This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

CommGNAS: Unsupervised Graph Neural Architecture Search for Community Detection

Jianliang Gao, Jiamin Chen, Babatounde Moctard Oloulade, Raeed Al-Sabri, Tengfei Lyu, Ji Zhang and Zhao Li

***Abstract*—Graph neural architecture search (GNAS) has been successful in many supervised learning tasks, such as node classification, graph classification, and link prediction. GNAS uses a search algorithm to sample graph neural network (GNN) architectures from the search space and evaluates sampled GNN architectures based on estimation strategies to generate feedback for the search algorithm. In traditional GNAS, the typical estimation strategy requires using labeled graph data to generate feedback, which plays a fundamental and vital role in the search algorithm to sample a better GNN architecture during the search process. However, a large portion of real-world graph data is unlabeled. The estimation strategy in traditional GNAS cannot use unlabeled graph data to generate feedback for the search algorithm, so the traditional supervised GNAS fails to solve unsupervised problems, such as community detection tasks. To solve this challenge, this paper proposed CommGNAS, an effective node representation learning method with unsupervised graph neural architecture search for community detection. In CommGNAS, we design an unsupervised evaluation strategy with self-supervised and self-representation learning. It represents the first research work in literature to solve the problems of unsupervised graph neural architecture search for community detection. The experimental results show that CommGNAS can obtain the best performance in community detection tasks on real-world graphs against the state-of-the-art baseline methods.**

***Index Terms*—Community detection, unsupervised graph neural architecture search, self-supervised learning, self-representation learning, graph neural network**

I. Introduction

COMMUNITY detection is one of the most essential unsupervised tasks in the field of network analysis for various real-world applications, such as anomaly detection and scientific discipline discovery [1]. The goal of community detection in graphs is to identify modules and possibly their hierarchies simply by using the encoding feature in the graph topology [2]. The unsupervised learning of node representation and the node clustering on graphs is a common method to implement community detection [3]. Graph neural networks (GNNs) are an effective tool for processing non-Euclidean graphs. Researchers increasingly focus on GNNs to explore communities in attributed graphs or online social networks [4]–[6].

Jianliang Gao, Jiamin Chen, Babatounde Moctard Oloulade, Raeed Al-Sabri and Tengfei Lyu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, Hunan, China. E-mail:{ gaojianliang, chenjiamin, olouladem octard, alsabriraeed, tengfeilyu}@csu.edu.cn.

Ji Zhang is with University of Southern Queensland, Australia. E-mail: Ji.Zhang@usq.edu.au.

Zhao Li is with Zhejiang University, Hangzhou 311121, Zhejiang, China. E-mail: zhao_li@zju.edu.cn.

(Corresponding Author: Zhao Li)

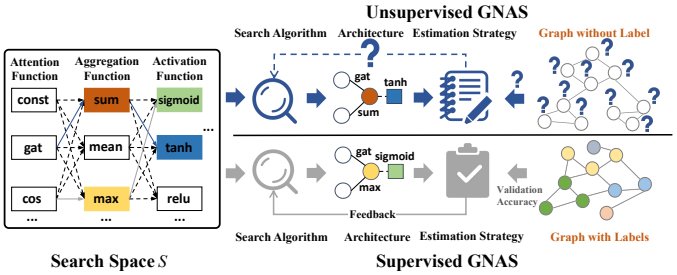

Fig. 1. The object of supervised GNAS is the graph with labels, and the estimation can use the available validation accuracy information to construct the feedback to guide the search algorithm. However, the unsupervised GNAS needs to process the graph without labels, and the estimation can't obtain practical information from the graph labels to produce feedback for the search algorithm. How do we construct useful information from unsupervised learning processes to generate feedback? It is the major challenge for unsupervised GNAS.

Although GNNs have succeeded in community detection in graphs, it is time-consuming and requires expert knowledge to construct and fine-tune graph neural network architecture for different graph data sets. Automatic machine learning can design good performance models based on different feature distributions and downstream tasks. Therefore, automatic machine learning methods based on GNN modeling have been used in an increasing number of scenarios [7]. Graph neural architecture search (GNAS) [8] is an effective method for designing GNN architecture to achieve good performance for graph data with different feature distributions. The GNAS mainly consists of the following four steps: (1) building the search space of the GNN architecture; (2) designing the search algorithm and using it to sample GNN architecture from the search space; (3) using an estimation strategy to evaluate the sampled GNN architecture to generate feedback; (4) using the feedback to iterate the search algorithm for improving the sampling performance. Traditional GNAS methods are designed for supervised learning problems. The estimation strategy uses labeled data to generate feedback to guide the search algorithm in the supervised learning process. However, there is much-unlabeled graph data in the real world, especially on social networks. As shown in Figure 1, the core challenge in achieving an unsupervised graph neural architecture search for community detection is that the estimation strategy cannot produce feedback based on unlabeled graph data for the search algorithm.

To solve this challenge, we propose CommGNAS, an effective node representation learning method with unsupervised graph neural architecture search for community detection.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

CommGNAS consists of the following four major modules: (1) The unsupervised neural architecture search. This module has an unsupervised estimation strategy based on self-supervised and self-representation learning for GNAS. Specifically, the strategy uses the loss change in self-supervised and self-representation learning to construct feedback. It can effectively guide the search algorithm to sample a GNN architecture with good performance for unsupervised tasks. (2) The self-supervised learning module. CommGNAS uses the GNN model to encode graph data G for obtaining the node embedding matrix Z and the self-supervised loss change in this module. (3) The self-representation learning module. CommGNAS generates a similarity matrix S of node embedding matrix Z and self-representation loss change. (4) CommGNAS produces the cluster results by spectral clustering with the similarity matrix S for the community detection task.

The main contributions of this paper are summarized as follows: (1) We propose CommGNAS[1], an effective node representation learning method with unsupervised graph neural architecture search for community detection. (2) In CommGNAS, we propose an unsupervised estimation strategy based on self-supervised and self-representation learning for graph neural architecture search, which enables graph neural architecture search to effectively deal with unsupervised tasks. (3) We have conducted extensive experiments based on real graph data, and the experimental results show that CommGNAS can obtain the best performance on F1-score (F1), normalized mutual information (NMI), and cluster accuracy (Acc) compared to other baseline methods.

The remainder of this paper is organized as follows. In section one, we introduce the related work of community detection based on the GNN model and graph neural architecture search. In section two, we show our proposed CommGNAS framework and explain each part's working principle in detail. In section three, we present the experimental results and explain the effectiveness of the CommGNAS. The last section is the conclusion of this work.

II. RELATED WORK

This section highlights related work in community detection and graph neural architecture search.

A. Community Detection

In this section, we present the related work about community detection comprehensively. The idea of community structure in networks has been proposed by Girvan and Newman [9], which opened up a pertinent study in the field based on semantic information [10] and network structure [11].

1) Traditional Methods: The earlier methods for community detection were static methods that aim to study a community's structure from the current state of the networks. However, this method cannot be used in real-world dynamic networks. Static methods can further be categorized as graph partitioning, hierarchical clustering, partitional clustering, and spectral clustering. In graph partitioning, also known as graph clustering, vertices of a graph are divided into partitions of fixed size so that the number of edges between the partition is minimized [12]. Each node is associated with only one partition in this approach, and no overlaps occur. Spectral clustering enables dynamic node learning network structures and maps nodes to other nodes in a low-dimensional space. The modern spectral clustering algorithm consists of three steps, including regularization of a proper adjacency or Laplacian matrix, a form of spectral truncation, and a k-means algorithm in the truncated spectral-domain [13].

2) Deep Learning-based Methods: In recent years, deep learning (DL) has attracted much attention and has been shown to have great power in a wide variety of problems, including community detection [14]. Deep learning-based methods for community detection can be classified into three main classes, including auto-encoder-based methods [15], generative adversarial network-based methods [14] and convolutional network-based methods [16]. Auto-encoders (AEs) [17] are neural network models that aim to minimize the error between input feature representation and output feature representation based on cross-entropy loss. Thus Auto-encoders can learn the latent structural features inside the data and perform clustering based on the latent structural features. To directly discover communities, many researchers have integrated clustering into the model [18]. Depending on the type of auto-encoder used, existing models can be divided into four types: stacked AE-based, where stacked AEs depict multi-level and dynamic information to flexibly reinforce wide community detection implementations [19], sparse AE-based, where a sparsity penalty is introduced in network hidden layers, denoising AE-based, where it aims to minimize the reconstruction loss between the input feature and output feature, and variational auto-encoders that is an extension of AEs with variational inference. Convolutional neural networks (CNNs) and graph convolutional networks (GCNs) are popular methods for community detection. CNNs are an effective method to extract Euclidean data features, such as image data, the convolution kernels of different layers can learn feature representations of different levels to improve the performance of downstream tasks. Xin et al. [20] proposed the first community detection model based on CNNs. They use max-pooling operators for network representation and a fully connected layer for community detection. Later, Cai et al. [21] proposed Community Network Local Modularity R (ComNet-R), classifying edges within and between communities by CNNs. ComNet-R prepares the independent preliminary communities by removing inter-community edges and merging communities based on local patterns. Many other CNNs-based methods have been proposed [22], but they showed some limitations when applied to non-Euclidian data and have yet to be up to the success they had known until then on Euclidean data. This is because non-Euclidian data does not have the spatial translation invariance of Euclidean data. Graph neural network [7] performs convolution operation based on message passing mechanism to successfully solve the problem that non-Euclidian data does not have spatial translation invariance. Sun et al. [23] propose a probabilistic generative model to collaboratively learn community membership and node representation. More recently, D. Bo et al. [16] presented GNN

[1]https://github.com/AutoMachine0/CommGNAS

algorithms that can operate directly for community detection in a graph. He et al. [24] derived node community membership in the hidden layer of an encoder and introduced a community-central dual decoder to reconstruct network structures and node attributes in an unsupervised manner. S. Bandyopad et al. [25] proposed the SEComm framework for the community detection task. The SEComm used self-supervised and self-representation learning to construct an unsupervised learning training data set. It trained the GCN and MLP using the training data set based on the combined loss function for the downstream unsupervised learning task. Despite their success, these approaches require dense manual work and domain knowledge. Our work aims to automatically design a GNN architecture with good performance for different graphs without labels to achieve community detection tasks.

B. Graph Neural Architecture Search

The target of the graph neural architecture search (GNAS) is finding the optimal architecture of the GNN model that will have a competitive performance for the targeted task. The existing method usually consists of three components: a search space S, a search algorithm A, and an estimation strategy E. The search algorithm A selects an architecture s from a predefined search space S. The estimated performance of s is calculated using the estimation strategy E to generate feedback for the search algorithm A. This section presents the related work about graph neural network architecture search comprehensively.

1) Method based on Reinforcement Learning: GNAS models based on the reinforcement learning method use a recurrent network as a controller to sample the descriptions of GNN architecture from the search space. Then, they compute the rewards of the sampled GNN architecture as feedback to maximize the expected performance of sampled GNN architecture on the validation accuracy. This method has been used in many works for GNAS [8], [26] with slight differences. Y. Ga et al. [8] proposed the GraphNAS framework based on reinforcement learning and constructed the first general graph neural network search space. They also adopted the parameter-sharing strategy to optimize efficiency to avoid training each model from scratch to convergence. K. Zho et al. [27] proposed Auto-GNN that uses multiple recurrent networks to search for different GNN architecture components. R. S. Sutton. [28] proposed a reinforcement neural architecture search using a controller-based reinforcement rule of policy gradient to gradually validate architectures with small steps.

2) Method based on Genetic Algorithm: The method based on genetic algorithm [29]–[31] use a genetic population-based meta-heuristic optimization algorithm, which is inspired by the mechanics of natural selection and natural genetics. In these works, the individual is described by a GNN architecture. However, because of the large size of the search space, it is almost impossible to determine the fitness of all individuals as training one GNNs is time-consuming, especially with a large graph data set. Another limit of this method is the slow convergence [7]. To alleviate these limitations, GraphPAS [31] proposes a parallel genetic algorithm that explores the search space in parallel, and GNN architecture distribution information entropy is used to constrain the direction of genetic search, which improves the search convergence of the genetic method.

3) Method based on Bayesian Optimization: The method based on bayesian optimization (BO) uses a probabilistic-based method combining a prior probability with a likelihood function. The process of bayesian optimization is adaptive, with predictions and uncertainty estimates updated as new observations are made. These models work by building a probability distribution over sampled candidates tested by the surrogate function and using the acquisition function to interpret the response from the surrogate function and efficiently search for candidate samples before selecting them to evaluate the actual objective function. The acquisition function is optimized at each step to determine the best sample for the next evaluation. Then, the model is updated, and the process is repeated until convergence. Although BO has achieved good performances in NAS [32], Yoon et al. [33] has designed a search algorithm with BO for GNAS. BO is computationally expensive and almost impossible for large-scale search space to work efficiently [7].

4) Method based on Differentiable Search: The basic idea of differential search is to use trainable architecture parameters to continue the whole search space to construct a hypernet [34]. To improve the efficiency of GNAS, Zhao et al. [35] proposed the macro and micro architecture of the GNN search space with an efficient differential search process. Wei et al. [36] designed the specific GNN search space for graph classification. It optimizes the differential search process with the characteristics of graph pooling operations for graph classification. Qin et al. [37] further analyzed the relationship between the exploration direction of differential search and graph feature distribution, and discussed the feasibility of the GNN architecture search with graph data optimization. Compared with the previous search method, the differential search method effectively improves the efficiency of GNAS.

III. Problem Definition

Given a graph $G = (V, E)$ where V is a set of nodes, and E is a set of edges. $v \in V$ is used to denote a node and $e_{uv} = (u, v) \in E$ denotes an edge directed from u to v. We assume that each node u carries properties also called features denoted by vectors $\mathbf{x}_u$. The node features of a graph are represented by a matrix $\mathbf{X} \in^{n \times d}$ where n stands for the number of nodes in the graph and d represents a node feature dimension. The goal of our work is to learn a partition function $f : V \to [K]$, where $[K] = \{1, 2, ..., K\}$ is a set of the community (cluster) indices to map each node to a community by exploiting the graph representation learning. The partition of function f should follow these principles: (1) There should be more internal connections between nodes partitioned into a community. (2) Node representations processed by the partition function f in the same community should have higher spatial similarity than node representations in other communities.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

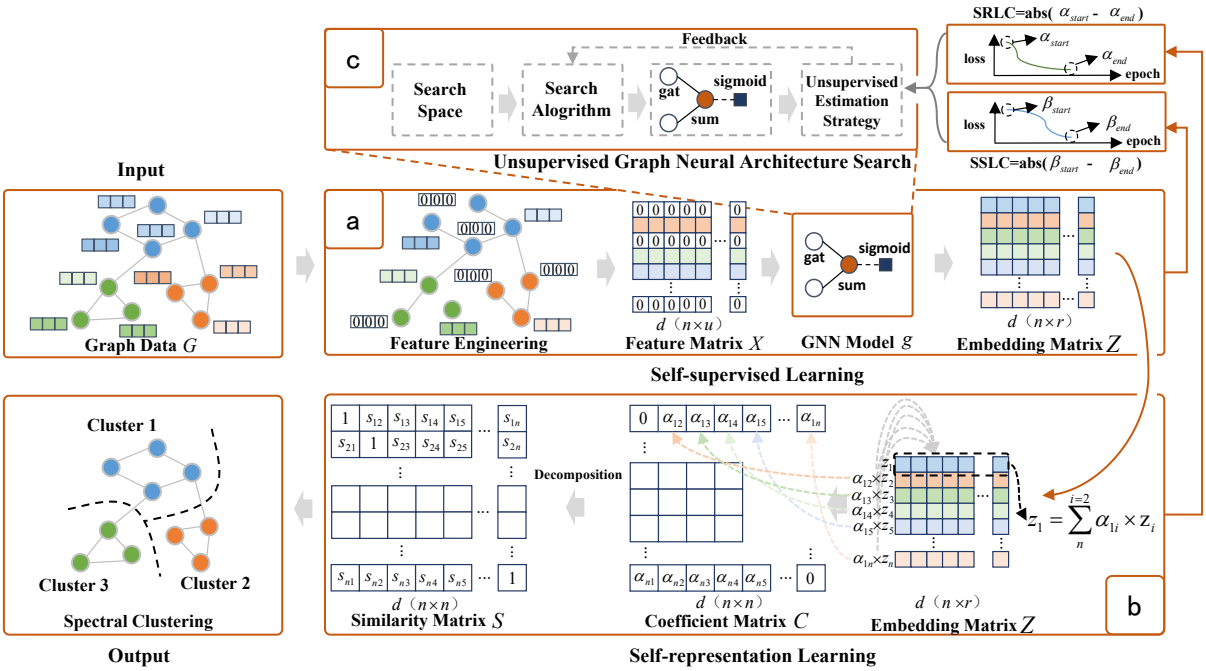

Fig. 2. CommGNAS framework. The unsupervised graph neural architecture search module uses the search algorithm to sample the GNN architecture from the search space and builds the GNN model g based on the sampled architecture. In the self-supervised learning module, CommGNAS gets the training graph data sets based on feature engineering. Then it uses the training graph data sets to train the GNN model g for getting self-supervised loss change (SSLC), where SSLC=abs(β_{start}-β_{end}), β_{start}, β_{end} stands for the loss value after the first training epoch and the last training epoch respectively in the self-supervised learning process. And using the trained GNN model g to encode the original node feature matrix X for obtaining the node embedding matrix Z. In the self-representation learning module, using a single-layer linear MLP to reconstruct the embedding matrix Z to get coefficient matrix C and self-representation loss change (SRLC), where SRLC=abs(α_{start}-α_{end}), α_{start}, α_{end} represents the loss value after the first training epoch and the last training epoch respectively in the self-representation learning process. Using matrix decomposition to obtain the similarity matrix S based on the coefficient matrix C. Finally, using similarity matrix S as input of spectral clustering to achieve community detection.

IV. Proposed Framework CommGNAS

CommGNAS is an unsupervised automatic graph neural network modeling framework for the community detection task, which consists of three parts. We first introduce the overall framework of CommGNAS and then introduce each part of the framework in detail. The first part is the self-supervised learning module, the second part is the self-representation learning module, and the last part is the unsupervised graph neural architecture search module.

A. Overview of CommGNAS

As shown in Figure 2, the graph data G with node features and topological structure is used as the input of CommGNAS. Then, the unsupervised graph neural architecture search module uses the search algorithm to sample the GNN architecture from the search space and builds the GNN model g based on the sampled architecture, as shown in sub-figure **c**. To encode the topological structure and node features of the graph data G, in the self-supervised learning module, first, we get training graph data sets X_1, X_2 from feature engineering and use them to train GNN model g for obtaining the self-supervised loss change (SSLC). We use the trained GNN model g to encode the original node feature matrix X for obtaining node embedding matrix Z and, as shown in sub-figure **a**. We construct a self-representation learning module to further enhance the spatial similarity of similar node features in the node embedding matrix Z. As shown in sub-figure **b**, we reconstruct the embedding matrix Z based on self-representation learning to get coefficient matrix C and self-representation learning loss change (SRLC), the goal of self-representation learning is to use other node feature vectors to represent the target node feature vector. We use the SSLC and SRLC to construct the unsupervised estimation strategy to generate feedback. The feedback is passed into the search algorithm and guides the search direction in the search space to get the optimal GNN g. Finally, we get the coefficient matrix C with the optimal GNN g sampled from the unsupervised graph neural architecture module, and use matrix decomposition to obtain the similarity matrix S based on the coefficient matrix C, we perform spectral clustering using the similarity matrix S as input to achieve community detection.

B. Self-supervised Learning

Self-supervised learning is a method for effectively mining the inherent characteristics of data without data labels, it can provide valuable feature representations for downstream tasks. Self-supervised learning [38] has been successfully applied to graph neural networks. Some previous works [39], [40] use it to obtain graph-level and node-level embedding representation. We use self-supervised learning based on the contrastive mechanism to encode the graph node feature matrix X. The formula is as follows:

$$Z = g(X, E), \tag{1}$$

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

where Z represents the node embedding matrix, g stands for the graph neural network model sampled by the unsupervised graph neural architecture search module, X is the node feature matrix, and E represents the edge relationship information after feature engineering. There are five parts in the self-supervised learning process of CommGANS, namely Feature Engineering, Feature Matrix X, GNN Model g, Embedding Matrix Z, and Self-supervised Loss Function.

1) Feature Engineering: Feature engineering consists of two sections. Section one is the node dropout operation, which randomly changes some node feature vectors to all zero vectors in the original node feature matrix X. The second section is the edge dropout operation, which randomly deletes some edges in the original graph data G.

2) Feature Matrix X: In the self-supervised learning module, to build a contrastive learning target, for the given graph data $G = (V, E)$, we perform feature engineering on the original graph data G to get different graph data G_1, G_2. We can obtain two different node feature matrices X_1, X_2 from the graph data G_1 and G_2. We use the node feature matrices X_1, X_2 as the input of the sampled GNN model g to get the output node embedding matrix Z_1 and Z_2. We construct a consistent number of positive and negative sample pairs as a training set for self-supervised learning based on node embedding matrix Z_1 and Z_2. Each positive sample pair consists of two nodes, which are from the same node corresponding to Z_1 and Z_2. The negative sample pair is composed of two cases. Case 1, the negative sample pair is composed of different nodes of the Z_1. Case 2, the negative sample pair is composed of different nodes of different node embedding matrices Z_1 and Z_2.

3) GNN Model g: In the unsupervised graph neural architecture search module, the search algorithm will sample GNN architecture from search space, and build sampled GNN model g based on the GNN architecture.

4) Embedding Matrix Z: When the training process of the sampled GNN model g is completed, we use the original node feature matrix X as the input of the sampled GNN model g to obtain the node embedding matrix Z.

5) Self-supervised Loss Function: We use cosine similarity to measure the distance between positive and negative sample pairs to construct a loss function. The goal of the loss function is to make the distance between the positive sample pairs closer and the distance between the negative sample pairs as far as possible, the formula is as follows:

$$L_{SS} = \sum_{i \in V} [log(\sum_{j \in V} e^{\frac{cos(z_{1,i}, z_{1,j})}{\tau}} + e^{\frac{cos(z_{1,i}, z_{2,j})}{\tau}}) - \frac{cos(z_{1,i}, z_{2,i})}{\tau}], \tag{2}$$

where $z_{1,i}$, $z_{1,j}$ represent the i-th node embedding feature vector and the j-th node embedding feature vector in the node embedding matrix Z_1. And $z_{2,i}$, $z_{2,j}$ stand for the i-th node embedding feature vector and the j-th node embedding feature vector in the node embedding matrix Z_2. The V is the set of nodes in graph G, and the τ is the temperature parameter. When the training is over, we use the first epoch loss value β_{start} to subtract the last loss value β_{end} to get the absolute value of the loss change $SSLC$, which will become part of the feedback signal.

C. Self-representation Learning

Inspired by the principle of self-representation learning [41], to further enhance the spatial similarity of features between similar nodes in the node embedding matrix Z obtained in the previous module, we use a single-layer linear MLP to build the self-representation learning module. We call the trainable parameter matrix of the single-layer linear MLP coefficient matrix C in this work. As shown in the formula (3).

$$Z_{sr} = MLP(Z; C), \tag{3}$$

where Z_{sr} represents the output feature matrix of self-representation learning based on the MLP with the coefficient matrix C. When the self-representation learning module training is completed, we can get the coefficient matrix C, which contains the intrinsic spatial similarity between each node in the graph. Then, based on the decomposition of the coefficient matrix C, we can obtain the similarity matrix S of the node embedding matrix Z. The self-representation learning of CommGNAS contains two important matrices and one loss function, namely Coefficient Matrix C, Similarity Matrix S, and Self-representation loss function.

1) Coefficient Matrix C: The core idea of self-representation learning is to use the linear combination of other node features vector in the node embedding matrix Z to represent the target node feature vector $z_i = \sum_{j \in V} \alpha_{i,j} z_j$, $j \neq i$. The coefficient matrix C is composed of $\alpha_{i,j}$ where the coefficient matrix C is a square matrix and the diagonal elements are zero.

2) Similarity Matrix S: To use spectral clustering to achieve community detection, we need to obtain the similarity matrix S of the node embedding feature Z. Based on the coefficient matrix C, a heuristic method [42] to calculate the similarity matrix S, the calculation process is as follows:

(1) $C^* = \frac{1}{2}(C + C^T)$;
(2) Decomposing matrix C^* using SVD, $C^* = U \sum V^T$;
(3) Computing matrix $R = U \sum^{\frac{1}{2}}$ and normalize each row of R;
(4) Obtaining matrix R^* by setting negative values in R to zero;
(5) Building similarity matrix $S = \frac{R^* + R^{*T}}{\|R\|_\infty}$.

3) Self-representation Loss Function: We use the square of the Frobenius norm to measure the difference between the target node embedding vector z_i and the linear combination of other node embedding vectors $\sum_{j \in V} \alpha_{i,j} z_j$. The destination of the loss function is to minimize the difference between them. The loss function is shown in the following formula:

$$L_{SR} = \|Z - Z_{sr}\|_F^2 + \lambda \|C\|_F^2, \tag{4}$$

where Z represents the node embedding matrix, C stands for the coefficient matrix, $\|\cdot\|_F^2$ represents the square of matrix Frobenius norm, $\|C\|_F^2$ is the regularization term of the loss function, and λ is the regularization strength. When the self-representation learning training process is completed, we use

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

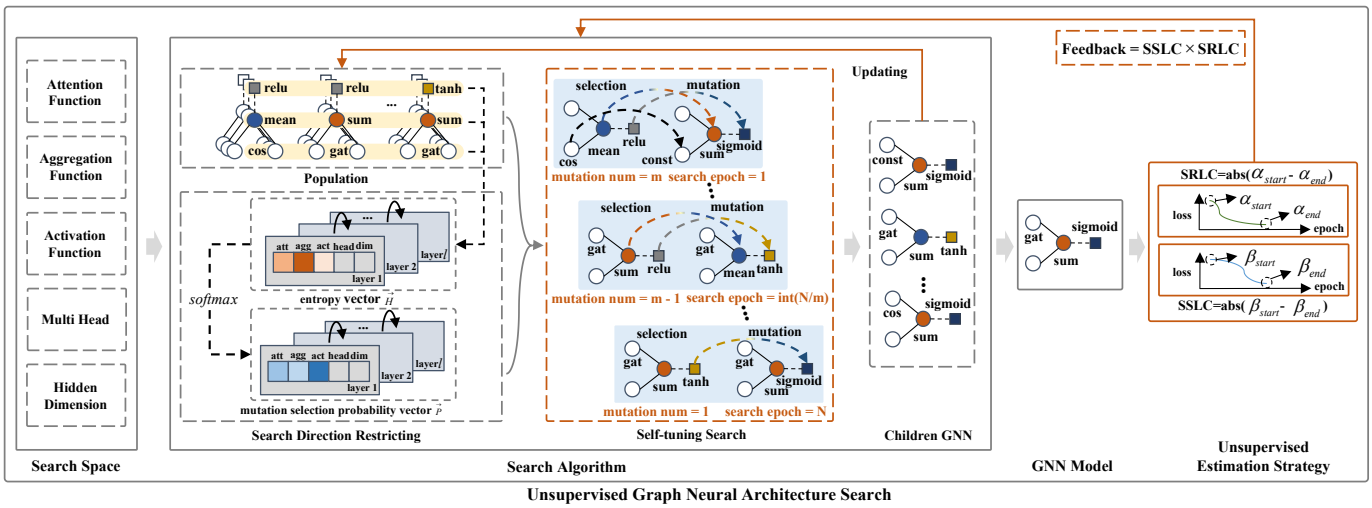

Fig. 3. Unsupervised Neural Architecture Search Framework. Randomly sampling N GNN architectures from the search space, and using an unsupervised estimation strategy to get the fitness of each architecture. Selecting the top k GNN architectures as the population based on fitness. Calculating the information entropy vector $\vec{H}$ using the architecture component value frequency distribution in the population. Computing mutation selection probability p_i based on information entropy vector $\vec{H}$. Getting the mutation selection probability vector $\vec{P}$. Using self-tuning search to sample child GNN architectures. Using the product of SSLC and SRLC as fitness for each child's GNN architectures. Where SSLC=abs(β_{start}-β_{end}), β_{start}, β_{end} stands for the loss value after the first training epoch and the last training epoch respectively in the self-supervised learning process. And SRLC=abs(α_{start}-α_{end}), α_{start}, α_{end} represents the loss value after the first training epoch and the last training epoch respectively in the self-representation learning process. Collecting the child GNN architectures generated by the self-tuning search process as the children. If the fitness of the child GNN architecture is greater than the threshold, adding the child architecture into the population.

the same method as the self-supervised module to calculate the self-representation loss change (SRLC). SRLC forms another part of the feedback signal generated by the unsupervised evaluation strategy.

D. Unsupervised Graph Neural Architecture Search

Inspired by the study of the learning rate decay strategy [43], we proposed the self-turning search strategy based on the GraphPAS [31] search algorithm for the CommGNAS framework to improve the search performance further. In this module, we construct an effective unsupervised evaluation using the product of the self-supervised loss change (SSLC) and the self-representation loss change (SRLC) for generating feedback to update the search algorithm. The framework of the self-tuning graph neural architecture search module is shown in Figure 3.

1) Search Space: In this work, we use a search space containing five graph neural network architecture components. Different graph neural network components have different functions.

- **Attention Function**: The attention function calculates the correlation coefficient $a_{i,j}$ between the center node and neighbor nodes, which can measure the contribution of different neighbor node features to the center node representation. The table I shows the different attention functions we used in this work, where d_i represents the degree of node i, h_i stands for the representation of node i, W is the learnable parameter vector, and $*$ represents the matrix multiplication.
- **Aggregation Function**: This function aggregates the neighbor node representations of the current layer with correlation coefficients to obtain the central node representation of the next layer. The aggregation function is the key operation in the graph neural network.
  Attention Head: Calculating multiple independent attention correlation coefficients for the central node can effectively stabilize the learning process, which is crucial for the rapid convergence of the GNN model.
- **Hidden Dimension**: Effectively transforming the dimensions of the original features based on the learnable matrix can enhance the expression of the hidden layer. It can help the GNN model achieve better performance.
- **Activation Function**: The activation function gives the model the ability to fit nonlinear data, and different activation functions have different effects on the performance of the GNN model.

TABLE I
ATTENTION FUNCTIONS

Attention Class	Function
const	$a_{ij}^{const} = 1$
gcn	$a_{ij}^{gcn} = 1/\sqrt{d_i d_j}$
gat	$a_{ij}^{gat} = reaky\_relu(W_c * h_i + W_n * h_j)$
sym-gat	$a_{ij}^{sym} = a_{ij}^{gat} + a_{ji}^{gat}$
linear	$a_{ij}^{lin} = tanh(sum(W_c * h_i))$
cos	$a_{ij}^{cos} =< W_c * h_i, W_n * h_j >$
gene-linear	$a_{ij}^{gen} = W_b * tanh(W_c * h_i + W_n * h_j)$

The functions of different graph neural network components in the search space are shown in the table II

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

TABLE II
SEARCH SPACE

Component	Candidate Values
Attention function	listed in TABLE I
Aggregation function	sum, $mean$, max
Attention head	1, 2, 4, 8
Hidden dimension	8, 16, 32, 64 128, 256
Activation function	$tanh$, $sigmoid$, $relu$, $linear$, $relu6$, elu, $leaky\_relu$, $softplus$

2) Search Algorithm: The learning rate decay strategy makes the learning rate decay automatically as the training epoch increases during the training process. Related studies [43] have proved that the learning rate decay strategy can make the optimization function converge faster and achieve better optimization results. Inspired by the learning rate decay strategy, we propose a self-tuning search algorithm with the exploration-intensity decay strategy. When the search algorithm samples the graph neural network architecture in the search space, the exploration intensity will decrease with the search epoch increases. The unsupervised graph neural architecture search module includes the following processes:

- **Population**: Randomly sampling N GNN architectures from the search space, and using an unsupervised estimation strategy to get the fitness of each architecture. Then, select the top k GNN architectures as the population based on fitness.
- **Search Direction Restricting**: Calculating the information entropy $h(c_i)$ by Eq. 5 using the architecture component value frequency distribution in the population.

$$\begin{aligned} h(c_i) &= -\sum_j f(v_j) log_2 f(v_j) \\ i &\in [1, 5 \times l], l \in N^*, \end{aligned} \tag{5}$$

  where $h(c_i)$ stands for the information entropy of the i-th architecture component, v_j represents the j-th value of the i-th component in the population, $f(v_j)$ denotes frequency of occurrence of v_j in the population, l is the number of GNNs layer. Next, achieving the information entropy vector $\vec{H} = [h(c_1), h(c_2), ..., h(c_i)]$. After, computing mutation selection probability p_i based on Eq. 6. Finally, getting the mutation selection probability vector $\vec{P} = [p_1, p_2, ..., p_i]$.

$$\begin{aligned} p_i &= softmax(h(c_i)) \\ i &\in [1, 5 \times l], l \in N^*. \end{aligned} \tag{6}$$

- **Self-tuning Search**: Using wheel strategy to select t parent individuals based on fitness from the population. Perform mutation operation on every parent individual based on mutation-selection probability vector P and mutation intensity m to get t new child GNN architectures, in which each parent GNN architecture will select m components based on probability vector P for random mutation. For the self-tuning search strategy, assuming that the total number of search epochs is N, in the search process, each time the search epoch increases by $int(\frac{N}{m})$, which int represents rounding, the mutation intensity m will automatically decrease by one until mutation intensity m is equal to one.
- **Children GNN**& **Updating**: Collecting the GNN architectures generated by the self-tuning search process as the children GNN. If the fitness of the children GNN architecture is greater than the threshold, add the children architecture to the population.

3) Unsupervised Estimation Strategy: The evaluation strategy of the traditional graph neural architecture search (GNAS) method is designed based on the supervised learning problem, so the traditional GNAS method cannot solve the problem of unsupervised learning. To overcome this challenge, we propose an unsupervised learning estimation strategy based on self-supervised and self-representation learning in CommGNAS. We use the product of the self-supervised loss change (SSLC) and the self-representation loss change (SRLC) to construct an unsupervised estimation strategy to generate feedback for the search algorithm.

In supervised learning, the loss convergence value and loss change are important indicators to measure whether the model is under-fitting. The loss change refers to the absolute value of the difference between the first epoch loss value and the last epoch loss value at the end of the model training. In our CommGNAS framework, the loss function of self-supervised and self-representation learning is similar to the optimization objective of the supervised learning loss function. When the loss value of self-supervised and self-representation learning converges to a smaller value or loss change is large, the model can better mine the inherent features and similarity relationships of unlabeled data. The CommGNAS includes self-supervised and self-representation learning processes. We need to use the loss information generated in the two unsupervised learning processes to guide the search algorithm to sample GNN architecture in the search space. The effective use of loss change information is the key to constructing an unsupervised evaluation strategy. To reduce the hyperparameters of the unsupervised graph neural architecture search module and to avoid the coupling problem caused by the inconsistent magnitude of the two different loss changes, we use the product of two-loss changes to construct our unsupervised evaluation strategy. The formula is as follows:

$$\begin{aligned} Feedback = &|\alpha_{start} - \alpha_{end}| \\ &\times |\beta_{start} - \beta_{end}|, \end{aligned} \tag{7}$$

where α_{start}, α_{end} represents the loss value after the first training epoch and the last training epoch respectively in the self-representation learning process. β_{start}, β_{end} stands for the loss value after the first training epoch and the last training epoch respectively in the self-supervised learning process. The meaning of $|\cdot|$ is to take the absolute value.

In this work, the optimal GNN architecture designed by CommGNAS for different data sets is shown in Figure 4.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

TABLE III
PERFORMANCE OF COMMUNITY DETECTION BY COMMGNAS AND OTHER BASELINE METHODS.

data set	Metrics	VAE	VGAE	MGAE	ARGE	ARVGE	AGC	GUCD	SEComm	**CommGNAS**
Cora	**F1-score**	41.97%	41.50%	38.01%	61.90%	62.70%	65.61%	-	73.94%	**74.50**%
	NMI	40.69%	38.45%	45.57%	44.90%	45.00%	53.68%	32.3%	56.04%	**58.68**%
	Acc	53.25%	55.95%	63.43%	64.00%	63.80%	68.92%	50.5%	75.92%	**75.96**%
Citeseer	**F1-score**	29.13%	31.88%	39.49%	54.60%	52.90%	62.48%	-	60.25%	**62.58**%
	NMI	18.34%	22.71%	39.75%	35.00%	26.10%	41.13%	27.43%	42.53%	**42.87**%
	Acc	41.26%	44.38%	63.56%	57.30%	54.40%	67.00%	54.47%	69.82%	**70.06**%
Wiki	**F1-score**	15.35%	20.49%	39.20%	38.27%	37.80%	40.36%	-	44.48%	**47.26**%
	NMI	17.33%	30.28%	47.97%	39.50%	40.01%	45.28%	-	51.38%	**54.48**%
	Acc	53.25%	28.67%	50.14%	41.40%	41.55%	47.65%	-	53.10%	**56.05**%

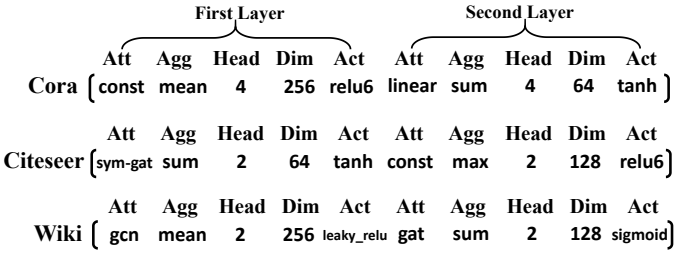

Fig. 4. An example of the optimal GNN architecture designed by CommGNAS for different data sets on community detection.

E. Spectral Clustering

Spectral clustering is an unsupervised clustering model with simple principles and better performance than traditional clustering methods such as K-means. When CommGNAS obtain the optimal GNN architecture based on an unsupervised graph neural architecture module, it will use the optimal GNN architecture to construct the GNN model g. Then, CommGNAS trains the optimal GNN model g with the self-supervised learning module and achieve the node embedding matrix Z. Next, the framework uses the node embedding matrix Z as the input of the self-representation learning module to get coefficient matrix C. After, based on the coefficient matrix C, CommGNAS performs the matrix decomposition operation to construct the similarity matrix S of the node embedding matrix Z. Finally, output the final community detection results based on the similarity matrix S with spectral clustering.

V. EXPERIMENT RESULTS

In this section, we first present the experimental data set. Then, we show the setup used in our experiment. Next, we briefly introduce the baseline methods. Finally, we will introduce our performance experiment and ablation experiment results.

A. Datasets

To evaluate the performance of the CommGNAS framework, we use three publicly graph data sets [44], [45]. Data set statistics are summarized in Table IV. Cora and Citeseer are citation data sets in which nodes correspond to papers and edges connect nodes if one paper references another paper in the data set. Cora contains seven ground truth labels for unsupervised learning performance evaluation. Each category indicates which machine learning method the paper belongs to, such as genetic learning methods, reinforcement learning neural networks, etc. Similarly, Citeseer includes six ground truth labels. Wiki is a collection of webpages in which nodes are individual webpages that are linked together if one links to another. And Wiki data set contains seventeen ground truth labels. Each of these data sets has an attribute feature vector for each node.

TABLE IV
BENCHMARK GRAPH DATA SETS.

data set	Nodes	Edges	Node Features	Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Wiki	2,405	17,981	4,973	17

B. Experiment Setup

For our proposed CommGNAS framework, the hyperparameters we used in this work for different modules are as follows:

- **Self-supervised Learning Module**: In order to get training graph data sets G_1 and G_2, we set the edge dropout rate to 0.2 and 0.4 respectively for G_1, G_2, and the node dropout rate to 0.6 for two graph data sets. For the training process of GNN model g, we set the model learning rate to 5×10^{-3} and the L_2 regularization intensity to 5×10^{-4}, the training epoch is 200. We choose $Adam$ as the optimization function. For the temperature parameter in the self-supervised loss function, we set it as 0.4.
- **Self-representation Learning Module**: In this module, we use $Adam$ as the optimizer, where the learning rate is set to 10^{-3}, and the L2 regularization strength is set to 10^{-5}. The regularization strength in the self-representation learning loss function is 0.5. The training epoch is 80.
- **Unsupervised Graph Neural Architecture Search Module**: We set the number of random initialization GNN architectures to 100, and the initial mutation m is set to 4, and the initial population size is 5. The parent size is 5 for every search epoch and the search epoch is 100. The sampled GNN architecture layer is fixed to 2. In

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

TABLE V
ABLATION STUDY OF COMMGNAS.

data set	Cora			Citeseer			Wiki		
Metrics	F1	NMI	Acc	F1	NMI	Acc	F1	NMI	Acc
CommGNAS-with-GCN	69.99%	58.11%	73.74%	58.09%	37.97%	62.70%	41.75%	52.44%	52.44%
CommGNAS-with-GAT	72.80%	56.26%	73.34%	59.30%	37.46%	61.98%	39.24%	46.11%	48.36%
CommGNAS-w/o-SR	65.86%	58.37%	72.30%	61.68%	40.76%	65.04%	45.71%	52.54%	53.47%
CommGNAS	**74.50%**	**58.68%**	**75.96%**	**62.58%**	**42.87%**	**70.06%**	**47.26%**	**54.48%**	**56.05%**

the GNN architecture search process, the training epoch of sampled GNN model g is 100 in the self-supervised learning module.

C. Method Baseline

To demonstrate the superiority of CommGNAS, we compare the optimal GNN architectures identified by our CommGNAS framework against a set of unsupervised GNNs for community detection tasks. GNN-based techniques have the advantage of being able to make use of both the topological structure and the node attribute of the graph. In our experiments, we use the methods that learn the node embeddings and then utilize clustering algorithms on the node embeddings to form groups including graph autoencoder (GAE) and graph variational autoencoder (VGAE) [46], adversarially regularized graph autoencoder (ARGE) and variational graph autoencoder (ARVGE) [47], and marginalized graph autoencoder (MGAE) [48]. We also use the recent community detection methods including GUCD [24], which employs an auto-encoder-based framework to obtain direct community category for every node in the graph, and AGC [44], which uses high-order graph convolution to get node embeddings and detect communities via spectral clustering on the embeddings. and SEComm [25], an end-to-end GCN model based on self-supervised and self-representation learning mechanisms for community detection.

D. Performance of Community Detection

The comparison of performance experiment results on three data sets is reported in Table III. We use three popular metrics to evaluate the quality of community detection including macro F1-score (F1), normalized mutual information (NMI), and clustering accuracy (Acc). We use ground truth community labels of the nodes only to get the performance of these quality metrics.

We have gotten the best results from the available literature [25]. We use the symbol $-$ to represent the result of that algorithm for the data set that is not publicly available. The average improvement of our framework to the best baseline is 0.56%, 2.64%, and 0.04% for F1-score, NMI, and Acc on the Cora data set and 0.1%, 0.34% and 0.24% on Citeseer data set, which justifies the effectiveness of our model. Particularly, our proposed model also outperforms other best baselines by 2.78% for F1-score, 3.1% for NMI, and 2.8% for Acc on the Wiki data set. The better performance of our framework is attributed to the fact that (a) CommGNAS can automatically design the optimal GNN model from the search space to encode the node feature, it can better mine the internal structure features of graph data, and (b) our proposed unsupervised estimation strategy can effectively guide the search direction of the search algorithm to sample good GNN architectures for the downstream unsupervised tasks. Specifically, in the self-supervised learning module, contrast-based self-supervised learning can effectively mine the internal structural features and correlations between nodes in graph data. In the self-representation module, the coefficient matrix obtained from the method of reconstructing the node embedding matrix can enhance the spatial similarity of the nodes with similar features in the graph data. At the same time, the similarity matrix generated by the coefficient matrix decomposition can well represent the internal spatial similarity between similar nodes in the graph data. In the unsupervised GNAS module, the unsupervised estimation strategy constructed by the self-supervised and self-representation learning loss changes can generate effective feedback to update the search algorithm for sampling the better GNN architectures. In the process of sampling the GNN architecture, the search algorithm based on the self-tuning strategy can automatically tune the intensity of the exploration, which enhances the performance of the search.

E. Ablation Study

Since there are different modules in CommGNAS, we analyze their impacts via an ablation study. Three variants of CommGNAS are designed to further explore the different contribution components of CommGNAS. Table V reports the performance of our framework and its several variants. We introduce the variants and analyze their effect respectively. CommGNAS-with-GCN is the model variant that we use the GCN model to replace the optimal GNN model sampled from the unsupervised graph neural architecture search module. The variant of CommGNAS-with-GAT is similar to the variant of CommGNAS-with-GCN, the only difference is that we use the GAT model instead of the optimal GNN model. CommGNAS-w/o-SR is the model variant in that we remove the self-representation module and use k-means to complete the community detection based on the node embedding matrix. It can be known from the results of ablation experiments, that compared with the manually designed GNN model, CommGANS can automatically sample the GNN model with better performance based on an unsupervised GNAS module for the community detection task. This is because our proposed unsupervised GNAS module can automatically design different graph neural network architectures based on different graph data distributions. The result based on CommGNAS-w/o-SR illustrates that self-representation learning can effectively enhance the spatial similarity of similar nodes in graph data.

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2023.3270181

F. Visualization of Node Embeddings

Finally, we provide qualitative results by visualizing the node embeddings learned with SEComm and CommGNAS. Specifically, we use t-SNE [49] to reduce the node embedding to a two-dimensional space from the high-dimensional space. As shown in Figure 5, each node is colored with its corresponding ground-truth class label for Citeseer. As is seen in Figure 5 (b), the node representations learned with CommGNAS exhibit better discernible clusters in the projected two-dimensional space than SEComm. The much clearer cluster structures of learned node representations compared to SEComm verify that the unsupervised estimation strategy we constructed can guide the CommGNAS to design an effective GNN model for community detection.

(a) SEComm(Citeseer) (b) CommGNAS(Citeseer)

Fig. 5. Visualization of node embeddings learned with SEComm and CommGNAS on the Citeseer. t-SNE is applied to project node embeddings into two-dimensional spaces. Each node is colored with its corresponding ground-truth class label.

VI. Conclusion

In this work, we propose CommGNAS, an unsupervised graph neural network automatic modeling framework for the community detection task. In the CommGNAS, we propose an unsupervised estimation strategy based on self-supervised and self-representation learning, which enables GNAS to solve unsupervised problems effectively. We use the product of self-supervised loss change and self-representation loss change to construct an unsupervised estimation strategy to generate feedback. The search algorithm based on the self-tuning search strategy can sample the better performance of GNN architecture for the community detection task. The experiment results show that CommGNAS can achieve state-of-the-art performance on all the data sets that we used on the community detection task.

References

[1] C.-C. Lin, D.-J. Deng, and S.-Y. Jhong, "A triangular nodetrix visualization interface for overlapping social community structures of cyber-physical-social systems in smart factories," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 1, pp. 58–68, 2020.

[2] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[3] T. Song, W. Zheng, S. Liu, Y. Zong, Z. Cui, and Y. Li, "Graph-embedded convolutional neural network for image-based eeg emotion recognition," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1399–1413, 2022.

[4] Z. Chen, L. Li, and J. Bruna, "Supervised community detection with line graph neural networks," in *Proceedings of the International Conference on Learning Representations*, 2019, pp. 1–12.

[5] L. Hutton and T. Henderson, "Toward reproducibility in online social network research," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 156–167, 2018.

[6] X. Huang, H. Cheng, and J. X. Yu, "Dense community detection in multi-valued attributed networks," *Information Sciences*, vol. 314, pp. 77–99, 2015.

[7] B. M. Oloulade, J. Gao, J. Chen, T. Lyu, and R. Al-Sabri, "Graph neural architecture search: A survey," *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 692–708, 2021.

[8] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2020, pp. 1403–1409.

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

[10] D. Jin, Z. Liu, W. Li, D. He, and W. Zhang, "Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 152–159.

[11] J. Kim and J.-G. Lee, "Community detection in multi-layer graphs: A survey," *ACM SIGMOD Record*, vol. 44, no. 3, pp. 37–48, 2015.

[12] M. E. Newman, "Spectral methods for community detection and graph partitioning," *Physical Review E*, vol. 88, no. 8, p. 042822, 2013.

[13] S. Soor, A. Challa, S. Danda, B. S. D. Sagar, and L. Najman, "Iterated watersheds, A connected variation of k-means for clustering GIS data," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 626–636, 2021.

[14] L. Yang, Y. Wang, J. Gu, C. Wang, X. Cao, and Y. Guo, "JANE: jointly adversarial network embedding," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2020, pp. 1381–1387.

[15] P. Zhang, F. Xiong, H. Leung, and W. Song, "Funkr-pdae: Personalized project recommendation using deep learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 886–900, 2021.

[16] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proceedings of the International World Wide Web Conference*, 2020, pp. 1400–1410.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] J. Cao, D. Jin, and J. Dang, "Autoencoder based community detection with adaptive integration of network topology and node contents," in *Proceedings of the International conference on knowledge science, engineering and management*, 2018, pp. 184–196.

[19] F. Liu, J. Wu, S. Xue, C. Zhou, J. Yang, and Q. Sheng, "Detecting the evolving community structure in dynamic social networks," *World Wide Web*, vol. 23, pp. 715–733, 2020.

[20] X. Xin, C. Wang, X. Ying, and B. Wang, "Deep community detection in topologically incomplete networks," *Physica A: Statistical Mechanics and its Applications*, vol. 469, pp. 342–352, 2017.

[21] B. Cai, Y. Wang, L. Zeng, Y. Hu, and H. Li, "Edge classification based on convolutional neural networks for community detection in complex network," *Physica A: Statistical Mechanics and its Applications*, vol. 556, p. 124826, 2020.

[22] A. D. Santo, A. Galli, V. Moscato, and G. Sperlì, "A deep learning approach for semi-supervised community detection in online social networks," *Knowledge-Based Systems*, vol. 229, p. 107345, 2021.

[23] F. Sun, M. Qu, J. Hoffmann, C. Huang, and J. Tang, "vgraph: A generative model for joint community detection and node representation learning," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2019, pp. 512–522.

[24] D. He, Y. Song, D. Jin, Z. Feng, B. Zhang, Z. Yu, and W. Zhang, "Community-centric graph convolutional network for unsupervised community detection," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2020, pp. 3515–3521.

[25] S. Bandyopadhyay and V. Peter, "Unsupervised constrained community detection via self-expressive graph neural network," in *Proceedings of the Uncertainty in Artificial Intelligence Conference*, 2021, pp. 1078–1088.

[26] H. Zhao, L. Wei, and Q. Yao, "Simplifying architecture search for graph neural network," in *Proceedings of the International Conference on Information and Knowledge Management*, 2020, pp. 1–12.

[27] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-gnn: Neural architecture search of graph neural networks," *arXiv preprint arXiv:1909.03184*, 2019.

[28] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 1999, pp. 1057–1063.

[29] M. Shi, D. A. Wilson, X. Zhu, Y. Huang, Y. Zhuang, J. Liu, and Y. Tang, "Evolutionary architecture search for graph neural networks," *arXiv preprint arXiv:2009.10199*, 2020.

[30] Y. Li and I. King, "Autograph: Automated graph neural network," in *Proceedings of the International Conference on Neural Information Processing*, 2020, pp. 189–201.

[31] J. Chen, J. Gao, Y. Chen, M. B. Oloulade, T. Lyu, and Z. Li, "Graphpas: Parallel architecture search for graph neural networks," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2182–2186.

[32] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 7603–7613.

[33] M. Yoon, T. Gervet, B. Hooi, and C. Faloutsos, "Autonomous graph mining algorithm search with best speed/accuracy trade-off," in *Proceedings of the IEEE International Conference on Data Mining*, 2020, pp. 751–760.

[34] H. Zhao, Q. Yao, and W. Tu, "Search to aggregate neighborhood for graph neural network," in *Proceedings of the IEEE International Conference on Data Engineering*, 2021, pp. 552–563.

[35] Y. Zhao, D. Wang, X. Gao, R. Mullins, P. Lio, and M. Jamnik, "Probabilistic dual network architecture search on graphs," *arXiv preprint arXiv:2003.09676*, 2020.

[36] L. Wei, H. Zhao, Q. Yao, and Z. He, "Pooling architecture search for graph classification," in *Proceedings of the ACM International Conference on Information & Knowledge Management*, 2021, pp. 2091–2100.

[37] Y. Qin, X. Wang, Z. Zhang, and W. Zhu, "Graph differentiable architecture search with structure learning," in *Proceedings of the Advances in Neural Information Processing Systems*, 2021, pp. 16 860–16 872.

[38] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.

[39] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*, 2019.

[40] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *arXiv preprint arXiv:1809.10341*, 2018.

[41] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[42] P. Ji, M. Salzmann, and H. Li, "Efficient dense subspace clustering," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2014, pp. 461–468.

[43] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?" *arXiv preprint arXiv:1908.01878*, 2019.

[44] X. Zhang, H. Liu, Q. Li, and X. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019, pp. 4327–4333.

[45] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017, pp. 1–12.

[46] Kipf, Thomas N and Welling, Max, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[47] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018, pp. 2609–2615.

[48] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: marginalized graph autoencoder for graph clustering," in *Proceedings of the ACM on Conference on Information and Knowledge Management*, 2017, pp. 889–898.

[49] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, pp. 2579–2605, 2008.

Jianliang Gao received the Ph.D. degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences, China. He is currently a professor at the School of Computer Science and Engineering, Central South University, China. He is the general chair of the 2016 IEEE Conference on Big Data. His main research interests include automatic machine learning and graph data mining.

Jiamin Chen received the B.S. degree in applied physics from Nanchang University, Nanchang, Jiangxi, China in 2014, and the M.S. degree in radio physics from Nanchang University, Nanchang, Jiangxi, China in 2017. He is currently a Ph.D. candidate at the School of Computer Science and Engineering at Central South University, Changsha, Hunan, China. His research interests are automatic machine learning and graph neural networks.

Babatounde Moctard Oloulade received a B.S. degree in computer science from the National School of Applied Economics and Management, University of Abomey-Calavi, Cotonou, Benin in 2012. He received the M.S. degree in Computer Science and Technology from Wuhan University of Technology, Wuhan, China in 2020. He is currently a Ph.D. student at the School of Computer Science and Engineering, Central South University, Changsha, China. His current research interests include automatic machine learning and graph neural networks.

Raeed Al-sabri received the B.S. degree in Information Technology from Thamar university, Thamar, Yemen in 2011, and the M.S. degree in Computer Science and Technology from Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China in 2020. He is currently a Ph.D. candidate at the School of Computer Science and Engineering, Central South University, Changsha, China. His research interests are natural language processing, machine learning, and graph neural networks.

Tengfei Lyu received the B.S. degree from Bohai University, Jinzhou, Liaoning, China, in 2019. He received his M.S. degree from Central South University, Changsha, China, in 2022. His research interests include machine learning and graph neural networks.

Ji Zhang is currently a full Professor in computer science at the University of Southern Queensland (USQ), Australia. He is an IET Fellow, IEEE Senior Member, ACM member, Australian Endeavour Fellow, Queensland International Fellow (Australia), and Izaak Walton Killam Scholar (Canada). His research interests are big data analytics, knowledge discovery and data mining (KDD), and computational intelligence. He received his degree of Ph.D. from the Faculty of Computer Science at Dalhousie University, Canada in 2009. He has published over 230 papers in major peer-reviewed international journals and conferences.

Zhao Li obtained his Ph.D. in computer science at the University of Vermont with an excellent graduate award in 2012. He is currently leading the adversarial intelligence team as a senior staff scientist and technical director in the Taobao division at Alibaba Group. His research interest lies in multi-agent reinforcement learning, big data-driven security, and large-scale graph computing. He has published several papers in prestigious conferences and journals including WWW, AAAI, ICDE, VLDB, TKDE, etc. He also won the CIKM Analyticup Champion in 2019.