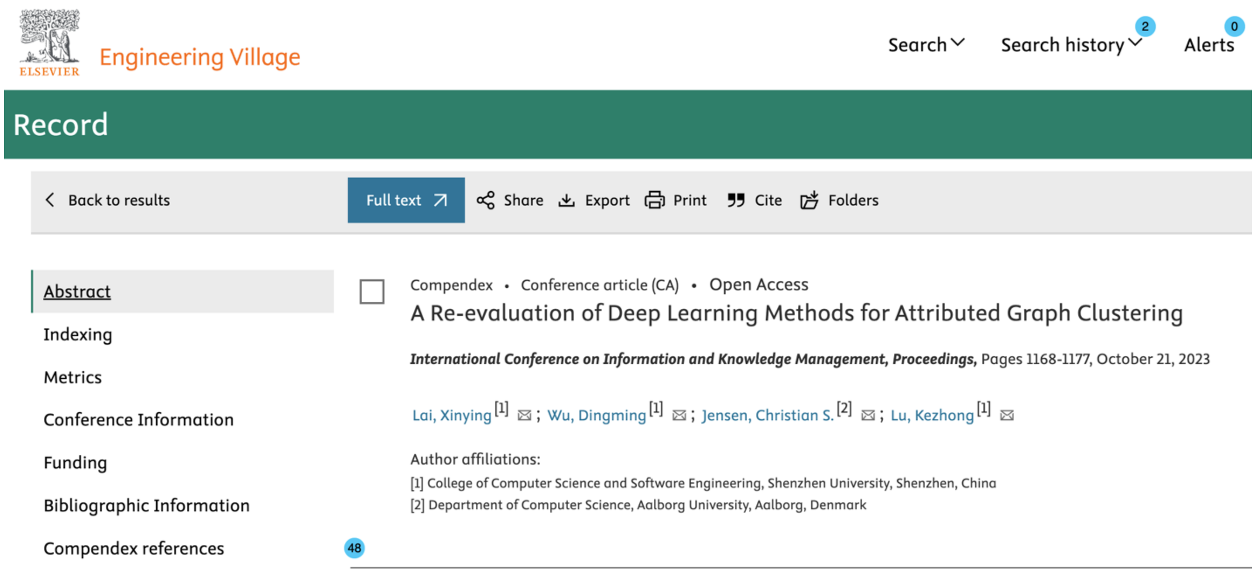


Record

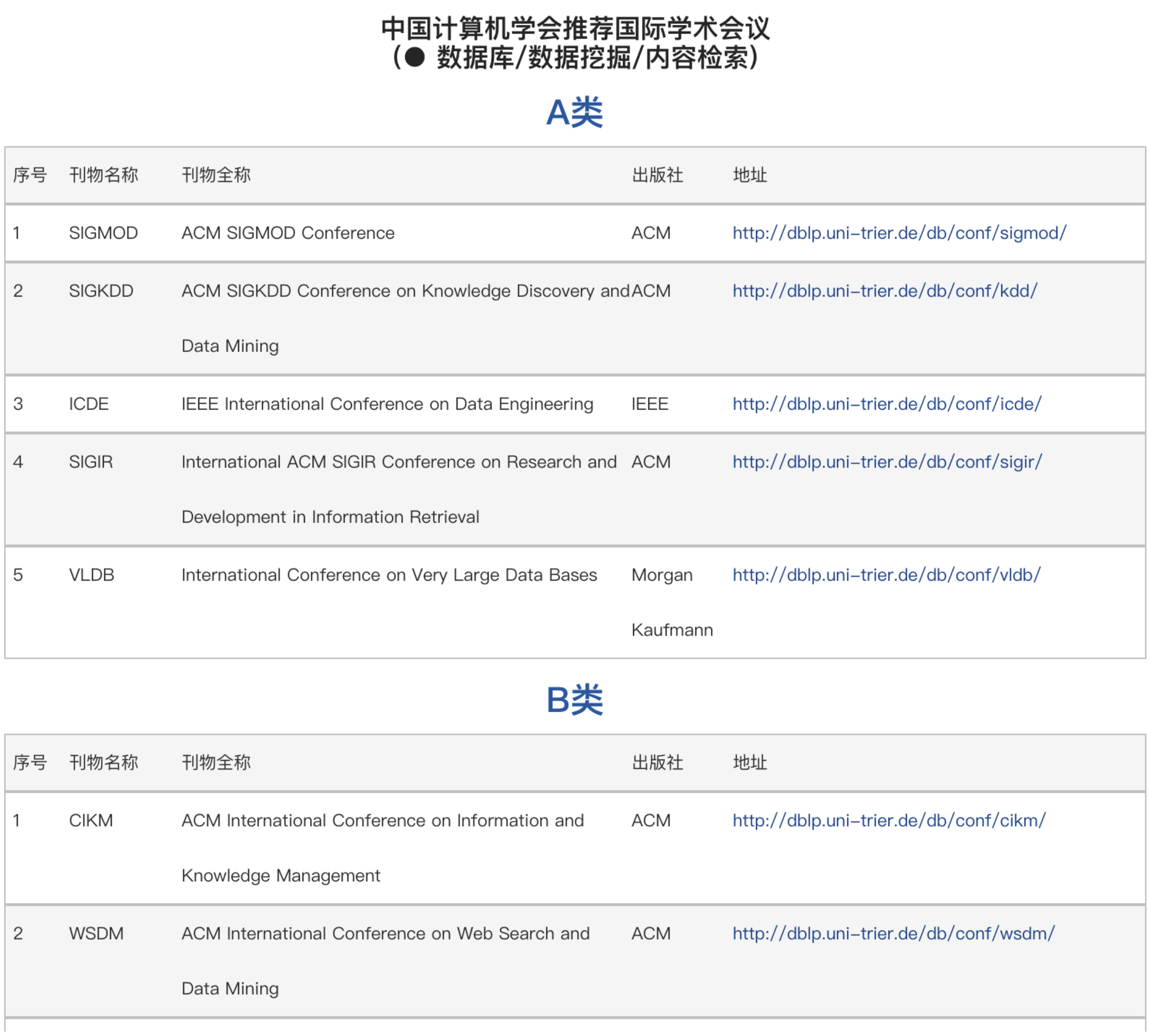
Compendex • Conference article (CA) • Open Access

A Re-evaluation of Deep Learning Methods for Attributed Graph Clustering

International Conference on Information and Knowledge Management, Proceedings, Pages 1168-1177, October 21, 2023

Lai, Xinying [1]; Wu, Dingming [1]; Jensen, Christian S. [2]; Lu, Kezhong [1]

Author affiliations:
[1] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[2] Department of Computer Science, Aalborg University, Aalborg, Denmark



中国计算机学会推荐国际学术会议
（● 数据库/数据挖掘/内容检索）

A类

序号	刊物名称	刊物全称	出版社	地址
1	SIGMOD	ACM SIGMOD Conference	ACM	http://dblp.uni–trier.de/db/conf/sigmod/
2	SIGKDD	ACM SIGKDD Conference on Knowledge Discovery and Data Mining	ACM	http://dblp.uni–trier.de/db/conf/kdd/
3	ICDE	IEEE International Conference on Data Engineering	IEEE	http://dblp.uni–trier.de/db/conf/icde/
4	SIGIR	International ACM SIGIR Conference on Research and Development in Information Retrieval	ACM	http://dblp.uni–trier.de/db/conf/sigir/
5	VLDB	International Conference on Very Large Data Bases	Morgan Kaufmann	http://dblp.uni–trier.de/db/conf/vldb/

B类

序号	刊物名称	刊物全称	出版社	地址
1	CIKM	ACM International Conference on Information and Knowledge Management	ACM	http://dblp.uni–trier.de/db/conf/cikm/
2	WSDM	ACM International Conference on Web Search and Data Mining	ACM	http://dblp.uni–trier.de/db/conf/wsdm/

A Re-evaluation of Deep Learning Methods for Attributed Graph Clustering

Xinying Lai
College of Computer Science & Software Engineering, Shenzhen University
Shenzhen, China
2100271064@email.szu.edu.cn

Dingming Wu
College of Computer Science & Software Engineering, Shenzhen University
Shenzhen, China
dingming@szu.edu.cn

Christian S. Jensen
Department of Computer Science, Aalborg University
Aalborg, Denmark
csj@cs.aau.dk

Kezhong Lu
College of Computer Science & Software Engineering, Shenzhen University
Shenzhen, China
kzlu@szu.edu.cn

ABSTRACT

Attributed graph clustering aims to partition the nodes in a graph into groups such that the nodes in the same group are close in terms of graph proximity and also have similar attribute values. Recently, deep learning methods have achieved state-of-the-art clustering performance. However, the effectiveness of existing methods remains unclear due to two reasons. First, the datasets used for evaluation do not support fully the goal of attributed graph clustering. The category labels of nodes are only relevant to node attributes, and nodes with the same category label are often distant in the graph. Second, existing methods for the attributed graph clustering are complex and consist of several components. There is lack of comparisons of methods composed of different components from existing methods. This study proposes six benchmark datasets that support better the goal of attributed graph clustering and reports the performance of existing representative methods. Given that existing methods leave room for improvement on the proposed benchmark datasets, we systematically analyze five aspects of existing methods: encoded information, training networks, fusion mechanisms, loss functions, and clustering result generation. Based on these aspects, we decompose existing methods into modules and evaluate the performance of reconfigured methods based on these modules. According to the experimental results on the proposed benchmark datasets, we identify two promising configurations: (i) taking the attribute matrix as input to a graph convolutional network and (ii) layer-wise linear fusing deep neural network and graph attention network. And we also find that complex loss function fails to improve the clustering performance.

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

evaluation, deep learning, attributed graph clustering

ACM Reference Format:
Xinying Lai, Dingming Wu, Christian S. Jensen, and Kezhong Lu. 2023. A Re-evaluation of Deep Learning Methods for Attributed Graph Clustering. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614768

1 INTRODUCTION

Graphs are generally used for capturing the interactions among entities in systems. In real world systems such as social networks, citation networks, and knowledge graphs, entities, modeled by nodes, are often described by attribute information. For instance, a user in a social network has a profile, and a paper in a citation network has an abstract. Graphs where nodes are associated with attribute information are called attributed graphs.

Attributed graph clustering [4] aims to partition the nodes in a graph into groups such that the nodes in a group are close in terms of graph proximity and also have similar attribute values. Such clustering has importance in community detection [21], recommendations [32], anomaly detection [8], etc.

Traditional clustering algorithms such as KMeans [29] often fail to take into account the topological structure of graphs. To exploit the topological information in graphs, graph embedding methods for graph clustering have emerged that aim to learn low dimensional representations that encode topological information. However, early graph embedding methods based on Laplacian eigenmaps [2], matrix factorization [26], and random walks [16] have limited representation power.

Recently, substantial research efforts have been devoted to deep learning methods [12, 34, 40] for attributed graph clustering, thanks to their abilities at modeling non-linear and complex relationships. Although promising performance has been reported for many methods, the effectiveness of existing methods remains unclear for two reasons. First, the datasets used for evaluation do not support the goal of attributed graph clustering well. The nodes in a cluster should be close in terms of both attribute similarity and graph proximity. However, in the six datasets that are often used for evaluation, the category labels of nodes are only relevant to node attributes and do not take node proximity into account. Thus, the performance of existing methods reported based on these datasets can only show that the nodes in the discovered cluster have similar attribute values. Even worse, many nodes with the same category label are distant in the graph (see Section 3.1). So it remains unclear that whether the existing methods are able to produce clusters of nodes that are close in terms of graph proximity. Second, methods for attributed graph clustering are complex and consist of several components. In evaluations, newly proposed models are often compared to previous models as a whole. Even when ablation studies show the performance of different components of a proposed model, there is lack of comparisons of different components across methods.

Hence, to be consistent with the goal of attributed graph clustering, we propose six benchmark datasets by refining the existing six popular benchmark datasets. Specifically, in each original dataset, we divide the nodes with the same category label into several groups such that the diameters of these groups of nodes are small. In original datasets, for the same category, the majority of node pairs have graph distances larger than 6. In the proposed datasets, around 80% of node pairs in the same group have graph distances less than 6. We re-evaluate the performance of 13 representative deep learning methods on the proposed benchmark datasets. We find that these methods perform worse on the new datasets than on the original datasets. Hence, there is room for better methods to cluster nodes in attributed graphs.

The challenge of attributed graph clustering is to use simultaneously the attribute and the topological information to perform effective node clustering. Existing deep learning methods adopt different ways to jointly capture and integrate attribute and topological information into learned representations. Given that existing methods leave room for improvement on the proposed benchmark datasets, to gain insights from previous designs, we systematically analyze five aspects of existing methods: (i) what information is encoded, (ii) which architecture is used for training, (iii) how are attribute and topological information fused, (iv) which loss functions are used, and (v) how are clusters generated. Based on these five aspects, we decompose the analyzed methods into modules and engineer new methods by combining these modules. Extensive experiments are conducted to evaluate the performance of the composed methods on the proposed benchmark datasets. According to the experimental results on the proposed benchmark datasets, we identify two promising configurations: (i) taking the attribute matrix as input to a graph convolutional network and (ii) layer-wise linear fusing deep neural network and graph attention network. And we also find that complex loss function fails to improve the clustering performance.

Although optimization techniques such as ones that address the interference between graph topology and attributes [43] and redundancy [12] have been shown to improve clustering results, this study focuses on evaluating the performance of fundamental components in existing methods and leaves the evaluation of optimizations as future work.

The rest of the paper is organized as follows. Section 2 presents the attributed graph clustering problem and the framework of the deep learning approach. Section 3 discusses the disadvantages of existing benchmark datasets and proposes six new benchmark datasets. Section 4 decomposes existing representative attributed graph clustering methods into modules and investigates different configurations of these. In Section 5, we report the performance of existing methods and engineered module configurations on the proposed benchmark datasets. We review surveys and evaluation studies related to deep learning methods for graphs in Section 6, and we conclude in Section 7.

2 DEEP LEARNING FOR ATTRIBUTED GRAPH CLUSTERING

2.1 The Attributed Graph Clustering Problem

An attributed graph is directed or undirected graph and is represented as $G = (V, E, X)$, where $V = \{v_1, v_2, \cdots, v_N\}$ is the node set, $E \subseteq V \times V$ is the edge set, and $X \in \mathbb{R}^{N\times d}$ is the attribute matrix. Each row x_i in X is a real-valued vector in $\mathbb{R}^d$ that records the attribute values of node v_i. Let $A \in \mathbb{R}^{N\times N}$ be the adjacency matrix of G, where $A_{ij} = 1$ if $(v_i, v_j) \in E$ and is 0 otherwise. Let $D \in \mathbb{R}^{N\times N}$ be the degree matrix of G. The normalized adjacency matrix of G is calculated as $\tilde{A} = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$, where $I \in \mathbb{R}^{N\times N}$ is an identity matrix. The attributed graphs investigated in this paper are connected graphs.

The goal of attributed graph clustering is to partition node set V into k groups $C = \{C_1, C_2, \cdots, C_k\}$, satisfying

(1) $V = \bigcup_{C_i \in C} C_i$
(2) $C_i \cap C_j = \emptyset, i \neq j, C_i, C_j \in C$
(3) nodes in the same group have similar attribute values and are close in G,
(4) nodes in different groups have dissimilar attribute values and are distant in G.

2.2 Deep Learning Approach

Given an attributed graph $G = (V, E, X)$ and the number k of clusters to produce, the deep learning approach aims to represent each node in G as a vector in f-dimensional Euclidean space. Specifically, the node embedding aims to learn a function $f(\cdot): G \longrightarrow Z \in \mathbb{R}^{N\times f}$, where $N = |V|$, $f \ll d$, and $Z = [z_1, z_2, \cdots, z_N]^T$, such that the result vectors z_i preserve both attribute similarity and graph proximity among the nodes in G.

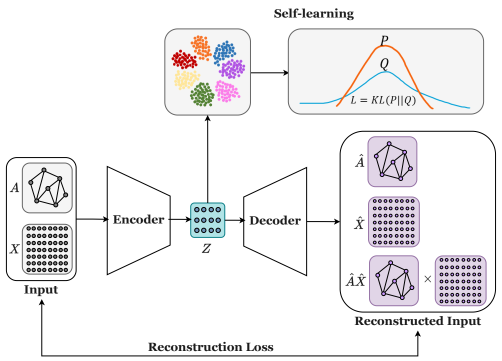


Figure 1: Framework of the deep learning approach.

Figure 1 illustrates the general deep learning framework. It takes adjacency matrix A and attribute matrix X as input. An encoder constructs an embedding matrix Z, where each row vector z_i represents a node. Then, using the node vectors, a decoder reconstructs input information such as adjacency matrix $\hat{A}$, attribute matrix $\hat{X}$, or weighted attribute matrix $\tilde{A}Z$. Various loss functions are designed to evaluate the similarity between the input and the reconstructed information. The output of the framework is the embedding matrix Z that minimizes the loss function.

In terms of how the clustering results are produced, some methods apply conventional clustering to the output node representations, while other methods adopt a self-learning mechanism that outputs the clustering results directly.

3 BENCHMARK DATASETS

3.1 Existing Benchmark Datasets

Six datasets [3, 6] are used widely as benchmarks to evaluate attributed graph clustering methods. We cleaned all the datasets by removing duplicate nodes, nodes with zero-valued attributes, isolated nodes, and self-loops. Table 1 shows the statistics of the cleaned datasets. All datasets caputure undirected graphs.

ACM is a network of papers that are classified into 3 categories based on research area: 'database,' 'wireless communication,' and 'data mining.' A link between a pair of papers indicates common co-authors. The set of keywords of a paper is taken as its attribute value. The number of weakly connected components (WCCs) is 156. The largest connected component contains 2,006 nodes.

DBLP is a network of authors. A link between two co-authors indicates co-authorship. The attribute value of an author is the union of the keywords of their papers. There are 4 categories in DBLP, including 'database,' 'data mining,' 'machine learning,' and 'information retrieval.' The number of WCCs is 335. The largest connected component contains 1,703 nodes.

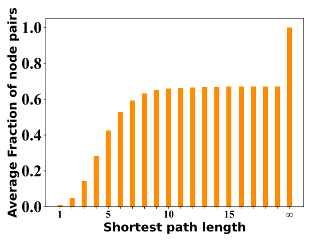
Wiki is a webpage citation network. The attribute value of a webpage is a set of keywords. The webpages are classified into 17 categories. The number of WCCs is 3. The largest connected component contains 2,257 nodes.

Cora, **CiteSeer**, and **Pumbed** are paper citation networks. The attribute value of papers contains keywords. Cora has 7 categories of papers: 'case-based', 'genetic algorithms,' 'neural networks,' 'probabilistic methods,' 'reinforcement learning,' 'rule-based learning,' and 'theory.' There are 6 categories of papers in CiteSeer: 'agent', 'artificial intelligence', 'database', 'information retrieval', 'machine language', and 'human-computer interaction.' The papers in Pumbed are classified into 3 categories: 'Diabetes Mellitus, Experimental', 'Diabetes Mellitus Type 1', and 'Diabetes Mellitus Type 2.' The numbers of WCCs in Cora and CiteSeer are 78 and 390, respectively. The largest connected components in these two datasets contain 2,485 and 2,110 nodes, respectively. Pumbed is a connected graph.

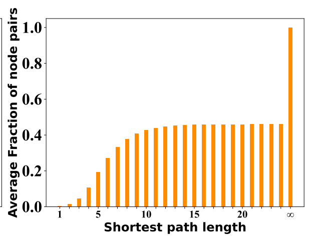
The goal of attributed graph clustering is to cluster nodes such that the nodes in the same group not only have similar attribute values but also are close in the graph. However, in these benchmark datasets, the category assigned to each node is only relevant to its attribute, e.g., the research areas of papers and authors and the keywords of webpages. It is not clear whether the nodes in the same category are close in the graph. To understand the proximities of nodes in the same category, we conduct the following analysis.

Table 1: Statistics of six existing benchmark datasets.

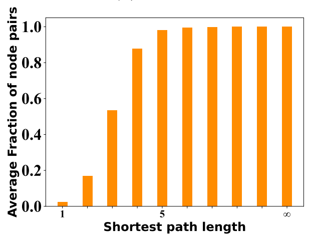
Dataset	$\|V\|$	$\|E\|$	#Dimensions	#Categories	#WCCs
ACM	2,464	13,128	1,870	3	156
DBLP	2,572	3,503	334	4	335
Wiki	2,263	10,269	4,973	17	3
Cora	2,708	5,278	1,433	7	78
CiteSeer	3,264	4,536	3,703	6	390
Pumbed	19,717	44,324	500	3	1



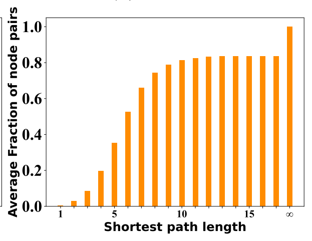
(a) ACM.



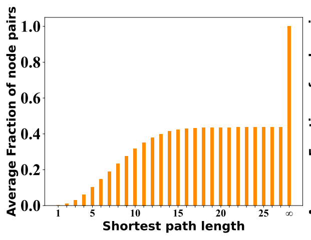
(b) DBLP.



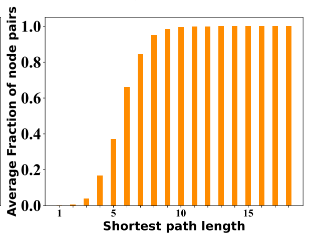
(c) Wiki.



(d) Cora.



(e) CiteSeer.



(f) Pumbed.

Figure 2: Average fraction of node pairs (u, v) in the same category whose $d_G(u, v) \leq x$ on existing benchmark datasets.

On each dataset, we calculate the lengths of the shortest paths between all pairs of nodes. We compute the fraction of node pairs (u, v) whose shortest path length $d_G(u, v) \leq x$ in each category. Figure 2 shows the average value of the fractions of all the categories in each dataset. Only in dataset Wiki, the nodes in the same category are closely connected, i.e., around 80% of the node pairs have graph distances less than 5, while in all the other datasets, the nodes in the same category are only loosely connected, e.g., in datasets CiteSeer and DBLP, more than 60% and 50% of the node pairs in the same category have graph distances larger than 10, respectively. In datasets ACM, Cora, and Pumbed, around 40% of the node pairs in the same category have graph distances less than 6.

The clustering results of existing algorithms are evaluated based on a set of metrics that use the category labels in the datasets as the ground-truth, including Accuracy (ACC) [42], Normalized Mutual Information (NMI) [10], F1-score (F1) [13], Precision (P) [41], Recall (R) [41], Average Entropy (AE) [41], and Adjusted Rand Index

(ARI) [44]. However, given the observations above, the majority of the nodes in the same category in these datasets are distant in terms of the graph proximity. Given the groups of nodes produced by an existing algorithm, current evaluation results only illustrate how similar node attribute values in the same group are and they do not reveal how close in the graph the nodes in the same group are. Hence, evaluating attributed graph clustering algorithms using these datasets is problematic.

3.2 Proposed Benchmark Datasets

Based on the observations in Section 3.1, we create new benchmark datasets by relabeling the nodes in the connected subgraphs in the existing benchmark datasets. The nodes in the new datasets are relabeled such that the attributes of the nodes in the same group are similar and the majority of node pairs in the same group are close in terms of graph proximity. We proceed to explain the relabeling process.

Given an existing benchmark dataset $\mathcal{D}$, each node is associated with an original category label. Let $V(L_j)$ be the set of nodes with label L_j. We extract the largest connected induced subgraph G_j based on node set $V(L_j)$ from the original dataset. In subgraph G_j, the attribute values of all nodes are similar because all nodes have the same original label. However, the nodes in G_j may be distant in terms of the graph proximity. We then try to partition the nodes in G_j such that the shortest path lengths of the node pairs in the same partition are less than parameter *dist*. Initially, the local clustering coefficients of all nodes in G_j are computed, and each node is considered as a subgraph. The key of a subgraph g is set to be the largest local clustering coefficient of the extendable nodes in g. A node is extendable if it has neighbors outside g. We then process all subgraphs in descending order of their keys. For each subgraph g, we take the extendable node v with the largest local clustering coefficient and find a set of subgraphs that contain one neighbor of v. Each of these subgraphs is added to g if the diameter of the combined graph is less than *dist*. When no subgraph can be merged with others, the nodes in the same subgraph are relabeled.

We apply this relabeling process to each existing benchmark dataset. Some result subgraphs are small, containing less than 20 nodes. We consider these small subgraphs as outliers and remove them. On datasets Wiki, Cora, and CiteSeer, we end up with some disconnected subgraphs. Then, we add several nodes as outliers to make the datasets connected. Table 2 shows the statistics of the new datasets. We also compute the fraction of node pairs (u, v) whose shortest path length $d_G(u, v) \leq x$ in each category. Figure 3 shows the average value of the fractions of all the categories in each new dataset. The shortest path lengths of around 80% of node pairs in the same category are below 6. Compared with the original datasets, the nodes belong to the same category are closer in terms of graph distance in the proposed datasets.

4 ANALYSIS OF ATTRIBUTED GRAPH CLUSTERING METHODS

We consider five aspects of 18 representative deep learning methods for attributed graph clustering.

- **Encoded information**: what information from an attributed graph is encoded in representations?
- **Training networks**: which kind of training network is used?
- **Fusion mechanisms**: how are the attribute information and the topological information fused?
- **Loss functions**: how are the differences between the reconstructed information by the learned representations and the original information in the attributed graph measured?
- **Clustering result generation**: how are clustering results obtained?

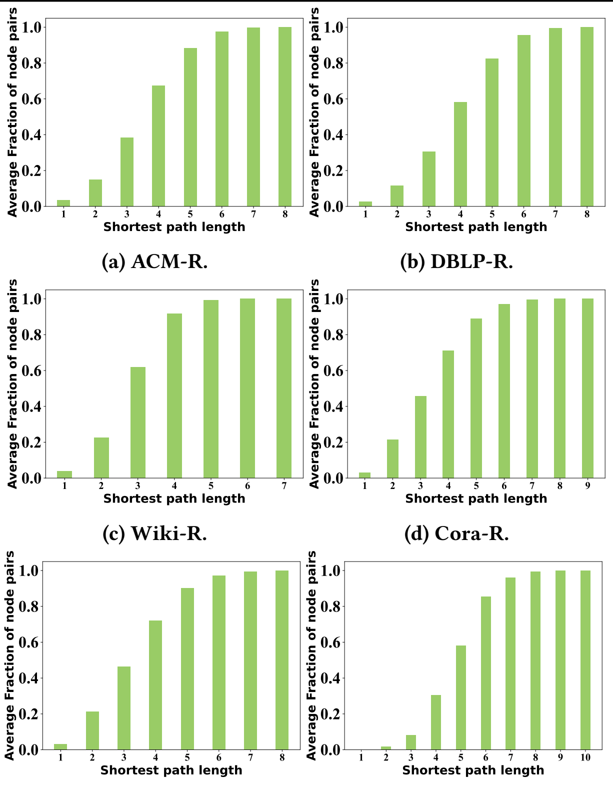
Table 3 profiles the 18 methods in terms of the above five aspects, and Sections 4.1–4.5 detail the techniques used in the methods.

4.1 Encoded Information

A variety of information in an attributed graph can be used when designing an encoder. The analyzed methods each considers some of the following four types of information in their encoders.

Table 2: Statistics of proposed benchmark datasets.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	#Dimensions	#Categories
ACM-R	1,718	11,778	1,870	12
DBLP-R	1,104	2,012	334	17
Wiki-R	1,837	8,868	4,973	17 (5 outliers)
Cora-R	1,218	2,101	1,433	18 (7 outliers)
CiteSeer-R	675	1,146	3,703	12 (1 outlier)
Pumbed-R	15,607	37,162	500	26



(a) ACM-R. (b) DBLP-R. (c) Wiki-R. (d) Cora-R. (e) CiteSeer-R. (f) Pumbed-R.

Figure 3: Average fraction of node pairs (u, v) in the same category whose $d_G(u, v) \leq x$ on proposed benchmark datasets.

Table 3: Characteristics of attributed graph clustering methods.

	Encoded information				Training networks			Fusion mechanisms				Loss functions					Clustering Results	
Methods	Attribute information	One-hop neighbors	High-order neighbors	Similarity information	DNN	GCN	GAT	Attributes as input	Linear fusion	Layer-wise linear fusion	Layer-wise non-linear fusion	X	A	$\tilde{A}X$	KL	Cross entropy	Two-phase	Self-learning
Graph Encoder [33]	✓	✓		✓	✓			✓				✓					✓	
MGAE [37]	✓	✓				✓		✓				✓					✓	
DAEGC [36] DNENC [38]	✓		✓				✓	✓					✓		✓			✓
SDCN [3]	✓	✓			✓	✓				✓		✓			✓			✓
AGE [6]	✓	✓		✓	✓			✓								✓	✓	
DFCN [34]	✓	✓		✓	✓	✓			✓			✓	✓	✓	✓			✓
CaEGCN [20]	✓	✓			✓	✓					✓	✓	✓		✓			✓
AG-cluster [40]	✓	✓				✓	✓			✓					✓			✓
DLGAMC [27]	✓	✓		✓	✓			✓								✓	✓	
AGC-DRR [12]	✓	✓				✓		✓										✓
DCRN [28]	✓	✓		✓	✓	✓			✓			✓	✓	✓	✓			✓
EFR-DGC [17]	✓		✓		✓		✓			✓		✓	✓		✓			✓
HiAN [47]	✓	✓					✓	✓					✓		✓			✓
CDBNE [48]	✓		✓				✓	✓					✓		✓			✓
DGCSF [25]	✓	✓				✓		✓					✓				✓	
GRACE [11]	✓		✓			✓		✓									✓	
R-GAE [30]	✓	✓				✓		✓					✓		✓			✓

(1) Attribute information, captured by the attribute matrix X, where each row contains the attribute values of a node.
(2) One-hop neighbors, captured by the adjacency matrix A, where each row contains the adjacent nodes of a node.
(3) High-order neighbors, captured by the t-th order transition matrix B^t, where $B_{ij} = \frac{1}{d_i}$ if $A_{ij} = 1$ and is 0 otherwise; d_i is the degree of node i.
(4) Similarity information, captured by the similarity matrix S, where each row indicates the similar nodes of a node. There are two ways of constructing a similarity matrix. One way is to compute the similarity between two nodes based on external information. GraphEncoder uses this way and initializes the attribute matrix to $D^{-1}S$, where D is the degree matrix. The other way is to measure the similarity between two nodes in terms of the dot product of their embeddings. AGE, DLGAMC, DFCN, and DCRN use this approach, but they incorporate the similarity information into the training process differently. AGE and DLGAMC determine positive training samples based on the similarity matrix, while DFCN and DCRN combine the similarity matrix S and the embedding matrix Z linearly, i.e., $Z \longleftarrow \beta SZ + Z$.

4.2 Training Networks

Three types of training networks dominate in the analyzed methods. We describe the main idea of each network.
(1) Deep Neural Network (DNN) [22]. Both the encoder and the decoder are fully connected L-layer networks. Node embeddings are encoded as $H^{(l)} = \sigma(W_1^{(l)} H^{(l-1)} + C_1^{(l)})$, where $H^{(0)} = X$, $W_1^{(l)}$ is the l^{th} layer trainable parameter matrix, $C_1^{(l)}$ is a trainable bias parameter, and $\sigma(\cdot)$ is a non-linear activation function, such as ReLU or Tanh. Embeddings are decoded as $\hat{X} = \sigma(W_2 H + C_2)$.
(2) Graph Convolutional Network (GCN) [23]. A GCN follows the structure of the input graph. The embedding of a node is an aggregation of its neighbors, i.e., $Z^{(l)} = \sigma(LZ^{(l-1)} W^{(l)})$, $Z^{(0)} = X$, where $W^{(l)}$ is a trainable parameter matrix, L is the Laplacian matrix caculated as $L = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I$, the $\tilde{d}_{ii}$ in $\tilde{D}$ is calculated as $\sum_j \tilde{A}_{ij}$, and I is the identity diagonal matrix containing self-loops.
(3) Graph Attention Network (GAT) [35]. A GAT is similar in structure to a GCN, but has a different aggregation mechanism. The neighbors of a node are treated differently in the aggregation:

$$z_i^{(l)} = \sigma(\tfrac{1}{K} \textstyle\sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^k W^k z_j^{(l-1)}), \alpha_{ij} = \frac{\exp(\sigma(\vec{a}^T [Wx_i \| Wx_j]))}{\sum_{r \in N_i} \exp(\sigma(\vec{a}^T [Wx_i \| Wx_r]))},$$

where N_i denotes the set of neighbors of node i, α_{ij}^k is the attention coefficient that indicates the importance of neighbor j to node i, and $\sigma(\cdot)$ is a non-linear activation function. Parameter K enables the multi-head attention mechanism in GAT. The neighbors of node i can be aggregated in K different ways. The average of these embeddings is the final embedding of node i.

4.3 Fusion Mechanisms

One of the challenges in attributed graph clustering is how to combine attribute and topological information so that the learned representations are able to preserve both types of information. In the analyzed methods, we observe four fusion mechanisms that combine these two types of information. Figure 4 illustrates the architectures of these fusion mechanisms, with $Encoder_A$ being the encoder for the attribute information and $Encoder_T$ being the encoder for the topological information. $Encoder_A$ and $Encoder_T$ could be any one type of the training network reviewed in Section 4.2.
(1) Attributes as input: $X \longrightarrow Encoder_T$. $Encoder_T$ is either a GCN or a GAT. This way of combining takes the attribute matrix as the input to the encoder, shown in Figure 4(a).
(2) Linear fusion (LF): $Z \longleftarrow (1 - \varepsilon)Z + \varepsilon H$. The attribute and the topological information are first encoded separately, yielding H and Z, respectively. These are then combined by a linear function, shown in Figure 4(b).

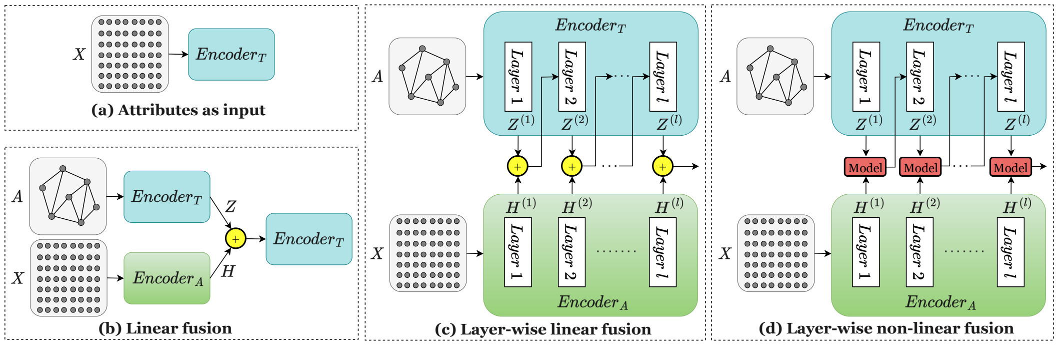


Figure 4: Architectures of fusion mechanisms.

(3) Layer-wise linear fusion (LLF): $(1-\varepsilon)Z^{(l)}+\varepsilon H^{(l)} \rightarrow Encoder_T$. The encoders for the attribute and the topological information both consist of multiple layers. Let H^l and Z^l be the outputs of the two encoders at layer l. The input to $Encoder_T$ at layer $l+1$ is a linear combination of H^l and Z^l, shown in Figure 4(c).

(4) Layer-wise non-linear fusion (LnLF): $Model(Z^{(l)}, H^{(l)}) \rightarrow Encoder_T$. The outputs at layer l of the encoders for the attribute and the topological information are combined by a non-linear *Model* and the result is then fed into $Encoder_T$ at layer $l+1$, shown in Figure 4(d).

4.4 Loss Functions

To quantify the difference between reconstructed and original information in an attributed graph, the analyzed methods consider the following loss functions.

(1) Reconstructing the attribute matrix: $\|X - \hat{X}\|_F^2$, where $\hat{X}$ is the output of the decoder.

(2) Reconstructing the adjacency matrix: $\|A - \hat{A}\|_F^2$, where $\hat{A} = Sigmoid(Z^T Z)$ and Z is the learned representation matrix.

(3) Reconstructing the weighted adjacency matrix: $\|\tilde{A}X - \hat{Z}\|_F^2$, where $\hat{Z}$ is the output of the decoder for learned matrix.

(4) Cross entropy: $\sum -l_{ij} \log s_{ij} - (1 - l_{ij})(1 - \log s_{ij})$. This is calculated using a set of positive and negative samples selected based on similarity information from the attributed graph.

(5) KL divergence [42]: $KL(P||Q) = \sum_i \sum_u p_{iu} \log \frac{p_{iu}}{q_{iu}}$, where q_{iu} uses the Student's t-distribution as a kernel to evaluate the similarity between the node representation and the cluster center vector, and where $p_{iu} = \frac{q_{iu}^2/\sum_i q_{iu}}{\sum_k (q_{ik}^2/\sum_i q_{ik})}$. KL divergence measures the similarity between the clustering result distribution Q and the target distribution P.

4.5 Clustering Result Generation

The existing methods use two ways of obtaining clustering results.

(1) Two-phase method. Node representations are learned first, and then node clusters are obtained by applying a conventional clustering algorithm such as K-means [18] to the learned representations.

(2) Self-learning method. Here, the node clustering is integrated into the learning process by means of a self-training module that optimizes the KL divergence between the clustering result distribution and target distribution. The representations and the assignments of nodes to clusters are learned together. The output of this method consists of the node embeddings and the centers of node clusters.

4.6 Summary

Table 3 summarizes the characteristics of the 18 attributed graph clustering methods. Regarding the encoded information, most of the methods consider the attribute information and the one-hop neighbors. A few methods take the high-order neighbors and the similarity information into account. The training networks adopted are usually hybrid that combine DNNs with GCNs or GATs. Regarding the fusion mechanism, most of the methods take the attribute matrix as the input to the encoder. DFCN and DCRN use linear fusion. Layer-wise linear fusion is adopted by SDCN, AG-cluster, and EFR-DGC. Only CaEGCN uses layer-wise non-linear fusion. Many methods utilize at least two loss functions. AGE and DLGAMC use only cross entropy in their loss functions. The majority of the methods employ the self-learning and output the clustering results directly. A few methods apply conventional clustering algorithms to the learned representations. GRACE is slightly different from the framework shown in Figure 1. It first converts attribute matrix X into Y by adopting a high-order graph convolution, so that the information of graph structure is integrated into node attributes. The rows in Y are the embeddings of nodes. Then, SVD is applied on Y to generate the first K eigen-vectors. At last, taking these eigenvectors as centers, the KMeans algorithm is applied to the node embeddings. Different from the other methods, AGC-DRR does not use the loss functions reviewed in Section 4.4. It adopts the graph contrastive learning (GCL) that aims to maximize the agreement between positive sample pairs and minimize the agreement between negative sample pairs. The agreement between sample pairs is commonly measured by mutual information maximization [19].

5 EVALUATIONS

5.1 Setup

Datasets. We conduct experiments on the six benchmark datasets proposed in Section 3.2: ACM-R, DBLP-R, Wiki-R, Cora-R, CiteSeer-R, and Pumbed-R.

Hyper-parameter setting. We observe that the hyper-parameter values reported by the analyzed methods are similar. Thus, we reuse commonly used hyper-parameter values. The dimensionality of the input vectors is 500. On DBLP-R, the dimensionality is set to 256 because there is fewer than 500 attribute features. We use 400 training epochs, a learning rate of 10^{-4}. We use a training batch size of 1000 in AGE. Since dataset Pubmed-R is larger than the other datasets, considering the main memory space and the training time, we use a training batch size of 2000 and evaluate every 20 training epochs for Pubmed-R.

Evaluation metrics. Following the evaluation settings in existing methods, we adopt four metrics: Accuracy (ACC) [42], Normalized Mutual Information (NMI) [10], Adjusted Rand Index (ARI) [44], and macro F1-score (F1) [13]. A good clustering result has a high value for these metrics.

Experiment environment. All methods are trained on a machine with an Intel(R) Xeon(R) E5-2680 v4@2.40GHz CPU, 128GB RAM, and four NVIDIA Tesla P100 16G GPUs. Among the 18 methods analyzed in Section 4, the source code of 13 methods are publicly available. The performance of the existing methods reported in Section 5.2 is evaluated using the publicly available source code. The results of the module evaluation in Section 5.3 are from our implementations using Python 3.6.7 and Pytorch 1.7.0. The new datasets and all source code are available [1].

5.2 Results of the Existing Methods

Table 4 shows the performance of 13 existing methods on the six proposed datasets. Overall, AGE, GRACE, and R-GAE outperform the other methods in most cases. DFCN, DCRN, and MGAE achieve good performance on some of the datasets. R-GAE is best on DBLP-R. AGE is best on Wiki-R, Cora-R, and Pubmed-R. GRACE is best on CiteSeer-R. There is no clear winner on ACM-R. The density of ACM-R is much higher than those of all the other datasets. Thus, DFCN and CDBNE perform better on dense graphs than do the other methods. AGE, GRACE, and R-GAE are more suitable for sparse graphs.

However, the performance on the proposed datasets is much worse than on the existing datasets as reported in the original papers of these methods. The category labels in existing datasets only relate to the node attributes. Many nodes with the same category label are distant in terms of graph distance, as shown in Section 3.1. In fact, the good performance on the existing datasets reported previously only shows that the existing methods are able to preserve attribute similarity. Our evaluations on the new datasets reveal that existing methods are not effective at capturing attribute and topological information simultaneously.

5.3 Performance of Modules

We decompose the analyzed methods into modules according to the five aspects reviewed in Section 4. Since we observe relatively low performance of the existing methods on the new datasets, we proceed to evaluate the effectiveness of each individual module by considering different module combinations and then provide recommendations for future attributed graph clustering methods.

In AGE, we obtain a smoothed attribute matrix $\tilde{X}$ by applying the Laplacian Smoothing Filter. All the engineered combinations evaluated in the following experiments use $\tilde{X}$ instead of the original attribute matrix X.

[1]https://github.com/2100271064/A-Re-evaluation-of-Deep-Learning-Methods-for-Attributed-Graph-Clustering

Effectiveness of training networks and fusion mechanisms. As discussed in Section 4.2, three fundamental types of training networks are used, i.e., DNN, GCN, and GAT. Section 4.3 summarizes four ways of fusing the topological and attribute information, i.e., $X \rightarrow Encoder_T$, LF, LLF, and LnLF. We generate 8 combinations of these modules (shown in Table 5) to understand which type of training network and fusion mechanism are effective. The generated methods all use the loss function that consists of reconstructing X and A. The clustering results are obtained by applying the KMeans algorithm on the learned representations. Considering the methods using one type of training network, GCN performs better than does GAT in most cases. For the methods that consist of two types of training networks, DNN+GAT is better than DNN+GCN on most datasets. Comparing GCN and DNN+GAT, the difference between their performance is insignificant. Regarding the fusion mechanisms, LLF is better than LF and LnLF.

Effectiveness of loss functions. According to the evaluation results of training networks and fusion mechanisms, we choose two winner modules to evaluate the effectiveness of different loss functions, i.e., Network-I: $\tilde{X} \rightarrow$ GCN and Network-II: DNN+GAT+LLF. We combine each of the two modules with the loss functions discussed in Section 4.4 and obtain 10 methods, shown in Table 6. The clustering results are obtained by applying the KMeans algorithm to the learned representations. We observe that in most cases the performance of the CrossEntropy loss is worse than the other loss functions. On Network-I, X+AX outperforms the other loss functions in most cases. On Network-II, X+A and A+AX perform better than the other loss functions. On ACM-R, Wiki-R, Cora-R, and CiteSeer-R, Network-I+X+AX is the best combination. On DBLP-R and Pumbed-R, Network-II+X+A is the best combination. We also observe that integrating X, A, and AX into the loss function fails to improve the performance of Network-I and Network-II.

Effectiveness of self-learning. According to the results of the above evaluations, we choose two winner methods, Network-I+X+AX and Network-II+X+A, to determine whether the self-learning module (SL) can improve performance. The SL adds KL divergence as mentioned in Section 4.4 to the loss function and directly outputs the clustering result. For each method, we evaluate the versions using only SL as the loss function and including SL into the loss function. Table 7 shows that the latter does not improve performance in most cases. Network-I taking SL as the loss function performs the best on Wiki-R, but its performance is close to the version without SL. The reason why SL is not effective is that there are more categories in the new datasets than in the existing datasets. Discovering more clusters and learning node representations together is more challenging.

Summary. Based on the evaluations in this section, we recommend taking the attribute matrix as the input to a GCN or layer-wise linear fusing DNN and GAT, which are two effective training architectures. It is not recommended to integrate more than two types of losses into the training process. On the new datasets, the existing self-learning module is not effective. We need to consider a better self-learning loss that is able to measure attribute similarity and graph proximity simultaneously.

Table 4: Performance of existing methods (bold indicates best, underlined indicates second best).

Dataset	Metric	AGE	SDCN	CaEGCN	DAEGC	AGC-DRR	DFCN	DCRN	CDBNE	EFR-DGC	GraphEncoder	MGAE	GRACE	R-GAE
ACM-R	ACC	35.90±1.50	39.55±4.56	37.93±1.63	33.45±2.13	35.97±1.60	51.33±0.34	38.54±1.29	**51.98 ± 2.09**	30.87±0.99	14.68±0.07	32.68±0.26	40.40±0.00	42.55±0.00
	NMI	39.75±1.06	32.33±5.09	29.55±2.51	35.53±1.88	40.04±2.05	46.62±0.15	40.94±2.08	39.10±0.51	24.48±0.56	2.99±0.02	32.07±0.29	**48.04 ± 0.00**	46.66±0.00
	ARI	23.77±1.73	20.67±6.31	16.51±3.13	21.55±2.73	22.37±1.92	**41.72 ± 0.35**	24.74±1.61	37.29±2.44	18.40±0.95	0.55±0.01	17.17±0.26	29.64±0.00	29.38±0.00
	F1	24.78±0.81	20.27±2.60	19.21±0.71	24.74±0.93	27.45±1.68	25.21±0.18	**28.56 ± 1.29**	24.62±1.02	19.56±0.57	10.37±0.07	27.23±0.23	26.95±0.00	28.09±0.00
DBLP-R	ACC	38.98±1.01	34.00±2.84	30.00±1.60	37.47±1.72	34.48±1.64	37.16±0.40	35.19±0.76	29.09±1.06	34.28±1.36	12.40±0.03	41.70±0.25	39.58±0.00	**43.61 ± 0.00**
	NMI	48.97±0.99	36.84±3.70	27.94±2.04	41.00±1.09	38.14±1.23	44.65±0.23	43.10±0.69	30.54±0.51	36.16±1.17	5.67±0.04	49.58±0.22	50.99±0.00	**52.16 ± 0.00**
	ARI	22.57±0.73	19.18±4.28	13.57±2.06	22.81±1.24	18.48±1.17	**26.46 ± 0.25**	20.07±0.58	14.42±1.14	16.44±0.92	0.53±0.01	25.34±0.26	23.22±0.00	25.47±0.00
	F1	37.70±1.06	18.39±1.78	14.56±1.24	27.74±1.26	29.92±2.35	26.37±0.33	29.45±1.04	17.83±0.88	29.80±1.64	10.69±0.03	31.30±0.25	37.32±0.00	**40.63 ± 0.00**
Wiki-R	ACC	50.31±1.24	16.53±0.01	22.48±1.73	29.45±0.94	28.38±1.82	32.40±0.11	28.44±0.09	23.10±0.57	35.05±0.57	12.94±0.03	**50.76 ± 0.39**	49.18±0.00	34.32±0.00
	NMI	**57.99 ± 0.91**	1.85±0.01	14.63±2.86	25.62±0.51	23.55±1.48	37.42±0.06	32.53±0.11	19.28±0.48	31.23±0.53	4.99±0.01	51.16±0.14	54.86±0.00	38.96±0.00
	ARI	23.43±1.27	0.06±0.01	4.04±1.82	11.84±0.44	12.72±1.87	12.15±0.13	8.60±0.03	5.96±0.30	14.72±0.72	1.00±0.01	**35.17 ± 0.32**	27.88±0.00	21.09±0.00
	F1	**42.72 ± 1.07**	1.87±0.01	12.83±2.44	26.00±0.85	23.05±1.11	19.93±0.09	18.04±0.08	18.42±0.69	27.70±0.77	10.14±0.03	40.31±0.29	40.47±0.00	27.56±0.00
Cora-R	ACC	**58.49 ± 1.43**	34.07±3.23	20.18±1.50	43.45±1.48	47.62±2.22	30.72±0.81	49.37±0.55	30.09±0.29	25.26±0.48	10.67±0.05	53.61±0.44	58.38±0.00	54.94±0.00
	NMI	**69.56 ± 0.56**	37.16±3.51	17.85±3.07	49.78±1.44	53.23±2.00	40.01±0.26	58.61±0.65	33.31±0.31	26.85±0.46	5.41±0.05	61.19±0.20	65.30±0.00	63.14±0.00
	ARI	**51.45 ± 0.93**	17.31±3.52	5.69±1.15	31.43±1.57	34.67±2.39	20.78±0.38	36.00±0.82	17.90±0.41	10.20±0.72	-0.08±0.02	41.57±0.37	44.69±0.00	42.60±0.00
	F1	52.26±1.86	22.39±2.46	9.41±1.68	36.55±2.14	42.29±2.13	20.10±0.56	44.26±0.48	16.23±0.74	20.12±0.66	9.15±0.07	45.66±0.35	**54.09 ± 0.00**	46.42±0.00
CiteSeer-R	ACC	59.19±2.83	38.16±3.34	34.44±2.87	43.71±1.49	47.97±3.62	43.37±0.90	63.71±2.32	42.33±1.36	35.45±1.14	14.13±0.06	58.86±0.58	**68.25 ± 0.00**	58.71±0.00
	NMI	63.19±1.88	36.42±4.87	26.18±3.61	44.88±1.51	50.21±2.20	46.15±1.12	66.54±1.31	35.92±0.56	36.52±1.12	3.64±0.07	65.87±0.27	**72.09 ± 0.00**	67.96±0.00
	ARI	44.74±3.87	20.68±5.68	13.56±3.28	31.25±2.07	31.91±4.17	28.32±1.37	48.34±1.02	27.06±2.15	16.94±0.93	-0.56±0.04	47.86±0.54	**64.91 ± 0.00**	42.45±0.00
	F1	53.15±2.89	26.08±3.48	19.31±2.31	36.76±1.50	43.88±2.44	22.49±0.72	54.32±3.30	23.34±1.19	29.55±1.30	11.89±0.09	54.11±0.49	53.97±0.00	**60.28 ± 0.00**
Pubmed-R	ACC	**38.87 ± 1.17**	30.64±1.36	24.44±1.96	30.85±0.97	28.67±0.90	18.84±0.19	26.05±1.37	29.48±0.25	18.08±5.58	8.71±0.11	22.68±0.08	30.76±0.00	18.84±0.00
	NMI	34.86±0.46	20.97±1.06	2.00±5.02	31.33±0.41	28.56±0.68	23.38±0.52	33.99±3.39	16.74±0.31	13.12±6.08	6.17±0.30	33.94±0.07	**36.32 ± 0.00**	27.37±0.00
	ARI	**22.60 ± 0.65**	11.35±1.05	0.96±2.90	15.02±0.45	13.66±0.82	8.54±0.15	12.87±1.32	11.65±0.68	3.55±5.76	1.60±0.10	11.13±0.06	16.28±0.00	10.62±0.00
	F1	22.89±0.54	8.28±0.60	2.24±1.31	13.98±0.28	13.93±0.62	13.11±0.14	**24.78 ± 1.85**	6.28±0.20	8.06±5.02	5.92±0.08	16.47±0.06	24.58±0.00	11.33±0.00

Table 5: Effectiveness of training networks and fusion mechanisms (bold is best, underlined is second best).

Dataset	Metric	$\tilde{X} \rightarrow$ GCN	$\tilde{X} \rightarrow$ GAT	DNN+GCN+LF	DNN+GCN+LLF	DNN+GCN+LnLF	DNN+GAT+LF	DNN+GAT+LLF	DNN+GAT+LnLF
ACM-R	ACC	40.23±1.95	28.41±0.53	**41.95 ± 4.79**	41.20±1.45	33.02±1.60	30.96±1.94	37.34±1.90	29.48±0.86
	NMI	**41.58 ± 2.01**	29.65±0.97	37.30±6.31	39.57±2.64	34.15±2.71	18.95±2.05	35.73±2.16	26.29±2.58
	ARI	27.13±1.98	14.86±0.62	24.55±7.35	**28.60 ± 2.37**	19.89±1.72	5.15±1.84	19.98±2.77	14.82±1.48
	F1	**26.17 ± 2.31**	21.80±0.44	18.15±1.85	19.96±0.90	23.68±1.26	12.43±1.43	19.84±1.60	20.94±0.55
DBLP-R	ACC	43.15±1.82	33.24±1.07	34.54±1.07	36.79±1.09	30.07±2.08	37.27±0.77	**43.50 ± 1.72**	27.17±1.57
	NMI	49.25±1.46	39.16±1.20	41.00±0.94	46.45±1.42	37.60±2.66	41.16±1.01	**51.12 ± 1.35**	32.80±2.39
	ARI	25.82±1.75	16.35±0.88	18.89±0.94	22.48±0.98	16.48±1.94	17.37±1.11	**27.05 ± 1.54**	13.59±1.64
	F1	40.78±1.76	30.60±1.24	28.66±0.99	33.74±0.88	26.26±2.38	34.60±1.33	**41.38 ± 2.12**	23.98±1.41
Wiki-R	ACC	38.00±1.68	26.01±2.05	29.03±1.15	**38.79 ± 1.57**	28.70±4.86	35.22±1.64	34.60±2.52	23.34±3.30
	NMI	42.99±2.20	21.68±1.85	28.00±2.13	**44.77 ± 1.34**	27.12±5.66	39.69±3.43	31.24±2.82	18.45±4.91
	ARI	14.32±1.15	8.80±2.14	5.80±0.76	**15.14 ± 1.19**	11.06±4.26	11.83±1.25	12.20±2.22	5.38±2.77
	F1	26.71±2.31	21.62±1.29	18.15±1.38	**27.66 ± 1.50**	21.06±5.18	23.41±1.66	23.34±3.02	13.49±3.86
Cora-R	ACC	**53.68 ± 1.77**	47.47±1.33	46.54±1.71	45.57±1.60	32.88±2.76	46.81±1.20	52.37±0.88	25.14±0.97
	NMI	60.90±1.50	52.49±1.22	55.46±1.32	54.92±1.11	39.89±3.16	52.63±1.04	**62.75 ± 0.85**	25.94±1.77
	ARI	40.02±2.21	32.04±1.43	32.38±1.67	31.82±1.27	19.49±2.71	26.06±1.89	**41.10 ± 0.89**	9.67±1.08
	F1	**49.50 ± 2.11**	42.82±1.11	40.67±1.69	42.24±2.51	29.04±3.25	40.98±0.97	47.90±1.31	22.32±1.34
CiteSeer-R	ACC	**59.16 ± 2.43**	40.04±1.35	49.27±1.70	51.86±2.26	41.13±3.13	39.54±1.62	56.07±2.95	31.99±2.45
	NMI	61.35±2.40	44.81±1.33	53.92±1.09	55.33±2.05	45.71±2.39	39.98±2.15	**61.79 ± 1.99**	28.96±3.74
	ARI	**45.96 ± 3.75**	22.73±1.17	29.25±1.99	36.57±2.34	27.48±3.74	12.39±2.68	40.30±3.41	15.34±2.70
	F1	**49.51 ± 2.62**	37.85±1.60	40.96±2.04	43.76±2.16	36.88±2.59	32.53±1.40	48.87±3.09	28.50±2.66
Pubmed-R	ACC	26.51±0.85	26.70±0.45	20.88±0.88	15.24±0.82	11.29±0.44	25.38±0.55	**27.42 ± 1.11**	16.58±1.00
	NMI	34.84±0.80	**35.98 ± 0.28**	22.89±1.37	21.44±1.53	11.10±1.43	34.00±0.52	34.48±0.92	13.69±1.80
	ARI	15.40±0.69	13.37±0.26	9.64±0.71	6.89±0.70	2.93±0.46	14.67±0.47	**15.83 ± 0.97**	5.13±1.05
	F1	21.66±1.16	20.67±0.55	13.41±1.11	11.24±0.95	7.38±0.38	19.54±1.29	**22.44 ± 0.87**	10.02±0.58

6 RELATED WORK

Existing surveys and evaluations related to graph embedding focus on the categorization of existing methods or conduct evaluations of functionality such as link prediction, classification, or different regression tasks. No studies provide detailed analyses or evaluations of attributed graph clustering. Cai et al. [5] compare graph embedding methods in terms of problem setting, analyze graph embedding methods based on how they preserve the information in graph, and categorize the graph embedding enabled applications as node related, edge related, or graph related. Goyal et al. [14] propose a taxonomy of graph embedding approaches. Existing embedding methods are evaluated according to graph reconstruction, visualization, link prediction, and node classification tasks. Pellegrino et al. [31] propose a framework to evaluate and compare graph embedding methods. The datasets for node clustering are retrieved by SPARQL queries on DBpedia and are not adopted by representative attributed graph clustering methods. The graph proximities of nodes in the same category are unknown. Ali et al. [1] re-implement and evaluate 21 graph embedding models using the PyKEEN software package. They report reproducibility results of all models and evaluate their performance on link prediction. Zhang et al. [46] conduct experiments with different knowledge graph (KG) embedding

Table 6: Effectiveness of loss functions (bold is best, underlined is second best).

Dataset	Metric	Network-I CrossEntropy	Network-II CrossEntropy	Network-I X+A+AX	Network-I X+A	Network-I X+AX	Network-I A+AX	Network-II X+A+AX	Network-II X+A	Network-II X+AX	Network-II A+AX
ACM-R	ACC	35.42±2.23	41.35±1.59	41.36±1.78	40.25±1.94	**43.85 ± 1.30**	40.27±1.92	40.46±1.86	37.34±1.90	40.33±1.22	40.30±1.98
	NMI	32.67±4.04	37.43±1.94	42.89±1.77	41.58±2.01	**48.10 ± 0.83**	41.65±1.96	44.31±1.21	35.73±2.16	42.43±2.09	43.79±1.24
	ARI	19.26±2.95	29.33±2.32	28.58±1.89	27.15±1.99	**31.89 ± 1.19**	27.21±1.91	28.76±2.85	19.98±2.77	25.84±1.63	28.16±2.66
	F1	25.35±2.76	25.79±1.40	27.08±2.03	26.17±2.34	**31.17 ± 0.85**	26.22±2.30	22.45±1.22	19.84±1.60	29.65±1.34	22.29±1.29
DBLP-R	ACC	31.09±1.82	30.65±1.78	**43.55 ± 1.71**	43.14±1.83	42.64±0.94	42.77±1.73	40.42±1.26	43.50±1.72	37.45±1.28	40.55±1.39
	NMI	35.14±3.15	34.86±1.72	50.09±1.26	49.25±1.47	50.57±0.63	49.32±1.42	49.34±0.89	**51.12 ± 1.35**	47.46±1.04	49.38±0.84
	ARI	15.37±2.02	17.94±1.66	26.35±1.63	25.81±1.76	26.51±0.52	25.65±1.68	23.91±0.95	**27.05 ± 1.54**	22.15±1.17	23.99±1.01
	F1	26.85±2.07	22.71±2.07	**41.72 ± 1.50**	40.77±1.78	40.44±1.50	40.69±1.64	40.34±1.46	41.38±2.12	36.20±1.45	40.29±1.41
Wiki-R	ACC	42.09±1.11	32.04±2.65	38.80±1.57	37.96±1.64	**45.13 ± 2.10**	37.82±2.05	34.22±2.37	34.60±2.52	36.21±3.08	32.70±2.50
	NMI	43.48±1.46	31.36±2.64	44.74±1.79	42.97±2.17	**46.66 ± 1.76**	43.25±2.28	37.50±2.41	31.24±2.82	36.24±3.70	36.16±3.52
	ARI	19.16±1.57	14.84±2.28	15.46±1.35	14.30±1.13	**24.27 ± 2.36**	14.55±1.44	11.90±1.86	12.20±2.22	15.68±2.49	11.19±2.30
	F1	32.20±0.92	26.41±2.89	28.12±1.88	26.69±2.28	**33.67 ± 1.57**	26.77±2.68	21.94±1.64	23.34±3.02	26.63±1.90	20.75±1.96
Cora-R	ACC	32.67±2.01	28.57±2.16	54.84±1.56	53.67±1.76	**56.62 ± 1.69**	54.39±1.64	56.18±0.87	52.37±0.88	54.04±1.45	56.34±0.85
	NMI	38.90±1.55	37.04±3.20	61.91±1.33	60.90±1.50	**64.70 ± 1.08**	61.17±1.41	64.33±0.55	62.75±0.85	61.73±0.73	64.38±0.55
	ARI	18.01±1.74	15.19±3.23	41.22±2.19	40.03±2.20	**44.62 ± 1.88**	40.19±2.23	43.37±0.80	41.10±0.89	39.47±1.13	43.53±0.79
	F1	29.25±2.65	20.68±2.63	50.33±1.96	49.49±2.13	**52.42 ± 1.63**	49.89±1.87	51.55±1.10	47.90±1.31	50.70±1.54	51.63±1.13
CiteSeer-R	ACC	40.23±1.05	43.04±4.17	60.82±2.28	59.18±2.42	**63.23 ± 1.26**	59.04±2.49	54.30±1.87	56.07±2.95	51.08±1.94	54.12±1.78
	NMI	42.30±2.49	49.38±3.29	63.23±2.10	61.35±2.39	**66.08 ± 0.73**	61.86±2.35	60.33±1.12	61.79±1.99	57.51±1.99	59.96±1.20
	ARI	24.86±2.37	32.14±6.76	48.50±3.78	45.98±3.74	**52.60 ± 1.44**	45.83±3.65	37.34±2.03	40.30±3.41	33.91±3.03	36.97±2.01
	F1	32.88±1.52	31.06±3.87	50.86±2.43	49.51±2.61	**52.46 ± 1.90**	50.23±2.64	50.72±1.40	48.87±3.09	47.92±1.81	50.59±1.39
Pubmed-R	ACC	13.47±0.30	19.50±1.85	22.43±1.30	26.51±0.85	14.25±0.74	25.71±1.01	26.44±0.76	**27.42 ± 1.11**	19.56±0.60	26.82±0.70
	NMI	18.22±1.10	24.32±3.57	30.90±1.17	**34.84 ± 0.80**	20.81±1.63	34.13±0.93	34.55±0.63	34.48±0.92	29.22±0.97	34.79±0.54
	ARI	5.27±0.43	9.18±1.67	12.77±0.88	15.41±0.69	6.33±0.71	14.87±0.76	15.35±0.59	**15.83 ± 0.97**	10.37±0.59	15.63±0.55
	F1	9.05±0.53	13.74±1.49	17.13±0.87	21.66±1.16	11.12±0.78	20.78±1.22	21.01±0.82	**22.44 ± 0.87**	15.03±0.74	21.19±0.84

Table 7: Effectiveness of self-learning (bold is best, underlined is second best).

Dataset	Metric	Network-I X+AX	Network-I X+AX+SL	Network-I SL	Network-II X+A	Network-II X+A+SL	Network-II SL
ACM-R	ACC	43.85±1.30	**49.42 ± 2.94**	48.57±2.14	37.34±1.90	41.18±2.34	46.51±3.87
	NMI	**48.10 ± 0.83**	37.96±3.78	36.14±3.01	35.73±2.16	36.16±2.19	34.55±3.18
	ARI	**31.89 ± 1.19**	30.57±4.85	28.98±3.25	19.98±2.77	25.64±3.75	24.85±2.33
	F1	**31.17 ± 0.85**	20.26±2.89	17.93±2.10	19.84±1.60	22.34±1.00	12.76±1.18
DBLP-R	ACC	42.64±0.94	35.15±2.51	32.92±1.90	**43.50 ± 1.72**	36.42±2.29	30.54±1.44
	NMI	50.57±0.63	38.40±3.62	34.93±3.96	**51.12 ± 1.35**	39.72±3.02	32.57±3.84
	ARI	26.51±0.52	19.49±3.63	18.22±3.40	**27.05 ± 1.54**	21.60±2.74	16.27±2.60
	F1	40.44±1.50	18.62±2.15	15.67±1.60	**41.38 ± 2.12**	20.38±2.94	8.45±0.97
Wiki-R	ACC	45.13±2.10	35.64±4.25	**45.24 ± 2.76**	34.60±2.52	24.43±2.26	29.07±4.64
	NMI	**46.66 ± 1.76**	33.54±4.50	45.59±2.32	31.24±2.82	17.46±4.00	24.67±5.66
	ARI	24.27±2.36	16.39±4.17	**25.70 ± 3.54**	12.20±2.22	4.82±2.77	11.51±4.61
	F1	33.67±1.57	23.13±3.46	**35.24 ± 1.86**	23.34±3.02	12.01±3.24	19.25±3.93
Cora-R	ACC	**56.62 ± 1.69**	31.06±1.97	29.72±1.68	52.37±0.88	33.66±3.89	24.83±1.43
	NMI	**64.70 ± 1.08**	39.10±2.46	36.99±2.11	62.75±0.85	41.46±4.42	29.69±3.16
	ARI	**44.62 ± 1.88**	18.49±1.85	17.43±1.68	41.10±0.89	21.21±4.68	11.59±2.11
	F1	**52.42 ± 1.63**	16.75±2.22	15.40±1.68	47.90±1.31	19.57±3.88	7.51±0.96
CiteSeer-R	ACC	**63.23 ± 1.26**	44.63±4.39	42.90±3.38	56.07±2.95	50.83±2.99	43.36±0.37
	NMI	**66.08 ± 0.73**	40.97±6.92	37.69±5.30	61.79±1.99	54.10±2.34	41.15±2.32
	ARI	**52.60 ± 1.44**	27.03±6.52	24.80±4.40	40.30±3.41	36.85±5.33	24.49±1.83
	F1	**52.46 ± 1.90**	21.93±4.20	19.39±2.56	48.87±3.09	33.10±3.43	13.43±0.59
Pubmed-R	ACC	14.25±0.74	30.64±0.32	**31.06 ± 0.41**	27.42±1.11	28.33±1.25	26.25±2.04
	NMI	20.81±1.63	14.72±0.54	18.59±1.20	**34.48 ± 0.92**	13.10±2.06	10.69±4.72
	ARI	6.33±0.71	7.72±0.39	9.85±0.74	**15.83 ± 0.97**	6.80±1.36	4.96±2.54
	F1	11.12±0.78	5.07±0.22	8.09±1.16	**22.44 ± 0.87**	5.50±0.47	5.73±1.03

methods and study their performance on KG-based question answering, recommendation, and natural language processing tasks. Knežević et al. [24] evaluate node2vec [16] and its two variants on a node clustering task using existing datasets. Wang et al. [39] survey studies on heterogeneous graph embedding, including representative methods, applications, available benchmark datasets, and open-source code/tools. Claudio et al. [7] introduce a taxonomy for dynamic graph embedding methods and explore different dynamic embedding behaviors.

A few studies propose benchmark datasets for evaluating graph embedding methods. However, they do not contain attribute information, and thus are not suited for attributed graph clustering. Goyal et al. [15] establish a benchmark comprising 100 real-world graphs that exhibit different structural properties and evaluate the graph embedding methods on a link prediction task. Dwivedi et al. [9] release a benchmark containing 12 graph datasets for evaluating regression, link prediction, and classification tasks. Zhang et al. [45] propose a benchmark for evaluating embedding-based entity alignment techniques.

7 CONCLUSION

We propose new benchmark datasets for attributed graph clustering, where nodes in the same category not only have similar attribute values but also are close in terms of graph distance, thus enabling better evaluation of clustering methods. We report an evaluation of existing representative methods on the proposed datasets, finding ample room for improvement. To gain insight from previous methods, we analyze five aspects of existing methods: what information is encoded, which training network is used, how are attribute and topological information fused, which loss function is used, and which approach is taken to generate clusters. Based on these five aspects, we decompose the analyzed methods into modules and engineer new methods by combining these modules. Performance evaluations of these engineered methods are reported and recommendations for future work are provided.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Science Foundation of Guangdong Province of China under Grant 2023A1515011619 and in part by the Shenzhen Science and Technology Program 20220818100205012.

REFERENCES

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. 2022. Bringing Light Into the Dark: A Large-Scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 12 (2022), 8825–8845.

[2] Mikhail Belkin and Partha Niyogi. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.* 15, 6 (2003), 1373–1396.

[3] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural Deep Clustering Network. In *WWW*. 1400–1410.

[4] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenková. 2015. Clustering attributed graphs: Models, measures and methods. *Netw. Sci.* 3, 3 (2015), 408–444.

[5] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Trans. Knowl. Data Eng.* 30, 9 (2018), 1616–1637.

[6] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive Graph Encoder for Attributed Graph Embedding. In *KDD*. 976–985.

[7] Claudio Daniel Tenorio de Barros, Matheus R. F. Mendonça, Alex Borges Vieira, and Artur Ziviani. 2023. A Survey on Embedding Dynamic Graphs. *ACM Comput. Surv.* 55, 2 (2023), 10:1—10:37.

[8] Kaize Ding, Jundong Li, and Huan Liu. 2019. Interactive Anomaly Detection on Attributed Networks. In *WSDM*. 357–365.

[9] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking Graph Neural Networks. *CoRR* abs/2003.00982 (2020).

[10] Pablo A. Estévez, M. Tesmer, Claudio A. Perez, and Jacek M. Zurada. 2009. Normalized Mutual Information Feature Selection. *IEEE Trans. Neural Networks* 20, 2 (2009), 189–201.

[11] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, and Bo Han. 2023. GRACE: A General Graph Convolution Framework for Attributed Graph Clustering. *ACM Trans. Knowl. Discov. Data* 17, 3 (2023), 31 pages.

[12] Lei Gong, Sihang Zhou, Wenxuan Tu, and Xinwang Liu. 2022. Attributed Graph Clustering with Dual Redundancy Reduction. In *IJCAI*. 3015–3021.

[13] Cyril Goutte and Éric Gaussier. 2005. A Probabilistic Interpretation of Precision, Recall and *F*-Score, with Implication for Evaluation. In *ECIR*. 345–359.

[14] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.* 151 (2018), 78–94.

[15] Palash Goyal, Di Huang, Ankita Goswami, Sujit Rokka Chhetri, Arquimedes Canedo, and Emilio Ferrara. 2019. Benchmarks for Graph Embedding Evaluation. *CoRR* abs/1908.06543 (2019).

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. 855–864.

[17] Jie Hao and William Zhu. 2023. Deep graph clustering with enhanced feature representations for community detection. *Appl. Intell.* 53, 2 (2023), 1336–1349.

[18] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.

[19] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. 4116–4126.

[20] Guangyu Huo, Yong Zhang, Junbin Gao, Boyue Wang, Yongli Hu, and Baocai Yin. 2023. CaEGCN: Cross-Attention Fusion Based Enhanced Graph Convolutional Network for Clustering. *IEEE Trans. Knowl. Data Eng.* 35, 4 (2023), 3471–3483.

[21] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip S. Yu, and Weixiong Zhang. 2023. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *IEEE Trans. Knowl. Data Eng.* 35, 2 (2023), 1149–1170.

[22] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *CoRR* abs/1611.07308 (2016).

[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. 1 pages.

[24] Dušica Knežević, Jela Babić, Miloš Savić, and Miloš Radovanović. 2022. Evaluation of LID-Aware Graph Embedding Methods for Node Clustering. In *International Conference on Similarity Search and Applications*. 222–233.

[25] Wang Li, Siwei Wang, Xifeng Guo, and En Zhu. 2023. Deep graph clustering with multi-level subspace fusion. *Pattern Recognition* 134 (2023), 10 pages.

[26] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. 2018. Community Detection in Attributed Graphs: An Embedding Approach. In *AAAI*. 338–345.

[27] Huifa Liao, Jie Hu, Tianrui Li, Shengdong Du, and Bo Peng. 2022. Deep linear graph attention model for attributed graph clustering. *Knowl. Based Syst.* 246 (2022), 13 pages.

[28] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. 2022. Deep Graph Clustering via Dual Correlation Reduction. In *AAAI*. 7603–7611.

[29] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–136.

[30] Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini. 2022. Rethinking Graph Auto-Encoder Models for Attributed Graph Clustering. *IEEE Trans. Knowl. Data Eng.* 10.1109/TKDE.2022.3220948 (2022), 15 pages.

[31] Maria Angela Pellegrino, Abdulrahman Altabba, Martina Garofalo, Petar Ristoski, and Michael Cochez. 2020. GEval: A Modular and Extensible Evaluation Framework for Graph Embedding Techniques. In *ESWC*. 565–582.

[32] Tribikram Pradhan and Sukomal Pal. 2020. CNAVER: A Content and Network-based Academic VEnue Recommender system. *Knowl. Based Syst.* 189 (2020), 25 pages.

[33] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning Deep Representations for Graph Clustering. In *AAAI*. 1293–1299.

[34] Wenxuan Tu, Sihang Zhou, Xinwang Liu, Xifeng Guo, Zhiping Cai, En Zhu, and Jieren Cheng. 2021. Deep Fusion Clustering Network. In *AAAI*. 9978–9987.

[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *CoRR* abs/1710.10903 (2017).

[36] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *IJCAI*. 3670–3676.

[37] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. MGAE: Marginalized Graph Autoencoder for Graph Clustering. In *CIKM*. 889–898.

[38] Chun Wang, Shirui Pan, Celina Ping Yu, Ruiqi Hu, Guodong Long, and Chengqi Zhang. 2022. Deep neighbor-aware embedding for node clustering in attributed graphs. *Pattern Recognit.* 122 (2022), 108230.

[39] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. 2023. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *IEEE Trans. Big Data* 9, 2 (2023), 415–436.

[40] Hui Xia, Shu-shu Shao, Chunqiang Hu, Rui Zhang, Tie Qiu, and Fu Xiao. 2023. Robust Clustering Model Based on Attention Mechanism and Graph Convolutional Network. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 5203–5215.

[41] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. 2014. Robust Multi-View Spectral Clustering via Low-Rank and Sparse Decomposition. In *AAAI*. 2149–2155.

[42] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *ICML*, Vol. 48. 478–487.

[43] Liang Yang, Wenmiao Zhou, Weihang Peng, Bingxin Niu, Junhua Gu, Chuan Wang, Xiaochun Cao, and Dongxiao He. 2022. Graph Neural Networks Beyond Compromise Between Attribute and Topology. In *WWW*. 1127–1135.

[44] Ka Yee Yeung and Walter Ruzzo. 2001. Details of the Adjusted Rand index and Clustering algorithms, Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data". *Bioinformatics* 17, 9 (2001), 763–774.

[45] Rui Zhang, Bayu Distiawan Trisedya, Miao Li, Yong Jiang, and Jianzhong Qi. 2022. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *VLDB J.* 31, 5 (2022), 1143–1168.

[46] Yuxin Zhang, Bohan Li, Han Gao, Ye Ji, Han Yang, Meng Wang, and Weitong Chen. 2021. Fine-Grained Evaluation of Knowledge Graph Embedding Model in Knowledge Enhancement Downstream Tasks. *Big Data Res.* 25 (2021), 9 pages.

[47] Qiqi Zhao, Huifang Ma, Lijun Guo, and Zhixin Li. 2022. Hierarchical attention network for attributed community detection of joint representation. *Neural Computing and Applications* 34, 7 (2022), 5587–5601.

[48] Xinchuang Zhou, Lingtao Su, Xiangju Li, Zhongying Zhao, and Chao Li. 2023. Community detection based on unsupervised attributed network embedding. *Expert Systems with Applications* 213 (2023), 10 pages.