GRACE: A General Graph Convolution Framework for Attributed Graph Clustering

BARAKEEL FANSEU KAMHOUA[*], The Chinese University of Hong Kong,
LIN ZHANG[*], The Hong Kong University of Science and Technology,
KAILI MA, The Chinese University of Hong Kong,
JAMES CHENG[†], The Chinese University of Hong Kong,
BO LI, The Hong Kong University of Science and Technology,
BO HAN, Hong Kong Baptist University,

Attributed graph clustering (AGC) is an important problem in graph mining as more and more complex data in real-world have been represented in graphs with attributed nodes. While it is a common practice to leverage both attribute and structure information for improved clustering performance, most existing AGC algorithms consider only a specific type of relations, which hinders their applicability to integrate various complex relations into node attributes for AGC. In this paper, we propose GRACE, an extended graph convolution framework for AGC tasks. Our framework provides a general and interpretative solution for clustering many different types of attributed graphs, including undirected, directed, heterogeneous and hyper attributed graphs. By building suitable graph Laplacians for each of the aforementioned graph types, GRACE can seamlessly perform graph convolution on node attributes to fuse all available information for clustering. We conduct extensive experiments on 14 real-world datasets of 4 different graph types. The experimental results show that GRACE outperforms the state-of-the-art AGC methods on the different graph types in terms of clustering quality, time, and memory usage.

CCS Concepts: • **Information systems** → **Clustering and classification**.

Additional Key Words and Phrases: Attributed graph clustering; Graph convolution

1 INTRODUCTION

Attributed graphs can be used to represent complex data in many application domains including social networks, web pages, and recommendation systems. Nodes in an attributed graph are associated with a number of attributes that describe the characteristics of objects, and edges (which can be of different types) represent the relationships among the objects.

Attributed graph clustering (AGC) is one of the most significant graph mining problems with many applications. AGC aims at partitioning the nodes in an attributed graph into a number of clusters such that (1) nodes in the same cluster should share similar attributes, and (2) nodes in different clusters should be minimally connected,

[*]Both authors contributed equally to this research and are listed in alphabetical order.
[†]Corresponding Author

Authors' addresses: Barakeel Fanseu Kamhoua, kamhoua@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, ; Lin Zhang, lzhangbv@connect.ust.hk, The Hong Kong University of Science and Technology, Hong Kong, ; Kaili Ma, klma@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, ; James Cheng, jcheng@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, ; Bo Li, bli@cse.ust.hk, The Hong Kong University of Science and Technology, Hong Kong, ; Bo Han, bhanml@comp.hkbu.edu.hk, Hong Kong Baptist University, Hong Kong,

1556-4681/2022/6-ART $15.00
https://doi.org/10.1145/3544977

i.e., a cluster should have minimal edges connecting it to other clusters. Clustering results are very helpful to study and understand the graph data. For example, AGC can be used to discover social circles for social network analysis [34] or improve the search quality by ranking web pages associated with their cluster information [61].

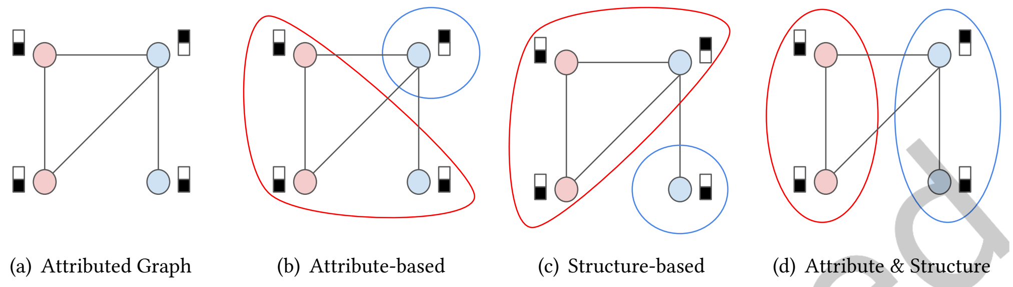


Fig. 1. An example of an attributed graph with different clustering results. In (a), four nodes are connected with undirected edges and associated with two-dimensional attribute vectors. In (b), only node attributes are used for clustering, in (c) only the graph structure is used, and in (d) both node attributes and graph structure are used. The nodes are partitioned into two clusters (red and blue circles), and different node colors represent their ground truth classes.

However, clustering attributed graphs poses significant challenges. On the one hand, traditional methods that focus on either node attributes or graph structure can lead to poor clustering results, as only one type of information is utilized in clustering the graph. An illustrating example of AGC is given in Figure 1. In attribute-based clustering, as shown in Figure 1(b), nodes with the same attributes are grouped together but many connections exist between these two clusters. In structure-based clustering, as shown in Figure 1(c), nodes belonging to different clusters are minimally connected, but nodes within the same cluster have different attributes. In addition, the clustering results obtained from attribute-based or structure-based clustering are inconsistent with each other, and there is no simple solution to choose one of these two different results, nor to combine them [21].

On the other hand, different approaches that combine node attributes and graph structure information have been proposed to produce better clustering results [77, 84, 88]. As shown in Figure 1(d), such approaches return better clusters by balancing the attribute and structure similarities properly. However, existing AGC approaches are mainly designed for specific types of attributed graphs and cannot be easily generalized to other types. For example, AGC methods designed for attributed graphs with simple undirected edges cannot be easily extended to capture asymmetric, higher-order, and heterogeneous relationships, which are represented by directed, hyper-, and multi-relational edges, respectively. This is because undirected edges are only used to represent symmetric pairwise relationships and do not capture higher order relationships, e.g., in a social network, friendship is undirected, while the influencer–follower relationship is directed, social tagging that involves a group of users is modelled by hyper-edges, and multi-relational edges can describe rich user interactions on different social media sites.

Some existing methods have been extended to handle more types of graphs, but the generalization to many types of graphs remains a challenging problem. For example, on heterogeneous graphs, the recent state-of-the-art SpectralMix [53] can be easily extended to undirected graphs as well as directed graphs by simply using a single adjacency in place of the sum in the structure part of its objective function. However, it is not clear how to extend it to hypergraphs since the structure part of its objective function assumes that an edge only connects exactly two nodes.

For AGC on undirected graphs (Undi-AGC), graph convolution-based models have been shown to achieve state-of-the-art performance [1] [49, 84]. In an effort to extend the graph convolution-based methods in order to tackle AGC on hyper-graphs (hyper-AGC), in our prior work [27] we provide solid theoretical analysis on the excellent performance of graph convolution on AGC for simple undirected graphs, and we extended the results to clustering hypergraphs and proposed a model (GRAC) for hyper-AGC.

In this paper, we further extend our prior work [27] to propose a general graph convolution-based framework, called GRACE[2], for AGC on different types of edges, i.e., a general framework for clustering attributed graphs with directed, hyper-, or multi-relational edges (or any graphs with a well defined graph Laplacian).

GRACE is the first graph convolution-based algorithm proposed to work on all types of graphs with a well defined Laplacian. The graph convolution is a filtering operation that is used to de-noise graph signals by allowing specific graph signals contained in the node attributes to pass through, where graph signals correspond to columns of the node attribute matrix and the graph convolution filter corresponds to functions of the graph Laplacian [58]. As such, GRACE consists mainly of two phases, a de-noising (graph convolution) phase and a clustering phase. To integrate both phases and decide how much de-noising is needed, GRACE utilizes the notion of cluster compactness on the de-noised node attributes (i.e., how close the nodes in the same cluster are to each other with respect to their node attributes).

Based on graph convolution, we first show from a Graph Signal Processing perspective [58] that GRACE can be interpreted as a two-step algorithm guided by the compactness measure, with the first step being the de-noising step and the second being the clustering step, where the compactness measure controls what level of de-noising is needed. Second, we show that GRACE is easy to understand with respect to its effects on AGC on different types of graphs. Specifically, GRACE helps find the trade-off between ensuring that nodes in different clusters are as minimally connected as possible, while at the same time ensuring that nodes in the same cluster have attributes that are not too dissimilar to each other. In addition, GRACE is easy to implement as it only consists of two (easy-to-implement) phases, a graph convolution phase and then a clustering phase, both of which have a relatively low time complexity.

2 BACKGROUND

In this section, we introduce the background of attribute-based and structure-based clustering algorithms. We also give the preliminaries of graph convolution and four different types of graph structures. We summarize the frequently used notations in Table 1.

2.1 Attribute-based Clustering

Clustering attributed graphs involves both node attributes and graph structure to measure the similarity between different nodes. In attribute-based clustering, as shown in Fig. 1(b), each node v_i is described by an attribute vector, denoted by $\mathbf{x}_i \in \mathbb{R}^d$, and nodes with similar attribute vectors should be assigned to the same cluster. For that purpose, the K-means [44] algorithm was proposed to find a number of clusters that minimize the intra-cluster feature distance (IntraFD) as follows:

$$\min_{C_1,\cdots,C_K} \sum_{k=1}^{K} \sum_{v_i \in C_k} ||\mathbf{x}_i - \mathbf{m}_k||^2, \tag{1}$$

where $\mathbf{m}_k = \sum_{v_i \in C_k} / |C_k|$ is the centroid of cluster C_k, and K is the number of clusters. Lower IntraFD means higher quality of clusters w.r.t. node attributes. In other words, nodes in the same cluster should be close to their centroid.

[1]https://paperswithcode.com/task/graph-clustering
[2]The code of GRACE can be found at https://github.com/BarakeelFanseu/GRACE.

Table 1. Notations

Name	Description
V	A set of nodes
v_i	The i-th node ($i = 1, 2, \cdots, n$)
$\mathbf{x}_i$	The node attribute vector of v_i
X	An attribute matrix of all nodes
E	A set of edges
e_j	The j-th edge ($j = 1, 2, \cdots, m$)
A	An adjacency matrix
B	An incidence matrix
D	A degree matrix
C_k	The k-th cluster ($k = 1, 2, \cdots, K$)
H	A cluster assignment matrix
g	A graph filter function
L_{sym}	A symmetric graph Laplacian matrix

Following the work of [15], it is useful to rewrite the IntraFD minimization problem in an alternative way:

$$\min_{H^T H = I} Tr(XX^T) - Tr(H^T XX^T H), \tag{2}$$

where X is the node attribute matrix whose rows are node attributes. H is the cluster assignment matrix such that $H_{ik} = (1/|C_k|)^{1/2}$ if node v_i belongs to cluster C_k. Otherwise, $H_{ik} = 0$. And $Tr(\cdot)$ is the trace of a matrix. In this case, H always fulfills the condition $H^T H = I$.

The IntraFD minimization problem is NP-hard. However, when the discreteness condition of H is relaxed (i.e., H can take arbitrary values in $\mathbb{R}^{n \times K}$ s.t. $H^T H = I$), the continuous optimal solution for H is given by the principal components of X via SVD. Therefore, it is suggested to perform SVD on node attributes before applying K-means to find high-quality clusters with low IntraFD [15].

2.2 Structure-based Clustering

In structure-based clustering, as shown in Fig. 1(c), only graph structure information (i.e., edge connections) is utilized to partition nodes into groups with dense edge connections internally and sparse edge connections between groups. Spectral clustering [26, 42] is one of the most popular graph clustering algorithms to find clusters that minimize the intra-cluster edge density (InterED), also known as NCut, as follows:

$$\min_{C_1, \cdots, C_K} \sum_{k=1}^{K} \frac{\sum_{v_i \in C_i, v_j \notin C_i} A_{ij}}{\text{volume}(C_k)}, \tag{3}$$

where A_{ij} is the edge weight between nodes v_i and v_j, and volume(C_j) measures the size of C_j by summing over the degrees of all nodes in C_j. Lower InterED implies higher quality of clusters w.r.t. graph structure. In other words, nodes in different clusters should be minimally connected, while the size of all clusters should be balanced.

To understand spectral clustering, it is helpful to rewrite the InterED minimization problem as follows [42]:

$$\min_{H^T H = I} Tr(H^T L_{sym} H), \tag{4}$$

where H is the cluster assignment matrix such that $H_{ik} = (\text{degree}(v_i)/\text{volume}(C_k))^{1/2}$ if $v_i \in C_k$. In other case, $H_{ik} = 0$. Obviously, $H^T H = I$ always holds. L_{sym} is the symmetric graph Laplacian matrix derived from the graph structure.

In a similar manner, the optimal continuous solution of H is given by the first K eigen-vectors of L_{sym} when its discreteness is removed. Therefore, spectral clustering is to compute the first k eigen-vectors of L_{sym} and feed them into K-means to find good clusters with low interED.

2.3 Graph Convolution on Node Attributes

In the task of Undi-AGC, as shown in figure 1(d), approaches that combine the information of node attributes and graph structure can achieve better clustering performance compared with either attribute-based or structure-based clustering methods (see also [21, 84]). Recently, graph convolution has gained much attention for integrating graph structure information on node attributes effectively, and improving the performance over many graph mining tasks [24, 27, 67, 84].

From the graph signal processing perspective [9, 58], graph convolution is regarded as a filter that removes the attribute noise at some frequencies in the spectral domain. Formally, given a specific filter function g, the graph convolution operation on node attributes X is defined as follows:

$$g * X = Ug(\Lambda)U^T X, \tag{5}$$

where Λ and U are eigenvalues and orthogonal eigenvectors of the symmetric graph Laplacian (i.e., $L_{sym} = U\Lambda U^T$) in increasing order. The graph convolution consists of three steps: (1) transform node attributes into the spectral domain (i.e., $Z = U^T X$); (2) remove attribute noise by scaling the spectrum with a scalar-valued filter function on eigen-values (i.e., $\overline{Z} = g(\Lambda)Z$, in which $g(\Lambda) = \text{diag}(g(\lambda_1), \cdots, g(\lambda_n))$); (3) perform inverse graph Fourier transform on the scaled spectrum of node attributes (i.e., $Y = U\overline{Z}$).

The effects of graph convolution on node attributes depend on the design of graph filters. For example, the graph filter in well-known GCNs [28] is $g(\lambda) = 1 - \lambda$, whose graph convolution on node attributes is as follows:

$$g_{GCN} * X = U(I - \Lambda)U^T X = (I - L_{sym})X = D^{-1/2}AD^{-1/2}X, \tag{6}$$

where A and D are adjacency matrix and diagonal degree matrix, and they are used to construct the symmetric graph Laplacian $L_{sym} = I - D^{-1/2}AD^{-1/2}$. However, the graph convolution in GCNs only utilizes graph structure information by aggregating the node attributes from their 1-hop neighbors. To capture the global graph structure, simplified graph convolution (SGC) performs the 1-hop graph convolution operation multiple times as follows [72]:

$$g_{SGC} * X = U(I - \Lambda)^L U^T X = (I - L_{sym})^L X = (D^{-1/2}AD^{-1/2})^L X, \tag{7}$$

where L is the length of hops that determines how far two nodes can aggregate information from each other. By performing SGC on attributed graphs, it has been shown to achieve the state-of-the-art performance on node classification tasks [38, 72].

2.4 Four Types of Graph Structures

In undirected or directed graphs, as shown in Figure 2(a) or (b), the edge connections are often represented by an adjacency matrix A, such that $A_{ij} = 1$ if there is an edge connecting between node v_i and v_j (or from node v_i to v_j). Otherwise, $A_{ij} = 0$.

In hyper-graphs, as shown in Figure 2(c), an adjacency matrix cannot be applied to express the higher-order relations when hyper-edges connect more than two nodes. Instead, an incidence matrix B is used to model hyper-edges, such that $B_{ij} = 1$ if node v_i belongs to hyper-edge e_j. Otherwise, $B_{ij} = 0$. Accordingly, the node degree$(v_i) = \sum_j B_{ij}$ represents the number of hyper-edges that are incident to node v_i, and the edge degree$(e_j) = \sum_i B_{ij}$ represents the number of nodes that belong to hyper-edge e_j.

In multi-relational graphs, as shown in Figure 2(d), there are a set of edge types, i.e., $\mathcal{R} = \{r_1, \cdots, r_R\}$, to describe different kinds of relationships. Therefore, we can use a set of adjacency matrices $\mathcal{A} = \{A^{r_1}, \cdots, A^{r_R}\}$ to model all types of edge connections, such that $A^r_{ij} = 1$ if there is an edge of type $r \in \mathcal{R}$ connecting node v_i and v_j.

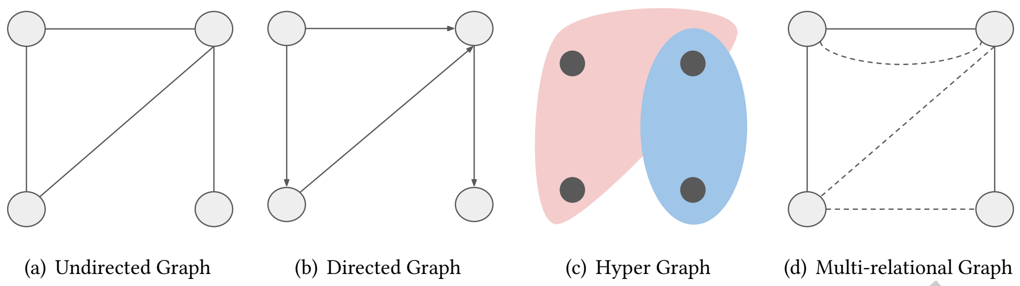


Fig. 2. Examples of graph structures with four types of edges. In (a) and (b), pairwise nodes are connected with undirected and directed edges, respectively. In (c), hyper-edges are used to connect two or more nodes. In (d), solid lines and dashed lines represent two types of relations.

Otherwise, $A^r_{ij} = 0$. In other words, an multi-relational graph can be decomposed as a set of simple undirected graphs of all edge types.

In summary, due to the different expressions among different types of graph structures, it is nontrivial to generalize the AGC framework from Undi-AGC to directed AGC (Di-AGC), Hyper-AGC, and multi-relational AGC (MR-AGC).

Discussion on Heterogeneous Graphs. Heterogeneous graphs contain multiple types of nodes and/or various types of edges. For example, in citation networks like ACM, there are three types of nodes (including author, paper, and subject), and there are two types of edges (representing author-write-paper and paper-belong-subject relations). To cluster nodes of a specific type (e.g., papers), it is a common practice to construct various meta-paths between two papers, such as paper-author-paper (describing two papers written by the same author), and paper-subject-paper (meaning that two papers belong to the same subject) [57, 70]. Therefore, heterogeneous graphs can be transformed into multi-relational graphs having homogeneous nodes (papers) and different types of edges (meta-paths). For ease of presentation, we use multi-relational graph structure to express heterogeneous graphs based on meta-paths.

3 PROBLEM STATEMENT

In this section, we formulate a general AGC problem. A general attributed graph is defined by $G = (V, E)$, where $V = \{(v_1, \mathbf{x}_1), (v_2, \mathbf{x}_2), \cdots, (v_n, \mathbf{x}_n)\}$ is a set of attributed nodes, each node v_i is associated with an attribute vector $\mathbf{x}_i$, and $E = \{(e_1, w_1), (e_2, w_2), \cdots, (e_m, w_m)\}$ consists of a set of weighted edges, each edge e_j is associated with a weight w_j.

The definition of edges depends on the specific graph structure. In undirected (or directed) graphs, each edge is denoted by a pair of nodes (v_i, v_j) connecting between nodes v_i and v_j (or from node v_i to v_j). In hyper-graphs, each hyper-edge is represented by a subset of nodes $e_j \subset V$. In multi-relational graphs, each edge is defined by (v_i, v_j, r) that connects nodes v_i and v_j with a edge type $r \in \mathcal{R}$.

Given an attributed graph G and the number of clusters K, the problem of AGC is to partition the nodes V into K disjoint subsets $C_1, C_2, \cdots, C_K$, such that $\cup_{k=1}^{K} C_k = V$ and $C_i \cap C_j = \emptyset$ for any $i \neq j$, with two objectives: (1) nodes in the same cluster have similar node attributes (i.e., low IntraFD), and (2) nodes from different clusters are sparsely connected (i.e., low InterED).

4 GRAPH CONVOLUTION BASED ATTRIBUTED GRAPH CLUSTERING

In this section, we present a graph convolution based algorithm for Undi-AGC specifically. Then we provide a theoretical analysis on how graph convolution improves the clustering performance.

4.1 Overview

Given an attributed undirected graph $G = (V, E)$, the node attributes and graph structure are represented by an attribute matrix X and an adjacency matrix A, respectively. The overview of our Undi-AGC algorithm, namely GRAC, is illustrated in Figure 3.

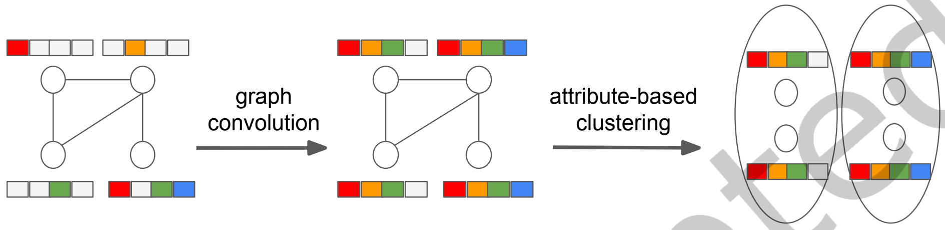


Fig. 3. The overview of graph convolution-based algorithm for Undi-AGC. It consists of two stages: (1) perform graph convolution on node attributes, and (2) feed processed node attributes into an attribute-based clustering algorithm.

The basic idea is to perform graph convolution on nodes attributes (i.e., $Y = g * X$), and then feed processed node attributes (Y) into an attribute-based clustering algorithm to find clusters. While it is simple to implement the algorithm, the challenge lies in how to select effective and efficient convolution and clustering approaches.

4.2 Convolution and Clustering Details

As we have discussed, the graph convolution in GCNs can only aggregate node attribute information from 1-hop neighbors. To capture the global structure information, we apply a generalized high-order graph convolution (HGC) on node attributes as follows [27]:

$$g_{HGC} * X = U(I - \alpha\Lambda)^L U^T X = (I - \alpha L_{sym})^L X, \tag{8}$$

where $\alpha > 0$ is the graph filtering rate that controls the degree of filtering attribute noise. L is the length of hops that determines how far two nodes can aggregate information from each other.

Even though the only difference between HGC (Eq. (7)) and SGC (Eq. (8)) is the graph filtering rate α, tuning this hyper-parameter is very important to improve the effectiveness of higher-order graph convolution. For example, Zhang et al. [84] set $\alpha = 1/2$, making HGC work well as a low-pass filter. On the other hand, HGC is different from GCN-based methods, as it has not introduced any trainable parameter and training process, so that it is much more time efficient than GCNs.

By performing HGC on attributed graphs, i.e., $Y = g_{HGC} * X$, we integrate the information of graph structure into node attributes, and the processed node attributes Y are readily fed into attribute-based clustering algorithms. As suggested in [15, 27], we apply SVD on Y to obtain the first K eigen-vectors, and then use them for K-means clustering. Note that K is the number of clusters.

In practice, the number of clusters K and graph filtering rate α are pre-defined, and the number of hops is determined by a stopping algorithm. That is, we perform HGC by repeating left multiplying $I - \alpha L_{sym}$ multiple times, until there is not much change in the stopping metric of consecutive iterations. The stopping metric that

Algorithm 1 GRAC: Graph convolution based Undi-AGC algorithm

```
    Input: Attributed graph $G = (V, E)$, number of clusters $K$, graph filtering rate $\alpha$,
           maximum iteration number $L_{\max}$, tolerance value $tol$
    Output: A set of clusters $C$
 1: Build adjacency matrix $A$ and attribute matrix $X$ from $G$
 2: Set $A_{ii} = 1$ for all nodes $v_i \in V$
 3: Compute symmetric graph Laplacian $L_{sym} = I - D^{-1/2}AD^{-1/2}$
 4: Initialize $Y^{(0)} = X$
 5: for $l = 1$ to $L_{\max}$ do
 6:     Perform $Y^{(l)} = (I - \alpha L_{sym})Y^{(l-1)}$
 7:     Obtain $K$ clusters $C^{(l)}$ by applying SVD and K-means on $Y^{(l)}$
 8:     if $Comp(C^{(l-1)}) - Comp(C^{(l)}) \leq tol$ then
 9:         count = count + 1
10:     if count $\geq$ 2 then
11:         break
12: Set $C = C^{(l)}$
```

measures the compactness of clusters is given as follows:

$$Comp(C) = \frac{1}{|C|}\sum_{k=1}^{K}\frac{1}{|C_k|(|C_k|-1)}\sum_{v_i,v_j\in C_k, v_i\neq v_j}||\mathbf{y}_i-\mathbf{y}_j||^2, \tag{9}$$

where $C = \{C_1, \cdots, C_K\}$ is a set of clusters. Each node v_i is associated with an attribute vector $\mathbf{y}_i$.

The pseudo-code of GRAC is described in Algorithm 1. First, we build an attribute matrix and graph Laplacian (with self-loops) in Lines 1-3, and then perform HGC on node attributes and obtain clusters via SVD and K-means on processed attributes $Y^{(l)}$, until the compactness of clusters changes very little or the maximum iteration number is reached. Given the dimension of node attributes as d, the number of iterations as L, the number of nodes as n, and the number of edges as m, the total time complexity of GRAC is $O((md + nd^2 + nK^2)L)$, where $O(md)$ is the complexity for one left-multiplication by sparse matrix $I - \alpha L_{sym}$, $O(nd^2)$ for SVD, and $O(nK^2)$ for K-means.

4.3 Properties of GRAC

We provide an analysis on how to improve clustering performance via graph convolution. In GRAC, node attributes are transformed to Y via HGC. As we have discussed in Section 2, performing SVD and K-means on Y is equivalent to optimizing the following problem:

$$\min_{H^TH=I} Tr(YY^T) - Tr(H^TYY^TH) = \min_{H^TH=I} ||YY^T - HH^T||_F^2, \tag{10}$$

where Y is the transformed node attributes, H is the cluster assignment matrix, and $||\cdot||_F$ is the Frobenius norm. The right-hand is derived from the fact that $||YY^T - HH^T||_F^2 = Tr(YY^TYY^T) + n - 2Tr(H^TYY^TH)$. Therefore, by solving the right-hand optimization problem [16], the optimal solution retains that $H_{opt}H_{opt}^T \approx YY^T$. Here, H_{opt} is a good approximate solution of clustering on Y.

On the other hand, the aim of Undi-AGC is to find clusters with low IntraFD and InterED. To understand the effects of graph convolution on the quality of clusters, we measure both IntraFD and InterED approximately with the solution H_{opt} as follows:

$$\text{IntraFD}(H_{opt}) = Tr(XX^T) - Tr(H_{opt}^T XX^T H_{opt}) \approx Tr(XX^T) - Tr(XX^T YY^T), \tag{11}$$

$$\text{InterED}(H_{opt}) = Tr(H_{opt}^T L_{sym} H_{opt}) \approx Tr(Y^T L_{sym} Y). \tag{12}$$

Both derivations are based on the properties of $H_{opt}H_{opt}^T \approx YY^T$, and $Tr(AB) = Tr(BA)$ for any two matrices. The approximate terms are called IntraFD_approx and InterED_approx, respectively.

We now provide a proof on the effects of HGC on both IntraFD_approx and InterED_approx.

Theorem 4.1. *The graph convolution based algorithm for attributed graph clustering produces clusters with lower InterED_approx and higher IntraFD_approx, compared to attribute-based clustering.*

Proof. In Algorithm 1, the transformed node attributes are obtained by $Y = (I - \alpha L_{sym})^L X$. Let $L_{sym} = U\Lambda U^T$ and $Z = U^T X$ (or $X = UZ$), we have $Y = U(I - \alpha\Lambda)^L Z$. Replacing X and Y in IntraFD_approx and InterED_approx, we have

$$\text{IntraFD\_approx} = \sum_{i=1}^{n} \mathbf{z}_i^T \mathbf{z}_i - \sum_{i=1}^{n}\sum_{j=1}^{n} (1 - \alpha\lambda_i)^L (1 - \alpha\lambda_j)^L (\mathbf{z}_i^T \mathbf{z}_j)^2, \tag{13}$$

$$\text{InterED\_approx} = \sum_{i=1}^{n} \lambda_i (1 - \alpha\lambda_i)^{2L} (\mathbf{z}_i^T \mathbf{z}_i), \tag{14}$$

where $\lambda_1, \cdots, \lambda_n$ are eigen-values of L_{sym} in ascending order. Each node v_i has its spectral attribute vector $\mathbf{z}_i$ (i.e., the i-th row in Z). Given any attribute graph, it is clear that the values of $\mathbf{z}_i^T \mathbf{z}_i$ and $(\mathbf{z}_i^T \mathbf{z}_j)^2$ are fixed and non-negative, and the eigen-values hold that $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$ [42]. Therefore, if we set a proper graph filtering rate, i.e., $\alpha \leq 1/\lambda_n$, the graph convolution based algorithm (when $L > 0$) produces clusters with higher IntraFD_approx and lower InterED_approx compared to attribute-based clustering (when $L = 0$). □

From the above analysis, we can conclude that our graph convolution based algorithm helps to improve the clustering performance by producing clusters with very low InterED, at the cost of increasing IntraFD. With proper hyper-parameters (α and L), it is effective to find good clusters that achieve a balance between IntraFD and InterED measures. Furthermore, this argument holds for any other graph structure when its graph Laplacian and graph convolution are properly designed.

5 EXTENSIONS TO OTHER TYPES OF GRAPH STRUCTURES

In this section, we develop an extension of GRAC algorithm (called GRACE) to other types of graph structures, including directed, hyper-, and multi-relational graphs. This means that GRACE is a general graph convolution based framework that is applicable to Undi-AGC, Di-AGC, Hyper-AGC, and MR-AGC tasks. By using a general framework, it is useful and flexible to model many different relations with specific edge types.

In our GRAC algorithm, it consists of building graph Laplacian, performing graph convolution on node attributes, and applying K-means with SVD to find clusters. To extend GRAC from Undi-AGC to other types of graph structures, we need to build a suitable graph Laplacian for other types of graph structures. For ease of presentation, we summarize different graph Laplacians in Table 2.

5.1 Graph Laplacians for Directed Graphs

So far we have built the symmetric graph Laplacian L_{sym} for undirected graphs. $L_{sym} = I - D^{-1/2}AD^{-1/2}$ was proposed in spectral clustering to find clusters by minimizing the NCut (Eq. 3 or Eq. 4), where A and D are the adjacency matrix and degree matrix, respectively. However, the symmetric graph Laplacian cannot be directly

Table 2. Graph Laplacians

Name	Definition
symmetric (undirected) graph Laplacian	$L_{sym} = I - D^{-1/2}AD^{-1/2}$
random walk directed graph Laplacian	$L_{rwd} = I - (\Pi^{1/2}P\Pi^{-1/2} + \Pi^{-1/2}P^T\Pi^{1/2})/2$
fast spectral normalized directed Laplacian	$L_{fsd} = I - (D_{out}^{-\frac{1}{2}}(A + A^T)D_{out}^{-\frac{1}{2}})/2.$
symmetric directed graph Laplacian	$L_{sd} = I - D_s^{-1/2}A_sD_s^{-1/2}$, where $A_s = A + A^T$
linear hyper-graph Laplacian	$L_{LH} = I - D_v^{-1/2}BD_e^{-1}B^TD_v^{-1/2}$
non-linear hyper-graph Laplacian	$L_{NLH} = I - D_s^{-1/2}A_sD_s^{-1/2}$, with mediators
multi-relational graph Laplacian	$L_{MR} = \sum_{r\in\mathcal{R}} \beta_r \cdot L_{sym}^r$

built for a directed graph, as its adjacency matrix is asymmetric and its degree matrix varies from in-degree and out-degree definitions.

In [11, 43, 86], the NCut objective has been extended to directed graphs using the concept of random walk. Accordingly, the random walk directed graph Laplacian has been proposed as follows:

$$L_{rwd} = I - (\Pi^{1/2}P\Pi^{-1/2} + \Pi^{-1/2}P^T\Pi^{1/2})/2, \tag{15}$$

where P is the transition matrix of a random walk, and Π is the diagonal matrix of its associated stationary distribution π. However, it is time-consuming to obtain the stationary distribution (e.g., via power iteration) that satisfies $\pi P = \pi$. Due to the complexity of computing the stationary distribtion, [35] proposed the fast spectral normalized directed Laplacian $L_{fsd} = I - \frac{1}{2}D_{out}^{-\frac{1}{2}}(A + A^T)D_{out}^{-\frac{1}{2}}$ as an approximation to L_{rwd}.

Moreover, a simple but effective method is to symmetrize a directed graph into an undirected graph [46, 55]. For example, we can derive the symmetric adjacency matrix (A_s) from the original asymmetric adjacency matrix (A) via $A_s = A + A^T$. In other words, the weight of an edge in the built symmetric graph is the sum of the weights on two directed edges. Note that this is different from simply ignoring the directionality of the edges. With A_s, we have symmetric directed graph Laplacian as $L_{sd} = I - D_s^{-1/2}A_sD_s^{-1/2}$, where D_s is the degree matrix built from A_s. In addition, D_s can be replaced by either out-degree matrix or in-degree matrix built from directed graphs [35]. In this work, we observe that L_{sd} with the degree matrix D_s performs best, which is adopted in GRACE. As the number of undirected edges is at most doubled by $A_s = A + A^T$, the time complexity of its corresponding graph convolution is $O(md)$ for one sparse matrix multiplication, where m is the number of directed edges and d is the dimension of node attributes.

5.2 Graph Laplacians for Hyper-Graphs

For hyper-graphs, each hyper-edge can connect more than two nodes. As discussed, we use an incidence matrix B to represent all hyper-edges, such that $B_{ij} = 1$ if and only if node v_i belongs to hyper-edge e_j. With the incidence matrix B, we have two diagonal matrices, node degree matrix D_v and edge degree matrix D_e, which represent the number of incident hyper-edges and nodes, respectively. Therefore, in hyper-graph clustering, the linear hyper-graph Laplacian was proposed to minimize the NCut on hyper-edges as follows [54, 85]:

$$L_{LH} = I - D_v^{-1/2}BD_e^{-1}B^TD_v^{-1/2}. \tag{16}$$

L_{LH} is the extension of L_{sym} on hyper-edges. Specifically, each hyper-edge is regarded as a clique, i.e., nodes inside a hyper-edge are fully connected. Then, when a hyper-edge e is cut, it is equivalent to partitioning the resulting clique. There are other variants of linear hyper-graph Laplacians [1, 52]. However, the construction of these Laplacians has suffered from inefficient graph convolution, with the time complexity of $O(m_{LH}d)$, where $m_{LH} = \sum_{e\in E}(|e|)(|e| - 1)/2$ is the number of the resulting clique edges.

To design an efficient hyper-graph convolution, we leverage the non-linear hyper-graph Laplacian [8, 27, 80] by converting an attributed hyper-graph into an undirected graph as follows:

- We find two farthest nodes inside each hyper-edge, i.e., $(i_e, j_e) = \arg\max_{v_i, v_j \in e} ||\mathbf{x}_i - \mathbf{x}_j||^2$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are attribute vectors of nodes v_i and v_j. The rest of nodes in each hyper-edge e is denoted as mediators, i.e., $P_e = \{p \in e, p \neq i_e, p \neq j_e\}$.
- We construct an edge (i_e, j_e) between two farthest nodes, and construct other edges between i_e and mediators of P_e, and between j_e and mediators of P_e for each hyper-edge $e \in E$. We choose each edge weight as $1/(2|e| - 3)$ as there are $2|e| - 3$ edges built for a hyper-edge e.
- We build the symmetric non-linear hyper-graph Laplacian as $L_{NLH} = I - D_s^{-1/2} A_s D_s^{-1/2}$, where A_s and D_s are the adjacency matrix and degree matrix of the constructed undirected graph.

In the construction of L_{NLH}, we only consider a part of representative edges that are incident to two farthest nodes, instead of using a fully connected sub-graph for each hyper-edge. This is because nodes with very dissimilar attributes are more likely partitioned into different clusters and then counted as a cut on this hyper-edge. As a result, the number of edges for constructing L_{NLH} is $m_{NLH} = \sum_{e \in E} 2|e| - 3$. The corresponding hyper-graph convolution becomes much more efficient with a time complexity of $O(m_{NLH} d)$.

The construction of L_{NLH} relies on the node attributes by choosing two farthest nodes for each hyper-edge. When applying high-order hyper-graph convolution, the node attributes are processed and changed at each iteration. Therefore, $L_{NLH}^{(l-1)}$ should be constructed dynamically based on the current node attributes $Y^{(l-1)}$, before we perform $Y^{(l)} = (I - \alpha L_{NLH}^{(l-1)}) Y^{(l-1)}$ at each iteration.

5.3 Graph Laplacians to Multi-relational Graphs

Finally, in a multi-relational graph, different types of edge connections can be modelled by a set of adjacency matrices $\mathcal{A} = \{A^{r_1}, \cdots, A^{r_R}\}$. Correspondingly, we can derive a set of symmetric graph Laplacians $\{L_{sym}^{r_1}, \cdots, L_{sym}^{r_R}\}$. To cluster a multi-relational graph, it is a common practice to find clusters by minimizing the weighted sum of heterogeneous NCuts of all edge types as follows:

$$\min_{H^T H = I} \sum_{r \in \mathcal{R}} \beta_r \cdot Tr(H^T L_{sym}^r H) = \min_{H^T H = I} Tr\big(H^T (\sum_{r \in \mathcal{R}} \beta_r L_{sym}^r) H\big) \tag{17}$$

where H denotes the cluster assignment matrix, $\beta_r > 0$ is the coefficient for each edge type $r \in \mathcal{R}$. The above equation implies that minimizing the weighted sum of heterogeneous NCuts is equivalent to minimizing the NCut with the weighted sum of symmetric graph Laplacians. And then we can define our multi-relational graph Laplacian as follows:

$$L_{MR} = \sum_{r \in \mathcal{R}} \beta_r \cdot L_{sym}^r. \tag{18}$$

It is worth noting that tuning the coefficients β_r separately can further improve the clustering performance, compared to simply setting them the same for all different relation types. When the L_{MR} is applied to multi-relational graph convolution, the time complexity is $O(md)$, where m is the number of edges of all types.

5.4 A General Graph Convolution Framework

Now we have studied different graph Laplacians (summarized in Table 2) for all four types of graph structures. We can put them into the GRACE framework for clustering attributed graphs with different types of graph structures. The pseudo-code of GRACE is given in Algorithm 2. First, we build a specific graph Laplacian L_G according to the graph type T_G in Lines 14-30. That is, we convert the given graph into the undirected graph (or graphs), and build the corresponding adjacency matrix (or matrices). And then we add self-loops before constructing the graph Laplacian in the form as we have discussed.

Algorithm 2 GRACE: A general graph convolution framework for AGC

```
Input: Attributed graph $G = (V, E)$, graph type $T_G$, number of clusters $K$,
       graph filtering rate $\alpha$, maximum iteration number $L_{\max}$, tolerance value $tol$
Output: A set of clusters $C$
1:  procedure FindClusters($G$, $T_G$, $K$, $\alpha$, $L_{\max}$, $tol$)
2:      Build node attribute matrix $X$ from $G$, and initialize $Y^{(0)} = X$
3:      Compute graph Laplacian $L_G$ = BuildLaplacian($G$, $T_G$, $Y^{(0)}$)
4:      for $l = 1$ to $L_{\max}$ do
5:          Perform $Y^{(l)} = (I - \alpha L_G) Y^{(l-1)}$
6:          Obtain $K$ clusters $C^{(l)}$ by applying SVD and K-means on $Y^{(l)}$
7:          if $Comp(C^{(l-1)}) - Comp(C^{(l)}) \leq tol$ then
8:              count = count + 1
9:          if count $\geq 2$ then
10:             break
11:         if $T_G$ is hyper-graph then
12:             Recompute $L_G$ = BuildLaplacian($G$, $T_G$, $Y^{(l)}$)
13:     Set $C = C^{(l)}$

14: function BuildLaplacian($G$, $T_G$, $S$)
15:     if $T_G$ is undirected then
16:         Build adjacency matrix $A$ from $G$
17:     else if $T_G$ is directed then
18:         Build adjacency matrix $A$ from $G$, and set $A = A + A^T$
19:     else if $T_G$ is hyper-graph then
20:         Initialize $A_{ij} = 0$ for $\forall v_i, v_j \in V$
21:         for $e \in E$ do
22:             $i_e, j_e = \arg\max_{i,j \in e} ||S_i - S_j||^2$
23:             $A_{i_e,j_e} = A_{j_e,i_e} = \frac{1}{2|e|-3}$
24:             $P_e = \{p \in e : p \neq i_e, p \neq j_e\}$
25:             $A_{i_e,p} = A_{p,i_e} = A_{j_e,p} = A_{p,j_e} = \frac{1}{2|e|-3}$ for $\forall p \in P_e$
26:     else if $T_G$ is multi-relational then
27:         Build a set of adjacency matrices $\mathcal{A} = \{A^{r_1}, \cdots, A^{r_R}\}$ from $G$
28:     Set $A_{ii} = 1$ (or $A^r_{ii} = 1$) for all nodes $v_i \in V$ (of all types $r \in \mathcal{R}$)
29:     Compute $L_G = I - D^{-1/2} A D^{-1/2}$ (or the weighted sum of Laplacians $L^r_{sym}$ of all types)
30:     Return $L_G$
```

Next, similarly to GRAC, we use the constructed graph Laplacian to perform graph convolution on node attributes, and then apply K-means with SVD on processed node attributes to find good clusters until the iteration is stopped. It is worth noting that L_G is built once at the beginning for all graph types except hyper-graphs. For hyper-graphs, we construct L_G dynamically at each iteration before the graph convolution. The time complexity of building graph Laplacian is $O(m)$, where m is the number of edges of the built graph. And the total time complexity of GRACE is $O((md + nd^2 + nK^2)L)$, where n, m, d, K, and L are the number of nodes, the number of edges, attribute dimension, the number of clusters, and iteration number, respectively.

Table 3. Statistics of the datasets.

Dataset	Type	#Node	#Edge	$\lvert\mathcal{R}\rvert$	#Attribute	#Class
Undirected Cora	Undi	2708	5429	-	1433	7
Citeseer	Undi	3327	4732	-	3703	6
Pubmed	Undi	19717	44338	-	500	3
Wiki	Undi	2405	17981	-	4973	17
Cora-ML	Di	2995	8416	-	2879	7
Directed Citeseer	Di	3312	4715	-	3703	6
Cora Co-citation	Hyper	2708	1579	-	1433	7
Citeseer Co-citation	Hyper	3327	1079	-	3703	6
Pubmed Co-citation	Hyper	19717	7963	-	500	3
Cora Co-authorship	Hyper	2708	1072	-	1433	7
DBLP Co-authorship	Hyper	43413	22535	-	1425	6
ACM	MR	8916	2240042	2	1870	3
IMDB	MR	9717	80216	2	2000	3
Hete DBLP	MR	26128	12055179	3	334	4

6 EXPERIMENTS

We conduct experiments to show the effectiveness, efficiency and generality of GRACE for clustering different types of attributed graphs.

6.1 Datasets

We used 14 datasets in our experiments with 4 different graph types: undirected (Undi), directed (Di), Hyper, and multi-relational (MR) graphs. The statistics of the datasets are summarized in Table 3. #Node and #Edge are the number of nodes and edges. $|\mathcal{R}|$ represents the number of edge types for MR graphs. #Attribute is the dimension of the node attribute vector. #Class is the number of classes i.e., the number of ground-truth labels.

For undirected graphs, Undirected Cora, Citeseer, Pubmed, and Wiki are commonly used to evaluate undirected AGC algorithms [66, 84]. The first three datasets are citation networks, where nodes are publications and they are connected if one cites another (ignoring the directionality). Wiki is a web-page network whose nodes are websites and are connected if one links another. The nodes in Undirected Cora and Citeseer are associated with binary word vectors, while the nodes in Pubmed and Wiki are associated with tf-idf weighted word vectors. Pubmed is connected, i.e., there is a path from any node to any other node, while Undirected Cora, Citeseer and Wiki are not. Moreover, the edges of Undirected Cora, Citeseer, and Pubmed are not weighted, i.e., $A_{ij} \in \{0, 1\}$, while the edges of Wiki are weighted, i.e., $A_{ij} \in [0, 16]$. **For directed graphs**, we used two citation networks Cora-ML and Directed Citeseer by maintaining the directionality [64].

For hyper-graphs, we used two Co-authorship networks: Cora Co-authorship and DBLP Co-authorship, and three Co-citation networks: Cora Co-citation, Citeseer Co-citation and Pubmed Co-citation [80]. For Co-authorship data, publications co-authored by an author are connected in one hyper-edge, while for Co-citation data, all publications cited by a publication are in one hyper-edge. Each publication is represented by bag-of-words attribute vector. Hyper-edges that have only one hyper-node are removed. The larger datasets DBLP Co-authorship and Pubmed Co-citation are typically used for time and memory efficiency validation.

For multi-relational graphs, we used three multi-relational graphs: ACM, IMDB, and Hete DBLP [53, 70], which are constructed from heterogeneous graphs. ACM consists of 3025 papers (P), 5835 authors (A) and 56 subjects (S). To cluster papers into different groups, we used the PAP (i.e., papers-authors-papers) and PSP (i.e., papers-subjects-papers) meta-paths to construct the two-type multi-relational (MR) graph for ACM. The IMDB consists of 3550 movies (M), 4441 actors (A), and 1726 directors (D). To cluster movies, we used the MAM and MDM meta-paths to construct the two-type multi-relational (MR) graph for IMDB. Finally, DBLP consists of 4057 authors (A), 14328 papers (P), 20 conferences (C), and 7723 terms (T). To cluster authors, we constructed the three-type multi-relational (MR) graph for DBLP based on the APA, APCPA, and APTPA meta-paths.

6.2 Evaluation Metrics and Experimental Setups

In our experiments, we used four unsupervised measures: IntraFD and InterED, and their approximations, i.e., IntraFD_approx and InterED_approx (see Section 4.3) to evaluate the clustering quality w.r.t. node attributes and graph structure, respectively. For all the measures, the lower their values the better is the quality of the clustering result. Lower IntraFD and IntraFD_approx mean that nodes with similar attributes are more likely grouped in the same cluster, while lower InterED and InterED_approx mean that less edges are connected between different clusters. In addition, we also used three commonly used supervised measures to evaluate the clustering results based on the ground-truth labels: clustering accuracy (Acc), normalized mutual information (NMI), and f1-macro score (F1) [49, 66, 67, 71, 74, 84]. For all supervised measures, a higher value indicates a better clustering quality.

We conducted all experiments on a Linux system with an Intel (R) Xeon (R) Silver 4114 CPU at 2.20GHz with 256GB of RAM, except the experiments with multi-relational graphs. This is because the state-of-the-art baseline SpectralMix [53] was implemented in Java [3] and incurred an execution error in our Linux server. To solve this issue, we ran all experiments for multi-relational graphs on a windows system with an Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz with 16G of RAM. We report the running time (in seconds) averaged over 10 runs of each dataset (excluding data loading and pre-processing), and the maximum process memory usage (in MB) of each algorithm. We will release the code and data for reproducibility of the results.

6.3 Baselines and Parameter settings

Table 4. Properties of the algorithms.

Baseline	Input	Undi	Di	Hyper	MR
K-Means	attribute	✓	✓	✓	✓
N-cut	structure	✓	✓	✓	✓
ARGE	both	✓			
ARVGE	both	✓			
AdaGCN	both	✓			
GNMF	both		✓	✓	
JNMF	both		✓	✓	
SpectralMix	both				✓
GRACE	both	✓	✓	✓	✓

We compared GRACE with the baselines on different types of graph structures. Their main characteristics are summarized in Table 4, and the parameter settings are introduced below.

[3]https://gitlab.cs.univie.ac.at/yllis19cs/spectralmixpublic/-/tree/master/SpectralMix

K-means [44] and N-cut [26] are clustering algorithms that use only node attribute or graph structure information. K-means can simply be applied on all datasets because it ignores any type of graph structures. N-cut was originally designed for clustering simple undirected graphs. However, as stated in Section 5, it can be used to cluster other types of graphs with properly designed graph Laplacians. Specifically, we selected the fast spectral normalized directed Laplacian [35] L_{fsd}, the linear hyper-graph Laplacian [31] L_{LH}, and multi-relational graph Laplacian L_{MR} for corresponding graph structures, as they performed better compared to other graph Laplacian candidates. For the L_{MR} laplacian, we used the same weights β that we used for GRACE.

ARGE [49], ARVGE [49], and AdaGCN [84] are the state-of-the-art Undi-AGC methods that combine both node attribute and graph structure information via graph convolution using the Laplacian L_{sym}. We tuned ARGE and ARVGE for better performance. We used the discriminator's learning rate as of 0.001, 0.0001, 0.005, and 0.0005 on Undirected Cora, Citeseer, Pubmed and Wiki, respectively. We equally set the learning rate with these same values on these datasets. For all datasets, we used 200 training epochs, 64 units in the third hidden layer and 32 in the first and 16 in the second, and set the drop out rate and weight decay to 0. As for AdaGCN, we used the same parameter settings following the original paper [84].

GNMF [6] and JNMF [17] are two non-negative matrix factorization (NMF) based methods used to cluster attributed directed or hyper-graphs. Both algorithms were *not* designed for directed or hyper-graphs. However, following the work of [27], we modified both algorithms by incorporating the L_{sd} or L_{LH} Laplacians for directed graphs and hypergraphs respectively into the objective functions, and applying the same multiplicative update rules as [27] for optimizations. On directed graphs, we set $\lambda = 0.53$ on Cora-ML and $\lambda = 20$ on Directed Citeseer for GNMF, and set $\alpha = 0.08$, $\beta = 0.8$ on Cora-ML, and $\alpha = 20$, $\beta = 1$ on Directed Citeseer for JNMF.

SpectralMix [53] is the state-of-the-art algorithm used for clustering attributed multi-relational graphs. We kept the same hyper-parameters as in [53] for ACM and IMDB. For Hete DBLP, we set dimensionality as 3 (we tried 2, 3 and 9, but 3 gave the best performance), $iter$ as 10 and $extra\_iter$ as 2.

For GRACE, we tuned hyper-parameters on different datasets. On undirected graphs, we used $\alpha = 0.2$ and $tol = 0.15$ for Undirected Cora, Citeseer, and Wiki, and used $\alpha = 0.5$ and $tol = 0$ for Pubmed. We applied L_2 normalization (for Undirected Cora) and L_1 normalization (for Wiki and Pubmed) on processed node attributes before feeding into SVD. On directed graphs, we used $\alpha = 0.1$ and $tol = 0$ for Cora-ML, and used $\alpha = 0.5$ and $tol = 0.15$ for Directed Citeseer. On hyper-graphs, we used $\alpha = 0.5$ and $tol = 0.1$ for all datasets except Cora Co-authorship and DBLP Co-authorship, where we set $tol = 0.12$ and $tol = 0.5$, respectively. We used L_2 normalization on Y for Cora Co-authorship, Cora Co-citation and Pubmed Co-citation datasets. On multi-relational graphs, we used β as 0.7 and 0.3 for the PAP and PSP relations on ACM, β as 0.5 and 0.5 for the MAM and MDM relations on IMDB, and β as 0.3 each for the APA, APCPA and APTPA relations on Hete DBLP. We set $\alpha = 0.1$ and $tol = 0$ and used L_2 normalization for all multi-relational graphs. The Ls selected by the stopping criterion or maximum power L for GRACE were: 40 for all heterogeneous graphs; 11, 16, 2, 23, and 17 for Cora Co-citation, Citeseer Co-citation, Pubmed Co-citation, Cora Co-authorship and DBLP Co-authorship; 26, 14, 21 and 40 for Citeseer, wiki, Undirected Cora, and Pubmed; and finally 18 and 40 for Directed Citeseer and Cora-ML, respectively. Concerning the parameter selection strategy, (a) we used a random search for α using values in the range $[0.1, 0.5]$ in order to keep the filters $g(L)$ low-pass i.e., decaying, (b) likewise we used a random search for tol using values in the range $[0, 1]$, and (c) for β, we started by giving equal importance to all relations using $\beta = \frac{1}{|\mathcal{R}|}$ for all relations, where $|\mathcal{R}|$ is the number of relations, and then used random search to give different importance to different relations while ensuring that $\sum_i^{|\mathcal{R}|} \beta_i = 1$. In general, any values below $\frac{1}{\lambda_{max}}$ will be suitable for α, where λ_{max} is the maximum eigen-value of the laplacian. Thus, we propose using values below 0.5 for α. However, for faster graph clustering, α should not be set too low. For β_i, we recommend assigning the same weights to all relations (except when the user has prior knowledge of which relation is most important), while ensuring that $\sum_i^{|\mathcal{R}|} \beta_i = 1$. The main challenge is how to select tol, for which values close to 0 are generally better according to our experiments.

6.4 Clustering Performance Comparison

6.4.1 Performance on undirected graphs. We first compared the clustering performance on four undirected graphs: Undirected Cora, Citeseer, Pubmed, and Wiki. The results are reported in Table 5 and Table 6. The best results amongst all algorithms are in bold.

Table 5. Clustering performance on Undirected Cora and Citeseer.

Baselines	Undirected Cora					Citeseer				
	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)
K-Means	37.22	16.64	32.09	01.11	172.58	43.04	20.68	39.26	03.61	234.76
N-cut	28.84	04.64	08.93	**00.56**	**156.83**	22.03	01.78	10.81	**01.26**	**151.14**
ARGE	62.19	44.13	63.60	128.32	5152.32	54.40	28.22	53.21	256.89	1608.60
ARVGE	68.24	50.49	68.04	111.10	4347.57	54.40	26.10	52.90	230.76	1694.24
AdaGCN	68.94	**53.74**	65.63	04.10	212.76	67.30	41.46	62.77	31.24	435.32
GRACE	**72.01**	53.30	**72.35**	07.18	240.00	**68.14**	**42.06**	**63.58**	16.34	427.00

Table 6. Clustering performance on undirected Pubmed and Wiki.

Baselines	Pubmed					Wiki				
	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)
K-Means	59.50	31.13	58.14	**02.23**	**268.83**	33.30	30.14	22.72	06.12	315.38
N-cut	58.96	18.30	43.53	26.63	359.00	35.67	35.51	27.90	**01.23**	**251.67**
ARGE	69.48	32.93	68.80	5667.13	15528.80	46.15	44.19	41.04	739.56	3997.67
ARVGE	67.99	30.39	67.37	4638.73	18451.83	45.29	44.03	39.90	728.08	3184.21
AdaGCN	69.82	31.56	68.77	32.77	421.04	46.90	44.30	38.51	08.98	412.83
GRACE	**70.40**	**32.60**	**69.68**	24.31	498.36	**62.50**	**54.43**	**46.19**	13.94	504.25

From the tables, we can conclude that the K-Means and N-cut algorithms, which utilize only node attribute or graph structure information respectively, produce poorer clustering results. K-Means outperforms N-cut on Undirected Cora, Citeseer, and Pubmed, but not on Wiki. This may be due to the fact that the node attributes provide rich information in the first three citation networks, while the weighted edges on Wiki help more for clustering. K-Means and N-cut are generally fast and use less memory as they do not combine the information. Moreover, the structure-based N-cut is more time and memory efficient on small graphs, e.g., Undirected Cora, Citeseer, and Wiki, but becomes more time and memory consuming when the graph size increases, e.g., on Pubmed.

The methods that combine node attribute and graph structure outperform K-Means and N-cut by a large margin, achieving 8.5% – 46.1% improvements in terms of Acc. The ARGE and ARVGE algorithms are the most time and memory consuming because they require training neural networks to combine node attribute and graph structure information e.g., for the large Pubmed dataset. The running time and memory usage of ARGE and ARVGE are 2000 times and 50 times, respectively more than those of K-Means. Using graph convolution instead, the state-of-the-art AdaGCN algorithm can produce better clustering results with much less time and memory cost.

Following the idea of AdaGCN, GRACE exploits effective and efficient graph convolution on undirected graphs to combine node attribute and graph structure information. It remarkably outperforms K-Means and N-cut, and significantly saves time and memory cost compared to ARGE and ARVGE. Furthermore, it improves the state-of-the-art performance of AdaGCN while being comparable in terms of time and memory cost. Especially for the Wiki dataset, the improvements of GRACE are 15.6%, 10.1% and 7.7% in terms of Acc, NMI, and F1, respectively. Importantly, GRACE can deal with various types of graph structures (as shown below), while AdaGCN was proposed for undirected graphs only.

6.4.2 Performance on directed graphs. We then evaluated the clustering performance on two directed graphs, Cora-ML and Directed Citeseer, and their results are given in Table 7.

Table 7. Clustering performance on directed datasets.

Baselines	Cora-ML					Directed Citeseer				
	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)
K-Means	50.62	32.30	47.00	03.40	**285.10**	41.06	18.74	37.99	04.98	**205.60**
N-cut	52.35	32.56	48.58	**00.37**	296.23	46.49	21.27	43.15	**00.40**	307.53
GNMF	68.21	43.58	68.09	03.95	240.93	63.86	36.97	60.57	10.23	295.55
JNMF	50.62	24.79	48.68	64.52	278.44	64.46	36.59	60.55	135.19	330.49
GRACE	**75.09**	**60.73**	**71.48**	22.79	309.64	**68.69**	**42.97**	**63.79**	10.98	421.81

The table shows that neither K-Means (attribute-based) nor N-cut (structure-based) algorithm can produce satisfying clustering results on attributed directed graphs. N-cut on these two datasets always outperforms K-Means when the directional information of graph structure has been utilized. The GNMF and JNMF algorithms, which combine node attribute and graph structure information using their respective objective functions, can generally obtain better clustering results compared to the K-Means and N-cut competitors. The results from GNMF and JNMF are comparable to each other, while GNMF usually takes less time and memory due to its simpler objective function and hence faster update rule. By applying graph convolution on directed graphs, GRACE consistently achieves the best clustering performance. The improvements of GRACE in terms of NMI are 17.2% on Cora-ML and 6.0% on Directed Citeseer. Though GRACE takes more time and memory space compared to GNMF, the differences are negligible as both directed datasets are very small. We will present the efficiency of GRACE later on larger hyper-graphs and multi-relational graphs.

6.4.3 Performance on hyper-graphs. We then analyzed the clustering results on two Co-citation datasets [4] (as shown in Table 8), and two Co-authorship datasets (as shown in Table 9).

Table 8. Clustering performance on Co-citation datasets.

Baselines	Cora Co-citation					Pubmed Co-citation				
	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)
K-Means	33.27	13.07	22.12	**01.48**	**179.64**	59.47	31.10	58.14	**02.64**	**195.45**
N-cut	30.02	01.12	07.39	02.41	343.35	39.95	00.07	19.09	105.20	12525.82
GNMF	43.91	22.58	40.56	20.92	337.52	55.61	20.84	56.98	147.76	8828.43
JNMF	45.31	23.01	42.61	36.95	382.82	51.61	14.04	49.22	603.02	10751.11
GRACE	**55.39**	**34.65**	**50.50**	04.88	283.75	**60.89**	**31.14**	**60.17**	04.23	448.30

[4]The comparison on the Citeseer Co-citation dataset is omitted, and it has very similar results as seen in [27].

Table 9. Clustering performance on Co-authorship datasets.

Baselines	Cora Co-authorship					DBLP Co-authorship				
	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)	Acc%	NMI%	F1%	Time (Sec)	Mem (MB)
K-Means	33.27	13.07	22.12	**01.46**	**183.30**	61.25	38.30	60.90	**21.34**	**1262.65**
N-cut	29.25	02.00	09.50	07.42	374.96	27.58	01.00	09.25	1751.96	54187.64
GNMF	51.22	27.49	44.57	21.01	327.47	63.71	47.91	58.38	1100.76	41845.62
JNMF	49.41	28.63	44.29	11.16	422.26	61.80	44.72	58.82	3642.91	48216.73
GRACE	**60.08**	**37.44**	**59.19**	11.10	276.61	**64.49**	**56.27**	**65.82**	92.65	1763.06

From the tables, we observe that the attribute-based K-Means algorithm always outperforms the structure-based N-cut algorithm. This implies that node attributes are more informative than hyper-graph structure information for clustering on both the Co-citation and Co-authorship networks. Especially on the Pubmed Co-citation and DBLP Co-authorship datasets, N-cut produces very poor clustering results. Moreover, due to the dense resulting edges, and the time complexity of building the traditional hypergraph laplacian, N-cut can be very time and memory consuming on large hyper-graphs, e.g., Pubmed Co-citation and DBLP Co-authorship.

As the hyper-graph structure contains very little information that helps for clustering (as shown by the poor N-cut results), the GNMF and JNMF algorithms that simply combine the node attribute and hyper-graph structure information in their objective functions perform even worse than the attribute-based methods. For example, the performance of GNMF and JNMF are 10.3% and 17.1% lower than K-Means in terms of NMI on the Pubmed Co-citation dataset. Moreover, both GNMF and JNMF algorithms like N-cut also become time and memory consuming for large hyper-graphs, e.g., GNMF is about 50 times slower and takes about 30 times more memory than K-Means on the DBLP Co-authorship and the Pubmed Co-citation datasets.

GRACE consistently outperforms all the baselines on all hyper-graph datasets. Even for the Pubmed Co-citation network, it still well utilizes the inferior hyper-graph structure information to improve clustering performance of the attribute based clustering. GRACE outperforms the GNMF and JNMF algorithms by a large margin, e.g., 8.4% – 17.1% in terms of NMI, and is time and memory efficient even for large hyper-graphs. On the DBLP Co-authorship datasets, GRACE achieves the best performance, while using only 93 seconds and 1763 MB of memory, which are 39 times and 27 times less than JNMF.

6.4.4 Performance on multi-relational graphs. We finally compared the clustering performance on the three multi-relational graphs (their results are reported in Table 10). Due to the space limitation, we only present the running time on the largest Hete DBLP dataset.

Table 10. Clustering performance on multi-relational datasets.

Baselines	ACM			IMDB			Hete DBLP			
	Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%	Time(Sec)
K-Means	67.93	32.03	68.10	52.45	14.63	53.14	36.80	08.47	28.46	**02.41**
N-cut	34.35	07.00	23.68	34.84	00.07	32.54	92.31	76.54	91.81	03.42
SpectralMix	**90.12**	**67.99**	**90.70**	55.46	**20.11**	56.98	40.55	16.05	32.27	22572.54
GRACE	88.89	65.08	89.08	**62.87**	18.44	**62.95**	**92.33**	**76.68**	**91.83**	74.63

The table shows that the K-Means algorithm with only node attribute information produces inferior clustering results. The structure-based method, N-cut, performs poorly on ACM and IMDB but performs exceptionally well on Hete DBLP. This is because the three-type relations of Hete DBLP provide more information for clustering compared with the two-type relations of the ACM and IMDB datasets. The clustering accuracy of using N-cut on the adjacency matrix of each individual relation of Hete DBLP is, 30% for the APA relation, 88.4% for the APCPA relation and 52.7% for the APTPA relation, while that for the combined relations is 92.31% as shown in Table 10. In comparison, the clustering accuracy of using N-cut on each individual relation of ACM and IMDB is: (ACM) 34.8% for PAP and 35.1% for PSP; and (IMDB) 37.5% for MDM and 37.7% for MAM. The results show that the relations of Hete DBLP capture more information for clustering than those of ACM and IMDB do. The results also indicate that each relation of Hete DBLP captures different information and combining them gives better clustering results.

On ACM, the state-of-the-art method, SpectralMix, outperforms GRACE in terms of clustering quality. However, GRACE still achieves competitive clustering quality, and we remark that GRACE is orders of magnitude faster than SpralMix, which took 13,558 seconds compared to the 10 seconds of GRACE for clustering ACM. However, on IMDB and Hete DBLP, GRACE outperforms SpectralMix (with a significant margin on Hete DBLP). The performance difference on Hete DBLP is because SpectralMix takes into account the multi-relational graph structure and node attributes alternatively in its update rule, and as such, the contribution from the informative graph structure of Hete DBLP is weakened. By using multi-relational graph convolution, GRACE can achieve comparable performance on ACM and IMDB, and produce much better performance on Hete DBLP (51.8% improvement in terms of Acc) compared with SpectralMix. Furthermore, GRACE is about 300 times faster than SpectralMix on the large Hete DBLP dataset. Note that the running time of SpectralMix on ACM and IMDB are 13,557.88 and 51.89 seconds, respectively, while those of GRACE are 9.60 and 2.88 seconds, respectively.

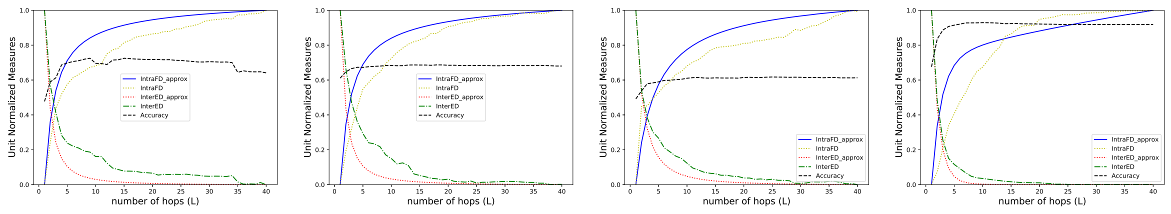
6.4.5 Conclusions of performance comparison. In conclusion, the clustering results on various datasets show that GRACE generally outperforms other algorithms on all types of attributed graphs, while remaining competitively time and memory efficient compared to methods that do not combine node attribute and graph structure. By using graph convolution, GRACE can significantly outperform the methods that focus on either node attribute or graph structure, and perform much faster and takes much less memory compared to other competitors that combine the information in other manners. The results demonstrate that our graph convolution based framework works well on various graph structures. This in fact shows the capability and universality of GRACE in capturing complex and diverse real-world relationships.

6.5 Sensitivity Study

Now we discuss the sensitivity of GRACE's hyper-parameters on different graph types. We report the trends of different measures (i.e., Acc, IntraFD, IntraFD_approx, InterED, and InterED_approx) when only varying the length of hops or the graph filtering rate using four representative datasets: Undirected Cora, Directed Citeseer, Citeseer Co-citation, and Hete DBLP. All measures (except Acc) are unit normalized to fit in the range [0, 1] for a fair comparison. Moreover, same Laplacians, normalizations and β parameters are used for the datasets as described in Section 6.3 unless mentioned otherwise.

6.5.1 Sensitivity study on the length of hops. We first study the impact of the length of hops while fixing the graph filtering rate (α) as 0.5 on all datasets, i.e., Undirected Cora, Directed Citeseer, Citeseer Co-citation, and Hete DBLP. The results are given in Figure 4.

From the above figures, one can observe that InterED_approx and InterED decrease exponentially and gradually stabilize at 0, while IntraFD_approx and IntraFD increase rapidly and gradually stabilize at 1 with respect to the increase in the length of hops. This shows that the two approximate metrics are capable of measuring the clustering quality in terms of structure and attribute. Moreover, their trends validate the claim of Theorem 4.1 that graph convolution based GRACE helps the clustering with lower InterED but harms with higher IntraFD. This explains that the clustering performance in terms of Acc first increases and then drops after the peak, e.g., the decreasing Acc after L=10 on Undirected Cora (Figure 4(a)) and Hete DBLP (Figure 4(d)). Therefore, it is

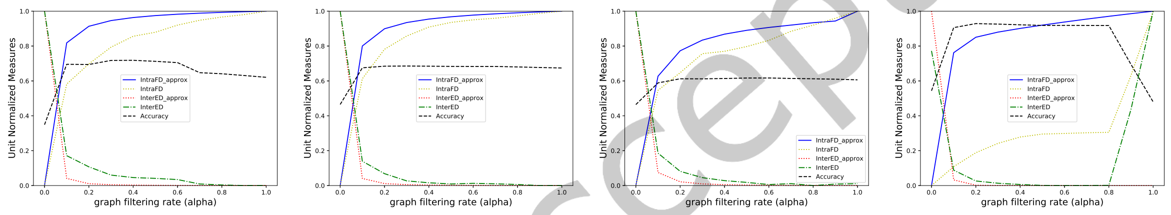


(a) Undirected Cora (b) Directed Citeseer (c) Citeseer Co-citation (d) Hete DBLP

Fig. 4. Effects of the length of hops.

important to determine a proper length of hops to achieve the best performance. As such, we will later check the effectiveness of the stopping algorithm used in GRACE in Section 6.6.

6.5.2 Sensitivity study on the graph filtering rate. We then investigate the impact of the graph filtering rate while fixing the length of hops as 25. The results are given in Figure 5.



(a) Undirected Cora (b) Directed Citeseer (c) Citeseer Co-citation (d) Hete DBLP

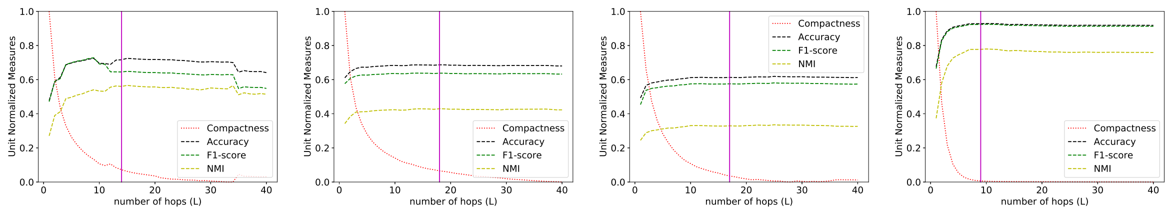
Fig. 5. Effects of the graph filtering rate.

The figures on different graph types show that InterED_approx and InterED decrease rapidly and gradually stabilize at 0 (except on Hete DBLP), while IntraFD_approx and IntraFD start growing from 0 and finally stabilize as the graph filtering rate increases. As expected, the clustering performance in terms of Acc increases and then decreases. The trends fit the analysis in Theorem 4.1 that graph convolution filters the attributive noise guided by graph structure. The higher graph filtering rate improves the clustering quality in terms of structure but harms in terms of attribute. On Hete DBLP, InterED starts growing and being discrepant from InterED_approx when the graph filtering rate becomes too large, which is caused by over-filtering or over-smoothing the node attributes. Therefore, we set a relatively small graph filtering rate and balance between the IntraFD and InterED measures by adjusting the length of hops.

6.6 Ablation Study

To measure the impact of the stopping algorithm and graph Laplacian selection, we conduct ablation studies on each component following the same experimental settings in Section 6.5.

6.6.1 Ablation study on the stopping algorithm. To measure the impact of the stopping algorithm, we let graph convolution run for 40 iterations, and report the trends of Acc, NMI, F1, and Compactness. We mark the iteration (with a pink line) when the stopping criterion is fulfilled, i.e., the compactness of clusters changes very little. The results are reported in Figure 6.

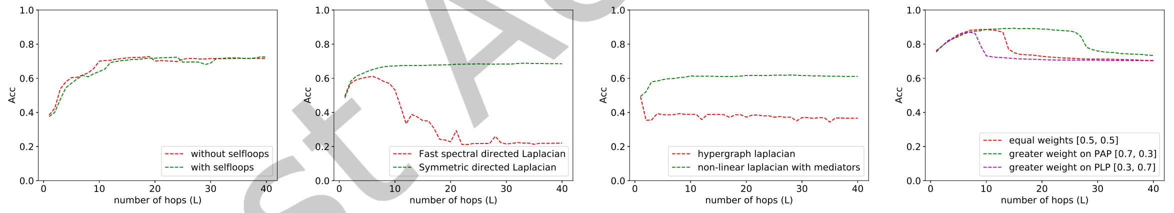


(a) Undirected Cora (b) Directed Citeseer (c) Citeseer Co-citation (d) Hete DBLP

Fig. 6. Effects of the stopping algorithm.

The figures show that the compactness of the clusters at each iteration indeed decreases exponentially. It can also be observed that as the compactness is deceasing, Acc, NMI, and F1 are all increasing and then reach the peak as the compactness stabilizes before they drop. Therefore, choosing the stopping criterion at the point when the compactness stabilizes enables us to pick a good balance point for the number of hops or iterations. The figures also validate the effectiveness of our stopping algorithm as the pink lines in the plots (corresponding to the number of hops chosen by our stopping algorithm) are always at the near-best Acc values. In some cases, as the graph filtering rate is set quite small, the performance will maintain after the pink lines, in which the early stopping algorithm helps save the running time by returning good clusters earlier.

6.6.2 Ablation study on graph Laplacian variants. We constructed variants of graph Laplacians to validate the effectiveness of GRACE's graph Laplacians choices on different graph types. The results are given in Figure 7, where the green curves represent the Acc results of GRACE, and the red curves (and magenta curve in Figure 7(d)) for the graph Laplacian variants. We used the same alpha values for each dataset as in Section 6.3.



(a) Undirected Cora (b) Directed Citeseer (c) Citeseer Co-citation (d) ACM

Fig. 7. Comparison of different graph Laplacians on different graph types.

On Undirected Cora, we compared the performance of using the symmetric graph Laplacian L_{sym} with self-loops (for GRACE) and without self-loops. One can observe from Figure 7(a) that both graph Laplacians produce comparable clustering results to each other. Thus, users may decide to use either. On Directed Citeseer, we compared the symmetric directed graph Laplacian of GRACE L_{sd} to the fast spactral normalized directed Laplacian L_{fsd} (with self-loops for both). Figure 7(b) shows that GRACE's symmetric directed graph Laplacian always outperforms the other and their performance gap increases drastically as the number of hops increases. On Citeseer Co-citation, we constructed the linear hyper-graph Laplacian L_{LH} as a competitor to GRACE's non-linear hyper-graph Laplacian L_{NLH}. It is illustrated in Figure 7(c) that the non-linear hyper-graph Laplacian outperforms the linear one in terms of Acc, and GRACE with sparser non-linear hyper-graph Laplacian is time and memory efficient as shown in Tables 9 and 8. Finally, on ACM, the multi-relational graph Laplacian L_{MR}, GRACE used different coefficients for two relation types ($\beta_{\text{PAP}} = 0.7$ and $\beta_{\text{PSP}} = 0.3$), and we constructed two variants, one that gives a greater weight to PSP ($\beta_{\text{PAP}} = 0.3$ and $\beta_{\text{PSP}} = 0.7$) and one that gives the same weights ($\beta_{\text{PAP}} = 0.5$ and $\beta_{\text{PSP}} = 0.5$). The results from Figure 7(d) show that the performance using the coefficients of GRACE, i.e., giving greater weight to PAP is better than the others, while the performance giving greater weight to PSP is worst (even worse than using same weights). This illustrates the fact that, the multi-relational edges may not have the same importance for clustering (e.g., in this case PAP is more important than PSP).

6.7 Case study

We notice that the same dataset used in our experiments can be constructed as different types of attributed graphs. For example, the Cora citation network can be modelled as an undirected graph (namely Undirected Cora) or hyper-graphs (namely Cora Co-citation and Cora Co-authorship). Prior works usually focus on a specific type and improve its clustering performance. As GRACE can generalize to different graph types, it offers an opportunity to analyze the effects of attributed graph modelling (storing relational information using one graph type over another) under the same framework. From previous results in Section 6.4, one can find that constructing an undirected graph on Cora can produce better clusters than building the Co-citation or Co-authorship hyper-graphs for GRACE (achieving 16.1% and 11.9% improvements respectively in terms of Acc). It infers that Co-citation and Co-authorship hyper-relations are less useful for clustering citation networks as one paper (Co-citation hyper-edge) can include references from different domains and one author (Co-authorship hyper-edge) can publish papers on different topics. This indeed shows that attributed graph modelling itself could be a relevant problem worth further investigation.

7 RELATED WORK

For AGC, there are two main approaches; (1) methods that do not combine graph structure and node attributes, and (2) methods that do.

7.1 Methods that do not combine graph structure and node attributes

Initially designed for undirected graphs, K-Means [45], Kernel K-Means [14] Symmetric-Non-Negative Matrix Factorization (Symm-NMF) [30], and Graph-regularised-Non-Negative Matrix Factorization (GNMF) [6] all focus on the node attributes only and seek to group nodes with similar node attributes in the same cluster by minimizing IntraFD. Two key mentions here are (a) Symm-NMF can focus either on the node attributes or on the graph structure only, but not both; and (b) we classify GNMF in this category because the Laplacian it uses is built from the node attributes rather than from the graph structure. All these methods can be used on all types of graphs since they do not care about the the graph structure. A classical method that only focuses on graph structure for undirected graphs is the well known Normalized-cut (N-cut) [42, 74], which seeks to group nodes in the same cluster by minimizing InterED. N-cut has been extended to directed graphs [46, 55][5], heterogeneous graphs [39, 56] and hypergraphs [85].

In this category, specific to undirected graphs only, we also have node representation learning methods such as Deepwalk [50] and Node2vec [20], which learn node embeddings from the graph structure and feed these embeddings into the traditional clustering algorithms (such as K-Means) to learn the clusters. Specific to hypergraphs, we have methods that only focus on the hypergraph structure such as the PageRank based algorithm by [63] (which extended the work of [3] to hypergraphs), Inhomogeneous hypergraph partitioning method [37] (which assigns different costs to different hyperedge cuts), and Efficient Hypergraph Clustering [33] (a method which updates the cluster membership of all points in parallel). Finally, specific to heterogeneous networks in

[5]Note that all undirected graph methods can easily be extended to directed graphs by making the directed graph structure undirected [55].

this category we have RankClus [60] which focuses on bi-typed heterogeneous networks by simultaneously addressing ranking and clustering on heterogenous graphs, NetClus [62] which extends RanClus to heterogenous graphs with more than two types of nodes, SI-Cluster [91] which introduces a vertex based similarity measure and proposes an algorithm to iteratively refine the clusters based on this measure, and NMF based methods [51].

There are two main drawbacks to these methods. First, the two sources of information, i.e., the node attributes and the graph structure, do not always contain exactly the same information, but rather are complementary to each other (as seen in Section 6, the results of K-Means are always different to those of N-cut). As such, focusing on one source of information only (e.g., graph structure) will neglect the information contained in the other (e.g., the node attributes). Moreover, once clusters are obtained independently from the different sources of information, it is not clear how to combine them both, nor which one to choose.

7.2 Methods that combine graph structure and node attributes

AGC methods that combine graph structure and node attributes can be categorized into four approaches: *hybrid distance-based, probabilistic model-based, nonnegative matrix factorization (NMF)-based, and node embedding-based* approaches.

Hybrid distance-based methods design a unified distance that combines node attributes and graph structure information. For example, the linear combination of attributive distance (e.g., euclidean distance) and structural distance (e.g., the shortest path). Based on the hybrid distance, there are two ways to perform clustering: (1) we can directly apply distance-based clustering algorithms (e.g., K-Medoids or K-Means [44]); (2) or we can utilize the hybrid distance information to construct a new graph (e.g., a K-NN graph [12] or edge-weighted graph [48]), and then apply graph clustering algorithms. This category cannot easily be extended to other types of graphs because the explicit structural distance measures are usually designed for a specific graph type, and to extend it to another graph type one will need to redefine the structural distance measure. For example, on undirected graphs, in SA-Cluster [89] and its extended versions [10, 90], the unified neighborhood random walk distance is computed on a node-augmented graph with new attributive nodes and edges, and the K-Medoids algorithm is then applied to find clusters. This new augmented graph is built on the assumption that the original graph structure was an undirected graph and it is not obvious how to build it on heterogeneous graphs or hypergraphs. Another example is the distance based state-of-the-art SpectralMix [53] on heterogeneous graphs. It can easily be reduced to work on undirected graphs, by setting the number of multi-relational graph adjacency matrices to 1 and using the undirected graph adjacency as the multi-relational graph adjacency matrix, but it is not clear how to extend it to hypergraphs, since in hypergraphs each edge can contain much more than a single node, but the structural distance used in SpectralMix' joint objective only accommodates simple edges linking two nodes. On heterogeneous graphs, MvAGC [40] denoises the graph signals (node attributes) and then proposes a graph learning algorithm to learn node similarities and uses the resulting similarity graph for clustering. The main difference between MvAGC and GRACE is that in MvAGC, the convolution (denoising) module is not directly connected to the graph learning and clustering modules. Thus, although MvAGC presents a novel approach to clustering, i.e., denoising and learning similarities and then clustering, their work is not directed towards understanding the effects of graph convolution for clustering. One of the limitations of MvAGC is that the graph filtering phase, which is shown to be essential for clustering in our work, is only treated as a prepossessing phase, and hence it is difficult to see why the optimal k as the power for their filter is 3 (i.e., $k = 3$ for the filter $(1 - 0.5L)^k$) as they observed. In addition, it is also non-trivial to generalize MvAGC to hyper-graphs because even if we use the de-noising (convolution) step of GRACE to prepossess the hyper-node embedding, it is still non-trivial to adapt the graph learning phase to hyper-graphs. Moreover, their multi-view learning approach differs from that of GRACE in that GRACE uses a combined laplacian for filtering, while they propose to filter individually and then use a weighted sum with learned weights to learn the similarity graph.

Probabilistic model-based methods consider both node attributes and graph structure as probabilistic variables, and model the community memberships of each node as latent variables. From this view, attributed graph clustering can be transformed into a community membership estimation problem. There are three ways to model the statistical relationships among node attributes (X), graph structure (G) and community memberships (F): (1) node attributes generate community memberships and then community memberships generate graph structure (i.e., $X \rightarrow F \rightarrow G$), such as [83]; (2) node attributes and community memberships together generate graph structure (i.e., $F \rightarrow G \leftarrow X$), such as [47]; (3) community memberships generate both node attributes and graph structure (i.e., $X \leftarrow F \rightarrow G$), such as [78, 79, 82]. The generative process in the model, though having intuitive explanations, requires specific decisions about statistical interactions and underlying distributions, where wrong decisions lead to highly biased models. Like hybrid distance-based methods, probabilistic methods are equally hard to extend to other types of graphs. For example, on undirected, directed and hypergraphs, we have the Popularity Conditional Link and Direct Content Model (PCL-DC) [83] which cannot be easily extended to heterogeneous graphs since it uses the content probability to estimate the probability of a node being connected to another without taking into account the type of link (i.e., it is designed to work only when there is one type of edge relations).

Nonnegative matrix-based methods introduce a nonnegative assignment matrix that associates each node with the strength of its belonging to each community. In contrast to probabilistic model-based methods, the nonnegative assignment matrix is not regarded as a random variable but can be obtained by nonnegative matrix factorization (NMF) [32]. NMF aims to find two nonnegative low-rank matrices whose product provides a good approximation to the original one, and their nonnegativity offers the interpretability for document co-clustering [76]. For undirected graphs, one can extend GNMF [7] to combine both node attributes and graph structure, by modifying the Laplacian regularizer it uses to the Laplacian built from the graph structure (rather than that built from the manifold graph constructed from the node attributes as in their original paper). Following this idea, different objectives that fuse both node attributes and graph structure information have been proposed on undirected graphs [21, 25, 69], and recently efforts have been made to extend to hypergraphs [17, 71]. In spite of these efforts to extend to other types of graphs, we remain unaware of any attempt on heterogeneous graphs.

Node embedding-based methods aim to learn a vector representation of each node in an attributed graph, and then the traditional clustering methods, such as K-Means, are applied to find clusters based on the learned node embeddings. Techniques to generate node embeddings include dimensionality reduction [5], random walk [20, 50] and neural networks [68]. However, these early works of node embedding only consider the graph structure. To learn node embeddings that incorporate node attributes information, a common practice is to simply concatenate the learned structural embeddings and node attributes. In addition, random walk-based methods are extended to attributed graphs such as [2, 81]. In recent years, graph neural networks [73, 87] have been demonstrated to effectively capture both node attributes and graph structure information into node embeddings. Though on undirected graphs various such methods have been proposed and used for clustering [13, 24, 28, 29, 49, 59, 65–67, 72, 75, 84], particularly related to our work and model GRACE are the graph convolution based based models [28, 29, 84] which have been shown to achieve the state-of-the-art-results for Undi-AGC[6]. Recently, efforts have also been made to extend graph neural networks to other types of graphs such as directed graphs [35, 43], hypergraphs [4, 19, 80], and more recently heteorgeneous graphs [18, 22, 23, 36, 41, 57, 70]. Amongst such extensions to heterogeneous graphs, we have GraphRec [18], KCGN [23], and FesoG [41]. GraphRec [18] is an attention network proposed for social recommendation. Given a user-user graph and a user-item graph, GraphRec learns user embedding (as a combination of the user embedding from the user-user graph and the user embedding from the user-item graph), as well as item embedding, and finally combine the learned user embedding and item embedding to obtain a user rating prediction for an item. Although GraphRec focuses on

[6]One can find the state-of-the-art models for Undi-AGC via the following link: https://paperswithcode.com/task/graph-clustering

social recommendation, one could use the combined user embedding for user clustering or the item embedding for item clustering. Compared with GRACE, it is non-trivial to generalize GraphRec to graph-types such as hypergraphs or multi-relational graphs, because in this case one would need to completely redesign the message passing scheme of GraphRec to adapt to multiple nodes in a single edge or message passing across different adjacency matrices. Moreover, it is also difficult to analyze what the effects of GraphRec on clustering may be (e.g., why and how it achieves a certain effect on the intra-cluster feature distance or inter-cluster edge density), even when it is paired with a suitable clustering loss function. This is because it is not clear how we may incorporate GraphRec's message passing scheme in the analysis even if we follow a similar approach as we do in our paper. FeSoG [41] proposes to employ federated learning to address the privacy concerns in social recommendation based on the fact that most graph neural networks for this task require a centralized storage of the social links and item interactions of users. The attention based graph neural network proposed in FeSoG is similar to GraphRec (discussed above), since FesoG is an attention network with a similar procedure to learn the embedding (besides the federated learning it employs). Therefore, FeSoG shares the same limitations as GraphRec compared with GRACE. KCGN [23] addresses issues that most social recommendation neural networks do not handle, i.e., (1) the interaction between items, (2) the presence of multi-typed user item interactions, and (3) the dynamic nature of user-item interactions. To address these issues, KCGN designs a new message passing framework and a new loss function that incorporate (a) multi-typed dynamic user-item interaction encoding, and (b) knowledge-aware user-user and item-item inter-dependent relations, where (a) captures the dynamic relationship between user and items and (b) captures the local relationship among users and the local relationship among items. KGCN uses a message passing framework to fuse user-item interactions and a graph leaning framework to fuse user-user interactions and item-item interactions in a coupled way. In contrast, GRACE uses graph convolution to fuse both user-user interactions and user-item (user-attributes in our case) interactions and then capture the interdependence in both by building the similarity matrix for clustering. An advantage of GRACE has over KCGN (and the previous discussed models as well) for the clustering task is that, as discussed in Sections 4 and 6, we show the direct effects of graph convolution on the fundamental measures of clustering and can thus use graph convolution to directly guide the clustering, in addition to the fact that no training is needed. Moreover, GRACE is easily generalized to different graph types so long as we have a well defined laplacian and a suitable low-pass filter, which is not the case for KCGN as it is hard to extend KCGN to another graph type such as hypergraphs. However, KCGN incorporates the dynamic nature of user-item interactions in their framework, which is an interesting direction to consider for GRACE as its future work.

Some major flaws of these models are: (1) some of these models as noted are proposed specifically for specific types of graphs and cannot be easily extended to other types, (2) Some of the state-of-the-art methods on each type of graphs (e.g., SpectralMix on heterogeneous graphs) are extremely time costly to run, (3) no study exists on how such a general method affects clustering on the different types of graphs and no performance nor analysis guarantees have been provided. As a result of these drawbacks, the benefits of different types of graph structures (undirected, directed, hyper, heterogeneous, etc.) used to store information have never been compared with each other. Thus, our aim in this work has been to address these challenges by (1) proposing an easy and general convolution framework for clustering on all types of graphs[7], (2) proposing a convolution model GRACE based on the framework which is both fast and memory efficient, and (3) providing theoretical and experimental analysis and guaranties of our model and framework in general on all types of graphs.

7.3 Differences between GRACE and GRAC

A preliminary work of this paper is GRAC [27], as GRACE can be considered as a generalization of GRAC. There are however key differences between GRACE and GRAC.

[7]An additional benefit of our framework is that it can be utilized by any convolution-based algorithms or NMF-based algorithms.

First, GRACE is a framework for handling AGC on different types of attributed graphs with different types of edges, while GRAC was tailored specifically for hypergraphs. To achieve this, we use the convolution proposed by GRAC and propose GRACE as a general framework. We make use different filters for GRACE for each different type of graphs in order to address the different challenges imposed by different types of relations (edges) on the different types of graphs.

Second, We provide theoretical guarantees for the effects of GRACE on AGC by following GRAC in analyzing the effects of the graph convolution on AGC. We modify the analysis from GRAC. Specifically, we modify the effects of the convolution on IntraFD. This can be seen by the fact that Equation (11) in this work is different from Equation (9) in [27]. This leads to the more precise conclusion that when α is sufficiently small such that $\alpha\lambda_i \leq 1$ for all i, then IntraFD is expected to increase exponentially as L increases. Otherwise, it may decrease exponentially (or increase more rapidly) with L depending on the products $(1-\alpha\lambda_i)^L(1-\alpha\lambda_j)^L$ in Equation (11), since IntraFD is assumed to follow the same trend with IntraFD_approx (as also shown in the experiments). We also provide detailed time complexity analysis for GRACE on each type of graphs used, and modify the complexity provided in [27] with sparse matrix operations.

Third, we conducted extensive experiments on 14 different datasets to show that the analysis on the effects of graph convolution holds on the four different types of graphs we used. The 14 datasets used include each of these four different types of graphs. We compared GRACE with the state-of-the-art methods to validate its superior performance. We also showed the effects of the different parameters of GRACE (as well as the filters we chose for GRACE) on the performance of AGC for each type of graphs.

Lastly, due to the different challenges that arise with different types of graph structure, we conducted an extensive survey of prior works for clustering the four types of graphs we studied. Following [27], we categorize these works into two categories; (1) methods that focus only on node attributes or graph structure, and (2) methods that combine the information from both graph structure and node attributes. We highlight the limitations of the methods in the first category, as well as discuss that the methods in the second category are not easy to generalize to clustering different types of graphs.

8 CONCLUSION

We studied the AGC problem with many different attributed graphs. Unlike existing AGC algorithms on only specific graph types, we provided a general framework for AGC on various graph types, including undirected, directed, hyper-, and multi-relational attributed graphs. We proposed effective and efficient graph convolution with well-designed graph Laplacians for each type of graph structures. By performing graph convolution to remove the attribute noises, the processed node attributes integrate the graph structure information and can be directly used for clustering. We conducted extensive experiments on 14 real-world datasets of different graph types. The results show that our algorithm outperforms the existing AGC methods while remaining fast and scalable.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. This work was supported by GRF 14208318 from the RGC of HKSAR and CUHK direct grant 4055146. Lin Zhang and Bo Li were supported in part by RGC RIF grant R6021-20 and RGC GRF grant under contracts 16209120 and 16200221. Bo Han was supported by the RGC Early Career Scheme No. 22200720 and NSFC Young Scientists Fund No. 62006202.

REFERENCES

[1] Sameer Agarwal, Kristin Branson, and Serge J. Belongie. 2006. Higher order learning with graphs. *Proceedings of the 23rd international conference on Machine learning* (2006).

[2] Esra Akbas and Peixiang Zhao. 2017. Attributed Graph Clustering: an Attribute-aware Graph Embedding Approach. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017*, Jana Diesner, Elena Ferrari, and Guandong Xu (Eds.). ACM, 305–308. https://doi.org/10.1145/3110025.3110092

[3] Reid Andersen, Fan Chung, and Kevin Lang. 2007. Using PageRank to Locally Partition a Graph. *Internet Mathematics* 4 (01 2007), 35–64. https://doi.org/10.1080/15427951.2007.10129139

[4] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2019. Hypergraph Convolution and Hypergraph Attention. *CoRR* abs/1901.08150 (2019). arXiv:1901.08150 http://arxiv.org/abs/1901.08150

[5] Mikhail Belkin and Partha Niyogi. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.* 15, 6 (2003), 1373–1396. https://doi.org/10.1162/089976603321780317

[6] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. 2010. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence* 33, 8 (2010), 1548–1560.

[7] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S. Huang. 2011. Graph Regularized Nonnegative Matrix Factorization for Data Representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 8 (2011), 1548–1560. https://doi.org/10.1109/TPAMI.2010.231

[8] T.-H. Hubert Chan and Zhibin Liang. 2020. Generalizing the hypergraph Laplacian via a diffusion process with mediators. *Theor. Comput. Sci.* 806 (2020), 416–428. https://doi.org/10.1016/j.tcs.2019.07.024

[9] Siheng Chen, Aliaksei Sandryhaila, José M. F. Moura, and Jelena Kovacevic. 2014. Signal denoising on graphs via graph filtering. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 872–876. https://doi.org/10.1109/GlobalSIP.2014.7032244

[10] Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. 2011. Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities. *ACM Trans. Knowl. Discov. Data* 5, 2 (2011), 12:1–12:33. https://doi.org/10.1145/1921632.1921638

[11] Fan Chung. 2005. Laplacians and the Cheeger Inequality for Directed Graphs. *Annals of Combinatorics* 9 (04 2005), 1–19. https://doi.org/10.1007/s00026-005-0237-z

[12] TA Dang and Emmanuel Viennet. 2012. Community detection based on structural and attribute similarities. In *International conference on digital society (icds)*. 7–12.

[13] Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. 2005. Efficient Non-Parametric Function Induction in Semi-Supervised Learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics* (proceedings of the tenth international workshop on artificial intelligence and statistics ed.). Society for Artificial Intelligence and Statistics, 96–103.

[14] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 551–556.

[15] Chris Ding and Xiaofeng He. 2004. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*. 29.

[16] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 606–610.

[17] Rundong Du, Barry L. Drake, and Haesun Park. 2017. Hybrid Clustering based on Content and Connection Structure using Joint Nonnegative Matrix Factorization. *CoRR* abs/1703.09646 (2017). arXiv:1703.09646 http://arxiv.org/abs/1703.09646

[18] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. 2022. A Graph Neural Network Framework for Social Recommendations. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2022), 2033–2047. https://doi.org/10.1109/TKDE.2020.3008732

[19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2018. Hypergraph Neural Networks. *AAAI 2019* (2018).

[20] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[21] Ting Guo, Shirui Pan, Xingquan Zhu, and Chengqi Zhang. 2018. CFOND: consensus factorization for co-clustering networked data. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (2018), 706–719.

[22] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. *CoRR* abs/2003.01332 (2020). arXiv:2003.01332 https://arxiv.org/abs/2003.01332

[23] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware Coupled Graph Neural Network for Social Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 4115–4122. https://ojs.aaai.org/index.php/AAAI/article/view/16533

[24] Po-Yao Huang, Robert Frederking, et al. 2019. RWR-GAE: Random Walk Regularization for Graph Auto Encoders. *arXiv preprint arXiv:1908.04003* (2019).

[25] Zhichao Huang, Yunming Ye, Xutao Li, Feng Liu, and Huajie Chen. 2017. Joint Weighted Nonnegative Matrix Factorization for Mining Attributed Graphs. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10234)*, Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon (Eds.). 368–380. https://doi.org/10.1007/978-3-319-57454-7_29

[26] Jianbo Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905. https://doi.org/10.1109/34.868688

[27] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, and Bo Han. 2021. HyperGraph Convolution Based Attributed HyperGraph Clustering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

[28] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[30] Da Kuang, Chris Ding, and Haesun Park. 2012. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*. SIAM, 106–117.

[31] Tarun Kumar, Sankaran Vaidyanathan, Harini Ananthapadmanabhan, Srinivasan Parthasarathy, and Balaraman Ravindran. 2018. Hypergraph Clustering: A Modularity Maximization Approach. *CoRR* abs/1812.10869 (2018). arXiv:1812.10869 http://arxiv.org/abs/1812.10869

[32] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, Todd K. Leen, Thomas G. Dietterich, and Volker Tresp (Eds.). MIT Press, 556–562. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization

[33] Marius Leordeanu and Cristian Sminchisescu. 2012. Efficient Hypergraph Clustering. *AISTATS* (01 2012).

[34] Jure Leskovec and Julian Mcauley. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc., 539–547. https://proceedings.neurips.cc/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf

[35] Chensheng Li, Xiaowei Qin, Xiaodong Xu, Dujia Yang, and Guo Wei. 2020. Scalable Graph Convolutional Networks With Fast Localized Spectral Filter for Directed Graphs. *IEEE Access* 8 (2020), 105634–105644. https://doi.org/10.1109/ACCESS.2020.2999520

[36] Jianxin Li, Hao Peng, Yuwei Cao, Yingtong Dou, Hekai Zhang, Philip S. Yu, and Lifang He. 2021. Higher-Order Attribute-Enhancing Heterogeneous Graph Neural Networks. *CoRR* abs/2104.07892 (2021). arXiv:2104.07892 https://arxiv.org/abs/2104.07892

[37] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous Hypergraph Clustering with Applications. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 2308–2318. https://proceedings.neurips.cc/paper/2017/file/a50abba8132a77191791390c3eb19fe7-Paper.pdf

[38] Q. Li, Z. Han, and X.-M. Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI.

[39] Xiang Li, Ben Kao, Zhaochun Ren, and Dawei Yin. 2019. Spectral Clustering in Heterogeneous Information Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 4221–4228. https://doi.org/10.1609/aaai.v33i01.33014221

[40] Zhiping Lin and Zhao Kang. 2021. Graph Filter-based Multi-view Attributed Graph Clustering. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2723–2729. https://doi.org/10.24963/ijcai.2021/375 Main Track.

[41] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S. Yu. 2021. Federated Social Recommendation with Graph Neural Network. *ACM Trans. Intell. Syst. Technol.* (nov 2021). https://doi.org/10.1145/3501815 Just Accepted.

[42] Ulrike Luxburg. 2004. A Tutorial on Spectral Clustering. *Statistics and Computing* 17 (01 2004), 395–416. https://doi.org/10.1007/s11222-007-9033-z

[43] Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. 2019. Spectral-based Graph Convolutional Network for Directed Graphs. *CoRR* abs/1907.08990 (2019). arXiv:1907.08990 http://arxiv.org/abs/1907.08990

[44] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[45] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[46] Fragkiskos D. Malliaros and Michalis Vazirgiannis. 2013. Clustering and Community Detection in Directed Networks: A Survey. *CoRR* abs/1308.0971 (2013). arXiv:1308.0971 http://arxiv.org/abs/1308.0971

[47] Julian J. McAuley and Jure Leskovec. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 548–556. http://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks

[48] Jennifer Neville, Micah Adler, and David Jensen. 2003. Clustering relational data using attribute and link information. In *In Proceedings of the Text Mining and Link Analysis Workshop, 18th International Joint Conference on Artificial Intelligence*. 9–15.

[49] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding.. In *IJCAI*. 2609–2615.

[50] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[51] Yaoyao Qin, Caiyan Jia, and Yafang Li. 2016. Community detection using nonnegative matrix factorization with orthogonal constraint. In *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*. 49–54. https://doi.org/10.1109/ICACI.2016.7449802

[52] J.A. Rodriguez. 2002. On the Laplacian Eigenvalues and Metric Parameters of Hypergraphs. *Linear and Multilinear Algebra* 50, 1 (2002), 1–14. https://doi.org/10.1080/03081080290011692 arXiv:https://doi.org/10.1080/03081080290011692

[53] Ylli Sadikaj, Yllka Velaj, Sahar Behzadi, and Claudia Plant. 2021. Spectral Clustering of Attributed Multi-Relational Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1431–1440. https://doi.org/10.1145/3447548.3467381

[54] Shota Saito, Danilo P. Mandic, and Hideyuki Suzuki. 2017. Hypergraph p-Laplacian: A Differential Geometry View. *CoRR* abs/1711.08171 (2017). arXiv:1711.08171 http://arxiv.org/abs/1711.08171

[55] Venu Satuluri and Srinivasan Parthasarathy. 2011. Symmetrizations for Clustering Directed Graphs *(EDBT/ICDT '11)*. Association for Computing Machinery, New York, NY, USA, 343–354. https://doi.org/10.1145/1951365.1951407

[56] Srijan Sengupta and Yuguo Chen. 2015. SPECTRAL CLUSTERING IN HETEROGENEOUS NETWORKS. *Statistica Sinica* 25, 3 (2015), 1081–1106. http://www.jstor.org/stable/24721222

[57] Jinghan Shi, Houye Ji, Chuan Shi, Xiao Wang, Zhiqiang Zhang, and Jun Zhou. 2020. Heterogeneous Graph Neural Network for Recommendation. *CoRR* abs/2009.00799 (2020). arXiv:2009.00799 https://arxiv.org/abs/2009.00799

[58] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98. https://doi.org/10.1109/MSP.2012.2235192

[59] Heli Sun, Fang He, Jianbin Huang, Yizhou Sun, Yang Li, Chenyu Wang, Liang He, Zhongbin Sun, and Xiaolin Jia. 2020. Network Embedding for Community Detection in Attributed Networks. *ACM Trans. Knowl. Discov. Data* 14, 3 (2020), 36:1–36:25. https://doi.org/10.1145/3385415

[60] Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. 2009. RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis *(EDBT '09)*. Association for Computing Machinery, New York, NY, USA, 565–576. https://doi.org/10.1145/1516360.1516426

[61] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based clustering of heterogeneous information networks with star network schema. *Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema*, 797–806. https://doi.org/10.1145/1557019.1557107

[62] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 797–806. https://doi.org/10.1145/1557019.1557107

[63] Yuuki Takai, Atsushi Miyauchi, Masahiro Ikeda, and Yuichi Yoshida. 2020. Hypergraph Clustering Based on PageRank. 1970–1978. https://doi.org/10.1145/3394486.3403248

[64] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. 2020. Digraph Inception Convolutional Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 17907–17918. https://proceedings.neurips.cc/paper/2020/file/cffb6e2288a630c2a787a64ccc67097c-Paper.pdf

[65] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ accepted as poster.

[66] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: a Deep Attentional Embedding approach. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Sarit Kraus (Ed.). Association for the Advancement of Artificial Intelligence (AAAI), United States of America, 3670–3676. https://doi.org/10.24963/ijcai.2019/509

[67] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 889–898.

[68] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 1225–1234. https://doi.org/10.1145/2939672.2939753

[69] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. 2016. Semantic Community Identification in Large Attribute Networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 265–271. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11964

[70] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised Heterogeneous Graph Neural Network with Co-contrastive Learning. *CoRR* abs/2105.09111 (2021). arXiv:2105.09111 https://arxiv.org/abs/2105.09111

[71] Joyce Whang, Rundong Du, Sangwon Jung, Geon Lee, Barry Drake, Qingqing Liu, Seonggoo Kang, and Haesun Park. 2020. MEGA: multi-view semi-supervised clustering of hypergraphs. *Proceedings of the VLDB Endowment* 13 (01 2020), 698–711. https://doi.org/10.14778/3377369.3377378

[72] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 6861–6871.

[73] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019). arXiv:1901.00596 http://arxiv.org/abs/1901.00596

[74] Rongkai Xia, Y. Pan, Lei Du, and J. Yin. 2014. Robust Multi-View Spectral Clustering via Low-Rank and Sparse Decomposition. In *AAAI*.

[75] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[76] Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, Charles L. A. Clarke, Gordon V. Cormack, Jamie Callan, David Hawking, and Alan F. Smeaton (Eds.). ACM, 267–273. https://doi.org/10.1145/860435.860485

[77] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12* (05 2012). https://doi.org/10.1145/2213836.2213894

[78] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 505–516. https://doi.org/10.1145/2213836.2213894

[79] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2014. GBAGC: A General Bayesian Framework for Attributed Graph Clustering. *ACM Trans. Knowl. Discov. Data* 9, 1 (2014), 5:1–5:43. https://doi.org/10.1145/2629616

[80] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS) 32*. Curran Associates, Inc., 1509–1520.

[81] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 2111–2117. http://ijcai.org/Abstract/15/299

[82] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. 2013. Community Detection in Networks with Node Attributes. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, Hui Xiong, George Karypis, Bhavani M. Thuraisingham, Diane J. Cook, and Xindong Wu (Eds.). IEEE Computer Society, 1151–1156. https://doi.org/10.1109/ICDM.2013.167

[83] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009. Combining Link and Content for Community Detection: A Discriminative Approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 927–936. https://doi.org/10.1145/1557019.1557120

[84] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. 2019. Attributed Graph Clustering via Adaptive Graph Convolution. 4327–4333. https://doi.org/10.24963/ijcai.2019/601

[85] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, 1601-1608 (2007)* 19, 1601–1608.

[86] Dengyong Zhou, Jiayuan Huang, and Bernhard Scholkopf. 2005. Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd international conference on Machine learning* (2005).

[87] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *CoRR* abs/1812.08434 (2018). arXiv:1812.08434 http://arxiv.org/abs/1812.08434

[88] Y. Zhou, Hong Cheng, and J. X. Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. *Proc. VLDB Endow.* 2 (2009), 718–729.

[89] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. *Proc. VLDB Endow.* 2, 1 (2009), 718–729. https://doi.org/10.14778/1687627.1687709

[90] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2010. Clustering Large Attributed Graphs: An Efficient Incremental Approach. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu (Eds.). IEEE Computer Society, 689–698. https://doi.org/10.1109/ICDM.2010.41

[91] Yang Zhou and Ling Liu. 2013. Social Influence Based Clustering of Heterogeneous Information Networks *(KDD '13)*. Association for Computing Machinery, New York, NY, USA, 338–346. https://doi.org/10.1145/2487575.2487640