Smoothness Sensor: Adaptive Smoothness-Transition Graph Convolutions for Attributed Graph Clustering

Chaojie Ji, Hongwei Chen, Ruxin Wang, Yunpeng Cai, and Hongyan Wu

Abstract—**Clustering techniques attempt to group objects with similar properties into a cluster. Clustering the nodes of an attributed graph, in which each node is associated with a set of feature attributes, has attracted significant attention. Graph convolutional networks (GCNs) represent an effective approach for integrating the two complementary factors of node attributes and structural information for attributed graph clustering. Smoothness is an indicator for assessing the degree of similarity of feature representations among nearby nodes in a graph. Oversmoothing in GCNs, caused by unnecessarily high orders of graph convolution, produces indistinguishable representations of nodes, such that the nodes in a graph tend to be grouped into fewer clusters, and pose a challenge due to the resulting performance drop. In this study, we propose a smoothness sensor for attributed graph clustering based on adaptive smoothness-transition graph convolutions, which senses the smoothness of a graph and adaptively terminates the current convolution once the smoothness is saturated to prevent oversmoothing. Furthermore, as an alternative to graph-level smoothness, a novel fine-grained nodewise-level assessment of smoothness is proposed, in which smoothness is computed in accordance with the neighborhood conditions of a given node at a certain order of graph convolution. In addition, a self-supervision criterion is designed considering both the tightness within clusters and the separation between clusters to guide the entire neural network training process. The experiments show that the proposed methods significantly outperform 13 other state-of-the-art baselines in terms of different metrics across five benchmark datasets. In addition, an extensive study reveals the reasons for their effectiveness and efficiency.**

Index Terms—**Adaptive graph convolutions, attributed graph clustering, graph neural networks, smoothness of graph signals.**

I. Introduction

Clustering techniques attempt to group objects with similar properties into a cluster [1], [2]. Various methods have been proposed to solve real-world problems via text [3] and image [4] clustering. Recently, with the emergence of graph-structured data, such as social networks and biological networks [5]–[7], the partitioning of the nodes of an attributed graph, in which each node is associated with a set of feature attributes, has attracted significant attention [8], [9]. For example, potential criminal organizations can be identified based on frequent contacts among known criminals [10]. In a graph, attributes represent the feature values of a vertex itself, while the structural information indicates the underlying similarity among graph nodes, including not only the relationships within a one-hop distance but also more complex relationships at higher order distances [11], [12]. The question of how to effectively integrate these two complementary factors of attributes and structural information for the task of clustering an attributed graph has attracted the interest of researchers.

Classical data clustering algorithms take the similarity matrix of node representations as the input [13]. Graph-structure-based approaches only exploit structural information to group nodes [14]. Although later research attempts to integrate both the node feature and network structure [15], [16], these methods explore deep representation learning less. Recently, deep-learning-related methods have been exploited to learn graph representations based on both node content and network structure information [17]. GCNs [18] are designed to naturally incorporate information on the nodes themselves and the relationships among nodes. Then, classical clustering techniques, for example, k-means and spectral clustering, can be stacked on the low-dimensional representations learned by deep learning networks.

Most of the existing methods rely on the application of fixed, usually shallow (low-order), graph convolutions. To capture structural information, a structural deep clustering network (SDCN) [19] was proposed with three layers of an autoencoder and a GCN. Wang *et al.* [20] proposed marginalized graph autoencoders (MGAEs), which increased the possible number of convolutional layers to three.

Manuscript received September 3, 2020; revised March 16, 2021; accepted May 31, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant U1913210; in part by the Shenzhen Science and Technology Program under Grant KQTD20190929172835662; in part by the Strategic Priority CAS Project under Grant XDB38050100; and in part by the Shenzhen Basic Research Projects under Grant JCYJ20180703145002040 and Grant JCYJ20180703145202065. This article was recommended by Associate Editor P. P. Angelov. *(Chaojie Ji and Hongwei Chen contributed equally to this work.) (Corresponding authors: Yunpeng Cai; Hongyan Wu.)*

Chaojie Ji, Ruxin Wang, Yunpeng Cai, and Hongyan Wu are with the Joint Engineering Research Center for Health Big Data Intelligent Analysis Technology, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: cj.ji@siat.ac.cn; rx.wang@siat.ac.cn; yp.cai@siat.ac.cn; hy.wu@siat.ac.cn).

Hongwei Chen is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: hw.chen@siat.ac.cn).

Digital Object Identifier 10.1109/TCYB.2021.3088880

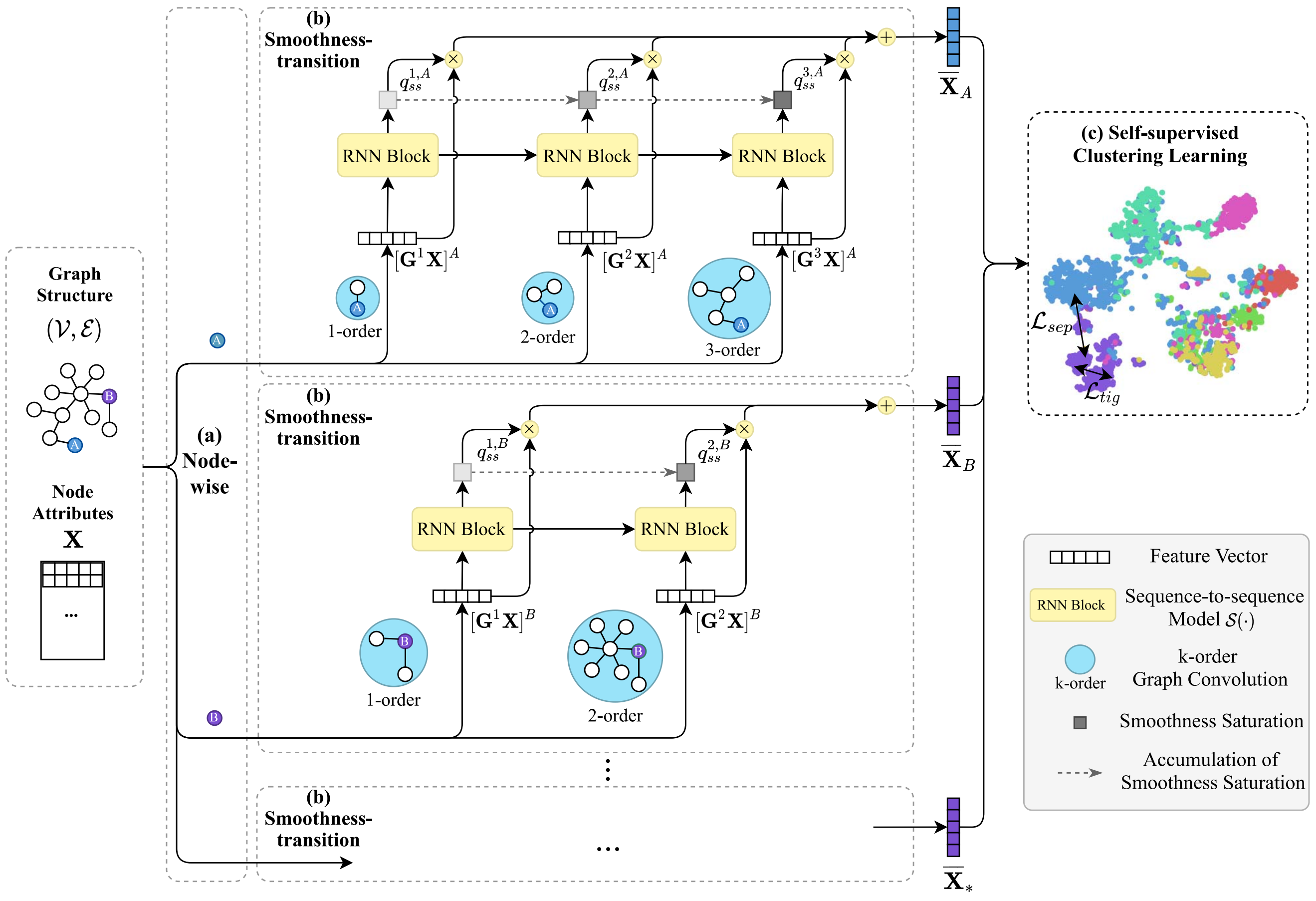

Graph Structure
$(\mathcal{V}, \mathcal{E})$
Node Attributes
$\mathbf{X}$
(a) Node-wise
(b) Smoothness-transition
RNN Block
$q_{ss}^{1,A}$
$q_{ss}^{2,A}$
$q_{ss}^{3,A}$
$[\mathbf{G}^1\mathbf{X}]^A$
$[\mathbf{G}^2\mathbf{X}]^A$
$[\mathbf{G}^3\mathbf{X}]^A$
1-order
2-order
3-order
$\overline{\mathbf{X}}_A$
$q_{ss}^{1,B}$
$q_{ss}^{2,B}$
$[\mathbf{G}^1\mathbf{X}]^B$
$[\mathbf{G}^2\mathbf{X}]^B$
$\overline{\mathbf{X}}_B$
$\overline{\mathbf{X}}_*$
(c) Self-supervised Clustering Learning
$\mathcal{L}_{sep}$
$\mathcal{L}_{tig}$
Feature Vector
Sequence-to-sequence Model $\mathcal{S}(\cdot)$
k-order Graph Convolution
Smoothness Saturation
Accumulation of Smoothness Saturation

Fig. 1. Framework of the proposed smoothness sensor of NAS-GC. (a) Perform convolution based on the node level and detect the fine-grained saturation of smoothness. (b) Core smoothness-transition component operates to quantify the saturation of smoothness in the form of standard sequence-to-sequence prediction, considering a gradual smoothness process. (c) Once clustering partitioning is achieved by applying the k-means algorithm to the learned representations, the proposed self-supervised clustering strategy is employed to each node pair in accordance with both the tightness within clusters and the separation between clusters.

Smoothness is an indicator for assessing the degree of similarity of feature representations among nearby nodes in a graph. With higher orders of graph convolution, smoother filtered graph signals are obtained. However, graph convolution with an excessively large order k results in oversmooth node representations. Oversmoothing has been identified as a major cause of performance degradation in deep graph convolutional networks and the downstream tasks thereafter [21].

Specifying a graph-level order of graph convolution is a general practice in GCNs. However, the node density in the entire graph can vary greatly in attributed graphs. A relatively isolated node with few neighbors could require a larger k (stronger smoothing) to introduce more distant nodes to obtain more information, while a node with more neighbors usually can efficiently gather information with a smaller k (weaker smoothing). Thus, fixed k-order graph convolution at the graph level is a coarse solution.

In this study, to address the problem of smoothness, we propose an adaptive smoothness-transition graph convolution method for attributed graph clustering as Fig. 1, which operates like a *smoothness sensor*. In particular, the smoothness is sensed at the fine-grained node level instead of at the level of the entire graph. Finally, a self-supervision strategy is designed to enable the training of our proposed model with respect to various graph structures.

Smoothness-Transition Adaptivity: To overcome the oversmoothing in GCNs, it is necessary to adaptively customize the order k of graph convolution. We explore how k-order filtered graph signals are transited from $(k-1)$-order filtered signals, and we define an assessment of smoothness saturation related to iterative graph convolution operations as a standard sequence-to-sequence prediction problem, in which the saturation of smoothness is taken as an indicator of whether the convolution process should be terminated as shown in Fig. 1(b).

Fine-Grained Smoothness: To further detect the saturation of smoothness at a fine-grained level that can adaptively consider the surrounding environment of each node, we first introduce a preliminary model—adaptive smoothness-transition graph convolution (AS-GC)—for detecting graph-wise smoothness and then evolve it into a mature version based on nodewise smoothness, that is, nodewise adaptive smoothness-transition graph convolution (NAS-GC), as illustrated in Fig. 1(a).

Self-Supervised Clustering Criterion: Clustering is a typical unsupervised learning problem. To adapt traditional GCNs, which requires semisupervision, to clustering scenarios, we propose a complete clustering criterion that can directly consider both the tightness within clusters and the separation between clusters to guide the self-supervised learning process of NAS-GC (AS-GC). It is integrated as a component of self-supervised clustering learning shown in Fig. 1(c).

We conduct a series of experiments to investigate the proposed approaches. The experiments prove that the proposed methods significantly outperform 13 other state-of-the-art baselines in terms of popular metrics across five benchmark datasets. Our extensive study shows the effectiveness and efficiency of our proposed methods and further reveals their implicit mechanisms.

The remainder of this article is organized as follows. Section II reviews the related work of attributed graph clustering in terms of machine learning and deep learning. We elaborate on the details of the procedure of evolution and details of the proposed AS-GC and NAS-GC in Section III. In Section IV, experimental results are given for evaluating the proposed models with respect to effectiveness and efficiency, and an extensive study is conducted to reveal the intrinsic mechanism of our methods. Finally, conclusion and future work are given in Section V.

II. Related Work

Our work focuses on attributed graphs, in which every node is associated with a set of feature attributes and the nodes are connected to each other [22], [23]. Then, these nodes are clustered in accordance with both their feature attributes and the structural information of the graph [24]. Related methods can be mainly categorized into two classes—1) machine-learning- and 2) deep-learning-based representations.

A. Graph Clustering Through Machine Learning

A series of classical methods based on node features has been proposed. Although the graph structure, in some cases, is not provided, a similarity matrix can be naturally constructed based on the node features. The *k*-means algorithm can be run directly on the given graph structure or on the constructed similarity matrix to obtain clustering results. Eigenvalue decomposition is introduced through a normalized graph Laplacian matrix, and then, eigenvectors with relatively small eigenvalues are fed into a clustering algorithm to obtain clusters [13], [25]. Newman [14] further used Laplacian eigenmaps to group nodes with a higher-than-average density of edges. In addition to eigenvalue decomposition, Girvan and Newman [26] proposed a method in which centrality indices are used to locate cluster boundaries. Wang *et al.* [27] exploited first-order and second-order proximity to jointly preserve the global and local structures of a network. Hastings [29] resorted to the techniques of belief propagation [28], observing that a graph has a low density of loops. By combining both node features and graph structure information, a non-negative matrix factorization method has also been proposed in which the node attribution matrix is decomposed and the graph structure is utilized to construct regularization terms [15], [16]. Because DeepWalk is considered equivalent to matrix factorization, Yang *et al.* [30] proposed text-associated DeepWalk (TADW) to incorporate text features of nodes into network representation learning within the matrix factorization framework. It is observed that there may exist considerable noise in real-world data, especially in each view of them [31], [32]. To avoid blindly combining the information from multiview data with potential noise, Xia *et al.* [33] proposed robust multiview spectral clustering, in which each transition probability matrix with respect to an individual view is decomposed into two parts: 1) a shared latent transition probability matrix and 2) a deviation error matrix that encodes the noise.

B. Deep Representations for Graph Clustering

With its powerful representation capabilities [34], deep learning has been widely applied in the graph domain [35], [36]. We intuitively split the deep learning methods used in this field into two main categories—1) autoencoders and 2) GCNs [37]. The autoencoders are usually designed as unsupervised learning models. Tian *et al.* [38] first proposed GraphEncoder, which learns a nonlinear embedding of the original graph by means of stacked autoencoders. This method differs from the singular value decomposition-based dimension reduction method in which the original representation space is merely projected into a new space with a lower rank through linear projection. Later, a random surfing model, called deep neural networks for graph representation (DNGR), combined with stacked denoising autoencoders was designed [39]. Another class of graph clustering methods is based on GCNs, in which node representations can be updated by aggregating messages from neighboring vertices. Graph autoencoders (GAEs) and variational graph autoencoders (VGAEs) apply GCNs to encode representations and an inner product decoder for learning [40]. To handle sparse data, second-order proximity (local pairwise second-order similarity within one hop) was exploited in [27] to preserve both global and local network structures through a deep structural network embedding method. By combining autoencoders and GCNs, Wang *et al.* [41] also proposed a goal-directed deep attentional autoencoder that simultaneously learns the importance of neighboring nodes and soft labels from the graph embedding itself.

Although deep learning techniques have been widely applied, the aforementioned methods can merely aggregate information from neighbors within a limited distance. To address this shortcoming, the SDCN was developed by designing a delivery action to construct a connection between an autoencoder and a GCN with three layers, combining both low-order and high-order information [19]. MGAEs were developed in an attempt to extend autoencoders to deep convolutions to learn more effective representations [20]. However, the number of layers in an MGAE is still limited to three. AGC was recently proposed to support higher order graph convolution with a novel low-pass graph filter, which utilizes the smoothness of the graph signals [42]. Our method, NAS-GC (AS-GC), is closely related to deep approaches of

TABLE I
IMPORTANT NOTATIONS USED IN THIS ARTICLE

Notation	Short explanation
$\boldsymbol{G}$	Frequency response function of a filter
$\boldsymbol{G}^k\boldsymbol{X}$	k-order graph convolution on the initial feature $\boldsymbol{X}$
$\boldsymbol{s}^k$	Smoothness state when k-order graph convolution is conducted
$p_{st}^{\mathcal{G}}(\cdot)$	A state transition (st) model for calculating the smoothness of an entire graph ($\mathcal{G}$)
q_{ss}^k	Smoothness saturation (ss) of k-order graph convolution
$\mathcal{K}$	Selected order of graph convolution for the graph
$\overline{\boldsymbol{X}}$	Updated representation of all nodes
$\boldsymbol{s}_i^k$	Smoothness state of node i when k-order graph convolution is conducted
$p_{st}^{\mathcal{V}}(\cdot)$	A state transition (st) model for calculating the smoothness of an individual node ($\mathcal{V}$)
$q_{ss}^{k,i}$	Smoothness saturation (ss) of node i with k-order graph convolution
$\mathcal{K}_i$	Selected order of graph convolution for node i
$\overline{\boldsymbol{x}}_i$	Updated representation of node i

this kind. Concretely, based on the smoothness saturation of attributed graphs and the surrounding environment of each node, we propose a smoothness sensor that can adaptively choose the appropriate order of graph convolution and a fine-grained nodewise mechanism for every vertex, along with an effective self-supervision criterion.

III. METHOD

A. Preliminaries

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, $\mathcal{V} = \{v_1, \dots, v_n\}$ is the set consisting of all nodes and $\mathcal{E}$ is the edge set that can be represented as an adjacency matrix $\boldsymbol{A}$. $\boldsymbol{X}$ is the feature matrix $[\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n\times m}$, where $\boldsymbol{x}_i$ denotes the feature representation of vertex v_i, and m represents the number of features. Our goal is to assign each node to a cluster. All possible clusters are collected in a set $\mathcal{C} = \{c_1, \dots, c_r\}$, where r is the number of candidate clusters. In addition, we define two operations that will be frequently used in our paper. $[\boldsymbol{X}]_j$ denotes the collection of the jth column in matrix $\boldsymbol{X}$, while $[\boldsymbol{X}]^j$ represents the extraction of the jth row. Table I lists some important notations that will be used throughout the rest of this article.

B. Smoothness of Graph Signals

We first introduce some basic notations. Given the adjacency matrix $\boldsymbol{A}$ of graph $\mathcal{G}$, the degree matrix and the graph Laplacian can be expressed as $\boldsymbol{D} = \mathrm{diag}(d_1, \dots, d_n)$ and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, respectively. This Laplacian can be eigen-decomposed as $\boldsymbol{U\Lambda U}^{-1}$, where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ is a diagonal matrix of the eigenvalues and $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_n]$ is the matrix of the corresponding eigenvectors. Moreover, the symmetrically normalized graph Laplacian is $\boldsymbol{L}_s = \boldsymbol{I} - \boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)}$, which can also be eigen decomposed in the same way as $\boldsymbol{L}$.

A *graph signal* can be represented as a vector $\boldsymbol{f} = [f(v_1), \dots, f(v_n)]^T$, where $f : \mathcal{V} \to \mathbb{R}$ is a real-valued function on the nodes of a graph. The input feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ can be split into m individual graph signals, where the column $[\boldsymbol{X}]_j$ corresponds to the jth signal.

Equipped with the Laplacian matrix, the smoothness of a graph signal $\boldsymbol{f}$ can be quantified by the graph Laplacian quadratic form [43] as

$$\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} = \frac{1}{2}\sum_{(v_i,v_j)\in\mathcal{E}} \big(f(v_i) - f(v_j)\big)^2. \tag{1}$$

It shows that smoother signals tend to assign similar values for connected nodes.

A graph signal can be decomposed into a linear combination of the eigenvectors [43]

$$\boldsymbol{f} = \sum_{i=1}^{n} e_i \boldsymbol{u}_i \tag{2}$$

where e_i is the coefficient of $\boldsymbol{u}_i$ for the graph signal, and the magnitude of the coefficient $|e_i|$ is proportional to the strength of the basis signal $\boldsymbol{u}_i$.

Then, the smoothness of a basis signal $\boldsymbol{u}_q$ can be calculated using the Laplacian–Beltrami operator [44]

$$\Omega(\boldsymbol{u}_q) = \frac{1}{2}\sum_{(v_i,v_j)\in\mathcal{E}} a_{i,j}\left\| \frac{\boldsymbol{u}_q^i}{\sqrt{d_i}} - \frac{\boldsymbol{u}_q^j}{\sqrt{d_j}} \right\|_2^2 = \boldsymbol{u}_q^T\boldsymbol{L}_s\boldsymbol{u}_q = \lambda_q \tag{3}$$

where u_q^i denotes the ith element of the eigenvector $\boldsymbol{u}_q$, and $a_{i,j}$ is the element located in the ith row and jth column of the adjacency matrix $\boldsymbol{A}$. It can be observed that the smoothness of a basis signal is equivalent to the corresponding eigenvalue, with smaller eigenvalues indicating smoother basis signals. We thus refer to signals with small eigenvalues as low-frequency (smoother) signals.

C. Nodewise Adaptive Smoothness-Transition Graph Convolution

In this section, we seek a graph convolutional approach to obtain better graph representations for clustering. We first propose a naive but intuitive prototype and gradually evolve it into its mature version, NAS-GC. The entire framework of NAS-GC is illustrated in Fig. 1.

1) Evolutionary Process of the Overall Objective of the Smoothness Sensor:

a) Order-fixed graph convolution: Smooth graph signals tend to cause nearby nodes to have similar representations, which is consistent with the way a cluster in a graph tends to be composed of adjacent nodes. Lower frequency signals correspond to smoother graph signals and, thus, help to form the nodes in an attributed graph into clusters more easily.

A low-pass graph filter is a function for producing low-frequency basis signals from relatively high-frequency signals for various downstream tasks. We first define a frequency response function for a filter as follows: $\boldsymbol{G} = \boldsymbol{U}p(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}$, where $p(\boldsymbol{\Lambda}) = \mathrm{diag}(p(\lambda_1), \dots, p(\lambda_n))$.

Given a specified graph filter, we can execute a first-order graph convolution as follows:

$$\left[\overline{\boldsymbol{X}}\right]_j = [\boldsymbol{G}\boldsymbol{X}]_j = \left[\boldsymbol{U}p(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}\boldsymbol{X}\right]_j = \sum_{i=1}^{n} p(\lambda_i)e_i\boldsymbol{u}_i \tag{4}$$

where $[\overline{\boldsymbol{X}}]_j$ is the filtered version of the jth graph signal from $\boldsymbol{X}$. $p(\lambda_i)$ is assigned to preserve low-frequency basis signals and remove high-frequency ones by scaling the values of e_i.

When a first-order convolution is conducted, the representation of every node is updated by aggregating its 1-hop neighbors. In this way, the information from long-distance neighbors is discarded, which could lead to severe problems in a large but highly sparse graph, resulting in an undersmooth graph signal. To alleviate this problem, the concept of k-order graph convolution is introduced.

We formulate the k-order graph convolution process as follows:

$$\left[\overline{\boldsymbol{X}}\right]_j = \left[\boldsymbol{G}^k\boldsymbol{X}\right]_j = \left[\boldsymbol{U}p(\boldsymbol{\Lambda})^k\boldsymbol{U}^{-1}\boldsymbol{X}\right]_j. \tag{5}$$

Finally, with a predefined k, the filtered graph signals can be fed to a downstream algorithm to perform clustering. $[\boldsymbol{G}^k\boldsymbol{X}]_j$ represents the jth filtered graph signal under k-order convolution.

b) Adaptive graph convolution: Oversmooth graph signals can exert a clear negative effect on clustering by causing two vertices that should belong to distinct clusters to possess similar representations with limited discriminability. To achieve a balance between undersmoothness and oversmoothness, the selection of a suitable k for the convolution process is naturally critical for learning effective representations of the graph and, consequently, for ensuring good performance in downstream tasks, for example, attributed graph clustering.

A naive paradigm for solving this problem is to adopt a score function or probability maximization model. We first define a maximum order K. Then, graph signals filtered by graph convolutions of different orders are fed into this model one by one, and the corresponding levels of smoothness saturation are evaluated. We can choose the order with the maximum smoothness saturation as the optimal order for graph convolution as follows:

$$\mathcal{K} = \underset{k=1,\ldots,K}{\arg\max}\, p_{\text{agc}}\left(\boldsymbol{G}^k\boldsymbol{X}\right) \tag{6}$$

where $p_{\text{agc}}(\cdot)$ denotes a score function or probabilistic model to output the value of smoothness saturation associated with k-order graph convolution.

c) Adaptive smoothness-transition graph convolution: The above solution simply assumes that there is no relationship among the filtered graph signals obtained under different orders of graph convolution. Considering that the low-pass frequency response function $p(\cdot)$ should be constantly nonincreasing and non-negative for all the inputs, the smoothness of the filtered node features will monotonically increase with increasing k [42]. This can be expressed as follows:

$$\Omega\left(\frac{[\boldsymbol{G}^k\boldsymbol{X}]_j}{\left\|[\boldsymbol{G}^k\boldsymbol{X}]_j\right\|_2}\right) \leq \Omega\left(\frac{[\boldsymbol{G}^{k-1}\boldsymbol{X}]_j}{\left\|[\boldsymbol{G}^{k-1}\boldsymbol{X}]_j\right\|_2}\right) \tag{7}$$

where $||\cdot||_2$ is used to project different graph signals to a common scale.

Thus, we can theoretically consider graph convolutions of increasing order as a process of a gradual change in smoothness. Inspired by this observation, we convert the aforementioned smoothness saturation objective equation (6) into a sequence prediction problem, in which the smoothness saturation associated with the current order of graph convolution depends on the previously filtered signals. We formulate a state transition model $p_{st}^{\mathcal{G}}(\cdot)$ for the entire graph as follows:

$$q_{ss}^k = p_{st}^{\mathcal{G}}\left(\boldsymbol{s}^{k-1}, \boldsymbol{G}^k\boldsymbol{X}\right) \tag{8}$$

$$\mathcal{K} = \underset{k=1,\ldots,K}{\arg\max}\, q_{ss}^k \tag{9}$$

where $\boldsymbol{s}^{k-1}$ denotes the smoothness state of the entire graph resulting from the last graph convolution.

Once $\mathcal{K}$ is fixed, we can achieve the final representation of the nodes through $\mathcal{K}$-order graph convolution

$$\overline{\boldsymbol{X}} = \boldsymbol{G}^{\mathcal{K}}\boldsymbol{X}. \tag{10}$$

Considering the possible variance caused by noise and the resulting accumulated influence on the gradients [45], we further propose a linear composition of multiple graph convolutions of distinct orders. We represent the accumulated value of smoothness saturation with respect to a threshold ϵ. The ultimate form of our task objective can be written as follows:

$$\mathcal{K} = \min\left\{k' : \sum_{k=1}^{k'} q_{ss}^k \geq \epsilon\right\} \tag{11}$$

$$\overline{\boldsymbol{X}} = \sum_{k=1}^{\mathcal{K}} q_{ss}^k \cdot \boldsymbol{G}^k\boldsymbol{X} \tag{12}$$

where q_{ss}^k is accumulated to ϵ.

d) Nodewise adaptive smoothness-transition graph convolution: Notably, in the above solutions, the optimal order $\mathcal{K}$ of graph convolution is determined only by the global smoothness saturation of the entire graph. In the k-order graph convolution, the node representations are updated by iteratively aggregating the features of all k-hop neighbors. However, the smoothness saturation of different nodes subjected to a convolution of the same order $\mathcal{K}$ could differ considerably. Intuitively, the density of the nodes can vary greatly in attributed graphs. A relatively isolated node with few neighbors could require a larger $\mathcal{K}$ to introduce more distant nodes to obtain more information, while a node with more neighbors usually can efficiently gather information in fewer hops. Thereafter, we should consider the surrounding environment of every node and adjust the optimal order $\mathcal{K}$ of graph convolution separately for each node to guarantee the collection of sufficient information from within an appropriate distance without introducing irrelevant noise. Accordingly, we further optimize our objective equation (8) in a more fine-grained, nodewise manner

$$q_{ss}^{k,i} = p_{st}^{\mathcal{V}}\left(\boldsymbol{s}_i^{k-1}, \left[\boldsymbol{G}^k\boldsymbol{X}\right]^i\right) \tag{13}$$

where the smoothness state of node v_i resulting from k-order graph convolution is denoted by $\boldsymbol{s}_i^{k-1}$, and $p_{st}^{\mathcal{V}}(\cdot)$ is the corresponding nodewise state transition function.

Driven by the motivation similar to that for (12), we accumulate the node representation of node v_i produced by graph convolutions of distinct orders k, combining with the corresponding smoothness saturation. We formulate this process as follows:

$$\mathcal{K}_i = \min\left\{k' : \sum_{k=1}^{k'} q_{ss}^{k,i} \geq \epsilon\right\} \tag{14}$$

$$\bar{\boldsymbol{x}}_i = \sum_{k=1}^{\mathcal{K}_i} q_{ss}^{k,i} \cdot \left[\boldsymbol{G}^k\boldsymbol{X}\right]^i. \tag{15}$$

2) Internal Structure of the Smoothness Sensor: With the established overall objective of customizing the order of graph convolution in (12) and (15), in this section, we illustrate the internal structure of our proposed AS-GC and NAS-GC methods. We build our methods upon an adaptive computation time mechanism that has been proposed for RNNs [46]. We first present the details of realizing adaptive graph convolution at the graph level, as in (12), and then extend the operations to the node level, as in (15).

To compute the accumulated smoothness saturation, the remaining challenge is to design an effective smoothness-transition model, as in (8), and adapt it to the graph domain with graph convolution.

First, we choose a low-pass graph filter such as that in [42]. The corresponding frequency response function is

$$p(\lambda_q) = 1 - \frac{1}{2}\lambda_q \tag{16}$$

where λ_q is the qth eigenvalue derived from the symmetrically normalized graph Laplacian. Accordingly, the filtered graph signals under k-order convolution can be specified as follows:

$$\boldsymbol{G}^k\boldsymbol{X} = \boldsymbol{U}\left(\boldsymbol{I} - \frac{1}{2}\boldsymbol{\Lambda}\right)^k \boldsymbol{U}^{-1}\boldsymbol{X}. \tag{17}$$

We model the gradual smoothing process as a sequence in which the smoothness under k-order graph convolution depends on the historical smoothness states, that is, 1, ..., $k-1$. A variable $\boldsymbol{s}^k$ records the state under k-order graph convolutions

$$\boldsymbol{s}^k = \begin{cases} \mathcal{S}\left(0, g\left(\boldsymbol{G}^1\boldsymbol{X}\right)\right), & \text{if } k = 1 \\ \mathcal{S}\left(\boldsymbol{s}^{k-1}, g\left(\boldsymbol{G}^k\boldsymbol{X}\right)\right), & \text{otherwise} \end{cases} \tag{18}$$

where $\boldsymbol{X}$ is the original feature matrix of the graph and $\boldsymbol{G}^k\boldsymbol{X}$ represents the filtered signals under k-order graph convolution. The function $g(\cdot)$ is a neural network projecting $\mathbb{R}^{n\times m}$ to $\mathbb{R}^m$. The model $\mathcal{S}(\cdot)$ can be implemented with RNNs [47] and GRUs [48], in which the first and second parameters correspond to the previous state under $(k-1)$-order graph convolution and the current filtered graph signal. In particular, the state in the initial step is set to a zero vector.

Once the current state is recorded, we introduce an extra unit to sense whether the smoothness is already saturated

$$h^k = \sigma\left(\boldsymbol{w}_h\boldsymbol{s}^k + \boldsymbol{b}_h\right) \tag{19}$$

where $\boldsymbol{w}_h$ and $\boldsymbol{b}_h$ represent the trainable parameters and bias, respectively. $\sigma(\cdot)$ denotes the sigmoid function, which projects the output to a fixed range of $(0, 1)$. We then accumulate the estimated $\{h^1, h^2, \ldots\}$. Once the accumulated value exceeds the threshold, the smoothness is saturated and $\mathcal{K}$ is achieved

$$\mathcal{K} = \min\left\{K, \min\left\{k' : \sum_{k=1}^{k'} h^k \geq \epsilon\right\}\right\} \tag{20}$$

where K is a hyperparameter to limit the maximum order of graph convolution.

To guarantee that the accumulated value reaches exactly ϵ, we especially address the $\mathcal{K}$-order graph convolution. The complete calculation of the smoothness saturation can be written as follows:

$$q_{ss}^k = \begin{cases} \epsilon - \sum_{k=1}^{\mathcal{K}-1} h^k, & \text{if } k = \mathcal{K} \\ h^k, & \text{otherwise.} \end{cases} \tag{21}$$

Thus, the smoothness-transition model is complete. Finally, an updated representation can be obtained via (12). The graph convolution based on the accumulated smoothness-transition model at the graph level is achieved. We can similarly adopt the aforementioned operations at the node level.

Given a specific node, a variable $\boldsymbol{s}_i^k$ is assigned to replace $\boldsymbol{s}^k$ for recording the smoothness state of node v_i under k-order graph convolution

$$\boldsymbol{s}_i^k = \begin{cases} \mathcal{S}\left(0, \left[\boldsymbol{G}^1\boldsymbol{X}\right]^i\right), & \text{if } k = 1 \\ \mathcal{S}\left(\boldsymbol{s}_i^{k-1}, \left[\boldsymbol{G}^k\boldsymbol{X}\right]^i\right), & \text{otherwise.} \end{cases} \tag{22}$$

Similarly, h^k is transformed into h_i^k, which is associated with node v_i

$$h_i^k = \sigma\left(\boldsymbol{w}_h\boldsymbol{s}_i^k + \boldsymbol{b}_h\right). \tag{23}$$

By accumulating the estimated values $\{h_i^1, h_i^2, \ldots\}$, the boundary of smoothness saturation can be drawn as follows:

$$\mathcal{K}_i = \min\left\{K, \min\left\{k' : \sum_{k=1}^{k'} h_i^k \geq \epsilon\right\}\right\}. \tag{24}$$

Finally, we utilize the maximum accumulated value ϵ to obtain the ultimate smoothness saturation of node v_i under k-order graph convolution

$$q_{ss}^{k,i} = \begin{cases} \epsilon - \sum_{k=1}^{\mathcal{K}_i-1} h_i^k, & \text{if } k = \mathcal{K}_i \\ h_i^k, & \text{otherwise.} \end{cases} \tag{25}$$

Thus, the nodewise adaptive smoothness-transition model becomes approachable.

D. Self-Supervised Clustering Learning

Although NAS-GC (AS-GC) has the potential to enable graph convolution to yield effective node representations for downstream clustering tasks, a supervision mechanism to guide the training process of the proposed model for a typical unsupervised clustering task is still lacking. Furthermore, it should be noted that NAS-GC provides a local perspective for evaluating smoothness in which the smoothness of each node is assessed in accordance with the surrounding environment within a relatively limited radius, but without any potential cluster-related information for the node. To be concrete, the global outlook—the distribution of the nodes with respect to clusters—is neglected, which leads to that node pairs separated by a long distance may nevertheless belong to the same cluster, whereas pairs located nearby may belong to different clusters. Based on this intuition, we propose and illustrate a self-supervision strategy that fills in the gap between the local and global perspectives and, thus, provides adaptability with respect to any graph structures. In addition, this learning tactic is also available for AS-GC.

Given the learned features $\overline{\boldsymbol{X}} = [\overline{\boldsymbol{x}}_1, \ldots, \overline{\boldsymbol{x}}_n]^T$, we apply a linear kernel $\boldsymbol{M} = \overline{\boldsymbol{X}}\overline{\boldsymbol{X}}^T$ to quantify the similarity of node pairs. Then, we use $\boldsymbol{W} = (1/2)(|\boldsymbol{M}| + |\boldsymbol{M}^T|)$ to make $\boldsymbol{M}$ symmetric and non-negative [49]. The function $|\cdot|$ represents taking the absolute value of each element in the matrix. Once the similarity matrix has been obtained, the eigenvectors associated with the r largest eigenvalues of $\boldsymbol{W}$, where r is the number of expected clusters, are calculated and passed to the k-means algorithm to obtain the ultimate cluster partitions. To more effectively perform the clustering task, the following factors should be considered.

Intracluster Tightness: A good cluster partition should have a small intracluster distance. It is natural to apply the indicator of tightness, which is the average length of all lines in $C(i)$ connected to node v_i [50]

$$\mathrm{tig}(i) = \frac{\sum_{v_j \in C(i)} \mathrm{dis}(i,j)}{|C(i)|} \tag{26}$$

where $C(i)$ denotes the node set belonging to the cluster to which vertex v_i is assigned and $\mathrm{dis}(\cdot)$ is a function for measuring the dissimilarity between two objects. Benefiting from this global measurement, the representations of two nodes situated far from each other but belonging to the same cluster can be detected and adjusted. We then extend $\mathrm{tig}(i)$ to the entire graph and adopt it as a part of our loss function, denoted by $\mathcal{L}_{\mathrm{tig}}$

$$\mathcal{L}_{\mathrm{tig}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \frac{1}{|c|(|c|-1)} \sum_{v_i, v_j \in c, v_i \neq v_j} \left|\left|\overline{\boldsymbol{x}}_i - \overline{\boldsymbol{x}}_j\right|\right|_2. \tag{27}$$

Intercluster Separation: A good cluster partition should have a large intercluster distance. Compared with the tightness, although the separation between clusters also plays a critical role, the intercluster separation is unfortunately neglected by other methods. For instance, in AGC [42], the node features become smoother as the order k of graph convolution increases, which will ultimately reduce both the intracluster and intercluster distances. Based on our proposed algorithm, however, intercluster separation can be equally considered in the opposite direction—the separation is expected to be large, while the tightness should be small. We formally define the intercluster separation as follows:

$$\mathrm{sep}(i) = \frac{\sum_{v_j \in C'(i)} \mathrm{dis}(i,j)}{|C'(i)|} \tag{28}$$

where $C'(i)$ represents the set of nodes belonging to a different cluster than that to which the vertex v_i belongs. Similar to the benefit offered by the intracluster tightness, the representations of nodes located nearby but assigned to different clusters are highlighted under this intercluster separation indicator. We also adopt this indicator as a part of our loss function, denoted by $\mathcal{L}_{\mathrm{sep}}$

$$\mathcal{L}_{\mathrm{sep}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \frac{1}{|c|(n-|c|)} \sum_{v_i \in c, v_j \notin c} \left|\left|\overline{\boldsymbol{x}}_i - \overline{\boldsymbol{x}}_j\right|\right|_2. \tag{29}$$

Tradeoff Between Tightness and Separation: Some bias between tightness and separation is inevitable with respect to the number of expected clusters and the distribution of nodes in the graph; this is the previously mentioned global structure information. From this global perspective, local observations can be compensated. To be precise, this adaptivity is empowered by the fact that the tradeoff between tightness and separation can be adjusted with respect to different graph structures. We consider the tightness and separation across all nodes for a cluster partition and combine them into an overall expression

$$\mathcal{L} = \lambda_{\mathrm{tig}}\mathcal{L}_{\mathrm{tig}} + \lambda_{\mathrm{sep}}\frac{1}{\mathcal{L}_{\mathrm{sep}}} \tag{30}$$

where a larger λ_{tig} drives the vertices within a cluster to be tighter and λ_{sep} drives the nodes to be well separated between clusters. λ_{tig} and λ_{sep} are adversarial parameters used to control the tradeoff between these two indicators and adapt the method to distinct graph networks.

Considerable effort could be required to determine the optimal parameters, λ_{tig} and λ_{sep}, through a comprehensive analysis of the nature of the data of interest. Therefore, we propose a practical and automatic selection strategy for seeking the optimal parameters for various datasets.

Instead of directly tuning the hyperparameters λ_{tig} and λ_{sep} by exploring the dataset characteristics, we start by observing the proportion of $\mathcal{L}_{\mathrm{tig}}$ with respect to $(1/\mathcal{L}_{\mathrm{sep}})$, which can be roughly approximated by the value obtained after executing the first epoch. Thereafter, we can balance the two terms $\lambda_{\mathrm{tig}}\mathcal{L}_{\mathrm{tig}}$ and $\lambda_{\mathrm{sep}}(1/\mathcal{L}_{\mathrm{sep}})$ in (30). We conduct a grid search on the proportion sequence, with the proportion between the two terms ranging from 1:3 to 1:50. Then, a favorable choice can be automatically revealed.

IV. Experiments

A. Data

We apply five datasets as our benchmark datasets. Cora, Citeseer, and Pubmed [51] can be characterized as citation networks in which nodes represent various documents and edges connect two documents with a citation relation. Each document is classified into a particular class. Wiki [30] is a webpage network in which webpages (nodes) are connected by page link relations (edges). In Cora and Citeseer, the initial node features are represented through the bag-of-words approach, while in Pubmed and Wiki, tf-idf word vectors are used. We completely inherit the format from the original works.

The dataset ogbn-arxiv [52] is utilized to verify the scalability of the proposed method, in which the numbers of the nodes, edges and, clusters are roughly 63, 215, and 6 times the size

TABLE II
BASIS STATISTICS OF DATASETS

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Wiki	2,405	17,981	4,973	17
ogbn-arxiv	169,343	1,166,243	128	40

of the Cora dataset, respectively. The ogbn-arxiv dataset[1] is a citation network of papers. Each paper is represented via a 128-D feature vector obtained by averaging the embeddings of words in its title and abstract. The Word2Vec model [53] is applied over the Microsoft Academic Graph corpus [54] to calculate the embeddings of individual words [52]. The details of the numbers of nodes, edges, features, and classes are listed in Table II.

B. Baselines and Evaluation Metrics

To evaluate the performance of the proposed method, 13 state-of-the-art methods are used as baseline methods, which can be grouped into three categories according to their inputs.

1) *Node Feature-Based Methods:* The similarity matrices are first constructed from the input node representations, and clustering techniques are then conducted on the constructed matrices. Typical examples are *k*-means [1] and spectral clustering (spectral-f).
2) *Graph Structure-Based Methods:* Adjacency matrices and other graph representations are mainly considered, such as structure-based spectral clustering (spectral-g), DeepWalk [55], and DNGR [39].
3) *Methods Based on Both Node Features and Graph Structures:* TADW [30] was designed to integrate these two kinds of information under the DeepWalk framework, while deep graph neural networks are employed to combine them in the GAE, VGAE, MGAE, adversarially regularized graph autoencoder (ARGE), adversarially regularized variational graph autoencoder (ARVGE) [56], SDCN, and AGC approach.

The parameter settings of the TADW,[2] SDCN,[3] and AGC[4] methods are consistent with their published codes. The implementations of the other baselines are inherited from their original papers and several parameters are optimized as mentioned in [42].

Two kinds of metrics are utilized for the evaluation. Ground-truth-based metrics compare the detected clusters with the ground-truth labels, such as clustering accuracy (Acc), normalized mutual information (NMI), and macro F1-score (F1), as in [42] and [57].

Modularity-based metrics [14], [58] are employed to handle the situation where ground-truth labels are not provided or ignored. The metric of Dist-Modularity (DM) [58] is applied in our work, in which both the structural closeness of graph and attribute homogeneity of a node are considered. The available range of DM is $[-1, 1]$, and a higher value indicates a greater performance.

[1]https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv
[2]https://github.com/benedekrozemberczki/TADW
[3]https://github.com/bdy9527/SDCN
[4]https://github.com/karenlatong/AGC-master

C. Implementation Details

For NAS-GC, the maximum order of graph convolution is set to 40 for Cora, Citeseer, Wiki, and ogbn-arxiv, and 120 for Pubmed. GRUs with a hidden size of 200 are chosen as the function $\mathcal{S}(\cdot)$ for Cora, Citeseer, and Wiki. RNNs equipped with 50 hidden units are employed for Pubmed and ogbn-arxiv. We uniformly set the smoothness saturation to 1 for all datasets. We train all models with the Adam optimizer [59]. The learning rate is 0.01 for Cora and ogbn-arxiv, 0.005 for Pubmed, 0.003 for Citeseer, and 0.0001 for Wiki. For Wiki, the learning rate is also annealed by 0.96 when the epoch index is larger than 10. The bias hyperparameter λ_{tig} is uniformly set to 1 for all datasets, while λ_{sep} is chosen to be 50, 350, 0.0005, 10, and 8 for Cora, Citeseer, Pubmed, Wiki, and ogbn-arxiv, respectively. These bias hyperparameters are automatically obtained via our proposed automatic selection strategy for self-supervised clustering learning. In addition, we will present a detailed analysis of this hyperparameter later.

The implementation details of AS-GC are similar to those of NAS-GC, with the exception that the maximum order of graph convolution is set to 25 and 30 for the Cora and Wiki datasets, respectively, and the learning rate for Citeseer is 0.03.

We uniformly customize an early termination mechanism such that if the standard deviation of the losses produced in the last five epochs is smaller than 0.001 (NAS-GC) or 0.1 (AS-GC), the training process will be terminated in advance. This mechanism is applied on all datasets, with a maximum of 200 epochs.

Our hardware environment consists of two 3.00 GHz Intel Xeon E5-2687W CPUs, 512 GB of memory, and two Nvidia Titan RTX GPUs. The code of the proposed model can be downloaded from https://github.com/aI-area/NASGC.

D. Results

Table III records the average performance over ten repetitions of the clustering task on each dataset.

Comparing the node feature-based methods with the graph-structure-based methods, we can observe that there is no clear superiority between these two approaches. The node feature-based spectral-f method performs better on Citeseer and Wiki, while the graph structure-based DNGR and Deepwalk method work better on Cora. On Pubmed, DeepWalk achieves better ACC and DM performance but worse NMI and F1 performance than spectral-f. Therefore, we cannot draw a clear conclusion regarding the superiority of either node- or graph-based methods.

In comparison with the baselines utilizing either node features or graph structure information alone, we note a significant improvement in the third group of methods, which combine these two kinds of information. On Cora, all of these methods outperform the methods in the first two groups by a considerable margin. As a reference, we average the scores of the methods in the first two groups and those of the methods in the third group. Compared to the first two groups, the

TABLE III
PERFORMANCE ON CLUSTERING TASKS

Method	Cora				Citeseer				Pubmed				Wiki			
	Acc%	NMI%	F1%	DM%	Acc%	NMI%	F1%	DM%	Acc%	NMI%	F1%	DM%	Acc%	NMI%	F1%	DM%
K-means	34.65	16.73	25.42	-16.48	38.49	17.02	30.47	-8.31	57.32	29.12	57.35	15.68	33.37	30.20	24.51	-29.96
Spectral-f	36.26	15.09	25.64	-17.69	46.23	21.19	33.70	-6.83	59.91	32.55	58.61	16.15	41.28	43.99	25.20	-16.53
Spectral-g	34.19	19.49	30.17	14.51	25.91	11.84	29.48	5.14	39.74	3.46	51.97	0.03	23.58	19.28	17.21	-18.73
DeepWalk	46.74	31.75	38.06	30.19	36.15	9.66	26.70	30.23	61.86	16.71	47.06	25.16	38.46	32.38	25.74	24.40
DNGR	49.24	37.29	37.29	-30.41	32.59	18.02	44.19	-27.24	45.35	15.38	17.90	-6.60	37.58	35.85	25.38	-24.54
TADW	56.03	44.11	51.80	34.79	45.58	29.14	43.53	36.67	51.25	24.32	49.85	26.20	30.96	27.13	26.68	29.48
GAE	53.25	40.69	41.97	35.29	41.26	18.34	29.13	31.55	64.08	22.97	49.26	36.17	17.33	11.93	15.35	-6.10
VGAE	55.95	38.45	41.50	36.64	44.38	22.71	31.88	34.57	65.48	25.09	50.95	**37.05**	28.67	30.28	20.49	26.21
MGAE	63.43	45.57	38.01	35.41	63.56	39.75	39.49	36.38	43.88	8.16	41.98	28.41	50.14	47.97	39.20	12.30
ARGE	64.00	44.90	61.90	35.27	57.30	35.00	54.60	34.63	59.12	23.17	58.41	35.99	41.40	39.50	38.27	27.41
ARVGE	63.80	45.00	62.70	34.28	54.40	26.10	52.90	36.45	58.22	20.62	23.04	3.67	41.55	40.01	37.80	15.45
SDCN	62.22	35.16	57.73	30.92	65.96	38.71	63.62	35.40	65.20	28.03	65.64	29.58	42.20	37.79	24.05	21.09
AGC	68.92	53.68	65.61	39.77	67.00	41.13	62.48	41.69	69.78	31.59	68.72	36.16	47.65	45.28	40.36	27.94
AS-GC	69.45	54.38	66.06	**39.90**	68.49	42.49	63.83	41.87	69.90	32.78	69.07	36.13	51.16	47.70	42.14	28.15
NAS-GC	**71.94**	**54.89**	**69.01**	39.84	**69.58**	**43.78**	**64.73**	**41.89**	**70.70**	**36.79**	**70.40**	36.42	**56.24**	**51.07**	**44.79**	**29.57**

note: The 1st group is feature-based; the 2nd is structure-based; the 3rd is feature and structure-based; the 4th is the proposed. Bold font indicates the best scores.

maximum improvement of the third group is by 20.73% (Acc), 19.38% (NMI), 21.34% (F1), and 39.27% (DM) on Cora, and the minimum improvement is on Wiki, with improvements of 2.63%, 2.65%, 6.67%, and 32.29%, respectively. It can be concluded that effectively combining node features and graph structure information can be beneficial for the node representations used for the downstream clustering task.

With respect to Acc, NMI, and F1, the proposed methods, both NAS-GC and AS-GC, achieve the best results across all datasets. Compared with AGC that obtains the best performance among the baselines, NAS-GC outperforms AGC by 8.59% (Acc), 5.79% (NMI), 4.43% (F1), and 1.63% (DM) on Wiki. We note that the improvement on Wiki is more significant than that on Cora, Citeseer, and Pubmed. Consider the numbers of classes, 7 in Cora, 6 in Citeseer, 3 in Pubmed, and 17 in Wiki, which means that the internal structures of clusters in Wiki may have larger variance. It can be recalled that the proposed method is more adaptive in dealing with the situation where the node density in a graph varies much in attributed graphs. NAS-GC gains more margin than AS-GC in Wiki, which is consistent with our core intuition that nodewise smoothness can further enhance the adaptiveness.

As for DM, following the paper [58], we specify $\exp(-(d/\sigma)^2)$ to measure node similarity, where d is the Euclidean distance between two nodes, and σ is set to $0.35\overline{d}$, in which $\overline{d}$ is the average distance across all node pairs. We also test the different values of σ, and the trend for performance evaluation does not change much. The proposed methods slightly lag behind VGAE on the Pubmed dataset, while they outperform the other baselines in the other three datasets.

In the third group, the numbers of convolutional layers applied differ among the different methods. The GAE, VGAE, ARGE, and ARVGAE methods all employ second-order convolutions, while the MGAE and SDCN methods rely on third-order convolutions; hence, these convolutions are relatively shallow. In contrast, AGC performs 12th-, 55th-, 60th-, and 8th-order graph convolutions on Cora, Citeseer, Pubmed, and Wiki, respectively; these convolutions are much deeper than those of the other baselines and achieve better performance. The maximum order of graph convolution in our proposed method NAS-GC is 40 for Cora, Citeseer, and Wiki and 120 for Pubmed. We will analyze the impact of the order of graph convolution in detail in the next section.

TABLE IV
PERFORMANCE ON CLUSTERING TASKS IN A LARGE GRAPH

Method	ogbn-arixv			
	Acc%	NMI%	F1%	DM%
K-means	17.69	22.15	12.70	0.40
Spectral-g	17.13	12.85	4.03	3.65
DNGR	16.86	29.41	10.75	-3.36
TADW	13.45	10.68	12.47	-3.46
MGAE	18.77	23.15	13.03	2.28
SDCN	26.18	21.54	8.90	27.03
AGC	22.13	26.71	17.34	19.68
AS-GC	29.30	29.19	17.37	**45.08**
NAS-GC	**29.44**	**32.41**	**20.12**	35.81

note: The 1st group is feature-based; the 2nd is structure-based; the 3rd is feature and structure-based; the 4th is the proposed. Bold font indicates the best scores.

E. Scalability Test

To verify the scalability of our proposed approach, we chose ogbn-arxiv [52] as our test target following the definition of a large graph in [60]. K-means, spectral-g, DNGR, TADW, MGAE, SDCN, and AGC were selected as our reference, due to the space and time cost limitations of our hardware environment even if we trained models on CPUs with 512 GB of memory.

The results are shown in detail in Table IV. The overall tendency of the performance keeps unchanged. The approaches that take both graph structures and node features as input achieve greater performance than others. The proposed AS-GC and NAS-GC take advantage over the baseline methods across all metrics. NAS-GC outperforms AGC by 7.31% (Acc), 5.70% (NMI), 2.78% (F1), and 16.13% (DM), respectively.

TABLE V
PERFORMANCE OF NAS-GC WITH DIFFERENT MAXIMUM ORDERS OF GRAPH CONVOLUTION

Method		Cora			Citeseer			Pubmed			Wiki		
		Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%
AGC		68.92	53.68	65.61	67.00	41.13	62.48	69.78	31.59	68.72	47.65	45.28	40.36
	2	60.83	39.75	60.6	62.33	36.47	58.97	61.21	34.21	59.96	45.15	43.35	34.68
	5	65.03	49.79	59.91	66.66	40.70	62.23	61.76	32.49	60.49	**48.83**	**48.89**	**40.77**
	7	65.36	49.62	60.02	**67.59**	**41.62**	**63.08**	62.3	31.73	61.13	-	-	-
	10	**70.22**	**54.10**	**67.44**	-	-	-	63.21	31.19	62.20	-	-	-
K	20	-	-	-	-	-	-	65.94	31.93	65.27	-	-	-
	30	-	-	-	-	-	-	68.03	33.35	67.46	-	-	-
	40	-	-	-	-	-	-	69.06	34.09	68.55	-	-	-
	50	-	-	-	-	-	-	69.69	34.85	69.12	-	-	-
	60	-	-	-	-	-	-	**70.26**	**34.50**	**69.66**	-	-	-

note: Bold font indicates the best scores.

This result is consistent with the forementioned explanation that the proposed algorithm performs better in relatively complex networks containing more diverse clusters. It also proves that the proposed methods can be applied to large graphs.

An interesting finding is that AS-GC significantly outstrips NAS-GC under the metric of DM. According to [61], in a large network, DM encourages an algorithm to merge the small clusters to form a larger group. The proposed AS-GC, in which smoothness saturation is shared across all nodes, results in more similar representations among nodes to form a bigger cluster and may obtain higher evaluation from DM.

F. Extensive Study

In this section, we will present several extensive experiments to further examine the proposed methods NAS-GC and AS-GC.

1) Does the Performance of NAS-GC Merely Depend on Higher Orders of Graph Convolution?: Although our method significantly outperforms all other baselines via relatively high orders of graph convolution, there is a concern of whether the performance of our approach merely depends on performing convolutions of a higher order. To eliminate this concern, we conduct a series of experiments to witness the performance changes as the order of graph convolution gradually increases. We follow the same hyperparameter settings as in the above evaluation test except for the order of graph convolution. We continuously increase the magnitude of the maximum order of graph convolution until the results outperform AGC, which almost achieves the best performance across all datasets. The results are recorded in Table V.

Our model performs better than AGC across all four datasets. To be precise, when 10th-, 7th-, and 5th-order graph convolutions are applied, the proposed method already outperforms AGC with 12th-, 55th- and 8th-order graph convolutions in the Cora, Citeseer, and Wiki dataset, respectively. This experiment proves that the superiority of our model does not completely rely on higher orders of graph convolution.

2) Does the Nodewise Mechanism Contributes to the Performance of NAS-GC?: We have proven that deeper graph convolution operations are not the sole reason that NAS-GC can significantly outperform the considered baselines. Another possible influential factor could be the nodewise mechanism, which is based on the premise that the order of graph convolution should be chosen for each node individually in accordance with its concrete surroundings. Although the superiority of this approach has already been verified through performance experiments in which NAS-GC outperforms AS-GC across all datasets, we further present an extensive investigation of the nodewise order of graph convolution, the variable $\mathcal{K}_i$, to more comprehensively illustrate this point.

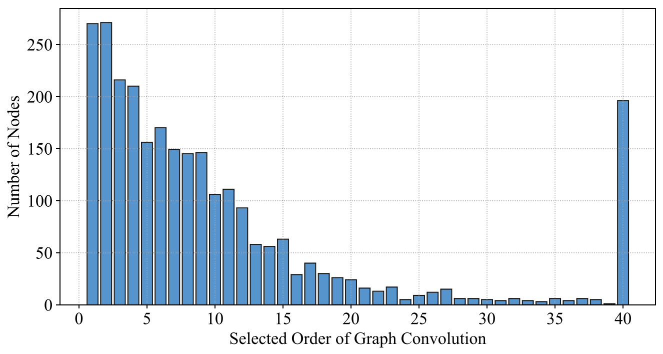


Fig. 2. Number of vertices with respect to the selected order of graph convolution for the Cora dataset.

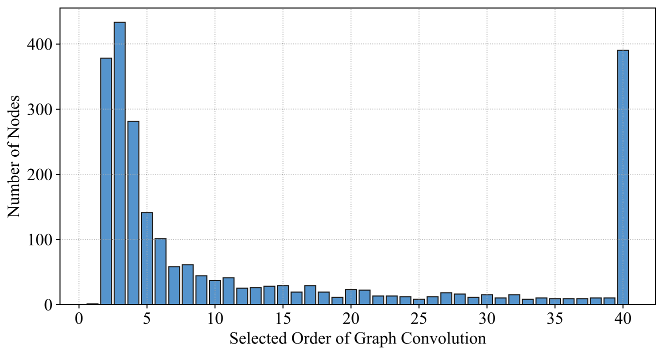


Fig. 3. Number of vertices with respect to the selected order of graph convolution for the Wiki dataset.

We first trained models on the Cora and Wiki datasets individually and recorded the selected order of graph convolution $\mathcal{K}_i$ for each node v_i. We then calculated the distribution of the number of corresponding nodes relative to the selected order. The results are depicted in Figs. 2 and 3. Note that our

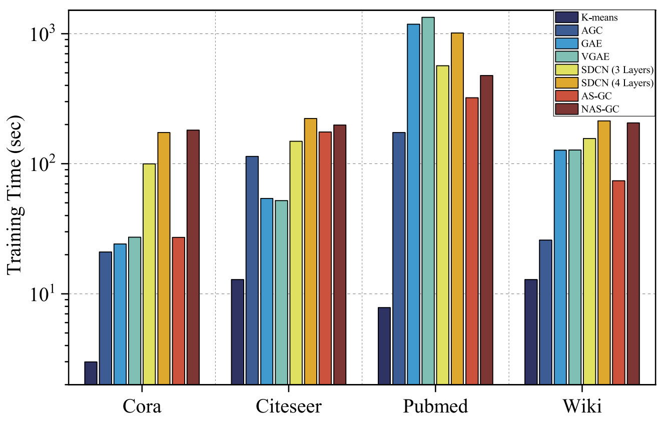


Fig. 4. Training times for the baselines and our proposed methods.

algorithm imposes a maximum value on the order of graph convolution, which is set to 40 for both Cora and Wiki.

We can observe that the finally selected $\mathcal{K}_i$ values differ tremendously among the nodes, and the distribution has a long tail. In Fig. 2, on Cora, although saturation can be reached for most nodes with an order of graph convolution that is below approximately 23, some nodes still require deeper convolution, even deeper than the maximum order of 40. These two distributions prove the necessity and effectiveness of our proposed nodewise mechanism.

Furthermore, we can also gain some interesting clues from these distributions regarding the possible reasons for the performance difference between our work and previous study. In the distribution obtained on Cora, the nodes for which the selected order of graph convolution is below 12 account for 75.44% of the entire graph. This result is surprisingly consistent with AGC, which chooses 12 as the order of graph convolution. Nevertheless, our method chooses to halt convolution at exactly the 12th order for only 3.43% of the nodes, while for the others, graph convolution is terminated at an order below 12. From another perspective, the results reveal that 24.56% of the nodes still require further convolution. However, if a higher order of graph convolution was to be applied in AGC, this would result in oversmoothness and worsen the performance in the clustering task because AGC determines the order of graph convolution only at the graph level. From Fig. 3, we can also learn the reason why the MGVAE method can achieve relatively high performance on Wiki compared with the other baselines. The frequency $\mathcal{K}_i$ peaks at 3, and then a sharp drop occurs, which is consistent with the selected order of graph convolution in the MGVAE method.

3) What Is the Efficiency of Our Proposed Method?: To compare the time efficiency among the baselines, we present experiments conducted to quantify the training time for each method. The detailed training times are depicted in Fig. 4. In this experiment, for a fair comparison, CPUs are uniformly adopted to train all models since some baselines cannot be trained on GPUs. It can be observed that the classical clustering method, k-means, has the lowest time consumption across all datasets and models, followed by AGC, in which no neural network parameters need to be trained. In contrast, there is a common characteristic of other deep learning methods—many parameters need to be learned. To be concrete, the GAE and VGAE models can be trained most quickly on the Cora and Citeseer datasets, respectively. AS-GC takes the least time on Pubmed and Wiki, followed by NAS-GC on the Pubmed dataset. Although it seems that the utilization of more fine-grained training targets—the nodes—instead of the entire graph in NAS-GC may lead to extra computations, the fact is that these more precise training units can be tremendously beneficial for speeding up the process of convergence. This superiority is especially obvious for large-scale graph structures, for example, the Pubmed dataset, which is a graph network composed of 19 717 nodes and 44 338 edges.

We also conducted a theoretical time cost analysis for the proposed method NAS-GC, given a graph with n nodes and m features. The entire process of the proposed method, for one input graph instance, can be divided into four subsequent steps: 1) obtaining the filtered graph signals under k-order convolution (17); 2) calculating the final node representations (15) and (22)–(25); 3) performing spectral clustering; and 4) computing loss value (27), (29), (30). The computation in the first step is related to eigenvalue decomposition and matrix multiplication, and the result keeps unchanged once calculated. Therefore, the time cost is constant and can be omitted here. For the second step, it is $\mathcal{O}(nKd^2)$ complexity, where K is the maximum number of convolution layers, and d is the dimension of hidden states of the state transition model (e.g., RNNs). The time complexity of performing spectral clustering is $\mathcal{O}(n^2m + n^2r)$ and can be considered as $\mathcal{O}(n^2m)$ since r is the number of clusters and usually much smaller than m. The action of computing intracluster tightness and intercluster separation cost is $\mathcal{O}(n^2m)$ complexity. In conclusion, the overall complexity is $\mathcal{O}(n^2m + nKd^2)$.

With respect to memory complexity, for simplicity, we omit the memory for storing the graph as [62]. $\mathcal{O}(dm + d^2)$ is for storing parameters inside the model. Besides, the variables used to compute the gradient require $\mathcal{O}(Knd)$ space. Thus, the overall complexity is $\mathcal{O}(dm + d^2 + Knd)$.

4) Does the Excellent Performance of NAS-GC Heavily Depends on Delicate Bias Hyperparameters?: The parameters λ_{tig} and λ_{sep} control the tradeoff between the tightness and separation of the clusters in a dataset. When λ_{tig} is equal to 1, a larger λ_{sep} places a higher emphasis on a greater separation among clusters, while a smaller λ_{sep} focuses on a closer intracluster relationship. Therefore, this hyperparameter is responsible for adapting this tradeoff to various graph networks with different characteristics. The problem addressed here is whether the performance of NAS-GC is tightly correlated with the selection of these bias hyperparameters.

As mentioned earlier, instead of directly tuning the hyperparameters through a painstaking exploration of the dataset characteristics, we can start by observing the proportion of $\lambda_{\text{tig}}\mathcal{L}_{\text{tig}}$ with respect to $\lambda_{\text{sep}}(1/\mathcal{L}_{\text{sep}})$ after executing the first epoch. We present a group of experiments conducted on Cora and Wiki by adjusting the proportion between the two terms from 1 : 3 to 1 : 50 by modifying λ_{tig} and λ_{sep}, with the corresponding performance results recorded in Table VI. A grid

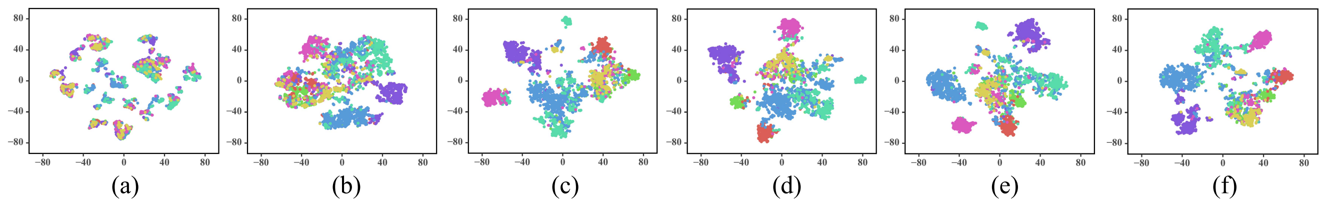


Fig. 5. 2-D visualizations of the learned representations from NAS-GC with an increasing number of epochs. (a) Before training. (b) Epoch 1. (c) Epoch 20. (d) Epoch 50. (e) Epoch 100. (f) Epoch 200.

TABLE VI
PERFORMANCE OF NAS-GC WITH VARIOUS SETTING OF λ_{tig} AND λ_{sep}

Method		Citeseer			Wiki		
		Acc%	NMI%	F1%	Acc%	NMI%	F1%
Baseline		67	41.13	62.48	50.14	47.97	40.36
Ratio	1:3	**69.58**	**43.78**	**64.73**	50.89	50.13	42.95
	1:4	68.41	42.76	63.62	51.88	50.15	42.79
	1:5	68.53	42.64	63.88	53.16	50.81	44.79
	1:10	68.86	43.16	64.06	**56.24**	**51.07**	**44.79**
	1:20	68.62	42.97	63.84	55.39	51.68	44.93
	1:30	68.74	43.09	63.95	53.86	48.99	42.22
	1:40	68.68	43.07	63.89	52.07	48.74	41.68
	1:50	68.71	43.11	63.92	53.42	49.13	42.01

note: "Baseline" records the best performance in all baselines. "Ratio" denotes the different proportions of the models, namely, $\lambda_{tig}\mathcal{L}_{tig} : \lambda_{sep}\frac{1}{\mathcal{L}_{sep}}$, after the first epoch. Bold type indicates the optimal scores.

search on the proportion sequence can automatically reveal the optimal hyperparameters settings, the results of which are indicated with bold font in this table. We also note that within this proportion range, NAS-GC, the proposed method, consistently outperforms the baseline method with the best performance—either AGC or MGAE. This consistent excellent performance proves that the proposed self-supervision strategy indeed seeks a "best" setting, not merely a "good" setting, and the excellent performance of NAS-GC is robust.

5) Can the Learned Representations Provide Intuitive Visualization?: A meaningful visualization of the learned representations can usually provide an indicator for qualifying the learned representations. We select the Cora dataset for such visualization because of its moderate number of classes.

We first map Cora into a low-dimensional space, project the learned representations into a similarity matrix through a linear kernel, and then apply the t-SNE algorithm [63] to further project the representations into a 2-D space. To show the gradual training process, we present visualizations of the representations learned in different epochs.

The results are depicted in Fig. 5, in which seven different colors denote the seven originally labeled document classes. We can observe that the clustering effect already seems reasonable, with separated pink and purple clusters, when the number of epochs reaches 20. However, there are still overlaps among the cyan and blue groups. These overlaps remain until approximately the 200th epoch when the cyan and blue nodes are approximately separated, although the clusters colored in red, green, and yellow still cannot be clearly separated yet. It can be concluded that these three categories are similar enough that they cannot be split in 2-D space. The subjects of genetic algorithms (red), reinforcement learning (green), and theory (yellow) are distinct, yet are inevitably connected to each other.

V. CONCLUSION AND FUTURE WORK

Oversmoothing in GCNs will cause the nodes of a graph to be grouped into fewer clusters and, thus, pose a challenge in terms of performance degradation. This article proposes a solution to overcome the oversmoothing in GCNs and the resulting performance degradation of downstream clustering for attributed graphs. Convolution at a fixed order k at the graph level tends to cause either undersmoothing or oversmoothing. In this study, we explore how k-order filtered graph signals can evolve via a transition from $(k-1)$-order filtered signals in terms of smoothness. We design a smoothness sensor to sense the graph smoothness and terminate the graph convolution process once the smoothness is saturated. Furthermore, we propose a nodewise smoothness-transition mechanism by adaptively customizing the order k of graph convolution for each node. Finally, a clustering criterion considering both the tightness within clusters and the separation between clusters is defined as the loss function to guide the training of the entire model. The experiments prove that the proposed methods significantly outperform 13 other state-of-the-art baselines in terms of different metrics across five benchmark datasets. In addition, an extensive study reveals the reasons for their effectiveness and efficiency. In the future, the potential of nodewise smoothness can be further utilized for other downstream tasks of GCNs, such as node classification.

REFERENCES

[1] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. Roy. Stat. Soc. C Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.
[2] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 849–856.
[3] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*. Boston, MA, USA: Springer, 2012, pp. 77–128.
[4] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2761–2773, Oct. 2010.
[5] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLoS ONE*, vol. 6, no. 4, 2011, Art. no. e18961.
[6] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.
[7] C. Ji, Y. Zheng, R. Wang, Y. Cai, and H. Wu, "Graph polish: A novel graph generation paradigm for molecular optimization," 2020. [Online]. Available: arXiv:2008.06246.

[8] Z. Bu, H.-J. Li, J. Cao, Z. Wang, and G. Gao, "Dynamic cluster formation game for attributed graph clustering," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 328–341, Jan. 2019.

[9] Y. Han, L. Zhu, Z. Cheng, J. Li, and X. Liu, "Discrete optimal graph clustering," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1697–1710, Apr. 2020.

[10] E. Ferrara, P. De Meo, S. Catanese, and G. Fiumara, "Detecting criminal organizations in mobile phone networks," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5733–5750, 2014.

[11] T. He, Y. Liu, T. H. Ko, K. C. C. Chan, and Y. Ong, "Contextual correlation preserving multiview featured graph clustering," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4318–4331, Oct. 2020.

[12] C. Pizzuti and A. Socievole, "Multiobjective optimization and local merge for clustering attributed graphs," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4997–5009, Dec. 2020.

[13] U. Von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[14] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, 2006, Art. no. 036104.

[15] D. Cai, X. He, X. Wu, and J. Han, "Non-negative matrix factorization on manifold," in *Proc. 8th IEEE Int. Conf. Data Min.*, 2008, pp. 63–72.

[16] Q. Gu and J. Zhou, "Co-clustering on manifolds," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 359–368.

[17] 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.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent*, 2017, pp. 1–6.

[19] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. Web Conf.*, 2020, pp. 1400–1410.

[20] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manag.*, 2017, pp. 889–898.

[21] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.

[22] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[23] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 927–936.

[24] Y. Li, C. Jia, X. Kong, L. Yang, and J. Yu, "Locally weighted fusion of structural and attribute information in graph clustering," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 247–260, 2019.

[25] M. E. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.

[26] 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.

[27] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 1225–1234.

[28] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

[29] M. B. Hastings, "Community detection as an inference problem," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, 2006, Art. no. 035102.

[30] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24nd Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.

[31] Y. Pan, C.-Q. Huang, and D. Wang, "Multiview spectral clustering via robust subspace segmentation," *IEEE Trans. Cybern.*, early access, Jul. 14, 2020, doi: 10.1109/TCYB.2020.3004220.

[32] Y. Liang, Y. Pan, H. Lai, and J. Yin, "Robust multi-view clustering via inter-and-intra-view low rank fusion," *Neurocomputing*, vol. 385, pp. 220–230, Apr. 2020.

[33] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 2149–2155.

[34] R. Wang, J. Fan, and Y. Li, "Deep multi-scale fusion neural network for multi-class arrhythmia detection," *IEEE J. Biomed. Health Inform.*, vol. 24, no. 9, pp. 2461–2472, Sep. 2020.

[35] X. Shen and F.-L. Chung, "Deep network embedding for graph representation learning in signed networks," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1556–1568, Apr. 2020.

[36] C. Ji, R. Wang, R. Zhu, Y. Cai, and H. Wu, "HopGat: Hop-aware supervision graph attention networks for sparsely labeled graphs," 2020. [Online]. Available: arXiv:2004.04333.

[37] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2475–2487, Jun. 2020.

[38] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.

[39] 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.

[40] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016, pp. 1–3.

[41] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3670–3676.

[42] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4327–4333.

[43] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[44] F. R. Chung and F. C. Graham, *Spectral Graph Theory*. Providence, RI, USA: Amer. Math. Soc., 1997.

[45] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in *Back-Propagation: Theory, Architectures and Applications*. Hillsdale, NJ, USA: Lawrence Erlbaum Assoc., 1995, ch. 13, pp. 433–486.

[46] A. Graves, "Adaptive computation time for recurrent neural networks," 2016. [Online]. Available: arXiv:1603.08983.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[49] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2429–2435.

[50] J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[51] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[52] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," 2021. [Online]. Available: arXiv:2005.00687.

[53] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013. [Online]. Available: arXiv:1310.4546.

[54] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 396–413, 2020.

[55] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 701–710.

[56] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2609–2615.

[57] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications* (Data Mining and Knowledge Discovery Series). New York, NY, USA: Chapman and Hall/CRC, 2014.

[58] X. Liu, T. Murata, and K. Wakita, "Extracting the multilevel communities based on network structural and nonstructural information," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 191–192.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent*, 2015, pp. 1–41.

[60] P. Chunaev, "Community detection in node-attributed social networks: A survey," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100286.

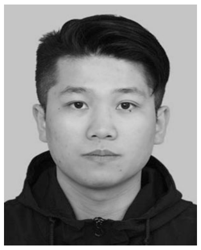
[61] M. Arab and M. Hasheminezhad, "Exploring the limitations of quality metrics in detecting and evaluating community structures," *Int. J. Web Res.*, vol. 1, no. 1, pp. 18–29, 2018.

[62] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2019, pp. 257–266.

[63] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Chaojie Ji received the M.S. degree in software engineering from Nanchang University, Jiangxi, China, in 2018.

He is currently an Engineer with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His current research interests include natural language processing, knowledge graph, and deep learning.



Hongwei Chen received the B.S. degree in network engineering from Tianjin University of Science and Technology, Tianjin, China, in 2019. He is currently pursuing the postgraduate degree with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.

His current research interests include data analysis and knowledge graph.

Ruxin Wang received the Ph.D. degree in operational research and cybernetics from the University of Chinese Academy of Sciences, Beijing, China, in 2017.

He is currently an Assistant Research Fellow with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His current research interests include artificial intelligence in medical big data, graph data mining, machine learning, and pattern recognition.

Yunpeng Cai received the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2007.

He is currently a Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His research interests include health big data, health informatics, bioinformatics, and machine learning.

Hongyan Wu received the Ph.D. degree in bioinformatics from The University of Tokyo, Tokyo, Japan, in 2011.

She is currently an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Her current research interests include big data analytics, health informatics, knowledge graph, and deep learning.