Toward Unsupervised Graph Neural Network: Interactive Clustering and Embedding via Optimal Transport

Liang Yang, Junhua Gu
School of Artificial Intelligence, Hebei University of Technology
Tianjin, China

Lu Zhai, Di Jin
College of Intelligence and Computing, Tianjin
University, Tianjin, China

Chuan Wang, Xiaochun Cao
SKLOIS, Institute of Information Engineering, Chinese Academy of
Sciences, Beijing, China

Yuanfang Guo
School of Computer Science and Engineering, Beihang
University, Beijing, China

***Abstract*—Graph Neural Networks (GNNs) have become a hot topic for their potentials in modeling the irregular data and providing state-of-the-art performance in many fields. Most of the existing GNNs are deliberately designed for semi-supervised learning tasks, where supervision information (labelled node) is utilized to mitigate the oversmoothing problem of message passing. Unfortunately, the oversmoothing problem tends to be more severe in unsupervised tasks, since supervision information is not available. Since community structure/cluster is an essential characteristic of network, a natural approach to reduce the oversmoothing problem is to also constrain the node embeddings to maintain their own characteristics to prevent all the node embeddings from becoming too similar to be distinguished. In this paper, a novel Optimal Transport based Graph Neural Network (OT-GNN) is proposed to overcome the oversmoothing problem in unsupervised GNNs by imposing the equal-sized clustering constraints to the obtained node embeddings. To solve the combinatorial optimization problem, the constrained objective function of unsupervised GNN is relaxed to an Optimal Transport problem, and a fast version of the Sinkhorm-Knopp algorithm is adopted to handle large networks. Besides of being employed to train existing GNNs, such as Graph Convolutional Network (GCN) and Graph Attention Network (GAT), for node embedding and clustering in an unsupervised manner, OT-GNN can also be exploited to regularize other unsupervised GNNs, such as Graph AutoEncoder, for the link prediction task. Extensive experiments on node clustering, classification and link prediction demonstrate the superior performance of our proposed OT-GNN.**

***Index Terms*—graph neural network, network embedding, unsupervised learning, node clustering**

I. Introduction

Graph Neural Networks (GNNs) [1], [2] have become a hot topic in deep learning for their potentials in modeling irregular data. GNNs have been widely used and achieved state-of-the-art performance in many fields, such as computer vision, natural language processing, traffic forecasting, chemistry and medical analysis, etc. Existing GNNs fall into two categories, spectral methods [3] and spatial ones [4], [5]. Graph Convolutional Network (GCN) [6], which is a simple, well-behaved and insightful GNN, bridges above two perspectives by proving that the propagation can be motivated from a first-order approximation of spectral graph convolutions. Many efforts have been paid to enhance GCN from the perspective of propagation, such as learnable propagation weights in Graph Attention Network (GAT) [7] and structural neighbourhood in Geom-GCN [8].

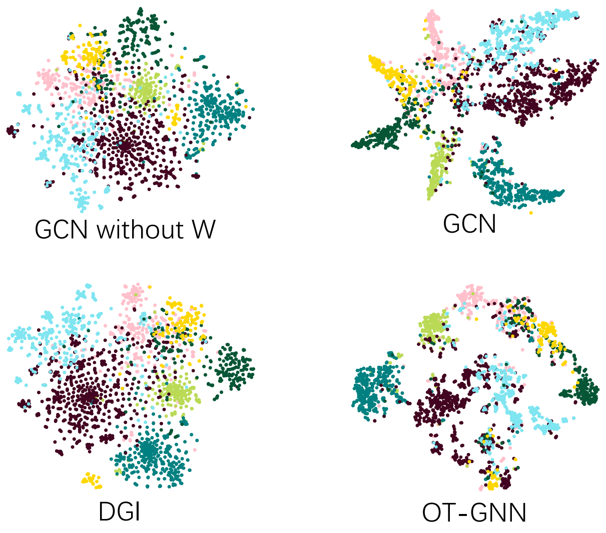

Fig. 1. Illustration of the oversmoothing in GNNs. By comparing the semi-supervised GCN and GCN without learnable parameter W, it can be observed that supervision information (labelled nodes) plays the role of mitigating the oversmoothing of graph convolution operation via the mapping function parameterized by W. This issue tends to be more severe in unsupervised GNNs, such as Deep Graph Infomax (DGI), where supervision information is not available. Our proposed OT-GNN can significantly alleviate it.

Most GNNs are deliberately designed for (transductive and inductive) semi-supervised learning tasks, where supervision information (labelled nodes) plays the important role of enhancing the distinguishability. In fact, Laplacian smoothing (spatial perspective) [9] and low-pass filtering (spectral perspective) [10], which have been accepted to be the reason for the success of GCNs, also significantly reduce the expressive power, i.e. data distinguishability, due to the over-smoothing problem [11]–[13]. Thus, the supervision information is utilized to mitigate this adverse effect. Fig. 1 illustrates this observation by comparing the semi-supervised GCNs and GCN without learnable parameter. Unfortunately, supervision information cannot fundamentally solve this problem for the following two reasons. First and foremost, supervision information is not available in unsupervised tasks, such as network embedding, node clustering and link prediction. Most existing unsupervised GNNs either reconstruct the original information (adjacency matrix and attribute matrix) [14]–[16] or maximize mutual information [17], [18] to retain as much information as possible. Fig. 1 visualizes the embeddings of Deep Graph Infomax (DGI) [17], which is a state-of-the-art unsupervised GNN. Thus, the oversmoothing problem tends to be more severe for unsupervised GNNs than for semi-supervised ones. Second, although learning the message aggregation from labelled nodes can alleviate smoothing problem, it also induces serious overfitting problem [19], [20], which also seriously effects the performance.

In this paper, a novel Optimal Transport based Graph Neural Network (OT-GNN) is proposed to overcome the oversmoothing problem in unsupervised GNNs, which constrains all the node embeddings to be similar. Since community structure/cluster is an essential characteristic of network [21], a natural approach to reduce the oversmoothing problem is to also constrain the node embeddings to maintain their own characteristics to prevent all the node embeddings from becoming too similar to be distinguished. According to this intuition, OT-GNN elegantly trains the GNNs by imposing the clustering constraints to the obtained node embeddings. Specifically, the objective function of unsupervised GNNs is firstly derived from the cross-entropy loss of the semi-supervised node classification. This objective function can be optimized with respect to the model parameters and node labels. Unfortunately, it leads to a degenerate solution which assigns all the nodes into a single class due to the oversmoothing problem. Then, to prevent this degenerate solution, clustering constraints are added to ensure the nodes being uniformly classified into classes of equal size. To solve the formulated combinatorial optimization problem, the constrained objective function of unsupervised GNN is relaxed to an Optimal Transport problem, which can be then solved via linear programming in a polynomial time. To speedup this process on large networks, a fast version of the Sinkhorm-Knopp algorithm, which employs a regularization term to the loss function, is adopted, and an iterative algorithm is proposed with additional complexity proportional to the network size. Besides of being employed to train the GNNs, such as Graph Convolutional Network (GCN) [6] and Graph Attention Network (GAT) [7] for node embedding and clustering in an unsupervised manner, OT-GNN can also be utilized to regularize other unsupervised GNNs, such as Graph AutoEncoder (GAE) [14], for the link prediction task.

The main contributions of this paper are summarized as follows:

- To overcome the oversmoothing problem in unsupervised GNNs, we propose a novel Optimal Transport based Graph Neural Network (OT-GNN) with interactive node embedding and clustering.
- To efficiently train the unsupervised GNNs, we relax its constrained objective function to an Optimal Transport problem and solve it via a fast version of the Sinkhorm-Knopp algorithm.
- We conduct extensive experiments on node clustering, classification and link prediction to demonstrate the superior performance of our proposed OT-GNN.

II. Related Work

In this section, recent progresses in graph neural networks are first reviewed. Then, oversmoothing problem in GNNs and methods used to alleviate it are provided. Next, unsupervised graph neural networks are elaborated. Finally, exiting methods, which combine clustering and embedding, are introduced.

Graph Neural Networks: Graph Neural Networks (GNNs) [1], [2] aim at applying the expressive representation power of deep learning to irregular data, i.e, graphs. GNNs fall into two categories, spectral methods [3] and spatial ones [4], [5]. Spectral methods [3], which originate from the spectral graph theory in graph signal processing, consider the node attributes as the signals over the graph and operate them in the spectral space of the graph. On the other hand, spatial methods [4] propagate the node attributes along the edge by leveraging the message passing mechanism [5]. Graph Convolutional Network (GCN) [6], which is a simple, well-behaved and insightful GNN, bridges above two perspectives by proving that the propagation can be motivated from a first-order approximation of spectral graph convolutions. Graph Attention Network (GAT) [7] improves the fixed propagation weights in GCN to learnable ones via attention mechanism. Geom-GCN [8] enhances the local propagation in both GCN and GAT to geometric aggregation in structural neighbourhood by leveraging network embedding.

Oversmoothing in GNNs: Li et al [9] and Wu et al. [10] interpret the success of GCN from Laplacian smoothing (spatial perspective) and low-pass filtering (spectral perspective), respectively. PageRank-GCN [12] integrates personalized PageRank to GCN to constrain the propagation. JKNet [11] employs dense connections for multi-hop message passing, while DeepGCN [22] incorporates residual layers, dense connections and dilated convolutions into GCNs to facilitate the development of deep architectures. Recently, [13] investigates the loss of expressive power of GNNs via their asymptotic behaviors by generalizing the forward propagation of a GCN as a specific dynamical system. PairNorm [23] proposes a normalization layer to prevents all the node embeddings from becoming too similar. DropEdge [20] randomly removes a certain number of edges from the input graph at each training epoch to act like a data augmenter and to reduce the adverse effect of message passing. However, existing methods neither work without label information, such as JKNet [11] and DeepGCN [22], nor exploit the property of the attributed graph, such as PairNorm [23] and DropEdge [20].

Unsupervised Graph Neural Networks: To extend GNNs for unsupervised tasks, there are few work on unsupervised

GNN [14], [15], [17], [18]. These methods fall in two categories, mutual information based methods and reconstruction based ones. Mutual information based methods train GNNs by maximizing the mutual information. Deep Graph Infomax (DGI) [17] maximizes mutual information between local embedding and global graph representation. Graph Mutual Information (GMI) [18] reconstruction based methods and maximizes mutual information between node embeddings before and after GNNs, and mutual information between reconstructed topology and original topology. Reconstruction based methods train GNNs by reconstructing original data. Graph AutoEncoder (GAE) and its variational version (VGAE) [6] take GCN [6] as encoder and reconstruct the adjacency matrix for link prediction. ARGAE [15] enhances GAE to seek robust embedding by adding an adversarial module on the obtained embedding. Co-embedding methods [16], [24] encode both node and attribute (co-embedding) from GCN and reconstruct topology and attribute matrix. According to Denoising AutoEncoders (DAE) [25], AutoEncoders can also be considered as maximizing a lower bound of mutual information to retain as much information as possible. Without supervision information (labelled nodes), the oversmoothing problem tends to be more severe for unsupervised GNNs than for semi-supervised ones.

Clustering based Network Embeddings: There are few work, which combines network embedding with clustering (community detection) [26]–[31]. Some of them use the modularity [21], which is widely adopted in community detection, to regularize the embedding learning [26], [27], while others employ the clustering error in K-means, i.e., summarization of distance between node and assigned cluster center, to regularize the embedding learning [29], [30]. ComE [31] and vGraph [28] assume that the node embeddings are drawn from a mixture distribution, and iteratively update node embedding and community assignment. Note that, oversmoothing problem often doesn't exit in most of the embedding methods adopted by them. For, example, the negative sampling has prevents DeepWalk, which employed by [29], [31], from oversmoothing. Thus, clustering strategy used in these methods play of role of enhancing cluster structure. However, our proposed OT-GNN constrains embeddings being uniformly partitioned into subsets of equal size to overcome oversmoothing, which makes the lose of cluster structure. Thus, the problem tackled by OT-GNN is too difficult to be handled by existing clustering strategy, such as (Gaussian) mixture model in [28], [31] and K-means in [29], [30], which can't prevent the degenerate solution where all the node embedding are classified into one clusters.

III. Preliminaries

In this section, the notations are first given. Then, the basic concepts in Graph Neural Networks are provided.

A. Notations

A network can be represented by an attributed graph $G = (V, E, X)$. $V = \{v_i | i = 1, ..., N\}$ is a set of $|V| = N$ vertices, where v_i is associated with a feature $x_i \in \mathbb{R}^K$. $X \in \mathbb{R}^{K\times N}$ represents the collection of the features. Each column of X corresponds to a node. E stands for a set of edges. Each of the edges connects two vertices in V. The adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{N\times N}$ is obtained according to the network topology. If an edge connects the vertices v_i and v_j, $a_{ij} = 1$, and vice versa. If self-edges are allowed in the network, then $a_{nn} = 1$. Otherwise $a_{nn} = 0$. $d_n = \sum_j a_{nj}$ stands for the degree of v_n while $D = \mathrm{diag}(d_1, d_2, ..., d_N)$ is the degree matrix of A. The graph Laplacian and its normalized form are defined as $L = D - A$ and $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, respectively.

For semi-supervised node classification task, a set of vertices $V_l \subset V$ are labelled. $y_i \in \{1, 2, 3..., F\}$ is utilized to represent the label of vertex $v_i \in V_l$, where F is the number of classes. A typical semi-supervised node classification algorithm classifies other nodes in $V - V_l$. For simplicity, the first $|V_l|$ nodes $\{v_i\}_{i=1}^{|V_l|}$ are assumed to be labelled, i.e., $\{y_1, y_2, ..., y_{|V_l|}\}$ are known.

B. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are designed from either the spatial or spectral perspective. Spectral ones originate from spectral graph convolution, while spatial ones are designed along attribution propagation pipeline. By simplifying the time-consuming spectral graph convolution operation of existing spectral methods, Graph Convolutional Network (GCN) [6] defines the graph convolution operation as

$$T^{GCN} = \sigma(H^{GCN}) = \sigma(WX\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}), \qquad (1)$$

where $\tilde{A} = A + I_N$ and $\tilde{D}_{nn} = \sum_j \tilde{A}_{nj} = d_n + 1$. W stands for the trainable weight matrix of a fully-connected layer. $\sigma(.)$ represents the nonlinear activation function, such as ReLU and softmax. $H^{GCN} \in \mathbb{R}^{K\times N}$ and T^{GCN} denotes the node representations (embeddings) before and after nonlinear activation function $\sigma(.)$, respectively. For clarity, H^{GCN} and T^{GCN} are called node feature and node representation, respectively. Eq. (1) can be formulated in a node-wise form as

$$\begin{aligned} t_i^{GCN} &= \sigma(h_i^{GCN}) \\ &= \sigma\left(W \sum_{j\in N(i)\cup\{i\}} \frac{1}{\sqrt{(d_i+1)(d_j+1)}} x_j\right), \end{aligned} \qquad (2)$$

where h_i^{GCN} and t_i^{GCN}, the i^{th} columns of H^{GCN} and T^{GCN}, are the feature and representation of node v_i, respectively. $N(i)$ represents the neighbourhood of vertex v_i. Some recent work interprets GCN from smoothing and low-pass filtering. Li et al. [9] concludes the mechanism and success of GCN as performing a symmetric Laplacian smoothing operation before the actual predictions. The performance based on the smoothed attribute obviously outperforms that based on the original attributes. Wu et al. [10] reduce the unnecessary complexity and redundant computation of GCN to Simplified Graph Convolution (SGC) by successively removing nonlinearities and collapsing weight matrices between consecutive layers, and show that it corresponds to a fixed low-pass filter followed by a linear classifier. Although GCN and SGC significantly improve the performance, its main drawback is the fixed propagation weight $\frac{1}{\sqrt{(d_i+1)(d_j+1)}}$, which is completely determined by the degrees of the two connected nodes.

To overcome that drawback, Graph Attention Network (GAT) [7] attempts to learn the propagation weight by leveraging the self-attention mechanism as

$$\begin{aligned}\alpha_{ij} &= \mathrm{softmax}\left(a(Wx_i, Wx_j)\right) \\ &= \frac{\exp\left(\mathrm{LeakyReLU}(b^T[Wx_i||Wx_j])\right)}{\sum_{k\in N(i)}\exp\left(\mathrm{LeakyReLU}(b^T[Wx_i||Wx_k])\right)},\end{aligned} \tag{3}$$

where $a(.,.)$ stands for a neural network, $||$ denotes the concatenate operator and $b \in \mathbb{R}^{2F}$ is the learnable vector for propagation weights. Then the node representation in Eq. (2) can be enhanced to

$$t_i^{GAT} = \sigma(h_i^{GAT}) = \sigma\left(W \sum_{j\in N(i)} \alpha_{ij} x_j\right), \tag{4}$$

where both b and W are the trainable parameters. h_i and t_i are adopted to represent the node feature and embedding by ignoring the superscript, if we don't emphasize the methods used to obtain them.

1) Semi-supervised GCNs: To obtain the learnable parameters for (transductive and inductive) semi-supervised node classification task, GNNs can be trained by minimizing the cross-entropy between the predictions and given labels of the labelled vertices $v_i \in V_l$ as

$$\mathcal{L}_{sup} = -\sum_{v_i\in V_l}\sum_{y=1}^{F} \delta(y_i - y)\log(t_{iy}), \tag{5}$$

where t_{iy} is the y^{th} element of node embedding t_i. $\delta(.)$ is Dirac delta function and $\delta(y_i - y) = 1$ if and only if $y = y_i$, otherwise $\delta(y_i - y) = 0$. By setting the dimensionality of t_i to the number of classes F and adopting softmax as the nonlinearity function $\sigma(.)$, t_{iy} can be considered as the predicted probability of node v_i being classified to class y as

$$t_{iy} = p(y|h_i) = \mathrm{softmax}_y(h_i) = \frac{\exp(h_{iy})}{\sum_k \exp(h_{ik})}. \tag{6}$$

2) Unsupervised GCNs: To utilize the GCNs without any supervision, unsupervised GNNs is often trained by reconstructing the given information, such as topology and node attribute. Graph AutoEncoder (GAE) [14] takes GCN as encoder and reconstructs the adjacency matrix from

$$\bar{a}_{ij} = \mathrm{sigmoid}(t_i^T t_j). \tag{7}$$

By minimizing the cross-entropy between the observed adjacency matrix A and the reconstructed adjacency matrix $\bar{A} = [\bar{a}_{ij}] \in \mathbb{R}^{N\times N}$

$$\mathcal{L}_{topo}(p) = -\sum_{i,j=1}^{N}\left(a_{ij}\log(\bar{a}_{ij}) + (1-a_{ij})\log(1-\bar{a}_{ij})\right), \tag{8}$$

the parameters W in GCN can be trained. GAE achieves superior performance in link prediction [14].

IV. Proposed Models

In this section, a novel unsupervised graph neural network (GNN), Optimal Transport based GNN (OT-GNN) is proposed. First, unsupervised GNN is derived from the semi-supervised one by adding clustering constraint to prevent oversmoothing, i.e., embedding collapsed. Then, the objective function of unsupervised GNN is interpreted from Optimal Transport perspective for efficient solution, and a fast version of Sinkhorn-Knopp algorithm is adopted to efficiently solve it. Finally, the proposed OT-GNN is utilized to enhance the existing unsupervised Graph AutoEncoder (GAE).

A. Unsupervised Graph Neural Networks

In this subsection, we derive unsupervised graph neural network from the semi-supervised one in Section III-B1. By integrating the probability interpretation of t_{iy} (Eq. (6)) into the loss function of semi-supervised node classification (Eq. (5)), the objective function can be reformulated as

$$\mathcal{L}\left(p|\{y_1, y_2, ..., y_{|V_l|}\}\right) = -\sum_{v_i\in V_l}\log\left(p(y_i|h_i)\right), \tag{9}$$

where $\mathcal{L}\left(p|\{y_1, y_2, ..., y_{|V_l|}\}\right)$ denotes that node labels $\{y_1, y_2, ..., y_{|V_l|}\}$ are required to train the GNN $t_i = p(.|h_i)$ parameterized by W and b in Eq. (6). After training, parameters W and b in Eqs (2) and (4) are obtained. Thus the labels of unlabelled nodes $v_j \in V - V_l$ can be obtained from $p(y|h_j)$, and the objective function in Eq. (9) can be extended to

$$\begin{aligned}&\mathcal{L}\left(p, \{y_j\}_{j=|V_l|+1}^{|V|}|\{y_i\}_{i=1}^{|V_l|}\right) \\ &= -\sum_{v_i\in V_l}\log\left(p(y_i|h_i)\right) - \sum_{v_j\in V-V_l}\log\left(p(y_j|h_j)\right) \\ &= -\sum_{v_i\in V}\log\left(p(y_i|h_i)\right) \\ &= -\sum_{v_i\in V}\sum_{y=1}^{F}\delta(y_i - y)\log\left(p(y|h_i)\right),\end{aligned} \tag{10}$$

where the term corresponding to the unlabelled nodes, i.e. $\sum_{v_j\in V-V_l}\log\left(p(y_j|h_j)\right)$, equals to 0 and achieves its minimum, since y_j is obtained from $p(y|h_j)$ for unlabelled nodes. Therefore, semi-supervised node classification is achieved by interactively optimizing the objective function in Eq. (10) with respect to the model $p(y|h)$ parameterized by W and b and node labels $\{y_i\}_{i=1}^{|V|}$. This process works if and only if part of node labels $\{y_i\}_{i=1}^{|V_l|}$ are known and utilized to constrain the optimization.

Unfortunately, optimizing this objective function leads to a degenerate solution if all the nodes are unlabelled. Eq. (10) can be trivially minimized by training the model to assign all the nodes into a single class. To prevent this degenerate solution, constraints must be added to the objective function. To facilitate it, we first represent the labels as a posterior distribution $q(y|h_i)$ instead of a deterministic Dirac delta function $\delta(y_i - y)$, and reformulate Eq. (10) as

$$\mathcal{L}(p, q) = -\sum_{v_i \in V} \sum_{y=1}^{F} q(y|h_i) \log \left(p(y|h_i)\right), \tag{11}$$

where $q(y|h_i) \in \{0, 1\}$ and $\sum_y q(y|h_i) = 1$. Note that $q(y|h_i)$ is utilized to denote the posterior distribution of node v_i's label instead of indicating label y is learned from h_i as $p(y|h_i)$. Then, we aim to add some constraints to ensure nodes being classified into F classes. Since the distribution of cluster size is unknown, we simply assume that all nodes are uniformly classified into F clusters of equal size to prevent the degraded situation where most nodes are classified into one big cluster. Note that although the equal-sized clustering doesn't perfectly meet the ground-truth, it is an effective way to prevent all the node embeddings from being too similar. In experiments, the number of clusters, i.e., F , is set larger than the ground-truth, and the obtained embeddings are adopted for clustering instead of the assigned clusters. Thus, the objective function for unsupervised graph neural network can be written as

$$\begin{aligned} \arg\min_{q,p} \quad & -\sum_{v_i \in V} \sum_{y=1}^{F} q(y|h_i) \log \left(p(y|h_i)\right) \\ \text{s.t.} \quad & \forall y : q(y|h_i) \in \{0, 1\} \text{ and } \sum_{v_i \in V} q(y|h_i) = N/F, \end{aligned} \tag{12}$$

where the constraint $\sum_{v_i \in V} q(y|h_i) = N/F$ makes sure that there are N/F nodes are assigned to class y, thus all the nodes are equally partitioned. Unfortunately, this objective function is difficult to minimize, since it is a combinatorial optimization problem with respect to q.

B. Unsupervised GNNs as Optimal Transport

In this subsection, to address the difficulty in optimization, we interpret the objective function of unsupervised graph neural network, i.e., Eq. (12), with respective to the assignment q from the perspective of Optimal Transport. Let $P = [p_{yi}] \in \mathbb{R}_+^{K \times N}$ and $Q = [q_{yi}] \in \mathbb{R}_+^{K \times N}$ be the joint probability matrix estimated from the GNN and the joint probability of assignment, respectively, where

$$p_{yi} = p(y, h_i) = p(y|h_i)p(h_i) = p(y|h_i)\frac{1}{N}, \tag{13}$$

$$q_{yi} = q(y, h_i) = q(y|h_i)p(h_i) = q(y|h_i)\frac{1}{N}. \tag{14}$$

The loss function in Eq. (12) can be rewritten as

$$-\sum_{v_i \in V} \sum_{y=1}^{F} q(y|h_i) \log \left(p(y|h_i)\right) = < Q, -\log P > \tag{15}$$

where log is an element-wise operation. $< A, B >= \sum_i \sum_j a_{ij} b_{ij}$ is the Frobenius inner product of two matrices. To alleviate the difficulty of combinatorial optimization problem with respect to q, matrix Q is relaxed to belonging to transportation polytope [32] as

$$U(r, c) := \{Q \in \mathbb{R}_+^{F \times N} | Q\mathbf{1} = r, \ Q^T\mathbf{1} = c\} \tag{16}$$

Algorithm 1: Optimal Transport based GNNs

```
Input: Adjacency matrix $A \in \mathbb{R}^{N \times N}$, feature matrix
       $X \in \mathbb{R}^{N \times K}$, the number of classes $F$ and
       hyper-parameter $\lambda$.
Output: Node representation matrix $H$.
1  Initialize $H$ via Eq. (1) where $W$ is set as identity
     matrix $I$, $\beta = c = \frac{1}{N}\mathbf{1}$, $\gamma = r = \frac{1}{F}\mathbf{1}$ ;
2  while not convergence do
3      %—— Updating $q$ with fixed $p$ ——
4      while not convergence do
5          Update $\gamma$ via Eq. (21);
6          Update $\beta$ via Eq. (22);
7          Update $Q$ via Eq. (20);
8      end
9      %—— Updating $p$ with fixed $q$ ——
10     Update $p$ in Eq. (6), which is parameterized by $W$
         and $b$, via batch gradient decent;
11 end
12 Obtain $H$ from Eqs (2) or (4) with learned $W$ and $b$;
13 return $H$.
```

where $\mathbf{1}$ stands for the vector of all ones with appropriate dimensionality. r and c are the fixed vectors used to constrain the summarizations of Q along row and column, respectively. Since $Q = [q_{yi}]$ denotes the joint distribution of label y and node v_i, where each column corresponds to a vertex, $c \in \mathbb{R}^N$ is set as $\frac{1}{N}\mathbf{1}$. According to the equal-size cluster constraints $\sum_{v_i \in V} q(y|h_i) = N/F$ in Eq. (12) and $q_{yi} = q(y|h_i)\frac{1}{N}$ in Eq. (14), $r \in \mathbb{R}^K$ is set as $\frac{1}{F}\mathbf{1}$, i.e.,

$$c = \frac{1}{N}\mathbf{1}, \quad r = \frac{1}{F}\mathbf{1}. \tag{17}$$

Thus, optimizing unsupervised GNNs in Eq. (12) with respective to the assignment Q can be relaxed to the following optimal transport problem [32]

$$\arg\min_{Q \in U(r,c)} < Q, -\log P >, \tag{18}$$

whose solution can be obtained via linear program problem in polynomial time.

C. Optimization

In this section, an algorithm, which iteratively updates q and p with another fixed, is proposed to minimize the objective function of unsupervised GNNs shown in Eq. (12). If GCN is adopted as the basic GNN, i.e., h_i is obtained from Eq. (2), the obtained unsupervised GNN is OT-GCN, while OT-GAT denotes that GAT is adopted as the basic GNN, i.e., h_i is obtained from Eq. (4). The overall algorithm is shown in Algorithm 1.

1) Updating p with fixed q: If q is fixed, optimizing Eq. (12) with respective to p, which is parameterized by W and b, is equivalent to optimizing Eq. (5) or (9), where labels of all the nodes $\{y_i\}_{i=1}^{|V|}$ are assumed to have been obtained from q. It can be achieved via gradient descent as existing semi-supervised GNNs, such as GCN [6] and GAT [7].

2) Updating q with fixed p: If p is fixed, optimizing Eq. (12) with respective to q can be achieved via Optimal Transport. By connecting the unsupervised GNNs optimization in Eq. (12) with respective to the assignment q with the optimal transport problem in Eq. (18) in Section IV-B, assignment can be obtained via linear programming in polynomial time. To further speedup this process on large networks, a fast version of the Sinkhorm-Knopp algorithm [33] is adopted. It employs a regularization term to the loss function of Eq. (18) as

$$\arg\min_{Q \in U(r,c)} < Q, -\log P > + \frac{1}{\lambda}\mathrm{KL}(Q||rc^T), \quad (19)$$

where $\mathrm{KL}(.||.)$ stands for the Kullback-Leibler divergence. $rc^T \in \mathbb{R}^{K\times N}$, each element of which is $1/(NK)$, can be considered as the non-informative uniform prior distribution of Q. λ balances the convergence speed and the closeness of Eq. (19) to the optimal transport in Eq. (18). According to [33], optimizing Eq. (19) is equivalent to optimizing Eq. (18) with moderate value of λ. By introducing this regularization, the optimal Q can be analytically obtained as

$$Q = \mathrm{diag}(\gamma)P^{\lambda}\mathrm{diag}(\beta), \quad (20)$$

where the exponentiation P^{λ} is element-wise operation. γ and β are the two vectors used to ensure that the obtained Q is a probability matrix. $\mathrm{diag}(.)$ creates diagonal matrix from vector. According to [33], γ and β can be iteratively updated as

$$\gamma_y = [P^{\lambda}\beta]_y^{-1}, \quad (21)$$
$$\beta_i = [\gamma^T P^{\lambda}]_i^{-1}, \quad (22)$$

each of which consists of a matrix-vector multiplication with complexity $\mathcal{O}(NK)$. This complexity is linear with the size of the network. At beginning, β and γ are initialized as c and r, respectively. In experiments, just a few iterations, which require only $\mathcal{O}(NK)$ operations, are needed to obtain Q.

D. Combination with Graph AutoEncoder

In the above sections, unsupervised OT-GNN is derived from semi-supervised node classification by leveraging the community (cluster) property of networks. Here, we show OT-GNN can also be employed to enhance exiting unsupervised GNNs, such as Graph AutoEncoder (GAE) [14]. Specifically, the topology reconstruction error in Eq. (8) can be combined with Eq. (12) as

$$\arg\min_{q,p} \quad \mathcal{L}_{topo}(p) + \varepsilon\mathcal{L}(p,q) \quad (23)$$
$$\text{s.t.} \quad \forall y : q(y|h_i) \in \{0,1\} \text{ and } \sum_{v_i \in V} q(y|h_i) = N/K,$$

where $\mathcal{L}(p,q)$ is shown in Eq. (11) and ε is the hyper-parameter to balance the impact of two parts. If $\varepsilon = 0$, Eq. (23) degenerates to the objective function of GAE in Eq. (8). Note that although $\mathcal{L}_{topo}(p)$ and $\mathcal{L}(p,q)$ are both unsupervised term, they play different roles. $\mathcal{L}_{topo}(p)$ trains encoder p in Eq. (6) for link prediction by reconstructing the topology. $\mathcal{L}(p,q)$ prevents the oversmoothing in GNNs, thus can be regarded as the regularization to topology reconstruction in $\mathcal{L}_{topo}(p)$. The optimization of Eq. (23) is similar to that of Eq. (12), i.e. iteratively updating q and p. Updating q with fixed p can be achieved via Eq. (20). Updating p, which is parameterized by W, with fixed q can be achieved via gradient descent. Thus, OT-GAE can also be optimized via Algorithm 1.

TABLE I
DATASETS.

Datasets	Nodes	Edges	Categories	Attributes
Texas	183	328	5	1,703
Cornell	195	304	5	1,703
Washington	217	446	5	1,703
Wisconsin	262	530	5	1,703
Cora	2,708	5,429	7	1,433
Citeseer	3,312	4,732	6	3,703
Pubmed	19,729	44,338	3	500

V. Evaluations

In this section, we validate the proposed OT-GNN by quantitatively evaluating the performances of its two instances, OT-GCN and OT-GAT, in the node clustering and classification, and the performance of OT-GAE in link prediction. Finally, a visualization of obtained embedding is provided to qualitatively demonstrate how the proposed OT-GNN alleviates the oversmoothing in unsupervised GNNs.

A. Datasets

The experiments are conducted on three commonly utilized citation networks, Cora, CiteSeer and PubMed. In each network, nodes and edges are research papers and undirected citations, respectively. In addition to the network structure, node content, which is represented by the bag-of-word representation of the documents, is available. According to the disciplines, papers are categorized into various classes. Besides, four more networks, including Cornell, Texas, Washington and Wisconsin, are employed. Each network, which is the collection of webpages from an American university, is the sub-network of the WebKB network. Nodes and edges are webpages and links between them, respectively. These webpages are manually classified into one of the following five classes: course, faculty, student, project and staff. Dataset statistics are summarized in Table I.

B. Baselines

To demonstrate the superiority of our proposed OT-GNN on representation learning, 11 state-of-the-art baselines are employed. These methods fall in two categories, network embedding methods and unsupervised graph neural networks. Network embedding methods can also be further classified into two sub-categories, methods based on topology and methods based on both topology and node attribute. The first sub-category includes DeepWalk [34], node2vec [35], LINE [36] and GraRep [37]. The second sub-category includes Text Augmented DeepWalk (TADW) [38], Tri-Party Deep Network Representation (TriDNR) [39], Accelerated Attributed Network Embedding (AANE) [40] and Attributed Social Network

TABLE II
COMPARISON ON NODE CLUSTERING IN TERMS OF AC AND NMI.

Metrics (%)	Methods	Cornell	Texas	Washington	Wisconsin	Cora	Citeseer	Pubmed
AC	DCSBM	37.95	48.09	31.80	32.82	38.48	26.57	53.64
	EdMot-SC	30.77	48.09	48.39	32.06	27.07	25.60	39.29
	LDA	44.62	56.28	44.62	44.62	37.19	31.34	46.30
	Block-LDA	46.15	54.10	39.17	49.62	25.52	24.35	49.01
	PCLDC	30.26	38.80	29.95	30.15	34.08	24.85	63.55
	SCI	45.64	62.30	51.15	50.38	40.62	27.98	47.39
	TLSC	47.69	**65.02**	51.61	49.23	47.62	35.74	61.38
	DeepWalk	36.05	46.72	40.76	38.76	45.61	36.21	64.84
	Node2Vec	33.85	47.54	37.33	49.62	56.30	40.76	65.56
	LINE	39.49	53.38	52.68	45.43	30.72	25.01	43.11
	GraRep	31.79	36.72	31.36	33.24	48.29	31.20	54.43
	TADW	47.69	59.23	50.30	55.00	55.39	36.19	57.46
	AANE	37.28	30.49	41.57	30.53	18.51	21.76	34.55
	TriDNR	38.21	47.54	43.59	43.70	31.56	34.44	59.29
	ASNE	41.08	41.53	48.80	55.30	39.44	31.17	65.13
	VGAE	36.72	48.35	43.73	43.28	57.06	53.46	58.64
	ARVGE	38.21	41.48	43.66	42.81	64.08	43.50	58.76
	DGI	38.46	51.91	48.85	45.80	63.51	67.54	64.07
	OT-GCN	51.40	63.76	55.27	54.41	64.62	68.92	66.36
	OT-GAT	**52.79**	64.67	**56.58**	**56.99**	**66.70**	**69.54**	**67.32**
NMI	DCSBM	9.69	16.65	9.87	3.14	17.07	4.13	12.28
	EdMot-SC	9.67	18.79	18.66	11.28	9.58	11.26	0.21
	LDA	21.09	31.29	38.48	46.56	14.61	9.13	10.55
	Block-LDA	6.81	4.21	3.69	10.09	2.42	1.41	6.58
	PCLDC	7.23	10.37	5.66	5.01	17.54	2.99	26.84
	SCI	11.44	17.84	12.37	17.03	19.26	4.87	5.59
	TLSC	13.61	23.92	17.63	16.65	33.2	23.16	19.63
	DeepWalk	7.06	6.16	5.66	7.65	31.51	10.58	25.55
	Node2Vec	6.65	4.49	2.94	7.86	42.02	12.99	25.02
	LINE	9.27	18.16	18.95	9.39	10.13	5.62	7.17
	GraRep	8.80	12.43	5.18	8.02	35.46	9.61	17.76
	TADW	11.13	10.90	11.63	17.52	31.60	37.92	20.11
	AANE	9.55	3.52	13.19	2.86	0.40	1.19	0.01
	TriDNR	7.20	4.32	8.10	6.60	12.19	9.59	19.28
	ASNE	11.11	12.63	17.43	18.94	16.28	7.31	25.61
	VGAE	7.77	8.52	9.03	9.31	42.92	27.93	17.83
	ARVGE	10.26	7.28	12.6	11.92	44.95	22.72	18.40
	DGI	12.52	13.98	15.64	13.69	49.76	42.74	26.64
	OT-GCN	31.66	23.17	45.76	47.19	50.89	43.74	27.62
	OT-GAT	**24.16**	**34.57**	**46.97**	**47.63**	**52.15**	**55.61**	**30.92**

Embedding (ASNE) [41]. The unsupervised graph neural networks includes (Variational) Graph AutoEncoder (VGAE) [14], Adversarially Regularized Graph Autoencoder (ARVGE) [15] and Deep Graph Infomax (DGI) [17]. Note that, both [14] and [15] include two version, i.e., one based on AutoEncoder and one based on Variational AutoEncoder. The methods based on Variational AutoEncoder, which often achieve better perforamnce, are adopted.

Additionally, another 7 state-of-the-art methods, which directly obtain clustering results without embeddings, are compared with our proposed OT-GNN in node clustering task. These methods are all generative models. Degree Corrected SBM (DCSBM) [42] and Edge enhanced Motif-aware community detection (EdMot-SC) [43] are only based on network topology. Latent Dirichlet Allocation (LDA) [44] is a topic model based on node content. The approaches based on both topology and node attribute include Block-LDA [45], PCLDC [46], Semantic Community Identification (SCI) [47] and Two-Level Semantic Community (TLSC) [48].

C. Experimental Settings

1) Parameter Settings: To ensure fairness, embedding dimension is uniformly set to 64 for all the methods on all the datasets. The parameters of the methods compared are set as what were used by their authors. All the results of the baseline methods are either from their original papers or produced by running the codes from the authors with their default settings. For DeepWalk, the walk length and window size are set as 40 and 5, respectively. For node2vec, walk length and window size are set as 80 and 10, respectively. For LINE, we set the number of negative samples as 5 and the starting value of the learning rate as 0.025. For GraRep, we set the maximum matrix transition step as 5. VGAE and ARVGE utilize two-layer GCN with dimensions of hidden and output layers as 512 and 64, respectively. DGI adopts one-layer GCN with dimension of output layer as 64.

In our proposed OT-GNN, Graph Convolutional Network (GCN) [6], Graph Attention Network (GAT) [7]. and Graph AutoEncoder (GAE) [14] are employed as basic graph neural networks and the resulted instances are named as OT-GCN, OT-GAT and OT-GAE, respectively. The number of clusters, i.e., F , varies from 10 to 20. Similar to DGI, one-layer GCN with dimension of output layer as 64 is adopted in OT-GCN, while one-layer GAT with the number of heads and the dimension of each head as 4 and 16, respectively, is adopted in OT-GAT. Similar to GAE, two-layer GCN with dimensions of hidden and output layers as 512 and 64 is adopted. The hyper-parameter ε is set to 0.23 in OT-GAE.

2) Evaluation Metrics: For node classification task, accuracy (AC) is employed as the metric to evaluate the performance of all methods. For the node clustering task, normalized mutual information (NMI), which is more sensitive and fair for unbalanced clusters situation, is also adopted as metric. For link prediction task, two metrics, i.e., Area Under Curve (AUC) and Average Precision (AP), are adopted to quantify the performances.

D. Node Clustering

For node clustering, the state-of-the-art methods fall in two categories, i.e., generative models and representation models. For generative models, clustering results are directly obtained by inferring the latent variables without clustering the node embeddings. For representation models (both network embedding methods and graph neural networks), k-means algorithm is utilized to the obtained embedding of nodes to classify them into clusters. The results are shown in Table II.

It can be observed that the two instances of OT-GNN, i.e., OT-GCN and OT-GAT outperform all other unsupervised GNNs on all the networks, especially on four webpage networks. OT-GAT achieves the best performance on all 7 networks in terms of NMI and achieves the best performance on 6 network in term of accuracy (AC). OT-GAT is slightly lower than recently proposed TLSC [48], which is a generative model combing topology and node content, in term of accuracy. This may attributes to its assumption of inconsistency between topology and node attribute, which makes its superiority on large cluster more remarkable. Factually, the superiority in terms of NMI indicates our proposed OT-GNN can obtain better performance on difficult small cluster, since NMI is a fair metric, which is more sensitive on the small cluster.

E. Node Classification

For node classification, both LibSVM and LibLINEAR are employed to classify these nodes according to the obtained embedding. On citation network, i.e., Cora, Citeseer and Pubmed, we adopt the experimental settings in [49], where 20 nodes per class, 500 nodes and 1,000 nodes are employed for training, validation and performance evaluation, respectively. For the four medium webpage networks, including Cornell, Texas, Washington and Wsicsonsin, we adopt 20% labelled nodes for training, 10% labelled nodes for validation, and the other nodes for testing. For each network, we used 10-fold cross-validation, and the average performances are reported in Table III.

Both OT-GCN and OT-GAT, which are two instances of our proposed OT-GNN, outperform the state-of-the-art baselines on all the 7 networks. These gains are more significant on four webpages networks, i.e., Cornell, Texas, Washington and Wisconsin, since the community structures on them are more clear than network homophily property, which is the theory basis of GNNs. Besides, the performances of OT-GAT are better than those of OT-GCN, since OT-GAT takes expressive GAT [7], which jointly learns mapping function with parameter W and propagation weights with parameter b, as basic component. Note that, to the best of our knowledge, OT-GAT is the first successful unsupervised GNN, which takes GAT as basic component. The failure of adoption GAT in previous unsupervised GNN may attribute to that it is more likely to lead to overfitting from the learnable propagation weights in GAT than from the fixed propagation weights in GCN. Therefore, our proposed OT-GNN makes it possible to adopt more powerful basic GNNs without oversmoothing.

F. Link Prediction

For the link prediction task, the proposed OT-GAE is compared to 6 state-of-the-art baselines on citation networks. For each citation network, the edges are randomly divided into three groups. 85%, 5% and 10% of the edges are utilized in training, validation (hyper-parameters tuning) and performance testing, respectively. Experiments are repeated 10 times on 10 different random edge partitions, and the average performances are reported in Table IV.

As can be observed, the performance improvement of ARGA over GAE is limited, only about 1%, because adoption of the adversarial mechanism can't alleviate the oversmoothing problem although it aims at seeking robust representations. The proposed OT-GAE consistently and significantly outperforms the state-of-the-arts. It achieves about 2.5% performance gains (both in AUC and AP) in average, compared to its basis GAE, which just reconstructs topology without considering the oversmoothing issue. These gains are attributed to that our proposed OT-GNN can correctly and effectively regularize the GAE by reducing the oversmoothing.

G. Visualization

To provide an intuitive illustration, embedding visualizations via t-SNE [50] on Cora and Citeseer are given in Fig. 2. In additional to our proposed OT-GNN, another two unsupervised methods, node2vec [35] and DGI [17], and two semi-supervised ones, GCN [6] and GAT [7], are employed for comparison. To demonstrate the role of the learnable mapping function parameterized by W in GCN, GCN without W, is also adopted as an unsupervised approach. It can be observed that both semi-supervised approaches obtain clear cluster structure, which effectively prevents all the node embeddings from becoming too similar to be distinguished induced by the oversmoothing.

TABLE III
COMPARISON ON NODE CLASSIFICATION IN TERMS OF AC (%).

Packages	Methods	Cornell	Texas	Washington	Wisconsin	Cora	Citeseer	Pubmed
LibSVM	DeepWalk	38.97	49.18	55.30	49.24	82.57	52.52	78.79
	Node2Vec	35.90	50.27	47.47	46.56	79.98	61.63	80.30
	LINE	43.59	68.85	59.91	54.58	30.20	41.07	75.47
	GraRep	53.33	62.68	52.07	59.16	73.41	54.28	80.64
	TADW	64.10	67.76	59.45	64.50	69.83	68.70	85.37
	AANE	51.80	56.28	64.06	43.13	30.20	24.70	78.63
	TriDNR	37.95	48.09	47.01	40.46	43.27	54.47	79.07
	ASNE	48.21	57.92	54.38	59.54	49.00	44.74	78.37
	VGAE	45.13	55.00	54.38	53.82	81.05	65.97	83.42
	ARVGE	42.56	56.28	58.99	49.26	80.42	65.10	80.64
	DGI	42.56	56.28	47.47	45.42	80.21	70.07	74.57
	OT-GCN	63.02	69.07	65.66	67.43	82.73	70.65	86.77
	OT-GAT	**65.91**	**69.98**	**66.17**	**69.50**	**83.82**	**72.62**	**87.63**
LibLINEAR	DeepWalk	38.46	48.09	53.92	49.62	82.04	48.42	78.36
	Node2Vec	37.95	50.27	45.62	46.95	80.79	52.44	80.08
	LINE	44.10	53.39	56.22	54.96	50.25	40.56	74.92
	GraRep	53.33	59.40	51.15	60.31	79.83	53.61	80.37
	TADW	61.03	67.76	64.98	67.56	72.53	68.50	86.80
	AANE	41.54	53.01	61.75	38.93	27.03	22.24	77.99
	TriDNR	34.87	42.08	43.32	41.60	53.39	52.91	78.40
	ASNE	45.64	59.02	55.76	59.92	54.46	44.35	77.20
	VGAE	45.64	51.91	54.84	54.49	79.13	69.25	83.81
	ARVGE	41.54	59.02	60.37	56.11	81.24	66.71	80.59
	DGI	43.08	56.28	55.31	48.86	84.71	70.85	78.11
	OT-GCN	62.10	68.21	65.47	67.93	84.60	71.36	86.95
	OT-GAT	**63.62**	**69.71**	**66.55**	**69.96**	**85.52**	**72.55**	**88.56**

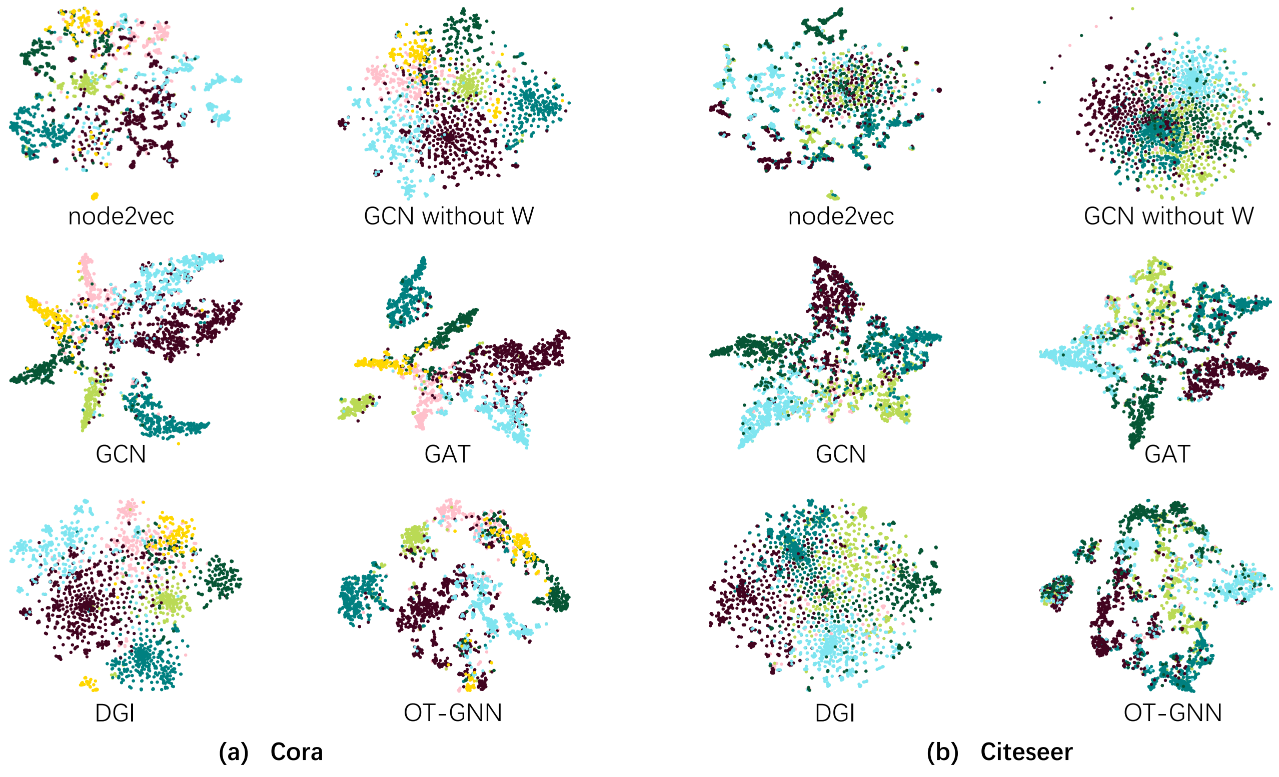

Fig. 2. Visualization of the embeddings on Cora and Citeseer. node2vec, DGI and our proposed OT-GNN are unsupervised approaches, while GCN and GAT are semi-supervised ones. GCN without learnable parameter W can also be considered as unsupervised method, since labelled nodes don't affect its results.

This is achieved by the mapping function parameterized by W trained from supervised information. This effect can also be verified by comparing semi-supervised GCN with GCN without W. However, existing unsupervised methods except our proposed OT-GNN lose this important structure due to the lack of supervision information. The cluster structure of

TABLE IV
LINK PREDICTION RESULTS IN TERM OF AUC AND AP.

Methods	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
DeepWalk	83.11	85.00	80.52	83.61	84.40	84.10
GAE	91.02	92.03	89.54	89.95	96.40	96.50
VGAE	91.41	92.61	90.82	92.02	94.42	94.72
ARGA	92.43	93.23	91.93	93.03	96.81	97.11
ARVGA	92.44	92.64	92.43	93.03	96.51	96.81
DGI	91.24	90.44	88.90	89.99	94.77	93.70
OT-GAE	**96.68**	**96.52**	**95.91**	**95.52**	**97.51**	**97.17**

our proposed OT-GNN is much clear, which indicates its effectiveness in preventing oversmoothing.

VI. CONCLUSIONS

To alleviate the oversmoothing issue in unsupervised graph neural networks (GNNs), a novel Optimal Transport based unsupervised GNN (OT-GNN) is proposed. It constrains the node embeddings to keep their community/cluster structures to prevent all the node embeddings from becoming too similar to be distinguished. By imposing the obtained node embeddings to be classified into clusters of equal size, OT-GCN performs interactive node embeddings and clustering. The constrained objective function of the unsupervised GNN is relaxed to an optimal transport problem, and a fast version of the Sinkhorm-Knopp algorithm is adopted to handle large networks. Besides of being utilized to train the GNNs for node clustering and embedding in an unsupervised manner, OT-GNN possesses the potential to be exploited to regularize other unsupervised GNNs, such as Graph AutoEncoder, for the link prediction task. Extensive experiments on node clustering, classification and link prediction demonstrate the effectiveness of our proposed OT-GNN in preventing from oversmoothing.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *TNNLS*, 2020.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.

[4] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017, pp. 1263–1272.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[8] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2020.

[9] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018, pp. 3538–3545.

[10] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019, pp. 6861–6871.

[11] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *ICML*, 2018, pp. 5449–5458.

[12] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR*, 2019.

[13] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.

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

[15] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *IJCAI*, 2018, pp. 2609–2615.

[16] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *WSDM*, 2019, pp. 393–401.

[17] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.

[18] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *WWW*, 2020, pp. 259–270.

[19] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Improving graph attention networks with large margin-based constraints," *NerIPS Workshop*, vol. abs/1910.11945, 2019.

[20] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.

[21] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[22] G. Li, M. Müller, A. K. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *ICCV*, 2019, pp. 9266–9275.

[23] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *ICLR*, 2020.

[24] Z. Meng, S. Liang, X. Zhang, R. McCreadie, and I. Ounis, "Jointly learning representations of nodes and attributes for attributed networks," *TOIS*, vol. 38, no. 2, pp. 16:1–16:32, 2020.

[25] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008, pp. 1096–1103.

[26] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *AAAI*, 2017, pp. 203–209.

[27] H. Sun, F. He, J. Huang, Y. Sun, Y. Li, C. Wang, L. He, Z. Sun, and X. Jia, "Network embedding for community detection in attributed networks," *TKDD*, vol. 14, no. 3, pp. 36:1–36:25, 2020.

[28] F. Sun, M. Qu, J. Hoffmann, C. Huang, and J. Tang, "vgraph: A generative model for joint community detection and node representation learning," in *NeurIPS*, 2019, pp. 512–522.

[29] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *TKDE*, vol. 31, no. 6, pp. 1051–1065, 2019.

[30] B. Rozemberczki, R. Davies, R. Sarkar, and C. A. Sutton, "GEMSEC: graph embedding with self clustering," in *ASONAM*, 2019, pp. 65–72.

[31] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *CIKM*, 2017, pp. 377–386.

[32] G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[33] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *NIPS*, 2013, pp. 2292–2300.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[35] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[36] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[37] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[38] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *IJCAI*, 2015, pp. 2111–2117.

[39] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *IJCAI*, 2016, pp. 1895–1901.

[40] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *SDM*, 2017, pp. 633–641.

[41] L. Liao, X. He, H. Zhang, and T. Chua, "Attributed social network embedding," *TKDE*, vol. 30, no. 12, pp. 2257–2270, 2018.

[42] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *PRE*, vol. 83, p. 016107, 2011.

[43] P. Li, L. Huang, C. Wang, and J. Lai, "Edmot: An edge enhancement approach for motif-aware community detection," in *SIGKDD*, 2019, pp. 479–487.

[44] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[45] R. Balasubramanyan and W. W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links," in *SDM*, 2011, pp. 450–461.

[46] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: a discriminative approach," in *SIGKDD*, 2009, pp. 927–936.

[47] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large attribute networks," in *AAAI*, 2016, pp. 265–271.

[48] G. Zhang, D. Jin, J. Gao, P. Jiao, F. Fogelman-Soulié, and X. Huang, "Finding communities with hierarchical semantics by distinguishing general and specialized topics," in *IJCAI*, 2018, pp. 3648–3654.

[49] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.

[50] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, pp. 2579–2605, 2008.