




# Boosting Nonnegative Matrix Factorization Based Community Detection With Graph Attention Auto-Encoder

Chaobo He , Yulong Zheng , Xiang Fei, Hanchao Li , Zeng Hu , and Yong Tang 

**Abstract**—Community detection is of great help to understand the structures and functions of complex networks. It has become one of popular research topics in the field of complex networks analysis. Due to the simplicity, flexibility, effectiveness and better interpretability, Nonnegative Matrix Factorization (NMF)-based methods have been widely employed for community detection. However, most existing NMF-based community detection methods are linear and their performance is limited when facing networks with diversified structure information. In view of this, we propose a nonlinear NMF-based method named NMFGAAE, which is composed of two main modules: NMF and Graph Attention Auto-Encoder (GAAE). This approach can boost the performance of NMF-based community detection methods by the aid of graph neural networks and deep clustering. More specifically, GAAE introduces an attention mechanism directed by NMF-based community detection to learn the node representations, while NMF can simultaneously factor these representations to uncover the community structure. We design a unified framework to jointly optimize GAAE and NMF modules, which is very beneficial to obtain better community detection results. We conduct extensive experiments on synthetic and real-world networks. The results show that our NMFGAAE not only performs better than state-of-the-art NMF-based community detection methods, but also outperforms some network representation based baselines. More importantly, NMFGAAE indeed can boost the performance of NMF-based community detection methods.

**Index Terms**—Community detection, nonnegative matrix factorization, graph attention auto-encoder, graph neural networks, deep clustering, complex networks

## 1 INTRODUCTION

COMMUNITY detection is a long-standing problem in the field of complex networks analysis. It aims to detect cohesive groups of nodes, where nodes in the same group connect to each other more densely than those in different groups [1], [2]. Effective methods for community detection not only can be served as the fundamental tools to characterize and understand the structures and functions of complex networks [3], but also having a great application value, such as helping to find groups with similar users in social networks [4], identify fraud groups in telecommunications networks [5], mine research teams in co-authorship networks [6] and discover functional units in protein-protein interaction networks [7]. Accordingly, community detection has attracted a lot of attention from both academia and industry. Moreover, related researchers cover a wide range

of disciplines, including network science, computer science, sociology and bioinformatics.

In the past decades, various methods for community detection have been proposed, such as modularity optimization based methods [8], [9], stochastic block model based methods [10], game theoretic model based methods [11], label propagation based methods [12], deep learning based methods [13] and so on. It is worth mentioning that Nonnegative Matrix Factorization (NMF) based methods are also widely adopted. Compared with other types of methods, NMF-based methods are often more simple and effective, and more interpretable [14]. Particularly, they are competent in almost all community detection tasks, including overlapping community detection [15], [16], semi-supervised community detection [17], [18] and detecting communities from a variety of networks (e.g., undirected networks [19], [20], attributed networks [21], [22], signed networks [23], [24], multi-layer networks [25], [26], dynamic networks [27], [28] and large-scale networks [29], [30]). It can be argued that NMF-based methods are very versatile to deal with the problem of community detection. Until now, NMF-based community detection is still a hot topic and many methods are constantly presented in high-prestige journals and conferences.

NMF-based community detection has exhibited its unique advantages, but it may be less effective when encountering complex networks with complicated and diversified structures. This is due to the intrinsic linearity of NMF. Most NMF-based methods assume that the original networks could be directly reconstructed using a linear

- Chaobo He and Zeng Hu are with the School of Information Science and Technology, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China. E-mail: hechaobo@foxmail.com, huzeng@zhku.edu.cn.
- Yulong Zheng and Yong Tang are with the School of Computer Science, South China Normal University, Guangzhou 510631, China. E-mail: 1457367033@qq.com, ytang@m.scnu.edu.cn.
- Xiang Fei and Hanchao Li are with the Department of Computing, Coventry University, Coventry CV15FB, U.K. E-mail: aa5861@coventry.ac.uk, lih30@uni.coventry.ac.uk.

Manuscript received 30 Apr. 2021; revised 10 July 2021; accepted 2 Aug. 2021. Date of publication 12 Aug. 2021; date of current version 9 July 2022.

(Corresponding author: Yong Tang.)

Recommended for acceptance by G. SONG.

Digital Object Identifier no. 10.1109/TBDA.2021.3103213

combination of community features. However, this linear reconstruction assumption is invalid for these networks mentioned above, because they often have various nonlinear features [31], [32]. To address this issue, a few methods have been proposed and can be roughly divided into the following two types:

- One is devising methods based on deep NMF [33], such as MvDGNMF [34] and DANMF [35]. These methods try to obtain better performance with the aid of deep NMF's expressive power enhanced by the multi-layer factorization structure. However, due to the linear decomposition feature of each layer in deep NMF, deep NMF-based methods are essentially still linear, and their performance improvements are limited.
- The other type of methods is constructing nonlinear community detection model. A representative method is NAGC [36] which is based on symmetric NMF with positive unlabeled learning. This method introduces a nonlinear projection function to learn the relationship between the different community assignments of the topology and the attributes of networks. Although NAGC achieves better performance than some linear NMF-based methods, it cannot be generalized to networks with only structure information. Besides, it is very time consuming and can not deal with networks beyond a certain scale.

In summary, these two kinds of methods above both try to improve NMF-based community detection model by enhancing its expressive power. Considering that nonlinear models often have stronger expressive power to learn complicated features of data samples, constructing nonlinear community detection model will be a better strategy. However, relatively little attention has been dedicated to this aspect, and more effective methods still need to be explored.

In recent years, Deep Neural Networks (DNNs) have revolutionized many machine learning tasks, including image classification [37], speech recognition [38] and machine translation [39]. Due to the powerful nonlinear expressive power, DNNs have also been widely used to boost the performance of tradition machine learning models. Among them, deep clustering which combines clustering model (e.g., K-means and spectral clustering) with DNNs is very attractive and many effective methods have been proposed, such as DCN [40], DEC [41] and SC-EDAE [42]. Comparing with traditional clustering methods, deep clustering methods demonstrate great superiorities, especially in terms of the clustering performance. On the other hand, currently, as a special extension of DNNs to graph data, Graph Neural Networks (GNNs) have received increasing interest and many variants have been presented, such as famous Graph Convolutional Network (GCN) [43] and Graph Attention Network (GAT) [44]. Existing works show that GNNs are more suitable and powerful to conduct nonlinear feature learning on graph data than other general DNNs (e.g., auto-encoder and convolutional neural networks) [45], [46]. This makes them have great potential to improve many graph-related tasks, such as community detection focused in this paper. All of these above give us a

good inspiration that using GNNs can be expected to better boost NMF-based community detection.

Motivated by the above-mentioned analyses, in this paper we propose a nonlinear NMF-based community detection method named NMFGAAE, which is powered by a specially designed variant of GNNs: graph attention auto-encoder (GAAE). To the best of our knowledge, it is the first time to utilize GNNs to boost the performance of NMF-based community detection. Our contributions are summarized as follows:

- NMFGAAE provides a unified framework to integrate GAAE with NMF-based community detection model. Using this framework, GAAE and NMF-based community detection model can be optimized jointly, which enables GAAE to generate more beneficial node representations for NMF-based community detection.
- For the GAAE module in NMFGAAE, we design an attention mechanism which specially serve NMF-based community detection. Comparing with the attention mechanism used in GAT, our proposed attention mechanism can help NMFGAAE obtain better performance.
- We conduct extensive experiments to demonstrate the effectiveness of NMFGAAE on both synthetic and real-world networks. The results show that NMFGAAE not only outperforms state-of-the-art NMF-based community methods, but also outperforms some widely used network representation based methods.

The rest of this paper is organized as follows. We first briefly review the related work in Section 2, and then respectively detail the proposed method NMFGAAE and report the experiment results in Sections 3 and 4. Finally, we conclude this paper in Section 5.

## 2 RELATED WORK

In this section, we introduce some background knowledge and overview some work that relates to the features used in our method NMFGAAE: NMF-based community detection, GNNs and deep clustering.

### 2.1 NMF-Based Community Detection

NMF is a classical low-rank matrix factorization model proposed by Lee and Seung [47]. It can factor a nonnegative data sample matrix into the approximate product of two nonnegative matrices: the basis matrix and the coefficient matrix. Due to the nonnegativity constraint, every data sample can be represented as an additive linear combination of the basis features in the basis matrix, and the coefficients are from the corresponding vector in the coefficient matrix. Therefore, NMF is essentially a linear model.

In [48], Ding *et al.* proved that NMF has approximately equivalent relationships with K-means and spectral clustering models, which are both applicable to detect communities from complex networks. Besides, from the perspective of generative model, Psorakis *et al.* [14] demonstrated the powerful interpretability of NMF when used in community detection. These works provide a solid theoretical

foundation to NMF-based community detection, and meanwhile stimulate the research interest in this field. Recently, various NMF-based community detection methods have been proposed, such as symmetric NMF-based methods [16], [19], [20], semi-supervised NMF-based methods [17], [18], semi-NMF-based methods [23], [24], joint NMF-based methods [25], [26], [27], [28], [49] and deep NMF-based methods [34], [35]. These methods construct the community detection model by extending the basic NMF, such as relaxing nonnegative constraint (semi-NMF), imposing graph regularization (graph regularized NMF), stacking multi-layer factorization structure (deep NMF) and so on. With these strategies, they all have achieved good performance in some cases.

Although existing NMF-based community methods use various NMF variants, most of them are still within the category of linear methods due to the intrinsic linearity of NMF model. It is generally believed that linear community detection model may be less effective when facing networks with diversified structure information, which are considered to have various nonlinear features [31], [32]. In this paper, our proposed NMFGAAE is nonlinear due to the combination with a variant of GNNs: graph attention auto-encoder, which makes it fundamentally different from existing NMF-based community detection methods.

## 2.2 GNNs

The concept of GNNs was formally proposed in [50], which extended existing neural networks to process graph data. Up to now, lots of GNNs have been presented, which mainly include three categories: graph recurrent neural networks (GRNNs), graph convolutional networks (GCNs) and graph auto-encoders (GAEs). Among them, GCNs have received more attention in recent years, which is largely attributed to the great success of convolution neural networks in computer vision domain. According to the difference in how to define graph convolutions, GCNs are further categorized into two types: spectral-based and spatial-based. Spectral-based GCNs define the graph convolutions as low-pass filters of graph signals, such as spectral CNN [51], ChebNet [52] and GCN [43]. Spatial-based GCNs define graph convolutions as the process of information propagation. The key idea behind them is to aggregate feature information from neighbor nodes via specific aggregation schemes, such as LSTM pooling used in GraphSAGE [53], attention mechanism used in GAT [44] and max-pooling used in GIN [54]. In general, spatial-based GCNs are more efficient, because spectral-based GCNs often need to conduct the costly Laplacian matrix eigen-decomposition. Besides, spatial-based GCNs are more flexible, because aggregation schemes, especially attention mechanism, allow them to only focus on task-relevant parts instead of the whole graph structure.

As another type of GNNs, GAEs is applied to learn node low-dimensional representations by an unsupervised training manner. They are composed of two parts: Encoder and Decoder. Encoder maps nodes into a latent feature space and decoder reconstructs graph structure from latent representations. In particular, the encoder part can adopt various GNNs flexibly, such as GCN encoder [55], [56] and GAT

encoder [57]. Motivated by the unsupervised characteristic of community detection task here, and the effectiveness and flexibility of GAEs, we construct our method NMFGAAE based on the architecture of GAEs and specially design a spatial-based GCN with attention mechanism as its encoder.

## 2.3 Deep Clustering

Clustering with deep learning are called as deep clustering, which aims to improve the clustering performance via DNNs [58], [59]. In the early stage, most deep clustering methods comprise two steps, such as StructAE [60] and SAE [61]. These methods first use DNNs to learn data representations and then feed these representations into the specified clustering model (e.g., K-means) directly. The drawback of two-step methods is that the learned representations may not be the best fit for the subsequent clustering model, and the clustering model is not beneficial to the representation learning. To achieve mutual benefit for these two steps, one-step deep clustering is presented. It provides a unified framework to jointly train representation learning model and clustering model. Representative methods include DCN [40], DEC [41] and SC-EDAE [42], which all demonstrate that one-step deep clustering can learn more clustering-friendly data representations, resulting in more significant increase of the clustering performance.

It should be noted that most existing deep clustering methods are designed for data with grid or sequence structure (e.g., image and text data). More recently, with the popularity of GNNs, a few deep clustering methods for graph data are proposed, such as SDCN [56], DAEGC [57] and EGAE-JOCAS [62]. These methods all belong to one-step deep clustering approaches, and meanwhile use GAEs to learn node representations. Our method NMFGAAE adopts the similar architecture with them, but it still has some obvious differences: (1) NMFGAAE selects NMF as the clustering model, but DAEGC and SDCN select self-supervised clustering model, and EGAE-JOCAS simultaneously uses K-means and spectral clustering models. (2) NMFGAAE only exploits the structure information of the graph, but DAEGC, SDCN and EGAE-JOCAS all use the structure and attribute information. (3) NMFGAAE utilizes a different attention mechanism with DAEGC, whereas SDCN and EGAE-JOCAS do not use any attention mechanism. (4) More importantly, NMFGAAE is a goal-directed clustering method, which specially focuses on boosting NMF-based community detection.

To the best of our knowledge, there is still no work about applying the idea of one-step deep clustering to boost NMF-based community detection (i.e., integrating network representation model and NMF-based community detection model into a unified optimization framework), not to mention integrating promising GNNs-based network representation models. Therefore, NMFGAAE can be expected to perform better in practice.

## 3 THE PROPOSED METHOD: NMFGAAE

In this section, we will first introduce the notations and preliminaries used in this paper, next give an overview about



TABLE 1  
Notations and Descriptions

Notation	Description
$G$	The input network
$V$	Nodes set
$v_i$	The $i$ th node
$E$	Edges set
$ E $	The number of edges
$e_{ij}$	The edge from $v_i$ to $v_j$
$n$	The number of nodes
$k$	The number of communities
$C$	Communities set
$C_i$	The $i$ th community
$X$	The node feature matrix
$m$	The dimension of node feature
$A$	The adjacency matrix
$I$	The identity matrix
$Z$	The node latent representation matrix
$d$	The dimension of node latent representation
$U$	The community membership matrix
$V$	The community feature matrix
$\mathcal{N}_i$	The neighbor nodes set of $v_i$
$\mathbb{R}$	The real number set
$\mathbb{R}_+$	The nonnegative real number set

the proposed NMFGAAE framework, then detail its modules and finally present how to optimize NMFGAAE model.

### 3.1 Notations and Preliminaries

Throughout this paper, matrices are denoted by bold uppercase letters. For a given matrix  $Y$ , its  $i$ th row vector,  $(i, j)$ th element, trace, transpose and Frobenius norm are denoted by  $Y_i$ ,  $Y_{ij}$ ,  $tr(Y)$ ,  $Y^T$  and  $\|Y\|_F$ , respectively. For clarity, we summarized all frequently used notations in Table 1.

In this paper, we focus on non-overlapping community detection in undirected networks with only structure information. This is because most NMF-based community detection methods are suitable for this application scenario, and undirected networks with only structure information are more readily available in real-world.

Without loss of generality, we denote an undirected network as an undirected and unweighted graph  $G = (V, E)$ , where  $V = \{v_1, v_2, \dots, v_n\}$  is the nodes set and  $E = \{e_{ij} | v_i \in V \wedge v_j \in V\}$  is the edges set. To represent the structure information of  $G$ , we use a binary-valued adjacency matrix  $A = [A_{ij}]^{n \times n}$ , where  $A_{ij} = 1$  if  $e_{ij} \in E$  and  $A_{ij} = 0$  otherwise. Assuming that  $G$  comprises  $k$  non-overlapping communities, we denote the non-overlapping communities set as  $C = \{C_i | C_i \neq \emptyset, 1 \leq i \leq k\}$ , where  $C_i \cap C_j = \emptyset$  ( $i \neq j$ ). Based on the definitions above, the problem of community detection using NMFGAAE here is formally formulated as follows.

**Problem Statement.** Given a network  $G = (V, E)$ , using NMFGAAE to obtain the community membership matrix  $U = [U_{ij}]^{n \times k} \in \mathbb{R}_+^{n \times k}$ , where  $U_{ij}$  denotes the strength that  $v_i$  belongs to community  $C_j$ . For the given  $v_i$ , its non-overlapping community membership is determined by assigning it into the  $p$ th community satisfying the condition that  $U_{ip}$  is the maximum element in  $U_i$ .

### 3.2 An Overview of NMFGAAE Framework

The overall framework of NMFGAAE is shown in Fig. 1 and it consists of two main modules: GAAE (i.e., graph attention auto-encoder) and NMF. GAAE module learns node representation  $Z \in \mathbb{R}^{n \times d}$  of the graph  $G$  by minimizing the reconstruction loss. It takes  $A$  and a node feature matrix  $X \in \mathbb{R}^{n \times m}$  as the input. Because we focus on networks with only structure information, the attribute features adopted in most GCNs are unavailable here. However,  $X$  can be obtained by using any graph representation algorithm (e.g., DeepWalk [63], LINE [64] and DNGR [65]) or simply replaced with the identity matrix  $I$ , which is suggested in [66]. Based on the learned  $Z$ , NMF module iteratively factorizes it to obtain community membership matrix  $U \in \mathbb{R}_+^{n \times k}$ .

GAAE is optimized by minimizing the sum of the GAAE loss and NMF loss. Through this unified framework, we can learn node representations and perform community detection simultaneously, so that GAAE and NMF modules can benefit from each other. In the following two subsections, we will detail GAAE and NMF modules, respectively.

### 3.3 GAAE Module

**Encoder.** As shown in Fig. 1, Encoder attempts to learn the node representation  $Z$  via multiple graph convolutional layers with attention, which can adaptively measure the importance of neighbors to the target node. By borrowing from the attention mechanism used in GAT, we specially design an attention mechanism directed by NMF-based

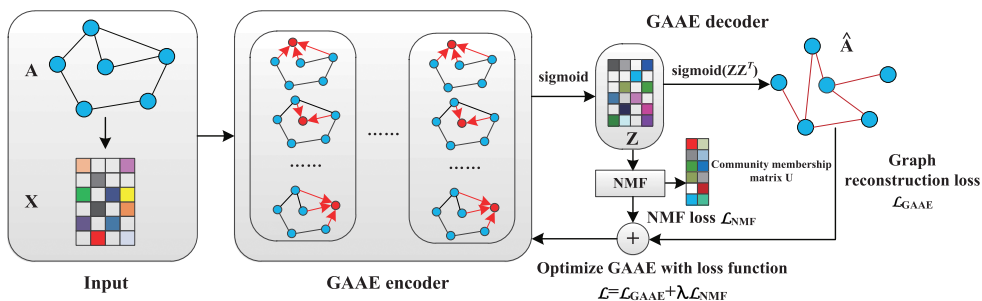


Fig. 1. The framework of NMFGAAE. Note that the input node feature  $X$  is not the attribute feature matrix, and it can be obtained from  $A$  by applying any network representation algorithm or simply replaced with the identity matrix  $I$ .

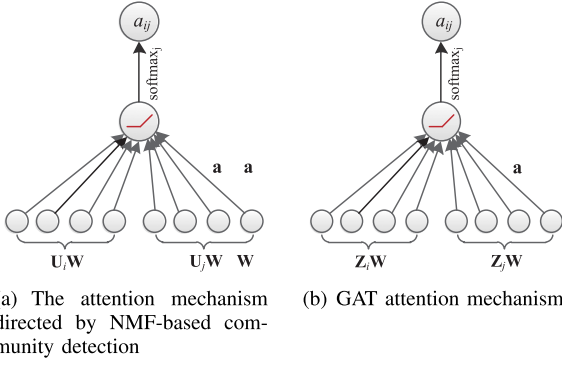


Fig. 2. Comparison of attention mechanism used in NMFGAAE and GAT.

community detection. Specifically, we define the attention coefficient  $\delta_{ij}$  from  $v_j$  to  $v_i$  as

$$\delta_{ij} = a(\mathbf{U}_i \mathbf{W} \parallel \mathbf{U}_j \mathbf{W}), \quad (1)$$

where  $a$  is the attention mechanism using a single-layer feedforward neural network on the concatenation ( $\parallel$ ) of  $\mathbf{U}_i \mathbf{W}$  and  $\mathbf{U}_j \mathbf{W}$  with weight vector  $\mathbf{a} \in \mathbb{R}^{2k}$ , and  $\mathbf{W} \in \mathbb{R}^{k \times k}$  is a weight matrix shared by the linear transformation of each  $\mathbf{U}_i$ . Note that we use the community membership representation  $\mathbf{U}$  instead of node representation  $\mathbf{Z}$  to compute attention coefficients, which is obviously distinct from the strategy used in GAT (Fig. 2).

We believe that this attention mechanism directed by NMF-based community detection can well incorporate the community membership information into the attention coefficients: for a target node  $v_i$  and  $\forall v_j \in \mathcal{N}_i$ , if  $\mathbf{U}_i$  is more similar to  $\mathbf{U}_j$ , then  $v_i$  and  $v_j$  are more likely to be in the same community and the corresponding attention coefficient  $e_{ij}$  will be bigger, which means that  $v_j$  will contribute more to the representation of  $v_i$ , or vice versa. By this attention mechanism, it can be expected to learn the node representation  $\mathbf{Z}$  which is more friendly to the community detection. This will be confirmed in our experiments.

To make attention coefficients easily comparable, we normalize them across all neighbor nodes with a softmax function

$$a_{ij} = \text{softmax}_j(\delta_{ij}) = \frac{\exp(\delta_{ij})}{\sum_{q \in \mathcal{N}_i} \exp(\delta_{iq})}. \quad (2)$$

Applying the LeakyReLU activation function, the coefficients can be finally expressed as

$$a_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}[\mathbf{U}_i \mathbf{W} \parallel \mathbf{U}_j \mathbf{W}]))}{\sum_{q \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\mathbf{a}[\mathbf{U}_i \mathbf{W} \parallel \mathbf{U}_q \mathbf{W}]))}. \quad (3)$$

We stack  $L$  graph attention layers with nonlinear activation function ReLU, and obtain the layer-wise representation for every node via the following aggregation scheme:

$$\mathbf{Z}_i^{l+1} = \text{ReLU} \left( \sum_{j \in \mathcal{N}_i} a_{ij} \mathbf{W}^l \mathbf{Z}_j^l \right), \quad (4)$$

where  $\mathbf{Z}_i^l$  and  $\mathbf{W}^l$  denotes the representation of  $v_i$  and the linear transformation weight matrix at  $l$ th layer, respectively.

Note that we set  $\mathbf{Z}^0 = \mathbf{X}$ , and at the final layer we use sigmoid function to obtain the latent representation  $\mathbf{Z}$

$$\mathbf{Z} = \text{sigmoid}(\mathbf{Z}^{(L-1)}), \quad (5)$$

where  $\mathbf{Z}$  is nonnegative for  $0 \leq \mathbf{Z} \leq 1$ , and will be factorized to uncover community structures in the following NMF module.

Similar to GAT, we also introduce multi-head attention mechanism to independently execute the transformation of Eq. (4) and average their layer-wise latent representations as

$$\mathbf{Z}_i^{l+1} = \text{ReLU} \left( \frac{1}{H} \sum_{h=1}^H \sum_{j \in \mathcal{N}_i} a_{ij}^h \mathbf{W}_h^l \mathbf{Z}_j^l \right), \quad (6)$$

where  $H$  is the number of attention heads,  $a_{ij}^h$  is the attention coefficient computed by the  $h$ th attention mechanism ( $a^h$ ), and  $\mathbf{W}_h^l$  is the corresponding linear transformation weight matrix.

*Decoder.* After obtaining  $\mathbf{Z}$ , decoder uses it to reconstruct the structure of  $G$ . Here, we devise the following inner product decoder:

$$\hat{\mathbf{A}} = \text{sigmoid}(\mathbf{Z} \mathbf{Z}^T), \quad (7)$$

where  $\hat{\mathbf{A}} \in \mathbb{R}_+^{n \times n}$  denotes the reconstructed structure matrix of  $G$ . The reconstruction loss  $\mathcal{L}_{\text{GAAE}}$  is measured via the following binary cross entropy function:

$$\mathcal{L}_{\text{GAAE}} = -\frac{1}{n^2} \sum_{i,j} (\mathbf{A}_{ij} \log \hat{\mathbf{A}}_{ij} + (1 - \mathbf{A}_{ij}) \log (1 - \hat{\mathbf{A}}_{ij})). \quad (8)$$

### 3.4 NMF Module

NMF module takes the output  $\mathbf{Z}$  of GAAE module as the input to detect communities from  $G$ . There are many NMF variants with better performance, such as graph regularized NMF [67] and orthogonal NMF [68], but we select the basic NMF model to factorize  $\mathbf{Z}$  to obtain the community membership matrix  $\mathbf{U} \in \mathbb{R}_+^{n \times k}$  and the community feature matrix  $\mathbf{V} \in \mathbb{R}_+^{k \times d}$ . In this way, we want to prove that even the combination of the basic NMF and GAAE also can achieve the goal of boosting the performance of NMF-based community detection. The corresponding NMF loss  $\mathcal{L}_{\text{NMF}}$  is denoted as

$$\mathcal{L}_{\text{NMF}} = \|\mathbf{Z} - \mathbf{U} \mathbf{V}\|_F^2, \quad (9)$$

where  $\mathbf{U}$  and  $\mathbf{V}$  are both nonnegative.

### 3.5 Optimization

To train the whole NMFGAAE model, we merge  $\mathcal{L}_{\text{GAAE}}$  and  $\mathcal{L}_{\text{NMF}}$  into a unified loss function  $\mathcal{L}$

$$\mathcal{L} = \mathcal{L}_{\text{GAAE}} + \lambda \mathcal{L}_{\text{NMF}}, \quad (10)$$

where  $\lambda$  is a tradeoff parameter. By minimizing  $\mathcal{L}$ , it can not only optimize GAAE, but also can simultaneously optimize NMF. Essentially, this is a joint optimization process, which can make GAAE and NMF benefit from each other. Namely, GAAE can output more friendly node latent representation

$\mathbf{Z}$  for NMF community detection module, meanwhile NMF module also can well improve the accuracy of attention coefficients in GAAE by Eq. (3).

Obviously, minimizing  $\mathcal{L}$  is a non-convex constrained optimization problem, which is hard to obtain its close-form solution. In view of this, we adopt the following alternating optimization strategy to solve this problem.

*Optimize GAAE With NMF Fixed.* When NMF related variables  $\mathbf{U}$  and  $\mathbf{V}$  are fixed as constants, we use  $\mathcal{L}$  to optimize GAAE and update its network parameters:  $\mathbf{W}$  and  $\mathbf{W}_h^l$  using the standard gradient decent

$$\mathbf{W} = \mathbf{W} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad (11)$$

$$\mathbf{W}_h^l = \mathbf{W}_h^l - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}_h^l}, \quad (12)$$

where  $\alpha$  is the learning rate, and the gradients  $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$  and  $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_h^l}$  can be computed by the back-propagation algorithm widely used in DNNs.

*Optimize NMF With GAAE Fixed.* When GAAE is fixed, minimizing  $\mathcal{L}$  is equivalent to

$$\begin{aligned} \min \mathcal{L}_{\text{NMF}} &= \|\mathbf{Z} - \mathbf{U}\mathbf{V}\|_F^2, \\ \text{s.t. } &\mathbf{U} \geq 0, \mathbf{V} \geq 0. \end{aligned} \quad (13)$$

To solve this constrained optimization problem, we use the Lagrange multiplier method and define the following Lagrange function:

$$\mathcal{F} = \|\mathbf{Z} - \mathbf{U}\mathbf{V}\|_F^2 + \text{tr}(\mathbf{\Phi}\mathbf{U}^T) + \text{tr}(\mathbf{\Psi}\mathbf{V}^T), \quad (14)$$

where  $\mathbf{\Phi}$  and  $\mathbf{\Psi}$  are the Lagrange multipliers. The derivatives of  $\mathcal{F}$  with respect to  $\mathbf{U}$  and  $\mathbf{V}$  are

$$\frac{\partial \mathcal{F}}{\partial \mathbf{U}} = -\mathbf{Z}\mathbf{V}^T + \mathbf{U}\mathbf{V}\mathbf{V}^T + \mathbf{\Phi}, \quad (15)$$

$$\frac{\partial \mathcal{F}}{\partial \mathbf{V}} = -\mathbf{U}^T\mathbf{Z} + \mathbf{U}^T\mathbf{U}\mathbf{V} + \mathbf{\Psi}. \quad (16)$$

Using the Karush-Kuhn-Tucker (KKT) conditions that:  $\mathbf{\Phi}_{ij}\mathbf{U}_{ij} = 0$  and  $\mathbf{\Psi}_{ij}\mathbf{V}_{ij} = 0$ , we can derive the following multiplicative update rules for  $\mathbf{U}$  and  $\mathbf{V}$ :

$$\mathbf{U} = \mathbf{U} \odot \frac{\mathbf{Z}\mathbf{V}^T}{\mathbf{U}\mathbf{V}\mathbf{V}^T}, \quad (17)$$

$$\mathbf{V} = \mathbf{V} \odot \frac{\mathbf{U}^T\mathbf{Z}}{\mathbf{U}^T\mathbf{U}\mathbf{V}}, \quad (18)$$

where  $\odot$  and  $\oslash$  are the element-wise product and division operators, respectively. Through iteratively execute Eqs. (17) and (18) until  $\mathcal{L}_{\text{NMF}}$  converge, we can obtain the final  $\mathbf{U}$  to uncover community structure of  $G$ . It should be pointed out that: after each complete update for NMF module, the output  $\mathbf{U}$  and  $\mathbf{V}$  should be preserved as the initial values of the next update. Only by this way,  $\mathbf{U}$  can become better and better during the whole optimization process. Based on the

analysis above, we summarize NMFGAAE community detection algorithm in Algorithm 1.

---

**Algorithm 1.** NMFGAAE Community Detection Algorithm

---

**Input:**  $G = (V, E), k, \lambda;$   
**Output:** Communities set  $C = \{C_1, C_2, \dots, C_k\};$

- 1 **repeat**
- 2    $\mathbf{Z} \leftarrow$  Optimize GAAE module via Eqs. (11) and (12);
- 3    $\mathbf{U}, \mathbf{V} \leftarrow$  Optimize NMF module via iteratively executing Eqs. (17) and (18);
- 4 **until**  $\mathcal{L}$  converges or exceed the maximum iterations
- 5 **for**  $v_i \in V$  **do**
- 6    $q = \arg \max_p \mathbf{U}_{ip};$
- 7    $C_q = C_q \cup \{v_i\};$
- 8 **return**  $C;$

---

### 3.6 Time Complexity Analysis

From Algorithm 1, we can observe that it is composed of two stages: iteratively optimizing GAAE and NMF, and extracting communities from  $\mathbf{U}$ . It is clear that the time complexity of the second stage is  $\mathcal{O}(nk)$ , and the first stage is the most time consuming. Assuming that the number of outer iterations is  $T$  and the number of inner iterations for NMF is  $t$ , the time complexity of the first stage can be expressed as  $\mathcal{O}(T(Hnmd + H|E|d + tndk))$ , where  $\mathcal{O}(T(Hnmd + H|E|d))$  and  $\mathcal{O}(Ttndk)$  are the time complexities of GAAE and NMF modules, respectively. By summing up and considering that  $m \ll n$ ,  $d \ll n$ ,  $k \ll n$  and  $H$  often takes a very small value, we can deduce that the overall time complexity of NMFGAAE is proportional to  $n$  and  $|E|$ . In real-world networks,  $n$  is often far fewer than  $|E|$ , so the time cost of NMFGAAE will be largely dominated by  $|E|$ .

## 4 EXPERIMENTS

In this section, to validate the effectiveness of our proposed method NMFGAAE, we conduct extensive experiments by comparing it with state-of-the-art methods on several synthetic and real-world networks. All methods are implemented in Python 3.5 and all of experiments are conducted on a PC with 64bits Windows 7 system, 3.4G Intel Core i7-6700 CPU and 32 GB RAM.

### 4.1 Baselines

Our basic hypothesis in this paper is that NMFGAAE is a non-linear NMF-based community detection method powered by graph attention auto-encoder, and is expected to achieve better performance than existing NMF-based community detection methods, almost all of which are linear. In view of this, we select eight representative NMF-based community detection methods as baselines. Besides, to further validate the superiority of NMFGAAE, we also compare it with five representative network representation based community detection methods. These baselines are detailed as follows.

The NMF-based community detection methods include:

- NMF. NMF is a module of NMFGAAE, and it can also be used to detect communities independently by replacing  $\mathbf{Z}$  in Eq. (9) with  $\mathbf{A}$ , i.e.,  $\|\mathbf{A} - \mathbf{U}\mathbf{V}\|_F^2$ .

- SNMF. SNMF is the symmetric NMF, which factorizes  $\mathbf{A}$  into the product of  $\mathbf{U}$  and  $\mathbf{U}^T$  with the objective function:  $\|\mathbf{A} - \mathbf{U}\mathbf{U}^T\|_F^2$ . It has been adopted for community detection in [19], [20].
- ONMF. ONMF is a variant of NMF by enforcing orthogonal constraints on community membership matrix  $\mathbf{U}$ , i.e.,  $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ . In [48], ONMF is proven to be equivalent to K-means.
- HPNMF [69]. HPNMF is based on symmetric NMF with graph regularized constraint. Through this framework, it can integrate structure information with node homophily information to detect communities.
- DANMF [35]. DANMF is based on deep NMF. It consists of an encoder component and a decoder component, which enables it to learn the hierarchical mappings between the original network and the final community structure.
- M-NMF [70]. M-NMF is a modularized NMF based community detection method. It provides a unified framework to jointly optimize NMF-based representation learning model and modularity-based community detection model.
- NSED [49]. NSED is similar to DANMF and also comprises an encoder component and a decoder component, but every component is still based on shallow NMF with only one layer factorization structure.
- BigClam [29]. BigClam is a cluster affiliation model. It uncovers community structure by incorporating NMF into the probabilistic generative framework, and meanwhile provides an efficient optimization algorithm.

The network representation based methods include:

- DeepWalk [63]. DeepWalk is a random walks based network representation learning method. It learns node representations by using local information obtained from truncated random walks.
- LINE [64]. LINE is an edge modeling based method. Through modeling a joint probability distribution and a conditional probability distribution on connected nodes, it can learn the first-order and the second-order proximity preserving node representations.
- DNGR [65]. DNGR is a deep learning based method. It first constructs the high-dimensional positive pointwise mutual information matrix representations, and then applies the stacked denoising auto-encoder to obtain node representations.
- GCN [43]. GCN is a spectral-based GNN method. Here, we use the Eq. (8) as its objective function to learn the unsupervised node representations.
- GraphSAGE [53]. GraphSAGE is an inductive GNN based method. It can learn a function that generates representations by sampling and aggregating features from the node's local neighborhood.

Note that some NMF-based methods (e.g., BigClam) claim to detect overlapping communities, but they are also very straightforward to detect non-overlapping communities by following the steps of the second stage in Algorithm 1. NMF-based methods can directly obtain community

TABLE 2  
Parameters of LFR Networks

Parameter	Explanation	Value
$n$	Number of nodes	To vary
$\mu$	Mixing paramter	To vary
$on$	Number of overlapping nodes	0
$om$	Number of overlapping memberships	0
$d_{avg}$	The average degree	4
$d_{max}$	The maximum degree	15
$\lambda_1$	Exponent for node degree distribution	2
$\lambda_2$	Exponent for community size distribution	1
$C_{min}$	Minimum community size	50
$C_{max}$	Maximum community size	100

detection results without seeking help from other clustering algorithms. For the network representation based methods, we utilize standard K-means algorithm to cluster the final node representations to obtain community detection results. For the Python implementations of SNMF, DANMF, M-NMF, NSED, BigClam and DeepWalk, we refer to the open-source Python framework for unsupervised learning on Graphs: Karate Club [71].

## 4.2 Evaluation Metrics

We select four widely used metrics to evaluate the performance of community detection. For the networks with ground-truth communities, we use accuracy (ACC), adjusted Rand index (ARI) and normalized mutual information (NMI). For the networks without ground-truth communities, we use modularity Q. Detailed definitions for these metrics can be found in [72], which is a comprehensive survey on metrics for community detection.

According to the definitions, we can obtain  $ACC \in [0, 1]$ ,  $ARI \in [-1, 1]$ ,  $NMI \in [0, 1]$  and  $Q \in [-1, 1]$ . For all these metrics, larger values indicate better performance.

## 4.3 Parameter Settings

For fair comparisons, the parameters of every baseline are set to be their optimal values. Specifically, for HPNMF, we set its regularization parameter  $\lambda$  to be 1. For DANMF, its regularization parameter  $\lambda$  and layer configuration are set to be 1 and  $n-256-128-k$ , respectively. For BigClam, we set its  $\ell_1$  regularization parameter  $\lambda$  to be 10. For DeepWalk, we set walks per vertex  $\gamma = 80$ , window size  $\omega = 10$  and walk length  $\ell = 40$ . For LINE, the number of negative samples  $K$  is set to be 5. For DNGR, its layer configuration is set to be  $n-256-128-64$ . For GCN, its layer configuration is set to be  $n-256-64$ . For GraphSAGE, the search depth  $D$  is set to be 2. For NMFGAAE, we set  $\lambda$ ,  $H$ ,  $t$  and the layer configuration of graph encoder to be 1, 3, 50 and  $n-256-64$ , respectively. The size of node representations of all the network representation based methods is uniformly set to be 64. Note that the number of communities  $k$  for all methods is set to be the same value and using the identify matrix as the input node feature matrix for GCN, GraphSAGE and NMFGAAE. Under the corresponding optimal settings for different



TABLE 3  
Performance Comparison on Synthetic Networks With Different  $n$  (Bold Numbers Denote the Best Results)

$n$	Metrics	NMF	SNMF	ONMF	HPNMF	DANMF	M-NMF	NSED	BigClam	DeepWalk	LINE	DNGR	GCN	GraphSAGE	NMFGAAE
1000	ACC	0.51	0.50	0.49	0.54	0.55	0.23	0.27	0.12	0.25	0.44	0.57	0.44	0.47	<b>0.59</b>
	ARI	0.29	0.28	0.29	0.33	0.32	0.07	0.09	-0.01	0.08	0.21	0.31	0.21	0.18	<b>0.37</b>
	NMI	0.39	0.37	0.39	0.42	0.41	0.18	0.20	0.03	0.19	0.35	0.45	0.35	0.41	<b>0.46</b>
2000	ACC	0.48	0.51	0.51	0.54	0.50	0.24	0.25	0.13	0.14	0.55	0.51	0.51	0.36	<b>0.56</b>
	ARI	0.30	0.31	0.35	0.34	0.29	0.09	0.10	0.02	0.03	0.34	0.33	0.31	0.15	<b>0.36</b>
	NMI	0.47	0.47	0.50	<b>0.52</b>	0.47	0.28	0.30	0.14	0.19	<b>0.52</b>	0.51	0.51	0.35	<b>0.52</b>
3000	ACC	0.49	0.51	0.51	0.53	0.53	0.14	0.23	0.11	0.16	0.59	0.49	0.53	0.36	<b>0.61</b>
	ARI	0.31	0.31	0.32	0.34	0.33	0.04	0.10	0.02	0.04	0.36	0.31	0.31	0.21	<b>0.39</b>
	NMI	0.44	0.50	0.50	0.54	0.53	0.24	0.34	0.20	0.19	0.57	0.51	0.54	0.39	<b>0.58</b>
4000	ACC	0.45	0.48	0.52	0.54	0.52	0.14	0.26	0.11	0.11	0.60	0.52	0.57	0.35	<b>0.61</b>
	ARI	0.29	0.30	0.34	0.36	0.34	0.04	0.11	0.02	0.01	0.38	0.33	0.38	0.18	<b>0.40</b>
	NMI	0.41	0.52	0.55	0.58	0.56	0.26	0.38	0.22	0.18	0.59	0.51	0.59	0.36	<b>0.60</b>
5000	ACC	0.51	0.53	0.51	0.53	0.53	0.25	0.25	0.10	0.09	0.58	0.55	0.54	0.36	<b>0.59</b>
	ARI	0.31	0.34	0.32	0.35	0.34	0.11	0.11	0.02	0.01	0.38	0.31	0.35	0.17	<b>0.39</b>
	NMI	0.53	0.56	0.53	0.59	0.58	0.40	0.39	0.22	0.18	0.59	0.52	0.59	0.41	<b>0.61</b>

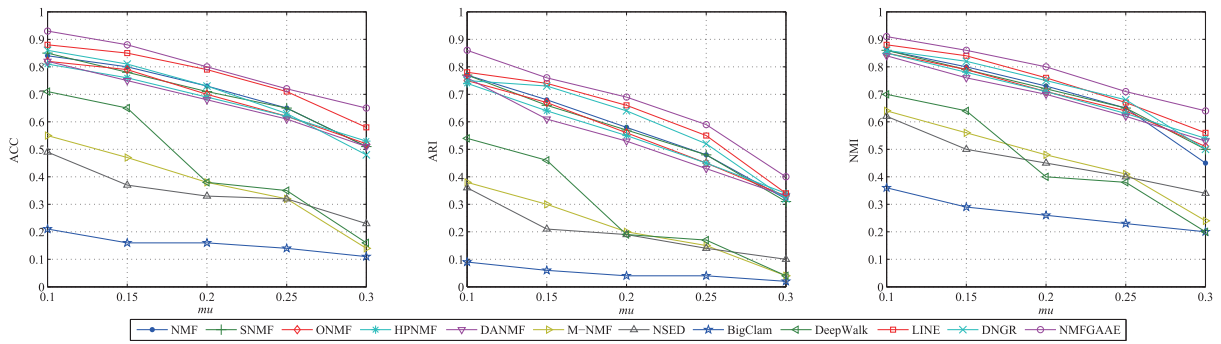


Fig. 3. Performance comparison on synthetic networks with different  $\mu$  in terms of ACC (Left), ARI (Middle), and NMI (Right).

methods, the experiments are repeated for 10 times and the average results are reported.

#### 4.4 Experiments on Synthetic Networks

1) *Datasets.* We utilize the well-known LFR tool [73] to generate benchmark synthetic networks with ground-truth communities. The setting of parameters of LFR networks is shown in Table 2. By respectively varying  $n$  and  $\mu$ , and fixing the other parameters, we generate 10 synthetic networks to test the performance of community detection. Because we only focus on non-overlapping community detection,  $on$  and  $om$  are both set to be 0.

2) *Experiment Results and Analysis.* In our experiments, we first generate 5 synthetic networks by varying  $n$  from 1,000 to 5,000 with step size 1,000 and fixing  $\mu$  at 0.3. For all methods,  $k$  is set to be the number of ground-truth communities. The results on these networks are shown in Table 3. We can observe that our proposed method NMFGAAE performs better than other methods in any case. Specifically, compared with the best NMF-based baseline: HPNMF, the average ACC, ARI and NMI scores of NMFGAAE respectively improve by 10.4, 11 and 4.5 percent. Compared with the best network representation based baseline: LINE, the average ACC, ARI and NMI scores of NMFGAAE respectively improve by 7.2, 14.4 and 5.7 percent.

We then generate another 5 synthetic networks by varying  $\mu$  from 0.1 to 0.3 with step size 0.05 and fixing  $n$  at 3,000. The results on these networks are shown in Fig. 3. As we can see, when  $\mu$  increases, the performance of all methods tends to decrease. This is because the community structures become more and more obscure, which makes community detection more and more difficult. However, NMFGAAE still performs better than other methods at any  $\mu$ . Even when  $\mu = 0.3$  the scores of ACC, ARI and NMI of NMFGAAE are still competitive and satisfactory. These results show that NMFGAAE has better ability to detect communities from networks with more complicated structure.

#### 4.5 Experiments on Real-World Networks

1) *Datasets.* In this part, we select 6 real-world networks without ground-truth communities, including WebKB, Cora, Citeseer, Polblog, Blogcatalog and Pubmed. These networks are from LINQS data repository.<sup>1</sup> For the sake of simplicity, the number of communities  $k$  on each network is set to be the number of node labels. The basic information of real-world networks is shown in Table 4.

1. <https://linqs.soe.ucsc.edu/data>



TABLE 4  
Statistics of Real-World Networks

Networks	$n$	$ E $	$k$
WebKB	877	1,388	4
Cora	2,708	5,278	7
Citeseer	3,327	4,552	6
Polblog	1,490	16,715	2
Blogcatalog	10,312	333,983	39
Pubmed	19,717	44,324	3

2) *Experiment Results and Analysis.* We run every method on the real-world networks. Since these networks have no ground-truth communities, we select modularity Q metric to evaluate the performance. The results are shown in Table 5. Although NMFGAAE achieves the same performance as most other methods on Polblog network, it performs the best on other networks and even exceed by a certain margin when comparing with NMF-based baselines. For example, on Citeseer network, the modularity Q scores of NMFGAAE are respectively 22, 18, 24.1, 10.8, 10.8, 14.3, 22 and 24.1 percent higher than those of NMF, SNMF, ONMF, HPNMF, DANMF, M-NMF, NSED and BigClam. Similar results can be found on WebKB, Cora and Pubmed networks. These results verify again that NMFGAAE indeed can boost the performance of NMF-based methods for community detection.

#### 4.6 Attention Mechanism Comparison

In NMFGAAE, we adopt specially designed attention mechanism directed by NMF-based community detection (Fig. 2a) instead of GAT mechanism (Fig. 2b). Results on synthetic and real-world networks have demonstrated its effectiveness. To further validate its superiority, we conduct comparative experiments between these two attention mechanisms on real-world networks. For convenience, we call the method using GAT attention mechanism as NMFGAT. The experiments results are shown in Fig. 4.

As we can see from Fig. 4, NMFGAAE performs better than NMFGAT on each network. The score of modularity Q of NMFGAAE increases by an average of 6.1 percent. This means that our proposed attention mechanism can better improve the performance of NMFGAAE than GAT attention mechanism.

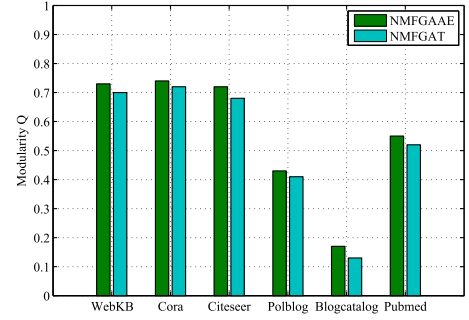


Fig. 4. NMFGAAE versus NMFGAT.

#### 4.7 Joint Optimization Analysis

In NMFGAAE, we jointly optimize GAAE and NMF models, which can make GAAE and NMF benefit from each other. One of the most obvious intuitions is that the output  $\mathbf{Z}$  of GAAE will be more discriminative, because it is improved by NMF community detection model simultaneously. To verify this intuition visually, we respectively run NMFGAAE on real-world networks, and apply t-SNE tool [74] to visualize  $\mathbf{Z}$  in two-dimensional space. As a comparison, we run GAAE individually (i.e., it is not jointly optimized with NMF), and also visualize its output  $\mathbf{Z}$ . Note that here we train GAAE only using the graph reconstruction loss. Because the results of different networks have similar trends, here we just show two networks with fewer communities: WebKB and Polblog in Figs. 5 and 6, respectively.

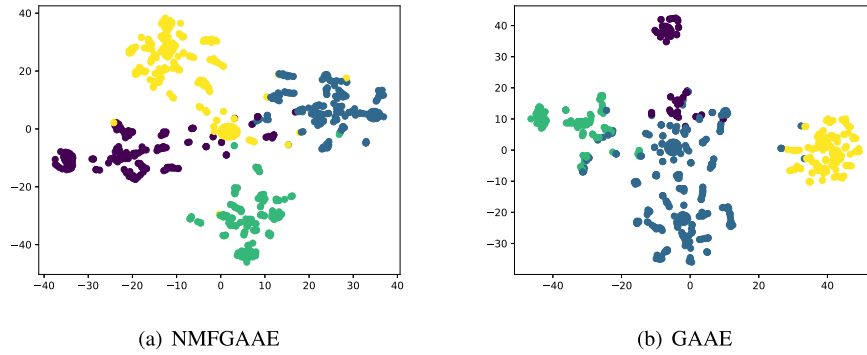
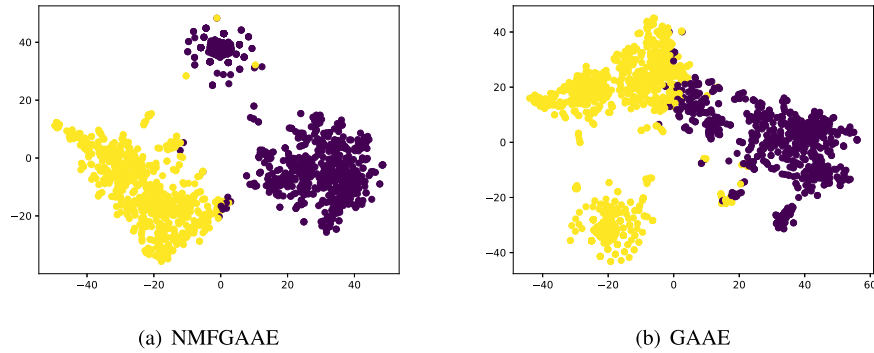
From Figs. 5 and 6, we can observe that the visualization results of  $\mathbf{Z}$  in NMFGAAE both have shown clear boundaries of clusters (i.e., communities), but those in GAAE are not distinguishable: points with different color are extensively mixed together. Bad node representations will deteriorate the performance of community detection. By applying NMF to  $\mathbf{Z}$  of GAAE, the modularity Q scores on WebKB and Polblog are 0.55 and 0.36, respectively, which are both inferior to those of NMFGAAE as shown in Table 5. These all fully demonstrate that optimizing GAAE and NMF jointly indeed can obtain better performance.

#### 4.8 Parameter $\lambda$ Sensitivity Analysis

There is one hyperparameter  $\lambda$  in the loss function of our NMFGAAE model. It is used to trade off GAAE reconstruction loss and NMF loss. To study its effects to the performance of NMFGAAE under different settings, we vary  $\lambda$  in

TABLE 5  
Performance Comparison on Real-World Networks in Terms of Modularity Q (Bold Numbers Denote the Best Results)

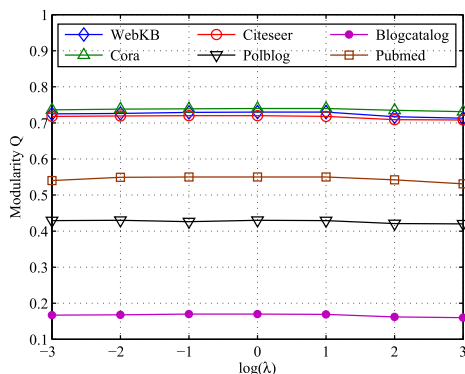
Networks	NMF	SNMF	ONMF	HPNMF	DANMF	M-NMF	NSED	BigClam	DeepWalk	LINE	DNGR	GCN	GraphSAGE	NMFGAAE
WebKB	0.57	0.63	0.64	0.68	0.35	0.62	0.66	0.30	0.61	0.45	0.58	0.44	0.48	<b>0.73</b>
Cora	0.53	0.52	0.62	0.71	0.71	0.70	0.69	0.47	0.65	0.61	0.71	0.62	0.53	<b>0.74</b>
Citeseer	0.59	0.61	0.58	0.65	0.65	0.63	0.59	0.58	0.49	0.67	0.65	0.68	0.61	<b>0.72</b>
Polblog	<b>0.43</b>	<b>0.43</b>	<b>0.43</b>	<b>0.43</b>	0.42	<b>0.43</b>	<b>0.43</b>	0.05	<b>0.43</b>	<b>0.43</b>	0.41	<b>0.43</b>	0.41	<b>0.43</b>
Blogcatalog	0.07	0.13	0.06	0.13	0.07	0.15	0.15	0.01	0.06	0.15	0.12	0.07	0.02	<b>0.17</b>
Pubmed	0.45	0.42	0.50	0.53	0.52	0.51	0.39	0.53	0.32	0.53	0.51	0.45	0.36	<b>0.55</b>

Fig. 5. 2D visualization of  $Z$  on WebKB.Fig. 6. 2D visualization of  $Z$  on Polblog.

the range of  $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$  and evaluate the corresponding performance on real-world networks. The results are shown in Fig. 7. As we can see, although there is degradation of performance as  $\lambda \geq 10$ , the degradation is mild and trivial. On the whole, when  $\lambda$  increases, NMFGAAE achieves consistent good performance. This shows that NMFGAAE is not sensitive to the exact value of  $\lambda$  and it can obtain satisfactory performance for a range of  $\lambda$ , e.g.,  $[10^{-3}, 10^1]$ . In all of our experiments, we equally treat the contributions of GAAE reconstruction loss and NMF loss, and thus set  $\lambda$  to be 1.

#### 4.9 Running Efficiency Analysis

As described in Algorithm 1, the results of community detection can be obtained by executing update rules iteratively. To investigate how many iterations NMFGAAE needs to take to obtain satisfactory results, we run

Fig. 7. The performance of NMFGAAE with different  $\lambda$ .

NMFGAAE on each real-world network and explore how performance varies with the number of iterations. To better illustrate this problem, we select the best NMF-based baseline HPNMF as the competitor. The results are depicted in Fig. 8. As we can see, on every network NMFGAAE only needs to take about 10 iterations to obtain stable and satisfactory results, but HPNMF needs at least 30 iterations, and even more than 100 iterations on Blogcatalog and PubMed networks. Actually, most existing NMF-based methods for community detection need to take many iterations to get the best results on these networks. However, nonlinear NMFGAAE utilizes graph attention auto-encoder to significantly reduce the number of iterations.

To further analyze the running efficiency of NMFGAAE, we specially compare it with NMF-based baselines in term of the runtime on two larger networks: Blogcatalog and Pubmed. In the previous experiments, we have found that DANMF is very time consuming, due to its multiple factorization structure. Therefore, to better illustrate the comparison results, in this experiment we do not select DANMF as the competitor. We set the number of iterations for NMFGAAE on Blogcatalog and Pubmed networks to be 10, and that of NMF-based baselines to be 100. Experiment results are shown in Fig. 9.

As we can see from Fig. 9, in terms of runtime, on Blogcatalog network, NMFGAAE is inferior to NMF, SNMF, NSED and BigClam, but performs better than ONMF, HPNMF and M-NMF. On PubMed network, NMFGAAE performs the best. Overall, NMFGAAE is very efficient, because it requires fewer iterations. On Blogcatalog and PubMed networks, it only takes about 100 and 800 seconds to converge to obtain satisfactory results, respectively.

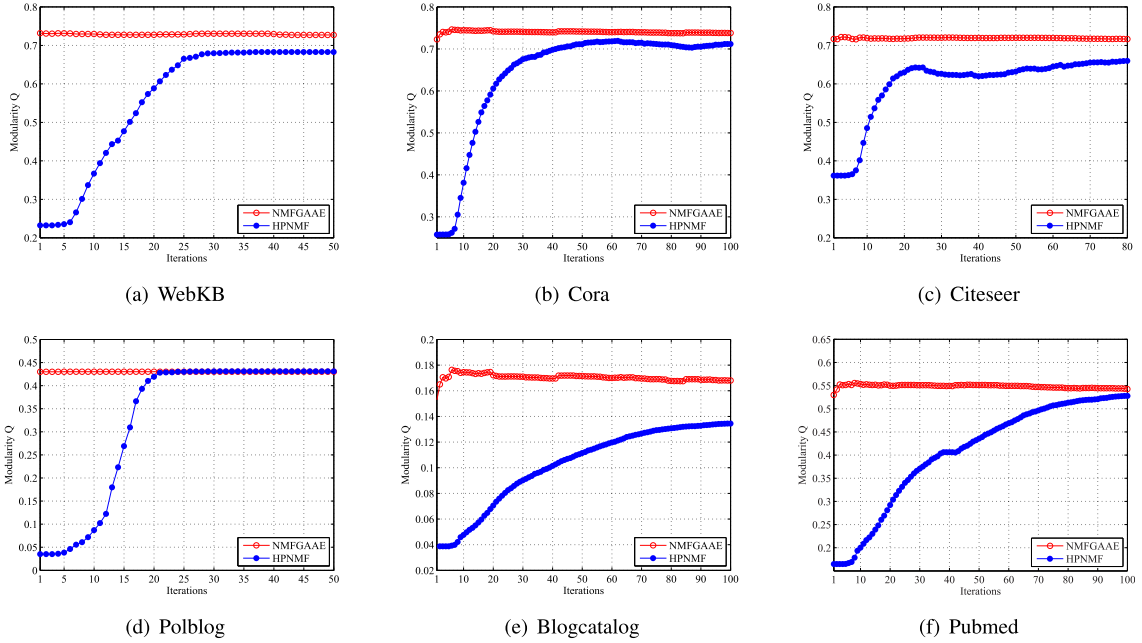


Fig. 8. The performance with respect to iterations on real-world networks.

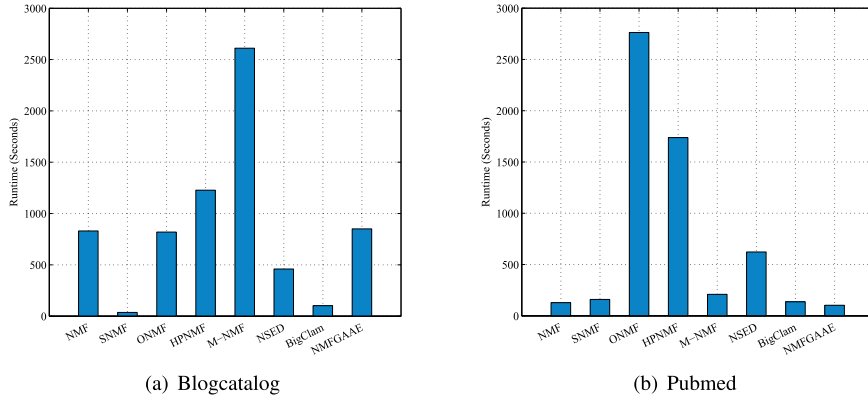


Fig. 9. Runtime comparison.

Besides, as we analyze in Section 3.6, the time cost of NMFGAAE is largely dominated by the number of edges  $|E|$ . This also shows that NMFGAAE can be expected to run faster on networks with fewer edges. As we can see, NMFGAAE run faster on PubMed than on Blogcatalog, because the number of edges on Blogcatalog network is about 7 times more than that of PubMed network.

## 5 CONCLUSION

NMF-based methods are widely applied to deal with the problem of community detection in complex networks. However, most of them are linear and still need to further boost the performance, especially when facing networks with diversified structure information. In this paper, we devise a nonlinear NMF-based method named NMFGAAE, which aims to combine NMF community detection and graph attention auto-encoder models to achieve better performance. We conduct extensive experiments on synthetic

and real-world networks. The results show that NMFGAAE is effective and efficient, and consistently performs better than state-of-the-art NMF-based community detection methods. In the future, we think there are two related topics which are worth further study:

- In essence, NMFGAAE provides us a general framework to boost the performance of NMF-based community detection by incorporating GNNs, and its two main components NMF and GAAE can be replaced with other advanced NMF and GNN models, respectively. There should be more advanced combinations that can be expected to obtain better performance.
- NMF-based community detection in attributed networks is also popular, but NMFGAAE here focuses on networks with only structure information. Therefore, extending NMFGAAE to attributed networks is promising. This leads to the study of how to

effectively fuse attributed information with structure information and how to learn the relevances of attributes to the communities.

## ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 62077045, Grant U1811263 and Grant 61772211, in part by the Humanity and Social Science Youth Foundation of Ministry of Education of China under Grant 19YJCZH049, in part by the Natural Science Foundation of Guangdong Province of China under Grant 2019A1515011292, and in part by the Science and Technology Support Program of Guangzhou City of China under Grant 201905010006.

## REFERENCES

- [1] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rep.*, vol. 659, pp. 1–44, 2016.
- [2] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, "Community detection in networks: A multidisciplinary review," *J. Netw. Comput. Appl.*, vol. 108, pp. 87–111, 2018.
- [3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.
- [4] G. Zhao, M. L. Lee, W. Hsu, W. Chen, and H. J. Hu, "Community-based user recommendation in uni-directional social networks," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 189–198.
- [5] Y. C. Chang, K. T. Lai, S. C. T. Chou, and M. Chen, "Mining the networks of telecommunication fraud groups using social network analysis," in *Proc. 9th IEEE/ACM Int. Conf. Advances Soc. Netw. Anal. Mining*, 2017, pp. 1128–1131.
- [6] A. P. Rodríguez, C. O. Gómez, and F. M. Anegón, "Detecting, identifying and visualizing research groups in co-authorship networks," *Scientometrics*, vol. 82, pp. 307–319, 2010.
- [7] S. S. Bhowmick and B. S. Seah, "Clustering and summarizing protein-protein interaction networks: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 638–658, Mar. 2016.
- [8] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
- [9] M. E. J. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Physical Rev. E*, vol. 94, 2016, Art. no. 052315.
- [10] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, pp. 1–86, 2018.
- [11] A. Jonnalagadda and L. Kuppusamy, "A survey on game theoretic models for community detection in social networks," *Soc. Netw. Anal. Mining*, vol. 6, no. 1, pp. 1–24, 2016.
- [12] S. E. Garza and S. E. Schaeffer, "Community detection with the label propagation algorithm: A survey," *Physica A, Statist. Mechanics Appl.*, vol. 534, 2019, Art. no. 122058.
- [13] F. Z. Liu et al., "Deep learning for community detection: Progress, challenges and opportunities," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 4981–4987.
- [14] I. Psorakis, S. Roberts, M. Ebdon, and B. Sheldon, "Overlapping community detection using Bayesian non-negative matrix factorization," *Physical Rev. E*, vol. 83, 2011, Art. no. 066114.
- [15] Y. Zhang and D. Y. Yeung, "Overlapping community detection via bounded nonnegative matrix tri-factorization," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 606–614.
- [16] F. H. Ye, C. Chen, Z. B. Zheng, R. H. Li, and J. X. Yu, "Discrete overlapping community detection with pseudo supervision," in *Proc. 19th IEEE Int. Conf. Data Mining*, 2019, pp. 708–717.
- [17] X. Liu, W. Wang, D. He, P. Jiao, D. Jin, and C. V. Cannistraci, "Semi-supervised community detection based on non-negative matrix factorization with node popularity," *Inf. Sci.*, vol. 381, pp. 304–321, 2017.
- [18] L. Yang, X. C. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2585–2598, Nov. 2015.
- [19] F. Wang, T. Li, X. Wang, S. H. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining Knowl. Discov.*, vol. 22, no. 3, pp. 493–521, 2011.
- [20] X. Luo, Z. G. Liu, L. Jin, Y. Zhou, and M. C. Zhou, "Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2020.3041360.
- [21] Y. Li, C. F. Sha, X. Huang, and Y. C. Zhang, "Community detection in attributed graphs: An embedding approach," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 338–345.
- [22] Z. C. Huang, Y. M. Ye, X. T. Li, F. Liu, and H. J. Chen, "Joint weighted nonnegative matrix factorization for mining attributed graphs," in *Proc. 21st Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2017, pp. 368–380.
- [23] C. B. He et al., "Similarity preserving overlapping community detection in signed networks," *Future Gener. Comput. Syst.*, vol. 116, pp. 275–290, 2021.
- [24] J. Y. Shi, K. T. Mao, H. Yu, and S. M. Yiu, "Detecting drug communities and predicting comprehensive drug-drug interactions via balance regularized semi-nonnegative matrix factorization," *J. Cheminformatics*, vol. 11, no. 1, pp. 1–16, 2019.
- [25] V. Gligorijević, Y. Panagakis, and S. Zafeiriou, "Non-negative matrix factorizations for multiplex network analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 928–940, Apr. 2019.
- [26] X. K. Ma, D. Dong, and Q. Wang, "Community detection in multi-layer networks using joint nonnegative matrix factorization," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 273–286, Feb. 2019.
- [27] X. K. Ma and D. Dong, "Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1045–1058, May 2017.
- [28] X. K. Ma, B. H. Zhang, C. Z. Ma, and Z. Y. Ma, "Co-regularized nonnegative matrix factorization for evolving community detection in dynamic networks," *Inf. Sci.*, vol. 528, pp. 265–279, 2020.
- [29] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.
- [30] C. B. He, X. Fei, H. C. Li, Y. Tang, H. Liu, and S. Y. Liu, "Improving NMF-based community discovery using distributed robust nonnegative matrix factorization with SimRank similarity measure," *J. Supercomput.*, vol. 74, no. 10, pp. 5601–5624, 2018.
- [31] L. Yang, X. H. Cao, D. X. He, C. Wang, X. Wang, and W. X. Zhang, "Modularity based community detection with deep learning," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 2252–2258.
- [32] R. Ibrahim and D. Gleich, "Nonlinear diffusion for community detection and semi-supervised learning," in *Proc. 30th World Wide Web Conf.*, 2019, pp. 739–750.
- [33] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 417–429, Mar. 2017.
- [34] J. Q. Li, G. X. Zhou, Y. N. Qiu, Y. J. Wang, Y. Zhang, and S. L. Xie, "Deep graph regularized non-negative matrix factorization for multi-view clustering," *Neurocomputing*, vol. 390, pp. 108–116, 2020.
- [35] F. H. Ye, C. Chen, and Z. B. Zheng, "Deep autoencoder-like non-negative matrix factorization for community detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1393–1402.
- [36] S. Maekawa, K. Takeuchi, and M. Onizuka, "Non-linear attributed graph clustering by symmetric NMF with PU learning," 2018, *arXiv:1810.00946*.
- [37] W. Rawat and Z. H. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 352–2449, 2017.
- [38] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.



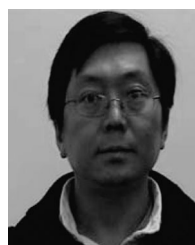
- [39] J. J. Zhang and C. Q. Zong, "Deep neural networks in machine translation: An overview," *IEEE Intell. Syst.*, vol. 30, no. 5, pp. 16–25, Sep./Oct. 2015.
- [40] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3861–3870.
- [41] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
- [42] S. Affeldt, L. Labiod, and M. Nadif, "Spectral clustering via ensemble deep autoencoder learning," *Pattern Recognit.*, vol. 108, 2020, Art. no. 107522.
- [43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–14.
- [44] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–12.
- [45] Z. H. Wu, S. Pan, F. W. Chen, G. D. Long, C. Q. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
- [46] Z. W. Zhang, P. Cui, and W. W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2020.2981333.
- [47] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.
- [48] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proc. 5th SIAM Int. Conf. Data Mining*, 2005, pp. 606–610.
- [49] B. J. Sun, H. W. Shen, J. H. Gao, W. T. Ouyang, and X. Q. Cheng, "A non-negative symmetric encoder-decoder approach for community detection," in *Proc. 26th ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 597–606.
- [50] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
- [51] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014, pp. 1–14.
- [52] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.
- [53] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 21st Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
- [54] K. Y. L. Xu, W. H. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–17.
- [55] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016, pp. 1–3.
- [56] D. Y. Bo, X. Wang, C. Shi, M. Q. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. 31st World Wide Web Conf.*, 2020, pp. 1400–1410.
- [57] C. Wang, S. R. Pan, R. Q. Hu, G. D. Long, J. Jiang, and C. Q. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3670–3676.
- [58] E. Min, X. F. Guo, Q. Liu, G. Zhang, J. J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.
- [59] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," 2018, *arXiv: 1801.07648*.
- [60] X. Peng, J. S. Feng, S. J. Xiao, W. T. Yau, J. T. Zhou, and S. F. Yang, "Structured autoencoders for subspace clustering," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5076–5086, Oct. 2018.
- [61] F. Tian, B. Gao, Q. Cui, E. H. Chen, and T. Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.
- [62] X. L. Li, H. Y. Zhang, and R. Zhang, "Embedding graph auto-encoder with joint clustering via adjacency sharing," 2020, *arXiv: 2002.08643*.
- [63] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
- [64] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
- [65] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning-graph representations," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.
- [66] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–13.
- [67] D. Cai, X. F. He, J. W. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.
- [68] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix tri-factorizations for clustering," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2006, pp. 126–135.
- [69] F. H. Ye, C. Chen, Z. Y. Wen, Z. B. Zheng, W. H. Chen, and Y. U. Zhou, "Homophily preserving community detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2903–2915, Aug. 2020.
- [70] X. Wang, P. Cui, J. Wang, J. Pei, W. W. Zhu, and S. Q. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.
- [71] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate club: An API oriented open-source python framework for unsupervised learning on graphs," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 3125–3132.
- [72] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, "Metrics for community analysis: A survey," *ACM Comput. Surv.*, vol. 50, no. 4, 2017, Art. no. 54.
- [73] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Rev. E*, vol. 78, no. 2, 2008, Art. no. 046110.
- [74] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.



**Chaobo He** received the BS, MS, and PhD degrees from South China Normal University, China, in 2004, 2007, and 2014, respectively. He is currently a professor with the Zhongkai University of Agriculture and Engineering, China, and is also a visiting scholar with the School of Computer Science, South China Normal University, China. His research interests are machine learning and social computing. He has published more than 30 papers on international journals and conferences.



**Yulong Zheng** received the BS degree from the School of Information Science and Technology, Zhongkai University of Agriculture and Engineering, China, in 2020. He is currently working toward the MSc degree with the School of Computer Science, South China Normal University, China. His research interests are machine learning and data mining.



**Xiang Fei** received the BS and PhD degrees from Southeast University, China, in 1992 and 1999, respectively. After graduation, he worked, as a postdoctoral research fellow, on a number of projects including European IST Programs and EPSRC. He is currently working as a senior lecturer for the School of Computing, Electronics and Maths, Coventry University, U.K. His current research interests include machine learning and data mining in cyber-physical systems.



**Hanchao Li** received the BS degree in mathematics from the University of Warwick, U.K., in 2013, and the MS degree in computing from Coventry University, U.K., in 2015. He is currently working toward the PhD degree with Coventry University, U.K., and worked on music information retrieval, i.e., data mining in music subject area. His research interests are big data, data mining, machine learning and any mathematics-related researches. He has published several conference and journal papers.



**Yong Tang** received the BS and MS degrees from Wuhan University, China, in 1985 and 1990, respectively, and the PhD degree from the University of Science and Technology of China, China, in 2001, all in computer science. He is currently a professor of the School of Computer Science, South China Normal University, China. His research interests are big data and collaborative computing. He has published more than 100 papers on international journals and conferences.

**Zeng Hu** received the BS and MS degrees from Xidian University, China, in 2008 and 2013, respectively. He is currently a lecturer with the School of Information Science and Technology, Zhongkai University of Agriculture and Engineering, China. His recent research interests include communication engineering and machine learning.

▷ **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/csdl](http://www.computer.org/csdl).**