Graph Entropy Guided Node Embedding Dimension Selection for Graph Neural Networks

Gongxu Luo[1,2] , **Jianxin Li**[1,2] , **Hao Peng**[1] , **Carl Yang**[3] , **Lichao Sun**[4] , **Philip S. Yu**[5] and **Lifang He** [4]

[1]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, China
[2]School of Computer Science and Engineering, Beihang University, China
[3]Department of Computer Science, Emory University, USA
[4]Department of Computer Science and Engineering, Lehigh University, USA
[5]Department of Computer Science, University of Illinois at Chicago, USA
{luogx, lijx, penghao }@act.buaa.edu.cn, j.carlyang@emory.edu, psyu@uic.edu,
{lis221,lih319}@lehigh.edu

Abstract

Graph representation learning has achieved great success in many areas, including e-commerce, chemistry, biology, etc. However, the fundamental problem of choosing the appropriate dimension of node embedding for a given graph still remains unsolved. The commonly used strategies for Node Embedding Dimension Selection (NEDS) based on grid search or empirical knowledge suffer from heavy computation and poor model performance. In this paper, we revisit NEDS from the perspective of minimum entropy principle. Subsequently, we propose a novel Minimum Graph Entropy (MinGE) algorithm for NEDS with graph data. To be specific, MinGE considers both feature entropy and structure entropy on graphs, which are carefully designed according to the characteristics of the rich information in them. The feature entropy, which assumes the embeddings of adjacent nodes to be more similar, connects node features and link topology on graphs. The structure entropy takes the normalized degree as basic unit to further measure the higher-order structure of graphs. Based on them, we design MinGE to directly calculate the ideal node embedding dimension for any graph. Finally, comprehensive experiments with popular Graph Neural Networks (GNNs) on benchmark datasets demonstrate the effectiveness and generalizability of our proposed MinGE.

1 Introduction

In recent years, Graph Neural Networks (GNNs) [Wu *et al.*, 2020; Xie *et al.*, 2020] have attracted tremendous attention from both research and industry, due to its powerful representation capability for large amounts of graph structured data in practice, *e.g.,* social networks, citation networks, road networks. GNNs are mostly used to compute distributed node representations (*a.k.a.* embeddings), as dense vectors, which

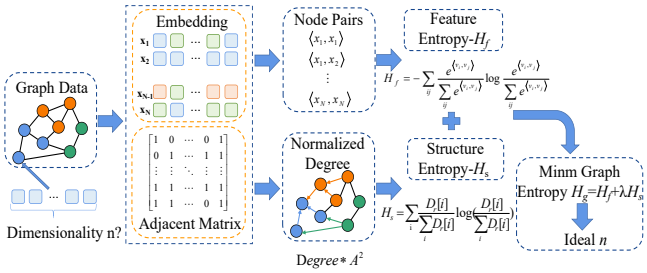

Figure 1: The overview of MinGE. The graph entropy considers both feature entropy and structure entropy to guide NEDS for a given graph. The feature entropy connects node features and link topology on graphs, while the structure entropy further measures the higher-order structure of graphs.

serves as the key to various downstream tasks in graph related applications [Peng *et al.*, 2019; Sun *et al.*, 2021]. The dimension of node embedding, as a crucial hyperparameter, has a significant influence on the performance of GNNs. First, node embeddings with too small dimensions limit the information expressiveness of nodes, whereas those with too large dimensions suffer from overfitting. Second, the number of parameters of GNNs that build on node embeddings is usually a linear or quadratic function of node embedding dimension, which directly affects model complexity and computational efficiency [Yin and Shen, 2018].

However, we often have no idea towards the appropriate dimension of node embedding and have to rely on grid search or domain knowledge to adjust it as a model hyperparameter given a new graph. These approaches often suffer from the following problems: **1)** Choosing node embedding dimension by empirical experience can easily lead to poor model performances. **2)** Experimental strategies such as grid search are resource-consuming and time-consuming. This raises the questions of what is the appropriate dimension of node embedding for given graphs and what should we base on to choose such dimension? Recently, several researchers [Liu *et al.*, 2020; Zhu *et al.*, 2018; Shen *et al.*, 2020] have attempt to tackle these problems

through dimensionality reduction. However, these methods focus on choosing important features or using complex matrix transformation to reduce from a large dimension instead of direct selection based on the graph. Moreover, different from sequence and image data, graph data contains rich link topology. The general dimensionality reduction methods only consider node features but ignore it for GNNs, which results in loss of useful information.

In this paper, inspired by recent dimension selection works [Yin and Shen, 2018; Hung and Yamanishi, 2020] in Natural Language Processing (NLP), we revisit NEDS from the perspective of minimum entropy principle [Zhu *et al.*, 1997]. Specifically, entropy is a natural way to measure the uncertainty of the dimension, the principle of minimum entropy motivates us to choose an ideal node embedding dimension for any given graphs by minimizing the uncertainty of the graph. As shown in Fig. 1, we design a novel graph entropy that contains feature entropy and structure entropy to account for the information regarding both node features and link structures on graphs. Subsequently, we propose a novel Minimum Graph Entropy (MinGE) algorithm[1] to directly select an appropriate node embedding dimension for a given graph through minimizing the graph entropy. As a consequence, MinGE leads to appropriate NEDS without the need of additional domain knowledge, and is obviously more efficient than grid search.

The main contributions of this paper are summarized as follows: **(1)** We stress the importance of direct NEDS for GNNs on arbitrary graphs and revisit it from an intuitive perspective based on the principle of minimum entropy. **(2)** We develop the MinGE algorithm based on our novel design of graph entropy to enable direct NEDS, which considers the rich feature and structure information on graphs. **(3)** Experimental results with several popular GNNs such as GCN, GAT, and GCNII over node classification and link prediction tasks on benchmark datasets demonstrate the effectiveness and generalizability of our proposed MinGE algorithm.

2 Related Work

Dimension Selection. How to choose the appropriate embedding dimension is always an open and challenging problem for deep learning. In NLP, motivated by the unitary-invariance of word embedding, [Yin and Shen, 2018] proposed the pairwise inner product loss to measure the dissimilarities between word embeddings. Inspired by the this work, [Wang, 2019] proposed a fast and reliable method based on Principle Components Analysis (PCA) to choose the dimension of word embedding, which use grid search to reduce the complexity of the algorithm. By rethinking the classical skip-gram algorithm, [Hung and Yamanishi, 2020] proposed a novel information criteria-based method to select the dimension of word embedding, and gave a theoretical analysis from the perspective of probability theory. Similarly, in Computer Vision (CV), researchers also try to reduce the representation dimensionality by sampling operation on the input data [Luo *et al.*, 2020]. In graph mining, existing work about embedding dimension selection is based on dimensionality reduction to select important features. Many previous dimensionality reduction algorithms are based on a predefined graph, however, there are noise and redundant information in original data. [Xiong *et al.*, 2017] proposed the linear manifold regularization with adaptive graph to directly incorporate the graph construction into the objective function to solve the problem. Similarly, [Liu *et al.*, 2020] proposed the termed discriminative sparse embedding to learn a sparse weight matrix to reduce the effects of redundant information and noise of original data. In order to avoid choosing the dimension that is outliers, [Zhu *et al.*, 2018] proposed a robust graph dimensionality reduction algorithm to map high-dimension data into lower-dimensional intrinsic space by a transformation matrix. Although there are amounts of fantastic work for dimension selection. However, these methods focus on studying how to choose important features to reduce the dimensionality instead of direct selection based on the graph. Moreover, these methods ignore the rich link structures of graph data in dimensionality reduction, which results in loss of information.

Structure Entropy. The conception of structure entropy originate from the research of [Shannon, 1953], who establish the structural theory of information to support the analysis of communication system. Subsequently, a large number of methods are proposed about understanding the complexity of networks. [Mowshowitz and Dehmer, 2012] proposed the entropy of graph to measure the entropy of the distribution $p(G)$ at the global level. Besides, there were some methods to measure the structure entropy of nodes. [Raychaudhury *et al.*, 1984] first proposed the local measure of graph entropy, which is distance-based. Some extension works such as parametric graph entropy [Dehmer, 2008], Gibbs entropy [Bianconi, 2009], Shannon entropy, and Von Neumann entropy [Braunstein *et al.*, 2006] designed the structure information measurement of the network from different angles. Furthermore, [Li and Pan, 2016] first proposed the metric for structure information and defined the K-dimensional structure information of the graph, which can not only detect the natural or true structure but also can measure the complexity of dynamic evolving networks.

So far, there is still no effective solution to directly select an appropriate node embedding dimension for any given graph. Therefore, in this paper, we revisit the NEDS from the perspective of minimum entropy principle. Based on it, we design a novel MinGE algorithm that contains feature entropy and structure entropy for direct NEDS.

3 Methodology

In this section, we first introduce the basic notations used in this paper. Then we present the overall framework of our minimum graph entropy approach MinGE, followed by the detail of how to calculate the feature entropy and the structure entropy for graph data with rich link structures.

3.1 Notation

Suppose we are given an undirected graph $G = (V, E, A)$, where V denotes the node set of the graph, E denotes the

[1] https://github.com/RingBDStack/MinGE

set of edges, and $A \in \mathbb{R}^{N\times N}$ denotes the adjacency matrix with the element $A[i,j] = A[j,i]$ indicating whether edge (v_i, v_j) in E. N is the number of nodes. The link structures can be presented by the first-order adjacency matrix A and the second-order adjacency matrix A^2. The adjacency matrix after normalization is denoted as A_r. Similarly, the second-order adjacency matrix after normalization is denote as A_r^2. The degrees of all nodes are represented by a vector $D \in \mathbb{R}^{1\times N}$, and the normalized degree is D_r.

3.2 Graph Entropy

Information entropy proposed by Shannon [Shannon, 1948] is a measure of information uncertainty, which is defined as:

$$H = -\sum_{i}^{n} P_i log P_i, \tag{1}$$

where H denotes the entropy, P_i is the probability of event i, n is the number of events. It indicates that, the smaller the entropy, the lower information uncertainty, which equally means the more useful information. However, using single information entropy to measure graph data with rich link structures is brittle. Therefore, in this paper, we propose a novel graph entropy to consider both rich node features and link structures on graphs, which is defined as:

$$H_g = H_f + \lambda H_s, \tag{2}$$

where H_g represents graph entropy, H_f and H_s represents feature entropy and structure entropy respectively. λ is a hyperparameter that controls the ratio of structure entropy for NEDS. In the following, we will discuss how to define H_f and H_s in detail. Especially for the feature entropy H_f, it establishes the connection between dimension n and graph entropy. The structure entropy further search the ideal node embedding dimension. Finally, we can directly calculate the appropriate node embedding dimension n by setting $H_g = 0$.

Feature Entropy. Information entropy measures the uncertainty of information. The core of information entropy is how to define the basic unit of events. For graph data, we design a novel feature entropy as shown in Fig. 1. Based on the assumption that the node embeddings of adjacent nodes to be more similar, we use node embedding dot product of node pairs as the basic unit and the probability is defined as:

$$P(v_i, v_j) = \frac{e^{\langle v_i, v_j\rangle}}{\sum_{i,j} e^{\langle v_i, v_j\rangle}}, \tag{3}$$

where v_i and v_j are the corresponding node embedding, $\langle\cdot,\cdot\rangle$ is the dot product operation. For brevity, we denote $Z = \sum_{i,j} e^{\langle v_i, v_j\rangle}$. The feature entropy of graph data is defined as:

$$\begin{aligned} H_f &= -\sum_{ij} P(v_i, v_j) log P(v_i, v_j) \\ &= -\sum_{ij} \frac{e^{\langle v_i, v_j\rangle}}{Z} log \frac{e^{\langle v_i, v_j\rangle}}{Z} \\ &= logZ - \frac{1}{Z}\sum_{ij} e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle. \end{aligned} \tag{4}$$

To calculate Eq. (4), we approximate the sum operation with sampling, which is defined as:

$$\begin{aligned} Z &= \sum_{ij} e^{\langle v_i, v_j\rangle} = N^2 \frac{1}{N^2}\sum_{ij} e^{\langle v_i, v_j\rangle} \\ &\approx N^2 E_{v_i, v_j}(e^{\langle v_i, v_j\rangle}), \end{aligned} \tag{5}$$

$$\begin{aligned} \sum_{ij} e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle &= N^2 \frac{1}{N^2}\sum_{ij} e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle \\ &= N^2 E_{v_i, v_j}(e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle), \end{aligned} \tag{6}$$

where E is the expectation operation. Plugging the approximate expressions from Eqs. (5) and (6) into Eq. (4), we can calculate the feature entropy by

$$\begin{aligned} H_f =& logZ - \frac{1}{Z}\sum_{ij} e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle \\ =& logN^2 + log E_{v_i, v_j}(e^{\langle v_i, v_j\rangle}) - \frac{E_{v_i, v_j}(e^{\langle v_i, v_j\rangle}\langle v_i, v_j\rangle)}{E_{v_i, v_j}(e^{\langle v_i, v_j\rangle})}. \end{aligned} \tag{7}$$

However, $\langle v_i, v_j\rangle$ is hard to calculate directly in Eq. (7). In the experiments of graph representation learning, we observe that the absolute value distribution of the values of each dimension is uniform. Therefore, we assume that the absolute value of each element is 1, which is called the distributed hypothesis [Sahlgren, 2008]. Furthermore, to calculate $\langle v_i, v_j\rangle$, we map node embeddings to the n-dimensional hyper-sphere with radius $\sqrt{n}$. Then, $\langle v_i, v_j\rangle = n * cos\theta$ [Li *et al.*, 2020], where θ is the angle between any two vectors, and it connects the node embedding dimension n with entropy. Next, the probability distribution of the angle θ between two random vectors in the n-dimensional hyper-sphere is needed to keep n as the only variable. Because of the isotropy, we only need to consider the unit vector. Besides, we only need to fix one of the vectors and consider the random change of the other vector. Therefore, we set the n-dimensional random vector $x = (x_1, x_2, \cdots, x_n)$, the fixed vector is $y = (1, 0, \cdots, 0)$. Converting x to hyper-sphere coordinates is defined as:

$$\begin{cases} x_1 = r * cos(\varphi_1) \\ x_2 = r * sin(\varphi_1)cos(\varphi_2) \\ \quad\vdots \\ x_{n-1} = r * sin(\varphi_1)sin(\varphi_2)\cdots sin(\varphi_{n-2})cos(\varphi_{n-1}) \\ x_n = r * sin(\varphi_1)sin(\varphi_1)\cdots sin(\varphi_{n-2})sin(\varphi_{n-1}), \end{cases} \tag{8}$$

where $r = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = 1$ is a radial coordinate, $\varphi_{n-1} \in [0, 2\pi)$, others $\varphi \in [0, \pi]$. Note that $\|x\| = 1$ and $\|y\| = 1$, therefore the angle between x and y is

$$arccos(\frac{x^T y}{\|x\|\|y\|}) = arccos(cos(\varphi_1)) = \varphi_1. \tag{9}$$

where $arccos$ is the inverse $cosine$ ($arccosine$) function. Eq. (9) shows that the angle between x and y is φ_1. In particular, the probability of the angle between x and y that does not exceed θ is

$$P(\varphi_1 \le \theta) = \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})\sqrt{\pi}} \int_0^{\theta} sin^{n-2}\varphi_1 d(\varphi_1). \tag{10}$$

The probability density of θ is denoted (following [Henderson and Moura, 1996]) by

$$P_n(\theta) = \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})\sqrt{\pi}} sin^{n-2}\theta. \quad (11)$$

Therefore, plugging the probability density from Eq. (11) to Eq. (7), the feature entropy is updated by

$$H_f = logN^2 + logE(e^{ncos\theta})) - \frac{E(e^{ncos\theta}ncos\theta)}{E(e^{ncos\theta})}, \quad (12)$$

where the calculation of the expectation operation is

$$E(e^{ncos\theta}ncos\theta) = \int_0^\pi e^{ncos\theta}ncos\theta P_n(\theta)d_\theta, \quad (13)$$

$$E(e^{ncos\theta}) = \int_0^\pi e^{ncos\theta}P_n(\theta)d_\theta. \quad (14)$$

Structure Entropy. Structure entropy measures the complexity of networks, which is original from the Shannon's 1953 question [Jr., 2003]. For graph data, we design a novel structure entropy to measure the information of link structures as shown in Fig. 1. The structure entropy uses normalized node degree as basic unit to further search the ideal dimension, which considers two-hop neighbors. First, given a graph with the adjacency matrix A that contains first-order link structures, the second-order adjacency matrix is defined as $A^2 = A^T A$ to measure the second-order link structures. At the same time, the degree vector D is generated by the adjacency matrix A. Second, the normalized degree vector D_r, which contains first-order and second-order link structures, can be defined as:

$$D_r = D^T A_r^2, \quad (15)$$

where A_r^2 represents the normalized second-order adjacency matrix defined as:

$$A_r^2[i,j] = \frac{A^2[i,j]}{\sum_j A^2[i,j]}, \quad (16)$$

where $A^2[i,j]$ is the value of the i-th row and j-th column of the second-order adjacency matrix A^2. Following the paradigm of information entropy, the structure entropy with the normalized degree of nodes as a unit event is defined as:

$$H_s = -\sum_i^N P_i logP_i = -\sum_i \frac{D_r[i]}{\sum_i D_r[i]} log(\frac{D_r[i]}{\sum_i D_r[i]}), \quad (17)$$

where $D_r[i]$, calculated by Eq. (14), is the normalized degree of node i. Therefore, the graph entropy is defined as:

$$\begin{aligned} H_g &= H_f + \lambda H_s \quad (18)\\ &= logN^2 + log\int_0^\pi e^{ncos\theta}P_n(\theta)d_\theta \\ &- \frac{\int_0^\pi e^{ncos\theta}ncos\theta P_n(\theta)d_\theta}{\int_0^\pi e^{ncos\theta}P_n(\theta)d_\theta} \\ &- \lambda\sum_i \frac{D_r[i]}{\sum_i D_r[i]} log(\frac{D_r[i]}{\sum_i D_r[i]}). \end{aligned}$$

Algorithm 1: MinGE

Input: Graph $G(V, E, A)$; Hyperparameter λ;
Output: The ideal node embedding dimension n for the graph G

Initialize $A^2 \leftarrow A^T A$;
Calculate $A_r^2 \leftarrow Eq.\ (16)$;
Estimate the feature Entropy by $H_f \leftarrow Eq.\ (7)$;
Calculate $D_r \leftarrow Eq.\ (15)$;
Calculate the structure entropy by $H_s \leftarrow Eq.\ (17)$;
Calculate the graph entropy by $H_g \leftarrow Eq.\ (18)$;
Obtain the ideal node embedding dimension $n \leftarrow H_g = 0$.

Algorithm. We summarize the whole process of MinGE in Algorithm 1. To be specific, we first perform sampling to approximate the sum operation and map the node embedding to the n-dimensional hyper-sphere with radius $\sqrt{n}$ to estimate the feature entropy H_f by Eq. (7) and (12). Then we calculate the structure entropy H_s by Eq. (17) based on the normalized degree vector in Eq. (15). Finally, according to the minimum entropy principle [Zhu *et al.*, 1997], we can get the ideal node embedding dimension n by setting $H_g = 0$. The calculated dimension n is taken as the ideal embedding dimension that can carry all information of node features and link structures.

Time complexity analysis. In theory the time complexity of MinGE is $O(n^2)$, due to matrix multiplication. However, graphs are usually sparse in practice, and we optimize MinGE with sparse matrix computations, which results in rather efficient computation much less than $O(n^2)$ in practice (as analyzed in Fig. 3).

4 Experiments

Dataset	Cora	Citeseer	Pubmed	Airport
# Nodes	2708	3327	19717	3188
# Edges	5429	4732	44338	18631
# Features	1433	3703	4500	4
# Classes	7	6	3	4
# Validation Node	500	500	500	500
# Test Nodes	1000	1000	1000	1000

Table 1: Summary of datasets used in our experiments.

In this section, we conduct extensive experiments on benchmark datasets to demonstrate the effectiveness and generalizability of the proposed MinGE for GNN methods.

4.1 Experimental Settings

Datasets. In our experiments, we choose the most authoritative benchmark datasets, Cora, Citeseer, Pubmed proposed by [Sen *et al.*, 2008] and Airport [Chami *et al.*, 2019]. More details of the datasets are shown in Table 1. We conduct node classification and link prediction on above benchmark datasets, and evaluate the performance of our MinGE.

Task	**Node Classification**																	
Dataset	Cora									Citeseer								
Dim.	20	40	60	80	Apt (98)	120	140	160	180	20	40	60	80	Apt (101)	120	140	160	180
MLP	54.5	59.8	57.3	57.3	**60.6**	59.3	58.3	58.2	60.0	58.8	60.0	59.7	59.2	**60.3**	60.0	58.5	59.2	59.3
GCN	82.5	83.2	83.1	83.1	**83.5**	82.6	83.0	82.9	82.8	64.7	66.2	67.0	67.2	67.4	**67.6**	67.0	67.4	67.2
GAT	82.9	83.9	82.4	83.1	**84.3**	82.8	83.9	80.4	82.8	68.9	69.5	68.5	67.9	**69.6**	69.4	69.4	68.6	69.1
GCNII	83.9	84.3	84.8	84.3	**85.1**	85.0	84.6	83.9	84.6	72.4	72.6	72.4	72.4	**73.5**	72.0	72.9	72.3	71.4
Dataset	Pubmed									Airport								
Dim.	20	40	60	80	100	Apt (123)	140	160	180	20	40	60	80	Apt (100)	120	140	160	180
MLP	71.1	72.3	71.5	72.8	72.5	**74.5**	72.9	73.3	72.5	50.2	48.6	45.7	47.7	**54.1**	53.6	51.2	52.6	52.0
GCN	78.0	78.2	77.5	78.8	78.4	**79.2**	79.0	79.2	77.9	63.6	63.9	64.3	64.8	**65.8**	64.9	64.0	64.4	64.2
GAT	77.4	77.5	78.9	77.5	77.6	**79.6**	79.1	77.8	78.5	65.3	64	65.9	63.9	**67.9**	67.8	65.1	66.8	67.3
GCNII	78.6	78.8	79.5	80.0	79.5	**80.5**	80.0	79.8	79.7	65.5	67.9	64.1	68.8	**70.4**	68.9	67.1	69.2	68.6
Task	**Link Prediction**																	
Dataset	Cora									Citeseer								
Dim.	20	40	60	80	Apt (98)	120	140	160	180	20	40	60	80	Apt (101)	120	140	160	180
MLP	85.8	88.3	88.1	90.3	**90.7**	89.8	90.4	89.1	90.0	90.1	90.2	90.3	91.9	**92.5**	91.3	91.6	91.8	92.2
GCN	87.6	90.4	91.3	90.5	**92.6**	92.1	92.4	87.6	87.7	90.6	91.9	92.7	92.9	**95.4**	94.7	94.5	94.4	95.4
GAT	91.1	93.4	92.8	93.5	**94.4**	94.1	94.3	93.7	94.2	96.5	97.2	97.1	97.5	**97.7**	97.5	97.3	97.5	97.4
GCNII	94.1	95.8	95.7	96.1	**96.6**	96.4	96.2	95.9	96.2	98.9	99.2	99.3	99.5	99.7	**99.8**	99.5	99.5	99.6
Dataset	Pubmed									Airport								
Dim.	20	40	60	80	100	Apt (123)	140	160	180	20	40	60	80	Apt (100)	120	140	160	180
MLP	85.2	90.8	91.0	92.3	92.4	**92.6**	92.4	92.3	91.9	89.6	89.8	89.6	89.9	**90.2**	89.9	89.4	89.9	89.8
GCN	90.3	93.1	93.8	93.4	93.7	**94.1**	93.7	94.0	93.9	89.1	91.8	92.9	92.0	**93.2**	92.6	92.6	93.1	92.8
GAT	87.0	89.9	91.2	90.9	90.8	**92.7**	90.8	90.9	92.5	91.8	92.6	92.5	**92.9**	**92.9**	92.8	92.7	92.7	92.6
GCNII	95.4	97.6	98	98.1	98.3	**98.5**	98.5	98.4	98.2	92.5	93.0	93.1	93.5	**93.8**	93.7	93.5	93.6	92.8

Table 2: Performance on node classification and link prediction. *Apt* denotes the appropriate dimensions calculated by MinGE.

GNNs. We validate the effectiveness and generalizability of MinGE on most popular models, including Multi-Layer Perception (MLP), Graph Convolution Network (GCN) [Kipf and Welling, 2017], Graph Attention network (GAT) [Velickovic *et al.*, 2018] and Graph Convolution Network via Initial residual and Identity mapping (GCNII) [Chen *et al.*, 2020].

Protocols. For node classification task, following the experiment setting of GAT [Velickovic *et al.*, 2018], only 20 samples per class are allowed for training. We further choose 80, 140 per class for Cora, Citeseer and Airport, and 2000, 3000 per class for Pubmed to crop the data set at different scales and postpone the index to divide the validation set with 500 and a test set with 1000. For link prediction task, we randomly split the edges with a ratio of 85%, 5%, and 10% for training, validation and test sets. For both tasks, comparative experiments are constructed by setting dimension interval to 20 for GNNs. Early stopping strategy is used on validation set with a patience of 100 epoches for all experiments. All results are the average of 10 times. Furthermore, experiments are conducted to validate the efficiency of MinGE.

Hyperparameters. MinGE has only one hyperparameter λ to control the ratio of the structure entropy. We set $\lambda = 1$ in experiments and analysis it in Section 4.2. The values of node embedding dimension calculated by the MinGE algorithm are 98, 101, 123, 100 for Cora, Citeseer, Pubmed and Airport separately. Where these dimensions are the size of the last layer of GNNs representation model. In practice, we set hidden as a hyperparemeter to control the dimension size. For other hyperparameters, we follow the settings in the original text for most models, which are considered to be optimal. However, because of the differences between the datasets, we also make corresponding adjustments. For the MLP model, the optimizer is Adam with learning rate is 0.01 and weight decay is 0.0005, epoch is 250, drop out is 0.5. For GAT and GCN models, we fully keep the default settings. For GCNII model, the hyperparameters partly follow the default settings and early stopping strategy. The differences are that the learning rate is 0.008 for Cora, the lamda is 0.3 for Citeseer, the dropout is 0.5 and lamda is 0.4 for pubmed, and layer is 16 for all datasets.

4.2 Results and Analysis

Performance on node classification and link prediction. For node classification and link prediction tasks, the results of GNNs with different dimensions on benchmark datasets are shown in Table 2. Where *Apt* denotes the appropriate dimension calculated by the MinGE in all Tables. From Table 2, by comparing with other dimensions, we can see that GNNs with the appropriate dimension selected from MinGE achieve the best or near the best performance. Besides, from the perspective of overall trend, we can see that it always locates in the peak area of the result change curve. Moreover, we also compare with the experimental results of GCN, GAT, GCNII models in the original texts. We find that GNNs with MinGE can improve 2%, 1.3%, 1.6% on Cora and 0.6%, 0.6%, 0.2% on Pubmed respectively. The above experimental results verify the effectiveness and generalizability of MinGE in guid-

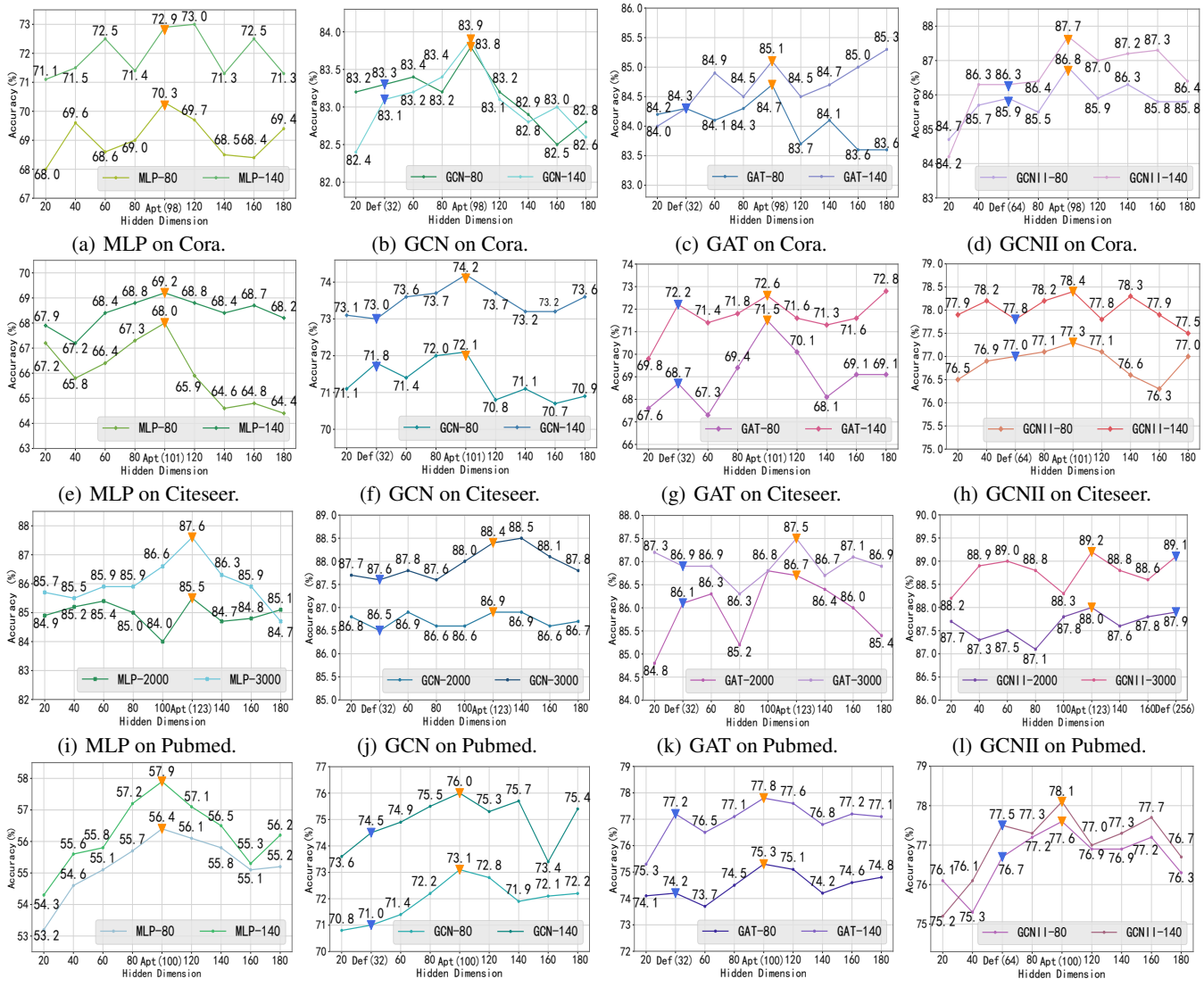

Figure 2: Node classification accuracy with different dimensions. Numbers like 80 and 140 in the legends denote the numbers of samples separately selected per class in the training set. The yellow inverted triangles denote the performance of GNNs with the appropriate dimensions selected by MinGE and the blue ones denote that with default dimensions mentioned in the original papers (MLP has no default dimension).

ing the NEDS for GNNs on different tasks and GNN models. The main reason is that around an appropriate dimension for a GNN model, the lower node embedding dimension is hard to cover all information due to the limitation of expression capacity, and the higher one tends to overfit the training data due to the excessive model complexity. In order to search an appropriate dimension for each model of each task, MinGE leverages the both feature entropy and structure entropy to measure the node features and link structures in the graph for NEDS, which leads to better performance.

Furthermore, in order to eliminate the influence of uneven data distribution and further verify the genralizability of MinGE, we conduct experiments on benchmark datasets cropped at different scales. The performance of GNNs with 80, 140 samples per class for Cora, Citeseer and Airport separately and 2000, 3000 samples per class for Pubmed are shown in Fig. 2. Where *Apt* denotes the ideal dimension selected by MinGE, and *Def* denotes the default dimension mentioned in original paper. According to the results, it is obvious that the performance of GNNs with the appropriate dimension selected by MinGE always locate in the peak center of the accuracy curves, and the accuracy of each model has been greatly improved as the training set increases. Furthermore, compared with GNNs with default dimension, GNNs with the ideal dimension all achieve better performance. The experimental results are in line with the conclusions verified in Table 2, which indicates generalizability of MinGE.

Memory Efficiency. GCNII is the current state-of-the-art model that can achieve better performance with deeper neu-

Layers	2	4	8	16	2	4	8	16
Dim.	Def	Def	Def	Def	Apt	Apt	Apt	Apt
Cora	80.2	82.3	82.8	83.5	82.5	83.0	83.9	85.1
Citeseer	66.1	67.9	70.6	72.0	72.1	72.7	73.2	73.5
Pubmed	77.7	78.2	78.8	80.3	79.7	79.9	80.0	80.5
Airport	64.1	64.3	65.0	66.7	66.5	67.8	69.2	70.4

Table 3: Node classification accuracy with varying numbers of GNN (GCNII) layers.

ral networks, as shown in Table 3. However, deeper network spends more memory and time during model training. Compared with the node dimension by default in original paper, MinGE can find an appropriate dimension for GCNII with different numbers of layers. In such a case, GCNII with MinGE can use fewer layers but achieve similar or better performance to a deeper GCNII with the node dimension by default setting. For example, the performance of GCNII with 16 layers and the node dimension by default is worse than GCNII with two layers and an appropriate dimension selected by MinGE on Citeseer. In summary, by using MinGE, GCNII can choose fewer layers to achieve good performance while memory limitation becomes crucial.

Time Efficiency. Considering fairness, we conduct experiments to compare the time that MinGE runs once with that GNNs run once. Fig. 3 shows the running time of MinGE and GNN methods. From these results, we can observe that MinGE costs much less runtime compared with training GNNs. It turns out our approach adds almost no additional computational overhead and is more efficient for an appropriate embedding dimension selection through MinGE. Note that, our approach only requires the runtime of MinGE with a single execution of different GNNs here, while finding a good dimension by grid search needs to run the GNNs multiple times, so as to cost more computational burden and time.

Hyperparameter Analysis. In our MinGE algorithm, there is only one hyperparameter λ, which controls the weight of structure entropy. The structure entropy provides additional help to the feature entropy. In order to determine the weight of the strcture entorpy, we use GCN to analyze λ on benchmark datasets. All dimensions calculated by MinGE

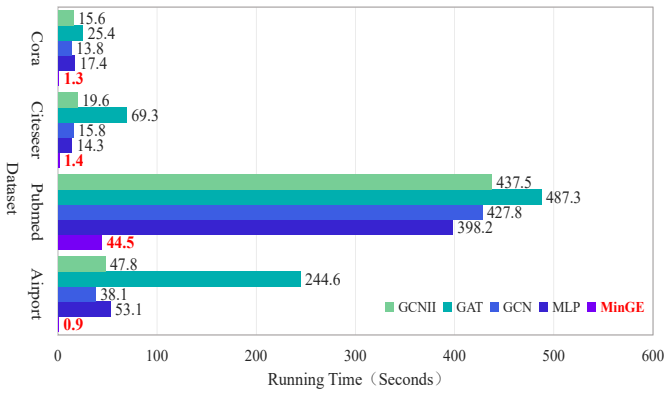

Figure 3: Running time of MinGE compared with GNNs.

Dataset	Cora				Citeseer			
λ	0.1	0.5	1	2	0.1	0.5	1	2
Dim	69	82	98	131	71	84	101	134
GCN	82.6	83.2	**83.5**	83.2	67.2	67.4	67.4	**67.6**
Dataset	Pubmed				Airport			
λ	0.1	0.5	1	2	0.1	0.5	1	2
Dim	86	102	123	164	71	84	100	133
GCN	78.8	78.4	**79.2**	79.1	64.5	64.7	**65.8**	64.3

Table 4: Node classification accuracy with varying hyperparameters.

are rounded up. From Table 4, it is not hard to see that the structure entropy is often equally important as the feature entropy for NEDS in general. Therefore, we can simply set λ to 1 by default given an arbitrary graph. We also observe that the graph entropy with $\lambda = 0.1$, which can be considered as feature entropy alone, can give pretty good results. Adding structure entropy with different weights can further improve the performance but with slight variance compared with experimental results as shown in Table 2.

5 Conclusion

In this paper, we revisit Node Embedding Dimension Selection (NEDS) for graph data from the perspective of minimum entropy principle. We proposed MinGE, a novel algorithm that combines well-designed feature entropy and structure entropy, to guide the NEDS for GNNs and addressed the challenge of how to directly select the appropriate node embedding dimension for graph data. We focused on MinGE that can maximize node information and structure information for appropriate NEDS by using minimum entropy principle. Moreover, to the best of our knowledge, MinGE is the first study that applies minimum entropy theory to NEDS of graph data. In practice, with this theory, we discovered the effectiveness and generalizability of our MinGE algorithm for popular GNNs. All of our discoveries were concretely validated on benchmark datasets.

Acknowledgements

The corresponding author is Jianxin Li. This work was supported by the NSFC program (No. U20B2053 and 62002007), S&T Program of Hebei through grant 20310101D, and SKLSDE-2020ZX-12. Lifang He is supported by NSF ONR N00014-18-1-2009 and NSFC of Guangdong Province under grant 2017A030313339. Philip S. Yu is supported by NSF under grants III-1763325, III-1909323, and SaTC-1930941.

References

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

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

[Chami *et al.*, 2019] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *NeurIPS*, pages 4869–4880, 2019.

[Chen *et al.*, 2020] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 2020.

[Dehmer, 2008] Matthias Dehmer. Information processing in complex networks: Graph entropy and information functionals. *Appl. Math. Comput.*, 201(1-2):82–94, 2008.

[Henderson and Moura, 1996] David Wilson Henderson and Eduarda Moura. *Experiencing geometry: on plane and sphere*. Prentice Hall, 1996.

[Hung and Yamanishi, 2020] Pham Thuc Hung and Kenji Yamanishi. Word2vec skip-gram dimensionality selection via sequential normalized maximum likelihood. *CoRR*, abs/2008.07720, 2020.

[Jr., 2003] Frederick P. Brooks Jr. Three great challenges for half-century-old computer science. *J. ACM*, 50(1):25–26, 2003.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

[Li and Pan, 2016] Angsheng Li and Yicheng Pan. Structural information and dynamical complexity of networks. *IEEE Trans. Inf. Theory*, 62(6):3290–3339, 2016.

[Li *et al.*, 2020] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *EMNLP (1)*, pages 9119–9130. Association for Computational Linguistics, 2020.

[Liu *et al.*, 2020] Zhonghua Liu, Kaiming Shi, Kaibing Zhang, Weihua Ou, and Lin Wang. Discriminative sparse embedding based on adaptive graph for dimension reduction. *Eng. Appl. Artif. Intell.*, 94:103758, 2020.

[Luo *et al.*, 2020] Gongning Luo, Wei Wang, Clara Tam, Kuanquan Wang, Shaodong Cao, Henggui Zhang, Bo Chen, and Shuo Li. Dynamically constructed network with error correction for accurate ventricle volume estimation. *Medical Image Analysis*, 64:101723, 2020.

[Mowshowitz and Dehmer, 2012] Abbe Mowshowitz and Matthias Dehmer. Entropy and the complexity of graphs revisited. *Entropy*, 14(3):559–570, 2012.

[Peng *et al.*, 2019] Hao Peng, Jianxin Li, Senzhang Wang, Lihong Wang, Qiran Gong, Renyu Yang, Bo Li, Philip Yu, and Lifang He. Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. *IEEE TKDE*, 2019.

[Raychaudhury *et al.*, 1984] C Raychaudhury, SK Ray, JJ Ghosh, AB Roy, and SC Basak. Discrimination of isomeric structures using information theoretic topological indices. *Journal of Computational Chemistry*, 5(6):581–588, 1984.

[Sahlgren, 2008] Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):pgs. 33–54, 2008.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

[Shannon, 1948] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[Shannon, 1953] Claude E. Shannon. The lattice theory of information. *Trans. IRE Prof. Group Inf. Theory*, 1:105–107, 1953.

[Shen *et al.*, 2020] Xiang-Jun Shen, Si-Xing Liu, Bing-Kun Bao, Chunhong Pan, Zheng-Jun Zha, and Jianping Fan. A generalized least-squares approach regularized with graph embedding for dimensionality reduction. *Pattern Recognit.*, 98, 2020.

[Sun *et al.*, 2021] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Phillip S Yu, and Lifang He. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. *arXiv preprint arXiv:2101.08170*, 2021.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR (Poster)*. OpenReview.net, 2018.

[Wang, 2019] Yu Wang. Single training dimension selection for word embedding with PCA. In *EMNLP/IJCNLP (1)*, pages 3595–3600. Association for Computational Linguistics, 2019.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[Xie *et al.*, 2020] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. When do gnns work: Understanding and improving neighborhood aggregation. In *IJCAI*, pages 1303–1309. ijcai.org, 2020.

[Xiong *et al.*, 2017] Kai Xiong, Feiping Nie, and Junwei Han. Linear manifold regularization with adaptive graph for semi-supervised dimensionality reduction. In *IJCAI*, pages 3147–3153. ijcai.org, 2017.

[Yin and Shen, 2018] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. In *NeurIPS*, pages 895–906, 2018.

[Zhu *et al.*, 1997] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.

[Zhu *et al.*, 2018] Xiaofeng Zhu, Cong Lei, Hao Yu, Yonggang Li, Jiangzhang Gan, and Shichao Zhang. Robust graph dimensionality reduction. In *IJCAI*, pages 3257–3263. ijcai.org, 2018.