Adversarial Graph Disentanglement With Component-Specific Aggregation

Shuai Zheng, Zhenfeng Zhu, Zhizhe Liu, Jian Cheng, and Yao Zhao, Fellow, IEEE

Abstract—A real-world graph has a complex topological structure, which is often formed by the interaction of different latent factors. Disentanglement of these latent factors can effectively improve the robustness and expressiveness of the node representation of a graph. However, most existing methods lack consideration of the intrinsic differences in relations between nodes caused by factor entanglement. In this article, we propose an adversarial disentangled graph convolutional network (ADGCN) for disentangled graph representation learning. To begin with, we point out two aspects of graph disentanglement that need to be considered, i.e., microdisentanglement and macrodisentanglement. For them, a component-specific aggregation approach is proposed to achieve microdisentanglement by inferring latent components that caused the links between nodes. On the basis of microdisentanglement, we further propose a macrodisentanglement adversarial regularizer to improve the separability among component distributions, thus restricting the interdependence among components. In addition, to reveal the topological graph structure, a diversity-preserving node sampling approach is proposed, by which the graph structure can be progressively refined in a way of local structure awareness. The experimental results on various real-world graph data verify that our ADGCN obtains more favorable performance over currently available alternatives.

Impact Statement—Graph representation learning aims to represent discrete irregular graph data as continuous vectors. However, most of the existing methods stay on the general utilization of the external topological structure to learn representation, ignoring the analysis of the internal causes of the graph. To tackle these concerns, disentangled graph representation learning has been seen as an effective solution. In this paper, we specify two aspects of graph disentanglement need to be considered and propose an adversarial disentangled graph representation learning method. Our extensive experiments show that the proposed method not only achieves comparable performance to state-of-the-art methods for various node classification tasks but also demonstrates some significant graph disentanglement properties. Owning to the effective graph disentanglement, our model also shows better defensive ability than previous graph neural networks. As a whole, the idea presented in this work provides a new way towards graph disentanglement.

Index Terms—Adversarial learning, graph disentanglement, graph neural networks, graph representation learning.

I. INTRODUCTION

The graph is a general description of data and their relations, which can model a variety of scenarios, from biomedical networks to citation networks, or other things defined by relations. There are a variety of graph-based applications in the machine learning field, such as link prediction [1], node classification [2], [3], and recommendation [4], [5]. As an important part of graph analysis, graph representation learning is the basis for graph data to be processed by conventional machine learning methods.

Most graph representation learning methods can be broadly classified into two categories: proximity-based methods and neural network-based methods. Proximity-based methods, such as DeepWalk [4], node2vec [5], and LINE [6], focus on extracting the patterns from the graph data. These methods tend to learn the node representation with various order proximity preserving by constructing the node paths through random walks or local neighborhoods. Besides, Qiu et al. [7] demonstrated that matrix factorization methods [8] are also equivalent to proximity-based approaches. While neural network-based methods usually apply graph convolution to aggregate neighborhood information [9], [10], [11], or use autoencoder to merge structure and node features [1], [2].

Significant progress in graph representation learning has been made recently, especially the emergence of graph convolutional networks (GCNs) [12]. Nevertheless, most of the existing methods learn node representation by treating the local neighborhood or node path as a whole, ignoring the subtle differences between node neighbors. Actually, such practices make the model prone to learning over-smoothed representations when a network goes deeper [13]. In general, a real-world graph is relatively complex and formed by the interaction of a variety of latent factors. In other words, links between a node and its neighbors may be caused by different components, which is why nuances are presented between node neighbors. However, these nuances are often overlooked by existing methods, limiting the expressiveness of learned node representations. Instead, we would like to consider the distinction of node neighbors and disentangle the node representation toward discovering those distinct latent
components with richer information, i.e., graph disentanglement. As an example in Fig. 1, unlike traditional undifferentiated aggregation of neighboring nodes by an aggregation function, graph disentanglement aims to learn the node representation that consists of some distinct latent components with each containing a certain semantic. Thus, how to disentangle the node representation into multiple components with distinct latent semantics is of great significance for graph representation learning, which will also be helpful for improving the robustness and interpretability of graph neural networks.

In the field of image analysis and understanding, disentangled representation learning has been successfully applied to a variety of tasks, such as foreground separation and visual reasoning [14]. It aims to separate the distinct, informative factors of variations in the data and learn component representations corresponding to the factors accordingly [15]. However, less focus has been put on disentangled representation learning of graph [16], [17], [18]. According to the abovementioned notion of disentanglement [15], there are two aspects to consider to implement node-level graph disentanglement. 1) For a given graph, we need to infer the informative component representation of each node in the graph corresponding to the different factors. 2) Besides, we need to ensure the component subspaces are as separable as possible. The former can be understood as representation-level microdisentanglement and the latter can be understood as subspace-level macrodisentanglement.

Challenges of Graph Disentanglement: Despite the essential explorations that have been made by [16] and [17], there are still some key issues that should be addressed.

1) The existing methods attempt to perform microdisentanglement through neighborhood routing mechanisms. Still, this practice lacks consideration that the relationship between nodes may be formed by a combination of multiple factors.

2) In addition to the microdisentanglement, the components should also be macroseparable at the component distribution level, while DisenGCN neglects to restrict interdependence among component distributions.

3) The disentangled representations should be inversely further reveal the intrinsic graph structure based on the given initial adjacency matrix.

To address the abovementioned key issues, we propose a novel adversarial disentangled graph convolutional network (ADGCN), for disentangled graph representation learning. Specifically, different from DisenGCN and IPGDN, to address the microdisentanglement of a combination of multiple factors, we consider the intracomponent correlation and intercomponent separability simultaneously, and achieve component-specific aggregation for the microdisentanglement of the node. On the basis of the microdisentanglement performed by the component-specific aggregation, conditional adversarial learning is further deployed to restrict the interdependence among components for the macrodisentanglement. To the best of authors’ knowledge, this is the first attempt to introduce adversarial learning for graph disentanglement. The learned disentangled representation is also leveraged to explore the underlying structure of the graph in a progressive manner. We propose a diversity-preserving node sampling method to guide the local structure-aware graph refinement and spread to the overall graph further. In summary, the main contributions can be highlighted as follows.

1) To achieve microdisentanglement for each node, a component-specific aggregation with a dynamic assignment mechanism is proposed, which ensures the compactness of the component space and allocates the neighbors of the target node to the corresponding component space to realize microdisentanglement.

2) For implementing macrodisentanglement, we propose a conditional adversarial regularizer to improve the separability among component distributions, thus learning the more distinct components.

3) To refine the global graph structure in a progressive way, we propose a diversity-preserving node sampling approach to perform local structure-aware refinement. Hence, the latent global topological graph structure can be well revealed to benefit graph disentanglement.

II. RELATED WORK

A. Graph Neural Networks for Graph Disentanglement

Currently, several studies on disentangled representation learning of graph data have emerged. DisenGCN [16] first studied the problem of graph disentanglement at the node level. On this basis, IPGDN [17] enhanced the independence constraint of graph disentanglement by using HSIC. Besides, FactorGCN [18] proposed graph-level disentangling for graph classification. SGCN [19] combines [18] and [16] in a unified framework. Recently, Guo et al. [20] proposed to leverage neighborhood routing locally and message passing globally to improve the graph disentanglement. While Zhao et al. [21] considered hierarchical edge disentanglement to assist node classification tasks. Meanwhile, DSSL [22] and DGCL [23] considered graph disentanglement in self-supervised learning and contrastive learning settings, respectively. In this article, according to the concept of disentanglement [15], we give two aspects that need to be considered for node-level graph disentanglement. Furthermore, the proposed ADGCN is the first work that introduces adversarial learning for graph disentanglement.
It not only considers the disentanglement of individual nodes but also imposes constraints at the component distribution level.

**B. Adversarial Learning on Graphs**

As a powerful generative model, generative adversarial networks have been proposed by Goodfellow et al. [24]. Then, the authors in [25] and [26] successfully applied adversarial learning to handle graph data for graph generation tasks. Besides, the success of [27], [28] shows that GAN and adversarial learning also have an excellent performance for data understanding. Benefiting from the great ability of distribution fitting, adversarial learning has also been adopted to the domain adaption field [29] as a critical part. Hence, some studies have also applied adversarial learning to the field of graph representation learning. Wang et al. [30] generated node pairs to approximate the linkage distribution, enhancing the structure-preserving capability of the node representation. And Lee et al. [31] also proposed to sample subgraph to learn node embedding based on adversarial learning. Another common practice, as in [32] and [33], is to utilize adversarial learning as a regularizer to constrain the representation distribution of nodes, thereby improving the generalization ability of the representations. In addition, DGI [34] constructed degraded graphs for adversarial learning with the original graph. Lu et al. [35] maintained consistency of node representation between multiple graphs through adversarial learning. Unlike them, we apply an adversarial regularizer to control the separability between component distributions for graph disentanglement.

**III. METHODOLOGY**

**A. Problem Formulation and Overview of Framework**

To begin with some definitions, an undirected graph is given as $G = (V, E)$, where $V$ consists of the set of nodes with $|V| = n$, and $E$ is the set of edges among nodes. To represent the link strength (similarity) among graph nodes, we use $A \in \mathbb{R}^{n \times n}$ to denote the adjacency matrix, and $D = \text{diag}([d_u]_{u=1,\ldots,n})$ to denote the corresponding degree matrix with $d_u = \sum_{i=1}^{n} A_{u,i}$. Meanwhile, let $X = [x_1, \ldots, x_n] \in \mathbb{R}^{n \times f}$ denote the node feature matrix with $x_u \in \mathbb{R}^f$, $u = 1, \ldots, n$, representing the raw feature of the $u$th node, and $H = [h_1, \ldots, h_n] \in \mathbb{R}^{n \times d}$ the node representation matrix. In addition, $\mathcal{N}(u)$ is defined as a node sub-set containing neighbors of node $u$ and $\hat{\mathcal{N}}(u) = \mathcal{N}(u) \cup \{u\}$.

1) Microdisentanglement: Assuming that $G$ is formed by $K$ latent factors corresponding to $K$ component spaces. For a specific node $u$, we want to decompose its latent representation $h_u$ into $K$ component representations $\{h_u^1, h_u^2, \ldots, h_u^K\}$, where $h_u^k \in \mathbb{R}^{\hat{\mathcal{N}}(u)}$ corresponds to the $k$th factor.

2) Macrodisentanglement: On the basis of microdisentanglement, for all nodes $V$ in the given graph, we still expect the component representations lying on different component spaces to be far away from each other, i.e., the difference between any two component distributions should be evident.

3) Graph Refinement: After the disentanglement mentioned above, the disentangled representations enable a more fine-grained exploration of node associations, which, allows for a further revelation of the intrinsic structure of the graph based on the given initial adjacency matrix.

4) Overview of Framework: The overall illustration of the proposed ADGCN model is shown in Fig. 2. For a specific node $u$ and its neighbors in the given graph, the initial embeddings $\{c^k_u\}_{k=1}^K$ are first obtained from $x_u$ through projection matrices $W = \{W_1, W_2, \ldots, W_K\}$, where $i \in \hat{\mathcal{N}}(u)$.

To perform the microdisentanglement, $\{c^k_u\}_{i \in \hat{\mathcal{N}}(u)}$ are aggregated into $h^k_u$ in the $k$th component space through the dynamic assignment mechanism and component-specific aggregation, then we have $h^k_u = \text{concat}(h^1_u, h^2_u, \ldots, h^K_u)$.

For a graph with the hard link, we have $A_{i,j} = 1$ if there exists an edge $(i, j) \in E$, otherwise $A_{i,j} = 0$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
To perform the macrodisentanglement, $e_u^k$ and $h_u^k$ are input into the discriminator $D$, where $e_u^k$ and $h_u^k$ are regarded as a “fake” sample and a “real” sample, respectively.

For graph refinement, if node $u$ would be selected by diversity-preserving node sampling, $h_u$ is also used to update $A^{(t-1)}$ to $A^{(t)}$.

### B. Component-Specific Aggregation for Microdisentanglement

Based on an important assumption, i.e., the links between nodes are formed by the interaction of different latent factors, our ADGCN learns disentangled components by aggregating the different neighborhood information of the node to achieve the microdisentanglement.

Intuitively, if node $u$ and its neighbor node $v$ are similar in the $k$th component space, the link between the two nodes is likely caused by the factor $k$. Furthermore, for a subset $N_k(u) \subseteq N(u)$, if all nodes in $N_k(u)$ are connected to node $u$ due to factor $k$, these nodes should also be similar and form a cluster with $u$ in the $k$th component space. It is worth noting that the abovementioned two points correspond to the first-order and the second-order proximities in the topological structure formed by the factor $k$, respectively [6]. Based on these two points, DisenGCN [16] proposed a neighborhood routing mechanism, which was also followed by [17]. However, the neighborhood routing mechanism merely considers the restriction on the separability of intercomponents. It also constrains that each neighbor should belong to only one component space, ignoring multirelations among nodes.

Motivated by the observations mentioned above, we propose a dynamic assignment mechanism as shown in Fig. 3 to achieve component-specific aggregation for microdisentanglement. To characterize the interactive relationship between node $u$ and its neighbor node $v \in N(u)$ in each component space, we define the hubness score $p_u^k v$ and the authority score $q_{u \rightarrow v}$ of node $u$ and $v$ in the $k$th component space

$$p_u^k v = \frac{\exp(s_{uv}^k)}{\sum_{k=1}^{K} \exp(s_{uv}^k)}, \quad q_{u \rightarrow v} = \frac{\exp(s_{uv}^k)}{\sum_{v \in N(u)} \exp(s_{uv}^k)}$$

(1)

where $s_{uv}^k = \cos(h_u^k, c_v^k)$ denotes the similarity between the target node $u$ and its neighbor $v$ in the $k$th component space.

In fact, as a bidirectional information interaction between node $u$ and its neighbor $v$, the hubness score $p_u^k v$ means the probability that $v$ is connected to $u$ by the factor $k$, and the authority score $q_{u \rightarrow v}$ refers to the aggregation importance of $v$ for $u$ in the $k$th component space. In particular, for a specific neighbor node $v$, the higher the hubness score $p_u^k v$, the more effective information between $u$ and $v$ has about factor $k$, therefore, $v$ will tend to be allocated to the $k$th component space for the aggregation of $u$. Meanwhile, a high authority score $q_{u \rightarrow v}$ means a high similarity between $u$ and $v$ in the $k$th component space.

Based on $p_u^k v$ and $q_{u \rightarrow v}$, the proposed dynamic assignment mechanism can be implemented in an iterative way to obtain the aggregated representation $h_u^k$ by

$$h_u^k = \tilde{h}_u^k + \alpha \sum_{v \in N(u)} q_{u \rightarrow v} c_v + \beta \sum_{v \in N(u)} p_u^k v c_v$$

(2)

where $\tilde{h}_u^k$ denotes the aggregated $h_u^k$ in previous iteration, $\alpha$ and $\beta$ are tradeoff coefficients. Notably, instead of treating them as hyperparameters, we set $\alpha$ and $\beta$ as two trainable parameters and scale their range to (0,1). The intracomponent part with $q_{u \rightarrow v}$ ensures that $h_u^k$ tends to be aggregated into the densest area in $k$th component space, and the intercomponent part with $p_u^k v$ constrains the separability between $\{h_u^k\}_{k=1}^K$. Meanwhile, different from the restricted single-relation constraint in [16] and [17], the coaggregation of intracomponent and intercomponent also allows for the existence of multiple relationships between nodes.

To make the proposed ADGCN be capable of inductive learning, we take a similar way as in [11] by randomly sampling the neighboring nodes of $u$ to get $N(u)$ with a fixed size. Notably, average sampling is used for unweighted graphs, and probability sampling based on edge weights is applied for weighted graphs. The pseudocode of the component-specific node representation aggregation is summarized in Algorithm 1.

### C. Adversarial Learning for Macrodisentanglement

In essence, the goal of disentangled representation learning is to make a specific component be only sensitive to the variation in the corresponding factor, while not affected by the one in other factors [15], [36]. In addition to the microdisentanglement mentioned above, there should exist only weak correlation, or strong separability in other words, among component distributions corresponding to different factors.

2To avoid numerical instability caused by inconsistent norms between different component spaces, $L_2$-normalization is adopted for $h_u^k$ and $c_v^k$ to facilitate the calculation of similarity.
GANs have demonstrated the strong ability of distribution fitting in various fields, such as domain adaption and neural language processing [29], [37]. Adversarial learning provides an efficient and trainable way to constrain the distances between distributions. However, adversarial learning is nontrivial to directly apply to graph disentanglement. Therefore, we propose a conditional adversarial regularizer for macrodisentanglement at the distribution level by explicitly imposing constraints on the dependence among components.

To enable conditional adversarial learning of ADGCN, we introduce a discriminator D with two branches D_{src} and D_{cls}. Concretely, the purpose of the discrimination branch $D_{src}$ is to distinguish whether the input representation is a disentangled component representation. Relying on $D_{src}$, we have the adversarial loss term as

$$\min_{W} \max_{D} \mathcal{L}_{adv} = \mathbb{E}_{h}[\log D_{src}(h)] + \mathbb{E}_{c}[\log(1 - D_{src}(c))].$$

(3)

Here, $h_u$ and $c_u$ are, respectively, taken as the “real” sample and the “fake” sample for the downstream task. In essence, D needs to maximize $\mathcal{L}_{adv}$ to distinguish between $\{c_u\}_{k=1}^{K}$ and $\{h_u\}_{k=1}^{K}$, while W will try to confuse D by minimizing $\mathcal{L}_{adv}$, thus enforcing $\{c_u\}_{k=1}^{K}$ to obey the distributions of $\{h_u\}_{k=1}^{K}$, where W can be regarded as the generator G in GANs. Meanwhile, due to the microdisentanglement achieved by the dynamic assignment mechanism, $\{h_u\}_{k=1}^{K}$ already has a certain intercomponent separability. Therefore, through adversarial optimization, W can learn to project $X$ to more distinguishable component subspaces under the guidance of minimizing $\mathcal{L}_{adv}$. In addition, it can also accelerate the convergence of Algorithm 1.

Another branch $D_{cls}$ as an auxiliary classification branch is used to identify, which component subspace the input representation comes from. It is added to incorporate the factor information into the discriminator, which allows a single discriminator to control multiple factors. Relying on $D_{cls}$, we have the component classification term as

$$\min_{W} \sum_{k=1}^{K} \mathbb{E}_{h}[-\log D_{cls}(k|h)] + \mathbb{E}_{h,c}[-\log(D_{cls}(k|c))].$$

(4)

By using auxiliary classifier to minimize $\mathcal{L}_{cls}$, D can learn to classify $h_u$ and $c_u$ to the corresponding k-th component space. W also tries to minimize $\mathcal{L}_{cls}$. Since $h_u$ is derived from $\{c_u\}_{i \in N(u)}$, it will be beneficial of boosting the separability among component spaces and assisting D to minimize $\mathcal{L}_{cls}$.

Due to the microdisentanglement achieved by the dynamic assignment mechanism, $\{h_u\}_{k=1}^{K}$ already has a certain intercomponent separability. Furthermore, with the support of the proposed conditional adversarial learning, W can learn to seek more distinguishable component subspaces and also accelerate the convergence of dynamic assignment through adversarial optimization. Besides, the auxiliary classification branch $D_{cls}$ provides a way to make it possible to explicitly constrain the separability between components via minimizing $\mathcal{L}_{cls}$. The discriminator with two branches for the proposed conditional adversarial learning yields a strong connection to the methods for adversarial domain adaptation [38] with two-branches discriminators. Similarly to the one in [38], $D_{src}$ in the ADGCN is also used for domain-level alignment. The difference is that $D_{cls}$ in [38] is directly related to the downstream tasks, whereas the $D_{cls}$ in the proposed conditional adversarial learning is oriented toward component classification that is not explicitly associated with the downstream task. In other words, the proposed conditional adversarial learning focuses on distribution alignment and component classification of representations for disentanglement instead of downstream tasks. As shown in Fig. 4, the distributions of $\{c_u\}_{k=1}^{K}$ will become mutually independent as training progresses, i.e., $c_u$ will obey the distribution in the specific k-th component space and be independent of each other.

In this way, we can capture the multiple disentangled component distributions with weak correlations to each other to achieve macrodisentanglement.
Algorithm 2: Local Structure-Aware Graph Refinement in the $t$-th Epoch.

**Input:** $G = (V, E)$, $H^{(t-1)} = [h_1^{(t-1)}, \ldots, h_n^{(t-1)}]$, $A^{(0)}$;  
**Output:** $A^{(t)}$; **Hyperparameters:** $m$, $K$, $\gamma$

1. Initialization: $J_s = \phi$
2. for $u \in V$ do /* Degree-based Node Sampling */
   3. $d_u \leftarrow \sum_{v \in V} [A^{(0)}]_{uv}$
   4. $p_s(u) \leftarrow \frac{d_u}{\sum_{i=1}^{N_v} d_i}$
   5. end for
6. for $1$ to $m$ do
5. Take a sampling on node $u_s$ from $V$ according to $p_s(u_s)$
7. while $v \in \tilde{N}(u_s)$ do
8. Calculate $\Delta_{u_s,v} = \|h_{u_s}^{(t-1)} - h_v^{(t-1)}\|_2$
9. $p_s(v) \leftarrow p_s(v) \cdot f(\Delta_{u_s,v})$
11. end while
12. $J_s \leftarrow J_s \cup \{u_s\}$
13. end for
14. while sample node $u_s \in J_s$ do /* Graph Refinement */
15. for $k = 1$ to $K$ do
16. Select $k$ nearest neighbors of $h_{u_s}^{(t-1),k}$ as $N'(u_s)$
17. $\tilde{A}_{u_s,N'(u_s)}^{(t-1),k} \leftarrow \cos(h_{u_s}^{(t-1),k}, h_{N'(u_s)}^{(t-1),k})$
18. end for
19. end while
20. $A^{(t)} \leftarrow \text{maxpooling}(\tilde{A}_{u,N(u)}^{(t-1),1}, \tilde{A}_{u,N(u)}^{(t-1),2}, \ldots, \tilde{A}_{u,N(u)}^{(t-1),K})$
21. Obtain the refined $A^{(t)}$ by (5).

D. Local Structure-Aware Graph Refinement Based on Diversity-Preserving Node Sampling

For real-world graph data, the complex relationship between nodes is often represented as a hard binary link (i.e., 0 or 1). Obviously, it is a discrete and simplified form of continuous relationship between nodes, which seriously limits the expressibility of the learned node representation. On the other hand, the node representation obtained in the embedding space can in turn be used to reveal the intrinsic relationship between nodes. To better characterize the node relationships and further facilitate the learning of node representation, an intuitive way is to refine the originally given graph structure with the embedded node representations [39]. However, such global refinement of the relationships among all nodes without distinction will inevitably lead to some noisy edges, which may further confuse the training of the node representation learning model. In addition, it also has scalability problems on large graphs. To address these issues, we propose a local structure-aware graph refinement as shown in Fig. 5 to progressively reveal the latent relationships of nodes, thus achieving efficient and robust graph refinement.

The key to achieving local structure-aware graph refinement is how to obtain a diversified core subset $J_s \subset V$ for local refinement, so as to maintain the global coverage of the graph as much as possible. Determinant point process (DPP) [40] is a classical algorithm for diversity-preserving sample selection. However, the high computational cost of DPP is somewhat the opposite of what we intended. Hence, for local structure-aware graph refinement, we propose a degree-based node sampling method with good diversity preservation to serve as an efficient approximation of DPP sampling on graph data.

Considering the characteristics of the graph structure, the normalized degree centrality indicating the importance of a node is used as the sampling probability $p_s(\cdot)$ for node selection, that is to say, we have $p_s(u) = \frac{d_u}{\sum_{i=1}^{N_v} d_i}$ for $u$th node. Let $J_s \subset V$, $|J_s| = m$, denote the core set to be selected for local aware graph refinement. To realize the diversity of the selected core set $J_s$, we propose a sequential node selection method by means of soft-deflation. Specifically, for a node $u_s \in J_s$ that has been selected, a soft-deflation is applied to its every neighbor $v \in \tilde{N}(u_s)$, i.e., we will have $p_s(v) \cdot f(\Delta_{u_s,v}) \to p_s(v)$, where $f(\cdot) \in [0, 1]$ is the soft-deflation function and $\Delta_{u_s,v}$ denotes the distance between $u_s$ and its neighbor $v$. Here, we follow the definition of $f(\cdot)$ in [41] and consider $f(\Delta) = \sin^2(\Delta/\tau)$, where $\tau$ is the scaling factor to keep $\Delta/\tau$ ranging in $[0, \pi/2]$. Through the soft-deflation in sequential node selection, we can ensure that, for each node in the associated neighborhood node set $\Psi(J_s) = \bigcup_{u_s \in J_s} \tilde{N}(u_s)$, it will be further selected into the core set with less probability in the subsequent node selection. In this way, it means that the finally obtained core set $J_s$ can achieve the maximum coverage of the graph to a certain extent.

Then, with the assistance of the diversity-preserving core subset $J_s$, the local structure-aware graph refinement can be carried out as follows:

$$A_{u_s,N(u_s)}^{(t)} = (1 - \gamma)A_{u_s,N(u_s)}^{(0)} + \gamma \tilde{A}_{u_s,N(u_s)}^{(t-1)}$$  (5)
where \( u_k \in J_s \), \( A^{(0)} \) represents the initial adjacency matrix and \( A^{(t-1)} \) denotes the adjacency refining matrix in the \( t \)th epoch, and \( \gamma \) is a tradeoff coefficient.

The link relationship between two nodes may come from a certain component space. To capture the underlying topological structure of the graph, we first obtain \( K \) component adjacency matrices \( \{A^{(t-1),k}\}_{k=1}^{K} \) based on the associated node representations \( \{h_u^{(t-1),k}\}_{u \in V} \) in \( t \)th epoch, and then an element-wise max pooling to form a component sensitive adjacency matrix \( \tilde{A}^{(t-1)} = \text{maxpooling}(A^{(t-1),1}, A^{(t-1),2}, \ldots, A^{(t-1),K}) \). The adopted max pooling plays the role of integrating the specific structures of multiple latent components into a unified topological one, boosting the link characteristics of nodes.

Actually, it can be found from (5) that the local aware refinement guided by a core set \( J_s \) can spread to the overall graph, thus achieving global refinement of the graph structure. Meanwhile, it cannot only prevent the introduction of noisy edges in the refinement of the graph but also help to learn more robust disentangled representations progressively with the progress of training. Due to the frequent graph refinement, it may lead to an unstable training process. Therefore, in our practice, we perform graph refinement only in epochs when the model achieves better performance than before in the validation set. The detail about local aware graph refinement with diversity-preserving node sampling is illustrated in Algorithm 2.

IV. OPTIMIZATION

As we mainly focus on the task of node classification in this article, the last layer of ADGCN is designed as a fully-connected layer to output the predicted class label of nodes. Besides, rectified linear unit (ReLU) is used as activation to obtain the component representations \( \{c_k^{(t)}\}_{k=1}^{K} \) after projections. Specifically, the overall loss of ADGCN is

\[
L = L_t + \lambda (L_{\text{adv}} + \eta L_{\text{cls}}) \quad L_D = -L_{\text{adv}} + \eta L_{\text{cls}}
\]

(6)

where \( L_t \) denotes the task-aware loss, \( \lambda \) and \( \eta \) are hyperparameters of the adversarial regularizer. For the semisupervised node classification task, \( L_t \) is the cross-entropy loss. The same as [16], we use the multilabel soft margin loss as \( L_t \) for the multilabel node classification task

\[
L_t = -\frac{1}{C} \sum_{c=1}^{C} [y_u(c) \log(g(\tilde{y}_u(c))]
\]

\[
+ (1 - y_u(c)) \log(g(-\tilde{y}_u(c))]
\]

(7)

where \( g(\cdot) \) denotes the sigmoid function, \( y_u(c) \) and \( \tilde{y}_u(c) \) denote the true label and the predicted label of node \( u \), respectively. We optimize the proposed model with Adam optimizer [42].

Since \( h_k^{(t)} \) and \( c_k^{(t)} \) are dynamically correlated, the conventional GAN training method [24] is not suitable for such case since there are no generated samples and real disentangled samples for graph disentanglement as the “fake” and “real” samples in conventional GANs or the “source domain” and “target domain” in the field of domain adaption. Thus, an iterative adversarial training is applied to progressively improve the separability among component spaces via an “EM-like” procedure given by Algorithm 3. Here, the expression of “EM-like” refers to the iterative training process that is similar to EM. In fact, the process of aggregating \( \{c_i^{(t)}\}_{i=1}^{n} \) into \( \{h_i^{(t)}\}_{i=1}^{n} \) is equivalent to the E-step, where \( \{h_i^{(t)}\}_{i=1}^{n} \) can be seen as the achieved “expectation.” Then, the separability between components is expected to be maximized via adversarial learning, i.e., the “maximization” of the M-step.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Evaluation Setup

1) Datasets: We adopt eight real-world graph datasets and the statistics about them are summarized in Table I. On the evaluation of the semisupervised node classification and clustering, three citation networks Cora [44], Citeseer [43], and Pubmed [45] are used, whose links and labels represent citations and research fields, respectively. In addition, we also take three coauthorship and copurchase graphs, i.e., Coauthor CS [46], Amazon Computers [46], and Amazon Photo [46]. For multilabel node classification, Wikipedia [5] and Blogcatalog [48] are used. Wikipedia is a co-occurrence network whose labels

![Algorithm 3: EM-Like Procedure of the Proposed ADGCN.](image-url)
represent the part-of-speech tags. Blogdatalog is a social network, whose labels and links represent user interests and social relations, respectively. Since these two datasets do not contain node features, we use the rows of the adjacency matrix as node features.

2) Comparison Methods: We select several baselines for comparison:

1) general baselines, including multilayer perceptron (MLP), label propagation (LabelProp) [54], planetoid [56];
2) spectral convolution-based methods, including ChebNet [49], GCN [9], and ARMA [50];
3) spatial convolution-based methods, including GAT [10], MoNet [55], GraphSAGE [11], APPNP [51], SGC [52], and AP-GCN [53].

Besides, we also compare the most related state-of-the-art works that are focused on graph disentanglement in graph representation learning, i.e., DisenGCN [16], IPGDN [17], FactorGCN [18], SGCN [19], LGD-GCN [20], and DisGNN [21].

Due to the good performance on multilabel node classification, three probabilistic models DeepWalk [4], LINE [6], and node2vec [5] are also used for comparison.

3) Implementation Details: To detail the implementation, we define \( \Delta d = d / K \) as the dimension of node components, and \( K^{(l)} \) as the number of components in the \( l \)th layer, respectively. And \( \Delta T \) denotes the iterations of the dynamic assignment in the component-specific aggregation layer.

In semisupervised node classification, we fix \( K^{(l)} \) to \( K = 5 \) for the adopted six datasets and the dimension of \( h_{l} \) is set to \( K \cdot \Delta d \). For the multilabel node classification task, due to a large number of tags, we follow the setting in DisenGCN. Specifically, the number \( K \) of the intermediate layers is set to gradually decrease to learn the hierarchical disentangled representation, and we use skip-connection to preserve the representation of different levels. As for the output dimension of the first layer, it is set to \( K^{(1)} \cdot \Delta d = 128 \), where \( K^{(1)} \in \{4, 8, \ldots, 32\} \), \( K^{(l)} - K^{(l+1)} = \Delta K \in \{0, 2, 4\} \), and \( \Delta d \) is fixed to \( 128 / K^{(1)} \).

As hyperparameters, both \( K^{(1)} \) and \( \Delta K \) are searched through hyperopt [57]. For other hyperparameters, both \( \lambda \) and \( \eta \) are set to 1 empirically and we tune the remaining hyperparameters of ADGCN using hyperopt [57]. For each hyperparameter combination, we run 200 epochs and choose the best combination on the validation set. Then, we report the averaged results of 30 times of execution on each dataset.

B. Node Classification

1) Semisupervised Node Classification: In view of semisupervised node classification on Cora, Citeseer, and Pubmed datasets, we use the same fixed dataset split as GCN with each dataset containing only 20 labeled nodes for each class, 300 nodes for validation, and 1,000 nodes for testing. Meanwhile, we also conduct a random split as [51], and the random split has the same ratio as the fixed split. For Coauthor CS, Amazon Computers, and Amazon Photo, the random split is also adopted, that is, 20 labeled nodes per class are taken as the training set, 30 labeled nodes per class nodes as the validation set, and the rest nodes as the test set.

As shown in Tables II and III, the results with the best average performance are bolded and ADGCN-R denotes ADGCN with graph refinement. Due to the fact that FactorGCN is primarily devoted to graph-level disentanglement instead of node-level disentanglement, the performance of FactorGCN for node classification is unsatisfactory. Compared with the holistic approaches, such as GAT and GCN, DisenGCN, IPGDN, and ADGCN achieve a better performance. In particular, ADGCN outperforms over DisenGCN by about 0.8%, 1.0%, and 0.9% on Cora, Citeseer, and Pubmed datasets, respectively. Besides, ADGCN-R also achieves the best performance on three coauthor/purchase datasets. Meanwhile, the standard deviation of the results for ADGCN and ADGCN-R are smaller than that for DisenGCN, indicating that the performance of the proposed method is more stable as compared to DisenGCN. Furthermore, to verify the improvements achieved by ADGCN-R, we also perform a paired \( t \)-test between ADGCN and ADGCN-R with respect to the accuracy. The reported \( p \)-values in Tables II and III show that the results on different test datasets are statistically significant with \( p < 0.05 \) via the paired \( t \)-test.

2) Multilabel Node Classification: Following the experiment settings in [5], we report the performance of each method by increasing the number of nodes labeled for training from 10% to 80% of \( |V| \), and the other nodes are divided into two sets for verification and testing. Macro-F1 and Micro-F1 are used to evaluate the performance of each model. As can be observed from Fig. 6, ADGCN performs the best compared to all baselines including DisenGCN, especially on the Wikipedia dataset. Meanwhile, in terms of Macro-F1, graph disentanglement methods achieve a higher score than the others. It indicates that graph disentanglement can facilitate the understanding of the relation between nodes, so as to be more robust to graph data with class imbalance.

C. Clustering Analysis

For the node clustering task, we use the K-means [58] to cluster the learned representations as in [17]. The setting of the three citation datasets is the same as in the semisupervised node classification task with the fixed split. As shown in Table IV, the performances of graph disentanglement methods show a significant advantage over other compared methods. Moreover, our ADGCN achieves the best performance compared to DisenGCN and IPGDN. It means that the proposed macro- and microdisentanglement are greatly beneficial to exploring the inherent similarity between nodes. Meanwhile, the excellent performance of these graph disentanglement-based methods also demonstrates that graph disentanglement can indeed provide benefits in terms of graph clustering.

D. Ablation Study

We conduct an ablation study through the semisupervised node classification on Cora and Citeseer datasets. As shown in Fig. 7, the proposed macrodisentanglement, microdisentanglement, and diversity-preserving graph refinement are equally important to learning latent node representation. Here, the baseline model refers to the GCN with multicomponent. Specifically,
TABLE II
RESULTS ON CITATION DATASETS WITH BOTH FIXED AND RANDOM SPLITS FOR SEMISUPERVISED NODE CLASSIFICATION ACCURACY (%)

| Models       | Fixed  | Random | Fixed  | Random | Fixed  | Random |
|--------------|--------|--------|--------|--------|--------|--------|
|              | Cora   |        | CiteSeer |        | PubMed |        |
| Spectral-based |       |        |         |        |        |        |
| ChebNet [49]  | 80.5 ± 1.1 | 76.8 ± 2.5 | 69.6 ± 1.4 | 67.5 ± 2.0 | 78.1 ± 0.6 | 75.3 ± 2.5 |
| GCN [9]       | 81.3 ± 0.8 | 79.1 ± 1.8 | 71.1 ± 0.7 | 68.2 ± 1.6 | 78.8 ± 0.6 | 77.1 ± 2.7 |
| ARMA [50]     | 83.4 ± 0.7 | 79.7 ± 2.1 | 72.5 ± 0.4 | 68.9 ± 2.2 | 78.9 ± 0.3 | 78.4 ± 2.0 |
| APPNP [51]    | 83.3 ± 0.5 | **81.9 ± 1.4** | 71.8 ± 0.4 | 69.8 ± 1.7 | 80.1 ± 0.2 | 79.5 ± 2.2 |
| Spatial-based  |       |        |         |        |        |        |
| SGC [52]      | 81.7 ± 0.1 | 80.4 ± 1.8 | 71.3 ± 0.2 | 68.7 ± 2.1 | 78.9 ± 0.1 | 76.8 ± 2.6 |
| GAT [10]      | 83.1 ± 0.4 | 80.8 ± 1.6 | 70.8 ± 0.5 | 69.8 ± 1.7 | 79.1 ± 0.4 | 77.8 ± 2.1 |
| AP-GEN [53]   | 82.7 ± 0.3 | 79.8 ± 3.2 | 73.3 ± 0.6 | 69.2 ± 3.0 | 80.1 ± 0.7 | 79.2 ± 1.6 |
| Diverse-based  |       |        |         |        |        |        |
| DiversGCN [16]| 83.1 ± 1.1 | 80.3 ± 2.0 | 74.6 ± 0.8 | 70.1 ± 1.8 | 79.7 ± 0.7 | 78.7 ± 1.3 |
| iPGEN [17]    | 84.1    |        | 74.0    |        | 81.2    |        |
| FactorGCN [18]| 82.6 ± 1.0 | 78.1 ± 1.8 | 72.2 ± 0.9 | 69.1 ± 1.0 | 79.4 ± 1.1 | 77.8 ± 2.1 |
| SGCN [19]     | 82.2 ± 0.8 | 79.9 ± 1.5 | 74.6 ± 0.9 | 69.5 ± 1.1 | 80.3 ± 1.5 | 79.3 ± 0.7 |
| LGGC-GEN [20] | 84.2 ± 0.4 | 79.6 ± 2.1 | 74.3 ± 0.3 | 70.3 ± 1.6 | 80.3 ± 1.5 | 80.9 ± 0.4 |
| DisGNN [21]   | 82.8 ± 0.9 | 80.0 ± 1.9 | 73.4 ± 0.3 | 68.5 ± 1.7 | 80.3 ± 0.2 | 79.1 ± 1.1 |
| ADGCN-0 (Ours)| 84.3 ± 0.6 | 80.8 ± 1.7 | **74.3 ± 0.7** | 70.4 ± 1.5 | **81.2 ± 0.5** | 79.2 ± 1.7 |
| ADGCN-R (Ours)| **84.5 ± 0.7** | 81.0 ± 2.0 | **74.5 ± 0.8** | **71.0 ± 1.4** | **80.4 ± 0.4** | **79.9 ± 1.3** |

Boldface letters mark the best result, while underlined letters denote the second-best result.

![Fig. 6. Macro-F1 and Micro-F1 scores on the multilabel classification tasks. (a) Macro-F1(%), Wikipedia. (b) Micro-F1(%), Wikipedia. (c) Macro-F1(%), Blogdatalog. (d) Micro-F1(%), Blogdatalog.](image)

Fig. 6. Macro-F1 and Micro-F1 scores on the multilabel classification tasks. (a) Macro-F1(%), Wikipedia. (b) Micro-F1(%), Wikipedia. (c) Macro-F1(%), Blogdatalog. (d) Micro-F1(%), Blogdatalog.

TABLE III
RESULTS ON THE OTHER THREE DATASETS WITH RANDOM SPLITS IN TERMS OF SEMISUPERVISED NODE CLASSIFICATION ACCURACY (%)

| Models       | CS  | Computer       | Photo |
|--------------|-----|----------------|-------|
| MLP          | 88.3 ± 0.7 | 44.9 ± 5.8 | 69.6 ± 3.8 |
| LabelProp [54]| 73.6 ± 3.9 | 70.8 ± 8.1 | 72.6 ± 11.1 |
| LabelProp NL [54]| 76.7 ± 1.4 | 75.0 ± 2.9 | 83.9 ± 2.7 |
| GCN [9]      | 91.1 ± 0.5 | 82.6 ± 2.4 | 91.2 ± 1.2 |
| ARMA [50]    | 92.4 ± 0.3 | 80.7 ± 0.3 | 89.4 ± 0.3 |
| MoNet [55]   | 90.5 ± 0.6 | 78.9 ± 1.9 | 85.7 ± 20.3 |
| SGCN-mean [11]| 91.3 ± 2.8 | 82.4 ± 1.8 | 91.4 ± 1.3 |
| SGCN-maxpool [11]| 85.0 ± 1.1 | 90.4 ± 1.3 | |
| SGCN-meanpool [11]| 89.6 ± 0.9 | 79.9 ± 2.3 | 90.7 ± 1.6 |
| APPNP [51]   | 92.4 ± 0.2 | 80.1 ± 0.3 | 89.3 ± 0.2 |
| AP-GCN [53]  | 92.8 ± 0.2 | 82.1 ± 0.3 | 91.3 ± 0.9 |
| FactorGCN [18]| 90.9 ± 2.3 | 82.1 ± 1.5 | 88.2 ± 3.1 |
| SGCN [19]    | 91.1 ± 1.7 | 84.1 ± 1.1 | 90.5 ± 2.2 |
| LGGC-GEN [20]| 92.7 ± 0.3 | 83.8 ± 1.4 | 91.8 ± 0.3 |
| DisGNN [21]  | OOM  | OOM            | OOM   |
| ADGCN-0 (Ours)| 92.9 ± 0.3 | 84.7 ± 1.1 | 91.7 ± 0.4 |
| ADGCN-R (Ours)| 93.7 ± 0.4 | 85.2 ± 0.7 | 92.1 ± 0.3 |

P-value: $1.74 \times 10^{-5}$, $2.10 \times 10^{-5}$, $2.71 \times 10^{-10}$

The boldface text is the best result, while underlined text denotes the second-best result.

The performance on the two datasets is shown in Table V, where $A_{kNN}$ and $A^{(T)}$ denote the adjacency matrix constructed by the learned representation $H^{(T)}$ and the final adjacency matrix refined through progressive graph refinement, respectively. As we can see that $A^{(T)}$ performs better than both $A$ and $A_{kNN}$ by about 9.3% and 2.4%, respectively, on Cora dataset. The results show that the graph refinement strategy can effectively boost the node representation and the graph structure collaboratively.

We also evaluate the influence of the two regularization terms of conditional adversarial learning on our ADGCN, i.e., $L_{adv}$ and $L_{cls}$. As shown in Table VI, both $L_{adv}$ and $L_{cls}$ have an positive role to the performance of ADGCN. Concretely, the performance gain is just modest when only the $L_{cls}$ is adopted, and different forms of the adversarial objective $L_{adv}$ will bring various performance influences.

Both ADGCN w/o Macro-D and ADGCN w/o Micro-D achieve a better performance than the baseline. More importantly, the performance of the combination of Macro-D and Micro-D (i.e., ADGCN) verifies that both Macro-D and Micro-D are mutually reinforcing. Besides, it can be seen that the adoption of the progressive graph refinement is also helpful for further performance improvement.

To verify the effectiveness of the progressive graph refinement further, the LabelProp [54] is applied to evaluate the ability of the refined adjacency matrix to characterize node relationships. The performance on the two datasets is shown in Table V, where $A_{kNN}$ and $A^{(T)}$ denote the adjacency matrix constructed by the learned representation $H^{(T)}$ and the final adjacency matrix refined through progressive graph refinement, respectively. As we can see that $A^{(T)}$ performs better than both $A$ and $A_{kNN}$ by about 9.3% and 2.4%, respectively, on Cora dataset. The results show that the graph refinement strategy can effectively boost the node representation and the graph structure collaboratively.
TABLE IV
NODE CLUSTERING RESULTS (%)

| Datasets | Metrics | ChebNet [9] | GCN [9] | GAT [10] | DisenGCN [16] | IPGDN [17] | FactorGCN [18] | ADGCN |
|----------|---------|------------|---------|----------|---------------|-------------|----------------|--------|
| Cora     | ACC     | 71.9       | 73.5    | 75.2     | 75.5          | 76.1        | 75.1           | **81.0±1.5** |
|          | NMI     | 49.8       | 51.7    | 57.2     | 58.4          | 59.2        | 58.0           | **62.3±1.0** |
|          | ARI     | 42.4       | 48.9    | 54.1     | 60.4          | 61.0        | 59.3           | **62.1±1.2** |
| Citeseer | ACC     | 65.0       | 67.7    | 68.0     | 62.2          | 68.9        | 66.9           | **69.3±0.3** |
|          | NMI     | 42.6       | 42.8    | 43.7     | 43.7          | 44.3        | 40.7           | **46.6±0.5** |
|          | ARI     | 41.5       | 42.8    | 43.6     | 42.5          | 43.0        | 42.2           | **47.6±1.1** |
| Pubmed   | ACC     | 75.2       | 75.6    | 76.3     | 77.0          | 77.8        | 77.2           | **78.4±0.7** |
|          | NMI     | 35.6       | 35.0    | 35.0     | 36.1          | 37.0        | 36.9           | **39.2±1.3** |
|          | ARI     | 38.6       | 40.9    | 41.4     | 41.6          | 42.0        | 41.4           | **44.1±0.9** |

The boldface represents the better results.

Fig. 8. Visualization of the similarity matrix of node representations in different component spaces on the Cora dataset. We use the red and blue box markers to show the entangled relationship of nodes in different component spaces. (a) First component space. (b) Second component space. (c) Third component space. (d) Fourth component space. (e) Fifth component space.

TABLE V
NODE CLASSIFICATION ACCURACIES OF DIFFERENT GRAPHS

| Graph    | Datasets | ACC | NMI | ARI |
|----------|----------|-----|-----|-----|
| Cora     | WGAN-GP  | 80.8| 80.2| 81.4|
|          | LSGAN    | 79.4| 80.2| 80.3|
|          | Vanilla GAN | 80.3| 80.2| 80.8|
| Citeseer | WGAN-GP  | 70.2| 69.1| 70.8|
|          | LSGAN    | 70.0| 69.1| 70.4|
|          | Vanilla GAN | 69.6| 69.1| 70.4|
| Pubmed   | WGAN-GP  | 78.4| 78.3| 78.9|
|          | LSGAN    | 79.1| 78.3| 79.4|
|          | Vanilla GAN | 78.9| 78.3| 79.2|

The boldface represents the better results.

TABLE VI
EVALUATION OF ADVERSARIAL LEARNING (%)

| Datasets | Metrics | W/o $\mathcal{L}_{dis}$ | W/o $\mathcal{L}_{adv}$ | ADGCN |
|----------|---------|-------------------------|-------------------------|--------|
| Cora     | WGAN-GP | 80.8                    | 80.2                    | 81.4   |
|          | LSGAN   | 79.4                    | 80.2                    | 80.3   |
|          | Vanilla GAN | 80.3| 80.2| 80.8|
| Citeseer | WGAN-GP | 70.2                    | 69.1                    | 70.8   |
|          | LSGAN   | 70.0                    | 69.1                    | 70.4   |
|          | Vanilla GAN | 69.6| 69.1| 70.4|
| Pubmed   | WGAN-GP | 78.4                    | 78.3                    | 78.9   |
|          | LSGAN   | 79.1                    | 78.3                    | 79.4   |
|          | Vanilla GAN | 78.9| 78.3| 79.2|

E. Analysis of Disentanglement

To intuitively show the disentanglement achieved by ADGCN, we visualize the similarity matrix of the Cora dataset in both the original feature and the learned representation spaces. In contrast to the original feature in Fig. 10(a), the similarity matrix based on the learned representation has shown an explicit block effect as we can see from Fig. 10(b). It indicates that the representation learned by our model is capable of effectively capturing intraclass similarity and interclass differences.

More importantly, to reveal the intricate entanglement between different classes of nodes, we visualize the similarity matrices of the component level, thus showing the correlations between nodes in different component spaces. From Fig. 8, it can be observed that each component does reflect some entangled information among different classes at a certain level, and the complex entanglements between different two classes can be well unveiled by the different combinations of components. Taking the first three classes (marked in red box) in Fig. 8 as an example, class 1 and class 2 are significantly entangled but both can be well distinguished from class 3 in the first component space. These three classes also exhibit different entanglement relationships in the second and third component spaces, respectively. Similarly, the three classes marked by the blue box also show various entanglement relations in different component spaces.

As shown in Fig. 9, we also present the component confusion matrix $C$ to illustrate the correlations among the distributions in different component spaces. Here, we use the group average similarity to compute $C$. Specifically, the correlation score between the $i$th and $j$th component distributions is given by $C_{ij} = \frac{1}{|V|} \sum_{u,v \in V} \cos(h_i^u, h_j^v)$. The smaller the score, the weaker the correlation between the two component distributions. Obviously, the representations in different component subspaces learned by our ADGCN take weaker correlations compared to DisenGCN [16]. In particular, as we can find from Fig. 9(b), ADGCN only with microdisentanglement achieves considerable disentanglement performance compared to DisenGCN.
TABLE VII
GPU MEMORY COST AND AVERAGE TRAINING TIME PER EPOCH

| Dataset | DisenGCN | FactorGCN | DisGNN | ADGCN | ADGCN-R |
|---------|----------|-----------|-------|-------|---------|
| Cora    | 1847     | 1771      | 9239  | 1753  | 1753    |
| Citeese | 2179     | 1319      | 15471 | 2267  | 2267    |
| Pumbed  | 4495     | 1549      | 19695 | 3887  | 3887    |
| CS      | 2647     | 2473      | OOM   | 2365  | 2365    |
| Computers | 2275   | 1589      | OOM   | 2505  | 2505    |
| Photo   | 2031     | 1411      | OOM   | 1739  | 1739    |

Average training time per epoch (ms)

| Dataset | DisenGCN | FactorGCN | DisGNN | ADGCN | ADGCN-R |
|---------|----------|-----------|-------|-------|---------|
| Cora    | 201.6    | 227.3     | 502.1 | 190.5 | 212.0   |
| Citeese | 214.8    | 475.0     | 730.7 | 241.3 | 337.1   |
| Pumbed  | 698.6    | 818.9     | 10673.2 | 821.4 | 942.4   |
| CS      | 244.2    | 797.1     | 820.5 | 411.2 | 823.7   |
| Computers | 452.4   | 1091.3    | OOM   | 462.8 | 823.7   |
| Photo   | 191.0    | 721.0     | OOM   | 169.7 | 465.6   |

F. Analysis of Convergence and Efficiency

1) Efficiency Analysis: We evaluate the efficiency of the proposed ADGCN from the perspective of computational cost that consists of the GPU memory consumption and time consumption, respectively. As reported in Table VII, the costs in GPU memory consumption of DisenGCN and ADGCN are almost the same, while the cost is much higher for DisGNN. Particularly, ADGCN-R has the same GPU memory cost as ADGCN because no additional memory consumption is introduced in the graph refinement. In terms of time consumption from Tables VII and VIII, DisenGCN, ADGCN, and ADGCN-R are not as efficient as FactorGCN due to the involvement of the iterative microdisentanglement. In addition, for both DisenGCN and ADGCN, the time costs are comparable since only a little additional time cost with O(m) will be brought by the macro-D module. What is more, due to the graph refinement process, ADGCN-R is somewhat slower than ADGCN although it achieves a better performance.

2) Convergence Analysis: To get a clear view of the convergence of the dynamic assignment mechanism, we define the average difference at iteration $\tilde{t}$ as $b(\tilde{t}) = 1 - \frac{1}{|V|} \sum_{u \in V} \cos(h^K_{\tilde{t}-1}(u), h^K_{\tilde{t}}(u))$, where $h^K_{\tilde{t}}(u)$ denotes the kth component representation of node u at iteration $\tilde{t}$. As shown in Fig. 11, $b(\tilde{t})$ drops rapidly to a small constant only with a few iterations, which experimentally demonstrates that the convergent of our proposed dynamic assignment mechanism can be guaranteed. Besides, for all 5 components, it can be seen that the curves of ADGCN do decrease faster than the curves of ADGCN w/o $L_{adv}$ on both the Cora and Citeese datasets. On the Cora dataset, ADGCN is converged at the third iteration while ADGCN w/o $L_{adv}$ is converged at the fourth iteration. Besides, on the Citeese dataset, ADGCN w/o $L_{adv}$ is slower than ADGCN by 2 iterations to converge.

G. Defense Performance

To verify the robustness of ADGCN, we evaluate the node classification performance of ADGCN against two types of classical attacks, i.e., nontargeted attack and targeted attack. Specifically, two typical attacking methods, nettack [59] and metattack [60] are adopted for the targeted attack method and nontargeted attack method, respectively. We first use the attack method to poison the graph, and then we train ADGCN on the poisoned graph and evaluate the node classification performance. It can be seen from Fig. 12 that ADGCN outperforms the three baselines whether under metatack or nettack. In particular, the performance of the baselines varies greatly under different types of attacks, while the performance of ADGCN degrades less than other methods as the attack intensity increases under both attack methods.

H. Hyperparameter Sensitivity Analysis

We finally study the effect of several hyperparameters, including the number of layers, the iterations of dynamic assignment $T$, and the dimension of the component, on the Cora dataset with the fixed split. The results in Fig. 13 demonstrate that:

Fig. 11. Convergence analysis of the dynamic assignment on two datasets, where the solid lines "--" and the dotted lines "···" denote the curves of ADGCN and ADGCN w/o $L_{adv}$, respectively. (a) Cora. (b) Citeese.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
to 1 is a tradeoff with a good refinement. An appropriate number of nodes to sample is essential for graph representation learning. By microdisentangle-
ment, we innovatively introduce the dynamic assignment mechanism, ADGCN can efficiently complete the aggregation in the component space and still suffers from a slight limitation in efficiency caused by the graph refinement and the iterative microdisentanglement. To further reduce time consumption, a faster graph refinement strategy, such as the integration of graph sampling and refinement, is an interesting topic worth exploring.

VI. CONCLUSION

In this article, we propose an ADGCN model that takes both macrodisentanglement and microdisentanglement into account for graph representation learning. By microdisentangle-
ment, it achieves component-specific aggregation of intercomponent and intracomponent through the dynamic assignment of node neighborhood. Furthermore, we innovatively introduce the macrodisentanglement adversarial regularizer to explicitly constrain the interdependence between components. Meanwhile, we also propose a local structure-aware graph refinement based on diversity-preserving node sampling. With the training going on, the graph structure can be progressively refined to reveal the latent relations of nodes.

Although we attempt to carry out a balance of efficiency and effectiveness in graph disentanglement, the proposed ADGCN still suffers from a slight limitation in efficiency caused by the graph refinement and the iterative microdisentanglement. To further reduce time consumption, a faster graph refinement strategy, such as the integration of graph sampling and refinement, is an interesting topic worth exploring.

REFERENCES

[1] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 1225–1234.
[2] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, arXiv:1611.07308.
[3] L. Tang and H. Liu, “Leveraging social media networks for classification,” Data Mining Knowl. Discov., vol. 23, no. 3, pp. 447–478, 2011.
[4] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Inf. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.
[5] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proc. 22nd ACM SIGKDD Inf. Conf. Knowl. Discov. Data Mining, 2016, pp. 855–864.
[6] J. Tang et al., “Line: Large-scale information network embedding,” in Proc. 24th World Wide Web, 2015, pp. 1067–1077.
[7] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, “Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec,” in Proc. 11th ACM Int. Conf. Web Search Data Mining, 2018, pp. 459–467.
[8] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” in Proc. 24th ACM Conf. Inf. Knowl. Manage., 2015, pp. 891–900.
[9] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2016.
[10] P. Velickovi´c, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. Int. Conf. Learn. Represent., 2018.
[11] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Infor. Process. Syst., 2017, pp. 1024–1034.
[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.
[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” in Proc. Int. Conf. Learn. Represent., 2018.
[14] S. van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem, “Are disentangled representations helpful for abstract visual reasoning?,” in Proc. Adv. Neural Infor. Process. Syst., 2019, pp. 14245–14258.
[15] F. Locatello et al., “Challenging common assumptions in the unsupervised learning of disentangled representations,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 4114–4124.
[16] J. Ma et al., “Disentangled graph convolutional networks,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 4212–4221.

[17] Y. Liu, X. Wang, S. Wu, and Z. Xiao, “Independence promoted graph disentangled networks,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 4916–4923.

[18] Y. Yang, Z. Feng, M. Song, and X. Wang, “Factorizable graph convolutional networks,” Proc. Adv. Neural Inf. Process. Syst., vol. 33, pp. 20286–20296, 2020.

[19] L. Wu et al., “Learning the implicit semantic representation on graph-structured data,” in Proc. Database Syst. Adv. Appl., 2021, pp. 3–19.

[20] J. Guo, K. Huang, X. Yi, and R. Zhang, “Learning disentangled graph convolutional networks locally and globally,” IEEE Trans. Neural Netw. Learn. Syst., to be published, doi: 10.1109/TNNLS.2022.3195336.

[21] T. Zhao, X. Zhang, and S. Wang, “Exploring edge disentanglement for node classification,” in Proc. ACM Web Conf., 2022, pp. 1028–1036.

[22] T. Xiao et al., “Decoupled self-supervised learning for graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 620–634.

[23] H. Li, X. Wang, Z. Zhang, Z. Yuan, H. Li, and W. Zhu, “Disentangled contrastive learning on graphs,” Proc. Adv. Neural Inf. Process. Syst., vol. 34, pp. 21872–21884, 2021.

[24] I. Goodfellow et al., “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.

[25] N. De Cao and T. Kipf, “MolGAN: An implicit generative model for small molecular graphs,” 2018, arXiv:1805.11973.

[26] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, “Netgan: Generating graphs via random walks,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 610–619.

[27] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” in Proc. Int. Conf. Learn. Represent., 2016.

[28] V. Dumoulin et al., “Adversarially learned inference,” in Proc. Int. Conf. Learn. Represent., 2016.

[29] M. HassanPour Zonoozi and V. Seydi, “A survey on adversarial domain adaptation,” Neural Process. Lett., vol. 55, pp. 2429–246, 2023.

[30] H. Wang et al., “GraphGNN: Graph representation learning with generative adversarial nets,” in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 2508–2515.

[31] Y.-C. Lee et al., “ASiNE: Adversarial signed network embedding,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, 2020, pp. 609–618.

[32] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoders for graph embedding,” in Proc. Int. Joint Conf. on Artif. Intell., 2018, pp. 2609–2615.

[33] S. Zheng, Z. Zhuo, X. Zhang, Z. Liu, J. Cheng, and Y. Zhao, “Distribution-induced bidirectional generative adversarial network for graph representation learning,” in Proc. IEEE Comput. Vis. Pattern Recognit., 2020, pp. 7224–7233.

[34] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in Proc. Int. Conf. Learn. Represent., 2018.

[35] Q. Lu et al., “Exploiting node content for multiview graph convolutional networks locally and globally,” in Proc. 28th Int. Conf. Comput. Vis., 2019, pp. 6056–6065.

[36] D. Croce, G. Castellucci, and R. Basili, “GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguist., 2020, pp. 2114–2119.

[37] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in Proc. 32th Int. Conf. Mach. Learn., 2015, pp. 1180–1189.

[38] Y. Chen, L. Wu, and M. Zaki, “Iterative deep graph learning for graph neural networks: Better and robust node embeddings,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 19314–19326.

[39] J. B. Hough et al., “Determinantal processes and independence,” Probab. Surv., vol. 3, pp. 206–229, 2006.

[40] C. Wachinger and P. Golland, “Diverse landmark sampling from determinantal point processes for scalable manifold learning,” 2015, arXiv:1503.03506.

[41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent., 2015.

[42] P. Sen, G. Namata, M. Billic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective classification in network data,” AI Mag., vol. 29, no. 3, pp. 93–93, 2008.

[43] Q. Lu and L. Getoor, “Link-based classification,” in Proc. 20th Int. Conf. Mach. Learn., 2003, pp. 496–503.

[44] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU, “Query-driven active surveying for collective classification,” in Proc. 10th Int. Workshop Mining Learn. Graph., 2012, vol. 8, p. 1.

[45] O. Shchur et al., “Pitfalls of graph neural network evaluation,” 2018, arXiv:1811.05868.

[46] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol., 2003, pp. 173–180.

[47] L. Tang and H. Liu, “Relational learning via latent social dimensions,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2009, pp. 817–826.

[48] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3844–3852.

[49] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, “Graph neural networks with convolutional ARMA filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 7, pp. 3496–3507, Jul. 2022.

[50] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” in Proc. Int. Conf. Learn. Represent., 2018.

[51] F. Wu, A. Souza, T. Zhang, C. Firty, T. Yu, and K. Weinberger, “Simplifying graph convolutional networks,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 6861–6871.

[52] I. Spinelli, S. Scardapane, and A. Uncini, “Adaptive propagation graph convolutional network,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 10, pp. 4755–4760, Oct. 2021.

[53] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions,” in Proc. 20th Int. Conf. Mach. Learn., 2003, pp. 912–919.

[54] F. Monti, D. Boscaïni, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, “Geometric deep learning on graphs and manifolds using mixture model CNNs,” in Proc. IEEE Comput. Vis. Pattern Recognit., 2017, pp. 5115–5124.

[55] Z. Yang, W. Cohen, and R. Salakhudinov, “Revisiting semi-supervised learning with graph embeddings,” in Proc. 33th Int. Conf. Mach. Learn., 2016, pp. 40–48.

[56] J. Bergstra, D. Yamins, and D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in Proc. 30th Int. Conf. Mach. Learn., 2013, pp. 115–123.

[57] S. Lloyd, “Least squares quantization in PCM,” IEEE Trans. Inf. Theory., vol. 28, no. 2, pp. 129–137, Mar. 1982.

[58] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in Proc. 24th ACM SIGKDD Inf. Conf. Knowl. Discov. Data Mining, 2018, pp. 2847–2856.

[59] D. Zügner and S. Günnemann, “Adversarial attacks on graph neural networks via Meta learning,” in Proc. Int. Conf. Learn. Represent., 2019.