convergence
Output: $\theta^{(t+1)}$, $\phi^{(t+1)}$.
```

We introduce an inexact penalty algorithm (Algorithm 2) for solving the GeMax problem. By incrementally increasing values of μ and γ, the constraints are progressively satisfied. Utilizing Algorithm 2, we replicate the unsupervised experiments and present the findings in Table 8. The outcomes indicate that the regularized optimization yields representations comparable in quality to those obtained through constrained optimization.

D.5. Experiment: Sensitivity Analysis of Hyperparameters

In the alternative algorithm, as described in Algorithm 1, two critical hyperparameters require tuning: the orthonormal representation regularization parameter μ, and the subset of VP(G) objective regularization parameter γ. In this section, we analyze the sensitivity of these parameters using the average performance across methods such as InfoGraph, GraphCL, AD-GCL, JOAOv2, AutoGCL, GraphACL, and GCS, with different hyperparameter settings. We repeat each experiment ten times and plot the average accuracy with its variance on different datasets.

Table 8: Performance (ACC) of unsupervised learning. regularized opt. denotes the regularized algorithm 1 and constrained opt. denotes the exact algorithm 2.The **bold** numbers denote the better performances of the same method.

algorithms		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	regularized opt.	**92.44±1.23**	76.87±1.99	**75.43±1.10**	80.21±1.83	**73.95±1.17**	**74.31±1.53**	86.45±1.43	56.16±2.40
	constrained opt.	89.86±1.64	**77.60±1.05**	74.63±1.79	**81.62±1.89**	73.79±1.68	73.84±0.97	**87.29±1.20**	**57.93±1.44**
GraphCL	regularized opt.	88.83±1.10	**77.60±1.18**	78.24±1.84	80.89±1.03	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
	constrained opt.	**89.96±0.13**	75.30±0.52	**79.13±1.81**	**82.87±2.28**	72.13±1.79	73.86±0.80	89.45±1.93	56.42±1.07
AD-GCL	regularized opt.	88.37±1.05	75.96±2.30	**77.07±1.02**	77.15±1.83	**73.02±1.54**	73.62±1.56	**89.88±1.85**	56.61±1.38
	constrained opt.	**90.19±1.94**	**77.18±1.21**	75.76±1.30	**78.77±1.17**	71.09±0.74	**73.72±1.93**	87.97±0.25	**56.64±1.30**
JOAOv2	regularized opt.	89.07±1.19	**74.91±1.90**	74.68±1.91	**76.15±1.01**	**73.17±1.88**	72.62±1.18	87.19±1.89	55.24±1.28
	constrained opt.	**90.80±1.28**	74.29±1.09	**75.87±1.52**	74.87±1.75	71.70±1.16	**73.94±1.60**	**87.69±1.93**	**56.15±1.09**
AutoGCL	regularized opt.	90.63±1.15	77.12±1.75	**80.17±0.87**	81.11±1.85	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	57.86±1.72
	constrained opt.	**91.78±1.35**	**78.95±2.30**	77.43±1.04	**82.46±1.82**	70.13±1.28	72.65±1.32	91.64±1.19	**58.09±0.93**
GraphACL	regularized opt.	**91.86±0.70**	**77.29±1.14**	**81.08±1.09**	**81.19±1.86**	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
	constrained opt.	89.63±1.66	76.81±2.07	80.78±4.75	79.20±1.92	**76.32±1.22**	74.79±1.66	88.01±2.01	56.17±2.15
GCS	regularized opt.	**91.37±0.95**	76.12±0.76	78.82±0.76	80.08±1.50	**76.15±1.46**	75.19±1.76	**92.51±1.05**	**57.45±1.79**
	constrained opt.	89.88±2.10	**77.91±1.98**	**80.67±1.06**	**80.34±2.59**	74.47±1.08	**77.72±2.03**	89.51±1.24	55.82±1.29

D.5.1. Tuning the Orthonormal Representation Regularization Parameter μ

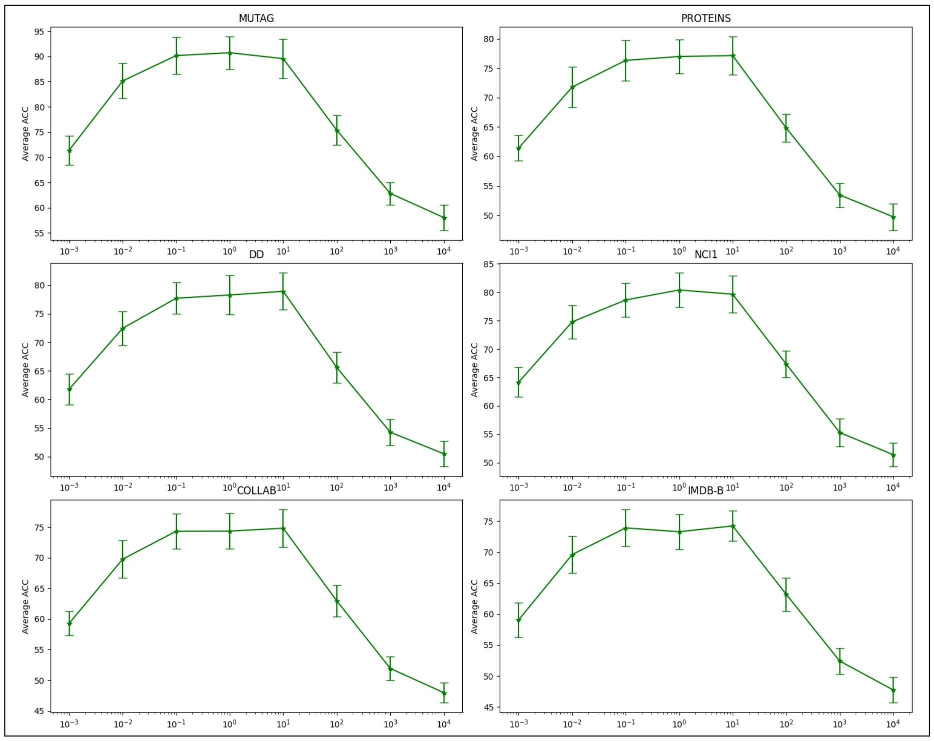


Figure 9: The average ACC of different μ values on various datasets.

With γ fixed at 0.5, we tune μ as the hyperparameter for orthonormal representation regularization. In Figure 9, we fix other hyperparameters and vary μ across $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3, 10^4\}$. The results demonstrate that μ is not sensitive within the range $0.1 \leq \mu \leq 10$. A too small μ leads to a decrease in performance due to inadequacies in achieving orthonormal representations, as postulated in Theorem 3.3. This inadequacy negatively impacts the model's overall performance. Conversely, an excessively large μ can be detrimental, as the orthonormal representation regularization starts to dominate the learning process, overshadowing other important aspects of the model.

D.5.2. Tuning the Subset of VP(G) Objective Regularization Parameter γ

Fixing μ at 0.5, we tune γ as the hyperparameter for the subset of VP(G) objective regularization. In Figure 10, other hyperparameters are kept constant while γ is varied across $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3, 10^4\}$. The findings indicate that γ is not particularly sensitive within the range $0.1 \leq \gamma \leq 10$. A very small γ value disrupts the definition of graph

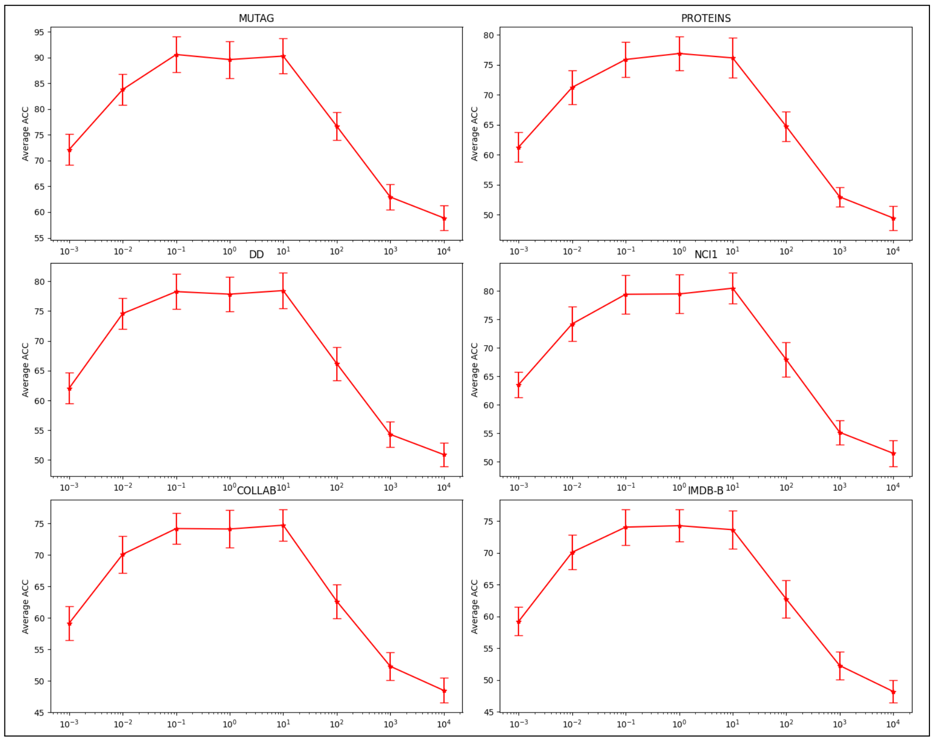


Figure 10: The average ACC of different γ values on various datasets.

entropy, as the subset of VP(G) loses significance. This loss adversely affects the overall performance of the model. On the other hand, a very large γ can be detrimental, as it causes the regularization parameter to dominate the representation learning, thereby impairing the model's performance.

D.6. Experiment: Ablation Study

In the ablation study, we analyze the importance of each part of GeMax objective $\mathcal{J}_1(\mathcal{G}; \theta, \phi, \mathcal{A})$ and $\mathcal{J}_2(\mathcal{G}; \theta, \phi, \mathcal{A})$.

D.6.1. REMOVE THE GRAPH ENTROPY LOSS $\mathcal{L}_{H_k}(\mathcal{G})$

We eliminate the graph entropy loss $\mathcal{L}_{H_k}(\mathcal{G})$ from the GeMax objectives $\mathcal{J}_1(\mathcal{G}; \theta, \phi, \mathcal{A})$ and $\mathcal{J}_2(\mathcal{G}; \theta, \phi, \mathcal{A})$. As shown in Table 9, omitting graph entropy maximization adversely impacts the training of the probability distribution for node and graph representations. Consequently, this results in a significant

Table 9: Performance (ACC) of unsupervised learning for Ablation study. The **bold** numbers denote the better performances of the same method.

remove $\mathcal{L}_{H_k}(\mathcal{G})$		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	no	**92.44±1.23**	**76.87±1.99**	**75.43±1.10**	**80.21±1.83**	**73.95±1.17**	**74.31±1.53**	**86.45±1.43**	**56.16±2.40**
	yes	63.42±3.63	54.24±1.97	53.82±3.01	61.71±4.42	52.25±4.83	59.80±3.90	57.54±5.66	44.20±4.30
GraphCL	no	**88.83±1.10**	**77.60±1.18**	**78.24±1.84**	**80.89±1.03**	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
	yes	63.15±2.90	54.09±6.60	59.86±5.31	60.40±5.09	54.30±3.34	53.50±3.87	62.81±3.43	41.75±2.74
AD-GCL	no	**88.37±1.05**	**75.96±2.30**	**77.07±1.02**	**77.15±1.83**	**73.02±1.54**	**73.62±1.56**	**89.88±1.85**	**56.61±1.38**
	yes	62.67±2.73	56.61±6.50	59.25±3.47	63.99±4.07	61.90±4.55	58.28±4.63	61.44±2.09	45.59±4.97
JOAOv2	no	**89.07±1.19**	**74.91±1.90**	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	**72.62±1.18**	**87.19±1.89**	**55.24±1.28**
	yes	66.37±3.55	65.59±5.21	63.01±1.14	60.28±2.73	62.17±1.26	63.14±2.24	58.42±0.75	47.42±3.20
AutoGCL	no	**90.63±1.15**	**77.12±1.75**	**80.17±0.87**	**81.11±1.85**	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	**57.86±1.72**
	yes	69.45±4.04	53.93±2.08	66.66±2.05	62.01±4.33	56.98±6.59	53.52±7.33	58.23±2.09	48.81±3.15
GraphACL	no	**91.86±0.70**	**77.29±1.14**	**81.08±1.09**	**81.19±1.86**	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
	yes	63.71±7.08	66.05±2.37	65.96±7.96	53.45±1.87	67.45±2.61	65.67±1.92	63.63±3.08	42.68±4.32
GCS	no	**91.37±0.95**	**76.12±0.76**	**78.82±0.76**	**80.08±1.50**	**76.15±1.46**	**75.19±1.76**	**92.51±1.05**	**57.45±1.79**
	yes	61.69±0.89	60.23±7.14	58.84±1.85	53.14±5.26	59.43±3.70	50.97±6.96	59.50±4.31	41.93±5.41

D.6.2. Remove the orthonormal representation objective $\mathcal{L}_{\text{ORTH}}(\mathcal{G};\phi)$

We exclude the orthonormal representation objective $\mathcal{L}_{\text{orth}}(\mathcal{G};\phi)$ from the GeMax objective $\mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A})$. According to Table 10, this exclusion compromises the effectiveness of the VP(G) regularization subset. This is due to Theorem 3.3, which is predicated on the existence of an orthonormal representation space. Consequently, the overall performance of the model is negatively impacted.

Table 10: Performance (ACC) of unsupervised learning for Ablation study. The **bold** numbers denote the better performances of the same method.

remove $\mathcal{L}_{\text{orth}}(\mathcal{G};\phi)$		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	no	**92.44±1.23**	**76.87±1.99**	**75.43±1.10**	**80.21±1.83**	**73.95±1.17**	**74.31±1.53**	**86.45±1.43**	**56.16±2.40**
	yes	75.45±1.10	61.24±1.18	57.21±1.84	69.44±1.03	58.83±1.55	56.33±1.74	70.07±1.47	47.26±1.91
GraphCL	no	**88.83±1.10**	**77.60±1.18**	**78.24±1.84**	**80.89±1.03**	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
	yes	71.04±1.19	59.48±1.90	61.31±1.91	64.12±1.01	60.26±1.28	61.74±1.18	72.07±1.89	44.93±1.28
AD-GCL	no	**88.37±1.05**	**75.96±2.30**	**77.07±1.02**	**77.15±1.83**	**73.02±1.54**	**73.62±1.56**	**89.88±1.85**	**56.61±1.38**
	yes	67.98±1.70	63.50±1.14	60.93±1.09	62.57±1.86	64.05±1.90	63.97±1.72	74.96±1.25	43.75±2.06
JOAOv2	no	**89.07±1.19**	**74.91±1.90**	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	**72.62±1.18**	**87.19±1.89**	**55.24±1.28**
	yes	70.80±1.28	59.29±1.09	61.87±1.52	60.87±1.75	61.70±1.16	67.94±1.60	66.69±1.93	42.15±1.09
AutoGCL	no	**90.63±1.15**	**77.12±1.75**	**80.17±0.87**	**81.11±1.85**	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	**57.86±1.72**
	yes	73.07±2.10	57.50±1.98	65.33±1.06	65.42±2.59	64.01±1.08	66.50±2.03	76.44±1.24	48.24±1.29
GraphACL	no	**91.86±0.70**	**77.29±1.14**	**81.08±1.09**	**81.19±1.86**	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
	yes	72.34±1.15	59.51±1.75	62.65±0.87	64.96±1.85	64.45±1.35	66.24±1.91	76.76±2.06	50.38±1.72
GCS	no	**91.37±0.95**	**76.12±0.76**	**78.82±0.76**	**80.08±1.50**	**76.15±1.46**	**75.19±1.76**	**92.51±1.05**	**57.45±1.79**
	yes	71.82±1.15	62.04±1.75	57.68±1.87	61.78±1.85	69.94±1.35	70.43±1.91	73.33±2.06	48.40±1.72

D.6.3. Remove the subset of VP(G) objective $\mathcal{L}_{\text{S-VP}}(\mathcal{G};\theta,\phi,\mathcal{A})$

We omit the VP(G) objective subset $\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})$ from the GeMax objective $\mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A})$ and $\mathcal{J}_2(\mathcal{G};\theta,\phi,\mathcal{A})$. As indicated in Table 11, the absence of the VP(G) objective leads to unconstrained vectors $\boldsymbol{a}_i$. The result shows its significant role in the model's effectiveness.

Table 11: Performance (ACC) of unsupervised learning for Ablation study. The **bold** numbers denote the better performances of the same method.

remove $\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})$		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	no	**92.44±1.23**	**76.87±1.99**	**75.43±1.10**	**80.21±1.83**	**73.95±1.17**	**74.31±1.53**	**86.45±1.43**	**56.16±2.40**
	yes	63.55±2.23	61.83±1.30	62.06±3.73	58.98±4.01	60.07±1.47	57.21±3.29	64.52±2.16	45.93±1.46
GraphCL	no	**88.83±1.10**	**77.60±1.18**	**78.24±1.84**	**80.89±1.03**	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	**57.87±1.91**
	yes	74.73±0.65	65.06±3.72	64.37±2.30	59.63±2.24	62.55±3.01	68.38±1.06	65.30±3.27	49.71±2.17
AD-GCL	no	**88.37±1.05**	**75.96±2.30**	**77.07±1.02**	**77.15±1.83**	**73.02±1.54**	**73.62±1.56**	**89.88±1.85**	**56.61±1.38**
	yes	73.19±1.94	61.18±1.21	65.76±1.30	60.77±1.17	61.09±2.74	65.32±1.93	74.25±1.25	45.64±1.09
JOAOv2	no	**89.07±1.19**	**74.91±1.90**	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	**72.62±1.18**	**87.19±1.89**	**55.24±1.28**
	yes	65.77±2.92	68.94±1.28	60.27±3.10	61.31±1.26	59.73±2.20	60.27±1.50	62.91±3.39	42.55±2.19
AutoGCL	no	**90.63±1.15**	**77.12±1.75**	**80.17±0.87**	**81.11±1.85**	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	**57.86±1.72**
	yes	73.04±1.47	60.92±1.36	63.64±2.79	67.62±1.79	57.56±3.28	61.41±1.48	67.21±0.90	41.97±2.75
GraphACL	no	**91.86±0.70**	**77.29±1.14**	**81.08±1.09**	**81.19±1.86**	**75.94±0.90**	**75.03±1.72**	**91.45±1.25**	**57.48±2.06**
	yes	74.23±1.46	60.74±4.55	64.97±3.61	64.31±2.76	64.64±3.09	65.08±2.29	74.11±2.05	47.52±1.09
GCS	no	**91.37±0.95**	**76.12±0.76**	**78.82±0.76**	**80.08±1.50**	**76.15±1.46**	**75.19±1.76**	**92.51±1.05**	**57.45±1.79**
	yes	74.96±2.21	68.20±3.58	62.97±2.13	66.78±3.20	59.68±0.61	67.86±1.92	74.25±3.08	43.97±2.32

D.7. Experiment: Distribution Comparison

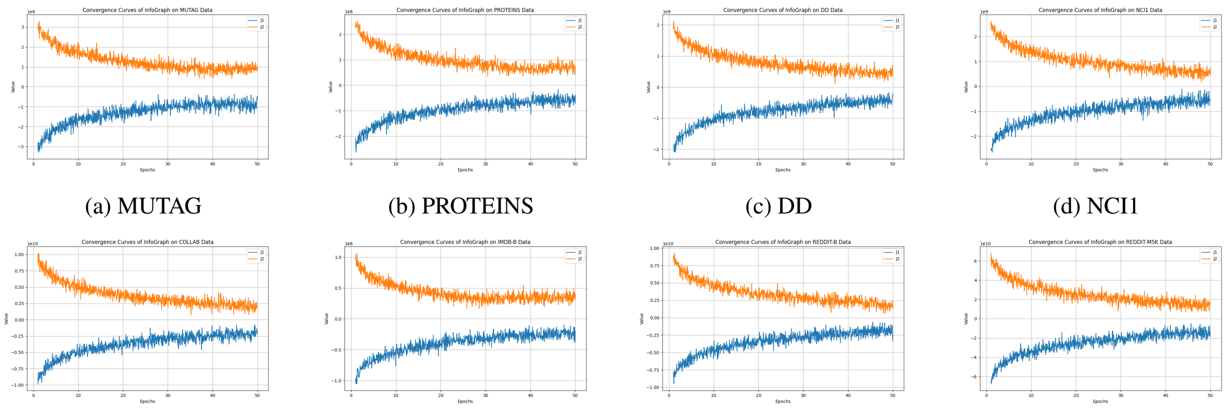
In Eq. (3), we propose a Boltzmann probability distribution derived from graph representations. Additionally, we suggest the use of a normalized exponential kernel to assign a distribution over the vertex set. In Table 12, we conduct a comparative analysis of these two distributions when applied to graph representation learning. The results indicate that both distributions perform well, with their performance metrics being closely aligned.

D.8. Experiment: Convergence Analysis

Figure 11. shows the convergence curves of InfoGraph on different datasets.

Table 12: Performance (ACC) of unsupervised learning using different distribution. The **bold** numbers denote the better performances of the same method.

algorithms		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
InfoGraph	Boltzmann	**92.44±1.23**	76.87±1.99	**75.43±1.10**	80.21±1.83	73.95±1.17	74.31±1.53	86.45±1.43	**56.16±2.40**
	exponential	89.45±4.23	**77.24±2.01**	75.21±1.73	**82.44±2.01**	**74.83±1.47**	**74.33±1.29**	**87.07±2.16**	53.26±1.46
GraphCL	Boltzmann	**88.83±1.10**	77.60±1.18	78.24±1.84	80.89±1.03	**74.21±1.55**	**74.31±1.74**	**90.76±1.47**	57.87±1.91
	exponential	86.03±0.47	**78.50±1.66**	**78.94±2.79**	**82.57±1.87**	71.05±1.28	73.97±1.48	89.96±2.30	**58.75±2.75**
AD-GCL	Boltzmann	88.37±1.05	75.96±2.30	**77.07±1.02**	77.15±1.83	**73.02±1.54**	73.62±1.56	**89.88±1.85**	56.61±1.38
	exponential	**90.19±1.94**	**77.18±1.21**	75.76±1.30	**78.77±1.17**	71.09±0.74	**73.72±1.93**	87.97±0.25	**56.64±1.30**
JOAOv2	Boltzmann	89.07±1.19	74.91±1.90	**74.68±1.91**	**76.15±1.01**	**73.17±1.88**	72.62±1.18	**87.19±1.89**	**55.24±1.28**
	exponential	**92.03±2.13**	**75.20±2.58**	73.97±1.13	73.78±2.20	71.68±2.61	**72.86±1.92**	84.25±1.08	53.97±1.32
AutoGCL	Boltzmann	90.63±1.15	**77.12±1.75**	**80.17±0.87**	81.11±1.85	**72.08±1.35**	**74.03±1.91**	**91.84±2.06**	57.86±1.72
	exponential	**91.16±1.92**	75.91±2.28	78.28±1.10	**82.50±1.24**	68.36±2.33	73.19±1.50	90.03±1.39	**58.25±1.19**
GraphACL	Boltzmann	**91.86±0.70**	**77.29±1.14**	**81.08±1.09**	81.19±1.86	75.94±0.90	75.03±1.72	**91.45±1.25**	**57.48±2.06**
	exponential	90.04±2.92	74.48±1.28	79.31±2.10	**82.12±2.10**	**78.26±2.05**	**77.24±4.51**	90.07±2.49	56.13±1.13
GCS	Boltzmann	91.37±0.95	**76.12±0.76**	78.82±0.76	**80.08±1.50**	**76.15±1.46**	**75.19±1.76**	**92.51±1.05**	**57.45±1.79**
	exponential	**92.61±1.46**	73.27±2.55	**79.10±1.61**	77.31±1.76	74.13±1.85	69.21±1.39	89.11±2.35	56.23±1.12



(a) MUTAG (b) PROTEINS (c) DD (d) NCI1

(e) COLLAB (f) IMDB-B (g) REDDIT-B (h) REDDIT-M5K

Figure 11: Convergence Curve of InfoGraph on different Dataset

In cases where a large dataset requires an excessive amount of time to meet convergence conditions, but where significant changes in the objective function primarily occur within the first 50 epochs, it may be practical to limit the plotting of the convergence curve to these initial 50 epochs. This approach can provide valuable insights into the majority of the convergence behavior without the need for extensive computation beyond the point of diminishing returns.

D.9. Experiment: Time Costs

We run experiments on a server with Intel 7 CPU and RTX 3090 GPUs. We repeat the experiment five times and report the average time. Since we only change the objective function while keeping other parts of the models unchanged, the time of each method is close.

D.10. Experiment: Code of Key Implementation

Here, we provide the key implementation code for the loss and objective functions of our paper. The complete code of our method is available at https://github.com/MathAdventurer/GeMax.

(1) Orthonormal representation learning loss is as follows:

$$\mathcal{L}_{\text{orth}}(\mathcal{G};\phi) := \sum_{j=1}^{N} \left\| \boldsymbol{M}_j \odot \left(\boldsymbol{Z}_j^{\phi} (\boldsymbol{Z}_j^{\phi})^{\top} - \boldsymbol{I}_n \right) \right\|_F^2, \tag{34}$$

The code of $\mathcal{L}_{\text{orth}}(\mathcal{G};\phi)$ is as follows:

```
def loss_orthogonal(Z, phi):
    batch_size = Z.batch_size
```

Table 13: Time cost. The h denotes hour.

tasks	methods and principles		MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
unsupervised learning	InfoGraph	InfoMax	0.10 h	0.47 h	2.46 h	1.20 h	2.66 h	0.28 h	6.06 h	13.34 h
		Lovász	0.04 h	0.41 h	2.43 h	1.26 h	2.03 h	0.30 h	6.05 h	13.41 h
		GeMax	0.08 h	0.42 h	2.89 h	1.79 h	2.34 h	0.32 h	6.25 h	13.18 h
	GraphCL	InfoMax	0.09 h	0.43 h	2.60 h	1.28 h	2.12 h	0.33 h	6.11 h	13.39 h
		Lovász	0.08 h	0.35 h	2.40 h	1.30 h	2.08 h	0.34 h	6.08 h	13.46 h
		GeMax	0.11 h	0.47 h	2.50 h	1.14 h	2.75 h	0.36 h	6.15 h	13.12 h
	AD-GCL	InfoMax	0.13 h	0.52 h	3.02 h	1.37 h	2.51 h	0.36 h	5.85 h	14.14 h
		Lovász	0.10 h	0.51 h	3.06 h	1.39 h	2.68 h	0.34 h	5.86 h	14.07 h
		GeMax	0.14 h	0.48 h	3.13 h	1.87 h	2.97 h	0.36 h	6.20 h	14.67 h
	JOAOv2	InfoMax	0.10 h	0.62 h	2.09 h	1.48 h	2.27 h	0.32 h	6.36 h	14.79 h
		Lovász	0.12 h	0.61 h	2.04 h	1.45 h	2.24 h	0.31 h	6.37 h	14.74 h
		GeMax	0.11 h	0.63 h	2.10 h	1.49 h	2.26 h	0.33 h	6.35 h	14.77 h
	AutoGCL	InfoMax	0.11 h	0.50 h	3.15 h	1.48 h	2.12 h	0.34 h	6.09 h	14.42 h
		Lovász	0.14 h	0.49 h	3.10 h	1.46 h	2.09 h	0.33 h	6.07 h	14.40 h
		GeMax	0.13 h	0.53 h	3.42 h	1.65 h	2.91 h	0.38 h	6.85 h	14.39 h
	GraphACL	InfoMax	0.18 h	0.63 h	2.11 h	1.45 h	2.24 h	0.32 h	6.41 h	14.78 h
		Lovász	0.16 h	0.62 h	2.10 h	1.48 h	2.23 h	0.31 h	6.39 h	14.77 h
		GeMax	0.17 h	0.66 h	2.30 h	1.54 h	2.69 h	0.33 h	6.19 h	14.32 h
	GCS	InfoMax	0.09 h	0.59 h	2.05 h	1.48 h	2.24 h	0.39 h	6.41 h	14.78 h
		Lovász	0.08 h	0.61 h	2.08 h	1.46 h	2.26 h	0.30 h	6.39 h	14.76 h
		GeMax	0.10 h	0.77 h	2.32 h	1.53 h	2.71 h	0.32 h	6.20 h	14.31 h
semi-supervised learning	InfoGraph	InfoMax	-	0.24 h	2.58 h	1.52 h	2.53 h	-	6.85 h	14.46 h
		Lovász	-	0.36 h	2.84 h	1.27 h	2.14 h	-	6.16 h	14.35 h
		GeMax	-	0.33 h	2.79 h	1.11 h	2.09 h	-	6.22 h	14.28 h
	GraphCL	InfoMax	-	0.49 h	2.71 h	1.05 h	2.45 h	-	6.92 h	14.53 h
		Lovász	-	0.63 h	2.18 h	1.39 h	2.26 h	-	6.47 h	14.08 h
		GeMax	-	0.56 h	2.15 h	1.42 h	2.29 h	-	6.41 h	14.14 h
	AD-GCL	InfoMax	-	0.56 h	2.94 h	1.10 h	2.76 h	-	6.65 h	14.72 h
		Lovász	-	0.61 h	3.23 h	1.79 h	2.08 h	-	6.63 h	14.49 h
		GeMax	-	0.59 h	3.27 h	1.88 h	2.02 h	-	6.75 h	14.58 h
	JOAOv2	InfoMax	-	0.61 h	3.16 h	1.72 h	2.40 h	-	6.35 h	14.06 h
		Lovász	-	0.91 h	3.77 h	1.89 h	2.70 h	-	6.57 h	14.30 h
		GeMax	-	0.89 h	3.79 h	1.80 h	2.72 h	-	6.61 h	14.20 h
	AutoGCL	InfoMax	-	0.59 h	3.15 h	1.12 h	2.91 h	-	6.90 h	14.89 h
		Lovász	-	0.51 h	3.38 h	1.87 h	2.50 h	-	6.23 h	14.75 h
		GeMax	-	0.52 h	3.41 h	1.91 h	2.45 h	-	6.20 h	14.81 h
	GraphACL	InfoMax	-	0.61 h	3.14 h	1.75 h	2.30 h	-	6.01 h	14.11 h
		Lovász	-	0.92 h	3.84 h	1.85 h	2.75 h	-	6.75 h	14.27 h
		GeMax	-	0.91 h	3.80 h	1.88 h	2.78 h	-	6.71 h	14.24 h
	GCS	InfoMax	-	0.77 h	3.13 h	1.05 h	2.29 h	-	6.21 h	14.48 h
		Lovász	-	0.73 h	3.77 h	1.03 h	2.32 h	-	6.49 h	14.83 h
		GeMax	-	0.79 h	3.88 h	1.50 h	2.14 h	-	6.94 h	14.92 h

```
loss_orth = torch.tensor(0.0, dtype=torch.float32, requires_grad=True)

start_idx = 0
for j in range(batch_size):
    num_nodes = Z.batch_num_nodes()[j]
    Z_j = Z.ndata['h'][start_idx:start_idx+num_nodes]
    start_idx += num_nodes
    n_j = Z_j.size(0)
    M_j = torch.eye(n_j, device=Z.device) -\
                Z.adjacency_matrix().to_dense()[:n_j, :n_j]
    term = torch.matmul(Z_j, Z_j.t()) - torch.eye(n_j, device=Z.device)
    loss_orth = loss_orth + torch.norm(M_j * term, 'fro')**2

return loss_orth
```

(2) The subset of VP(G) loss is:

$$\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A}) := \sum_{j=1}^{N} \left\| \boldsymbol{D}_{a_j} \left(\boldsymbol{Z}_j^{\phi} (\boldsymbol{Z}_j^{\phi})^{\top} \right) \boldsymbol{D}_{a_j} - \boldsymbol{D}_{a_j}^2 \right\|_F^2 \tag{35}$$

The code of $\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})$ is as follows:

```
def loss_sub_vertex_packing(Z, A, theta, phi):
    batch_size = Z.batch_size
    loss_svp = torch.tensor(0.0, dtype=torch.float32, requires_grad=True)

    start_idx = 0
    for j in range(batch_size):
        num_nodes = Z.batch_num_nodes()[j]
        Z_j = Z.ndata['h'][start_idx:start_idx+num_nodes]
        start_idx += num_nodes
        a_j = A_set[j][:num_nodes]
        D_a_j = torch.diag(a_j)
        term = torch.matmul(torch.matmul(D_a_j, Z_j), Z_j.t()) -\
                                    torch.matmul(D_a_j, D_a_j)
        loss_svp = loss_svp + torch.norm(term, 'fro')**2

    return loss_svp
```

(3) The graph entropy loss is:

$$\mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) = \sum_{j=1}^{N} \sum_{i=1}^{n_j} -P_i(\mathbf{g}_j^{\theta}, \boldsymbol{Z}_j^{\phi}) \log(a_{j(i)}). \tag{36}$$

The code of $\mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A})$ is as follows:

```
def loss_entropy(Z, theta, phi, A_set):
    batch_size = Z.batch_size
    loss_entropy = torch.tensor(0.0, dtype=torch.float32, requires_grad=True)

    start_idx = 0
    for j in range(batch_size):
        num_nodes = Z.batch_num_nodes()[j]
        Z_j = Z.ndata['h'][start_idx:start_idx+num_nodes]
        start_idx += num_nodes
        a_j = A_set[j][:num_nodes]
        P_i = torch.sigmoid(torch.matmul(Z_j, theta.t()))
        a_j_expanded = a_j.unsqueeze(1).expand_as(P_i)
        loss_entropy = loss_entropy - torch.sum(P_i * torch.log(a_j_expanded))

    return loss_entropy
```

(4) The J_1 loss is:

$$\begin{aligned}\mathcal{J}_1(\mathcal{G};\theta,\phi,\mathcal{A}) :=& \mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) - \mu \cdot \mathcal{L}_{\text{orth}}(\mathcal{G};\phi) \\ & - \gamma \cdot \mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A})\end{aligned} \tag{37}$$

The code of J_1 objective is:

```
def objective_J1(Z, theta, phi, A_set, mu, gamma):
    loss_Hk = loss_entropy(Z, theta, phi, A_set)
    loss_orth = loss_orthogonal(Z, phi)
    loss_svp = loss_sub_vertex_packing(Z, A_set, theta, phi)

    J1 = loss_Hk - mu * loss_orth - gamma * loss_svp

    return J1
```

(5) The J_2 objective is:

$$\begin{aligned}\mathcal{J}_2(\mathcal{G};\theta,\phi,\mathcal{A}) :=& \mathcal{L}_{H_k}(\mathcal{G};\theta,\phi,\mathcal{A}) + \gamma\cdot\mathcal{L}_{\text{s-vp}}(\mathcal{G};\theta,\phi,\mathcal{A}) \\ &\text{s.t. } 0\le a_{ij}\le 1,\ \forall i\in[n_j],\ \boldsymbol{a}_j\in\mathcal{A},\end{aligned} \tag{38}$$

The code of J_2 objective is:

```
def objective_J2(Z, theta, phi, A_set, gamma):
    loss_Hk = loss_entropy(Z, theta, phi, A_set)
    loss_svp = loss_sub_vertex_packing(Z, A_set, theta, phi)

    J2 = loss_Hk + gamma * loss_svp

    return J2
```