Simultaneous Linear Multi-view Attributed Graph Representation Learning and Clustering

Chakib Fettal
Centre Borelli, Université Paris Cité
and Informatique CDC
chakib.fettal@etu.u-paris.fr

Lazhar Labiod
Centre Borelli UMR 9010
Université Paris Cité
lazhar.labiod@u-paris.fr

Mohamed Nadif
Centre Borelli UMR 9010
Université Paris Cité
mohamed.nadif@u-paris.fr

ABSTRACT

Over the last few years, various multi-view graph clustering methods have shown promising performances. However, we argue that these methods can have limitations. In particular, they are often unnecessarily complex, leading to scalability problems that make them prohibitive for most real-world graph applications. Furthermore, many of them can handle only specific types of multi-view graphs. Another limitation is that the process of learning graph representations is separated from the clustering process, and in some cases these methods do not even learn a graph representation, which severely restricts their flexibility and usefulness. In this paper we propose a simple yet effective linear model that addresses the dual tasks of multi-view attributed graph representation learning and clustering in a unified framework. The model starts by performing a first-order neighborhood smoothing step for the different individual views, then gives each one a weight corresponding to its importance. Finally, an iterative process of simultaneous clustering and representation learning is performed w.r.t. the importance of each view, yielding a consensus embedding and partition of the graph. Our model is generic and can deal with any type of multi-view graph. Finally, we show through extensive experimentation that this simple model consistently achieves competitive performances w.r.t. state-of-the-art multi-view attributed graph clustering models, while at the same time having training times that are shorter, in some cases by orders of magnitude.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; • **Information systems** → **Clustering**.

KEYWORDS

attributed networks, clustering, multi-view

ACM Reference Format:
Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2023. Simultaneous Linear Multi-view Attributed Graph Representation Learning and Clustering. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570367

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570367

1 INTRODUCTION AND RELATED WORK

Attributed graphs are graphs that contain features in their nodes. Under different approaches, they are used to model a wide variety of structured data [6, 7, 18] with applications in recommender systems [4, 19, 35], computer vision, [17, 22, 32], and physical systems [21]. However, capturing topological (or structural) information at the same time as capturing information about node-level features presents challenges, and in the last few years a number of methods have been proposed for tackling problems such as attributed graph representation learning and attributed graph clustering.

In some real-world applications, data is collected from a variety of different sources, which means that it can be characterized using different sets of information or *views*. This is the premise of *multi-view learning* [31], an area of considerable interest among the data mining and machine learning communities [1]. In the context of attributed graphs, a multi-view attributed graph is simply a set of attributed graphs; each attributed graph counts as a single view. For example, in the case of a recommender system, the relationship between users can be characterized using a two graphs, one representing their "friendship", and one representing their mutual interests; as for the features, one set of features could represent personal information and another set their past transactions.

The task of multi-view attributed graph clustering has lately received a lot of attention. The methods that have been proposed can be separated into two broad approaches. In the first approach, a consensus graph partition is learned directly from the data without explicitly learning an embedding of the graph. Methods adopting this approach include MvAGC [11], where a graph filter is proposed to perform the graph clustering, and MAGC [12], a similar method to MvAGC that uses a graph filter to learn a consensus graph before doing the clustering. The second approach is more flexible; it consists in learning a consensus representation or embedding before applying a simple single-view attributed graph clustering method. For example, DMGI [16] is an unsupervised network embedding method for attributed multiplex networks that uses the concept of mutual information, while O2MAC [3] is based on the graph autoencoder [9], it learns clustering-friendly embeddings through integrating a clustering loss in its objective.

These different methods have their shortcomings. First, the more flexible approaches that learn a consensus representation generally tackle the problems of representation learning and clustering separately, i.e. they learn representations that are not specifically tailored to clustering. Second, they often have unnecessarily complex architectures in comparison to simpler strategies. Finally, some of these methods are not generic, in the sense that they require the multi-view graph to be of a certain type: for example, a multi-view graph with a multiple structures and a single set of features (but not

the other way around) like for DMGI, or a graph with exactly two views, etc. As a way of addressing these shortcomings, we propose *LMGEC* (for *Linear Multi-view Graph Embedding and Clustering*), a simple yet effective linear model. LMGEC starts by applying a linear graph filter corresponding to a one-hop neighborhood propagation step in each individual view, and then applies a weighting scheme so that views are attended to in order of their perceived importance. This is followed by an iterative process of simultaneous clustering and representation learning, which gives rise to a consensus embedding and partition of the graph. The model is generic in the sense that it can deal with any number of graph structures and/or any number of feature sets. A high-level schematic representation of the model is shown in figure 1. Our contributions may be summarized as follows:

- We introduce a simple yet effective generic linear model for performing multi-view attributed graph representation learning simultaneously with clustering. The model is based on (1) a one-hop neighborhood propagation corresponding to a linear graph filter, (2) a view weighting scheme reminiscent of the attention mechanism in neural networks, and (3) a graph clustering and representation learning linear component that addresses both tasks via a unified framework.
- We carry out a theoretical study of the linear graph filtering, formulate the problem that we are seeking to solve, and propose an algorithm that we subject to a detailed computational complexity analysis.
- We showcase the efficiency and effectiveness of this model against the state of the art through extensive experimentation. We show that our model is both competitive and several magnitudes more efficient than current state-of-the-art multi-view attributed graph clustering.
- We release our code for reproducibility[1].

2 PRELIMINARIES

2.1 Definitions and Notations

An attributed graph is defined as a quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ where $\mathcal{V}$ represents the vertex set, $\mathcal{E}$ the edge, $\mathbf{X} \in \mathbf{R}^{n \times d}$ its node features matrix and $\mathbf{A}$ its adjacency matrix of size $n \times n$. A multi-view attributed graph is represented as a sequence of attributed graphs $\mathcal{M} := \left\{\mathcal{G}_v = (\mathcal{V}_v, \mathcal{E}_v, \mathbf{A}_v, \mathbf{X}_v)\right\}_{v=1}^{v=V}$. Matrices are denoted by boldface uppercase and vectors by boldface lowercase letters, $\mathbf{1}$ represents a column vector of ones. $\mathbf{I}$ denotes the identity matrix. Where $\mathbf{X}$ is a matrix, $\mathbf{x}_i$ is its i-th row. A matrix referenced as $\mathbf{X}_v$ means that it belongs to the v-th attributed graph, and its i-th row is referenced as $\mathbf{x}_i^v$.

2.2 Graph Filters and the Simple Graph Convolutional Network

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be an attributed graph whose symmetrically normalized Laplacian matrix is $\mathbf{L}^{\text{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the diagonal matrix of degrees of the graph such that $d_{ii} = \sum_j a_{ij}$. A graph signal can be seen as a vector $\mathbf{f} = [f(v_1), \ldots, f(v_n)]$ such that $f : \mathcal{V} \to \mathbf{R}$ is a real-valued function on the vertex set $\mathcal{V}$.

[1] https://github.com/chakib401/LMGEC

For any graph signal $\mathbf{f}$, we can quantify its smoothness using the Laplacian quadratic form [37]

$$\mathcal{S}(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}^{\text{sym}} \mathbf{f} = \frac{1}{2} \sum_{i,j}^{n} a_{ij} \left(\frac{f_i}{\sqrt{d_{ii}}} - \frac{f_j}{\sqrt{d_{jj}}} \right)^2. \tag{1}$$

Now let $\mathbf{L}^{\text{sym}} = \mathbf{U}\Lambda\mathbf{U}^\top$ be the eigendecomposition of the Laplacian and $\{\mathbf{u}_l\}_{l=1}^n$ and $\{\lambda_l\}_{l=1}^n$ the sets of eigenvectors and eigenvalues of $\mathbf{L}^{\text{sym}}$. Since $\mathbf{L}^{\text{sym}}$ is symmetric, these eigenvectors form a basis for $\mathbf{R}^n$, and we can therefore write $\mathbf{f} = \mathbf{U}\mathbf{c} = \sum_{l=1}^n c_l \mathbf{u}_l$. This implies that we can write the Laplacian quadratic form as

$$\mathcal{S}(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}^{\text{sym}} \mathbf{f} = \left(\mathbf{c}^\top \mathbf{U}^\top\right) \mathbf{U}\Lambda\mathbf{U}^\top (\mathbf{U}\mathbf{c}) = \mathbf{c}^\top \Lambda \mathbf{c} = \sum_{l=1}^{n} c_l^2 \lambda_l, \tag{2}$$

and that diagonal operators applied to the spectrum of the Laplacian modulate the smoothness of the signal; consequently, the eigenvalues can be seen as the frequencies of the signal [2]. Accordingly, if we wish to make a graph signal smoother, we should minimize this measure through removing frequencies that correspond to larger eigenvalues. This is done using a *low-pass filter*.

To low-pass filter a graph using a polynomial filter whose frequency-response function is g, we use the graph convolution operation which is defined as

$$\mathbf{f}_{\text{filtered}} = g\left(\mathbf{L}^{\text{sym}}\right) \mathbf{f} = \mathbf{U} g(\Lambda) \mathbf{U}^\top \mathbf{f} \tag{3}$$

such that $g(\Lambda) = \text{diag}\left(g(\lambda_1), ..., g(\lambda_n)\right)$. For example, in the case of a graph filter corresponding to a GCN [10] with p layers, or its simplified version [29] using a p-th order feature propagation, its frequency-response function is given as $g(\lambda) = (1-\lambda)^p$, or in matrix form as $g(\mathbf{L}^{\text{sym}}) = (\mathbf{I} - \mathbf{L}^{\text{sym}})^p = \mathbf{A}^p$, which is a polynomial filter that is low-pass for odd values of p and somewhat low-pass for even values of p since the filtering function is not strictly decreasing on the interval of definition of the eigenvalues $I = [0, 2]$. Note that the GCN also introduces added self-loops into the adjacency matrix $\mathbf{A}$. For more details about graph filtering and graph signal processing in general, we refer the reader to [15, 24].

3 PROPOSED MODEL

Now that we have introduced the necessary background, we can formulate our problem.

3.1 First-order Neighborhood Propagation and Linear Graph Filtering

As previously mentioned, the graph neighborhood propagation performed in the GCN acts as a filter on the graph signal and removes high-frequency noise. We argue, however, that these steps of neighborhood propagation as performed in the GCN are unnecessary and even counterproductive because of a risk of over-smoothing, which is when the signal becomes uniform over the different nodes. To support our argument, we would point to the performance of the linear graph autoencoder [20], which was competitive w.r.t more complex GCN-based models. In this paper we are seeking to show that a first-order (or one-hop) neighborhood propagation, when applied properly, is also sufficient for the task of simultaneous graph clustering and embedding. Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, let

$$\tilde{\mathbf{A}} \leftarrow \mathbf{A} + \beta\mathbf{I} \tag{4}$$

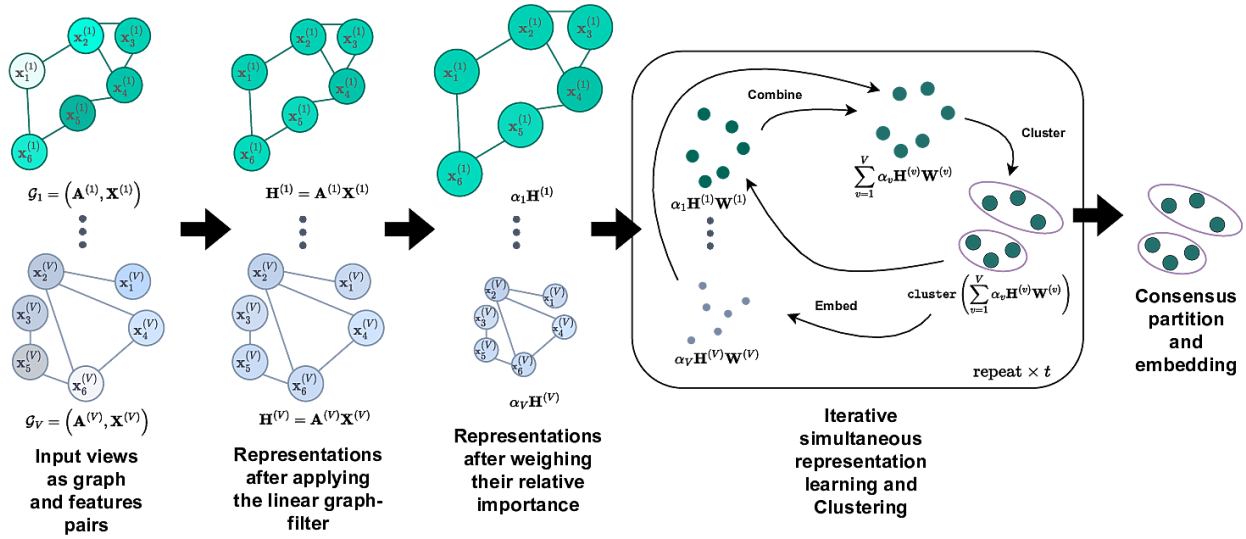


Figure 1: Schematic representation of LMGEC.

be the adjacency matrix with β added self-loops and $\tilde{\mathbf{D}}$ its diagonal matrix of degrees. We define our propagation matrix as

$$\mathbf{S} \leftarrow \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}. \tag{5}$$

The generalized Laplacian $\mathbf{L}^{\text{rw}}$ that corresponds to this propagation matrix is a random walk normalized Laplacian with added self-loops. The linear propagation operation we propose is

$$\mathbf{H} \leftarrow \mathbf{S}\mathbf{X} \tag{6}$$

where $\mathbf{H}$ are the new filtered features. The frequency-response function associated with this filter is $g(\lambda) = 1-\lambda$ or, in matrix form, $g(\mathbf{S}) = \mathbf{I} - \mathbf{L}^{\text{rw}}$, which is clearly a linear function that is decreasing in the interval of definition of the eigenvalues $I = [0, 2]$.

3.2 Simultaneous Multi-view Attributed Graph Representation Learning and Clustering

Given a multi-view attributed graph represented as a set of attributed graphs $\mathcal{M} := \{\mathcal{G}_v\}_{v=1}^{v=V}$, we define a preliminary version of the problem of simultaneous multi-view graph representation learning and clustering as

$$\begin{aligned} \min_{\substack{\mathbf{G},\mathbf{F}\\ \mathbf{W}_1,\ldots,\mathbf{W}_V}} \quad & \underbrace{\sum_v \underbrace{(\underbrace{\|\mathbf{H}_v - \mathbf{H}_v\mathbf{W}_v\mathbf{W}_v^\top\|^2}_{\text{reconstruction term}} + \underbrace{\|\mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F}\|^2}_{\text{clustering term}})}_{\text{individual view loss}}}_{\text{multi-view loss}} \\ s.t. \quad & \forall v \quad \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1} \end{aligned} \tag{7}$$

where $\mathbf{W}_v \in \mathbf{R}^{d_v \times f}$ such that d_v is the dimension of the features in view v, f is the dimensionality of the consensus representation we wish to learn and k is the number of clusters. The loss corresponding to each view consists of two terms, namely a reconstruction term and a clustering term:

- **The reconstruction term** can be seen as reconstructing the filtered graph signal of the v-th view $\mathbf{H}_v$ using a semi-orthogonal matrix $\mathbf{W}_v$ similar to what is done in principal component analysis. We may draw a parallel with autoencoders, where multiplying by $\mathbf{W}$ encodes the data and $\mathbf{W}^\top$ decodes it.
- **The clustering term** is similar to the k-means objective applied to the embeddings learned from the reconstruction process $\mathbf{H}_v\mathbf{W}_v$. $\mathbf{G}$ is a partition matrix and $\mathbf{F}$ is the centroids matrix.

Matrices $\mathbf{G}$ and $\mathbf{F}$ are the same for each matrix and represent the consensus partition and centroids respectively. There are, however, exactly V $\mathbf{W}_v$ matrices, due the fact that the features in the different views do not necessarily have the same dimensionality. Problem (7) can be rewritten in a way that combines the two terms as follows

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}_1,\ldots,\mathbf{W}_V} \quad & \sum_v \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|^2 \\ s.t. \quad & \forall v \quad \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \end{aligned} \tag{8}$$

This formulation is more intuitive, since we can see it as trying to minimize the discrepancy between each input vector $\mathbf{h}_i^v$ and its corresponding centroid learned in the latent embedding space after reconstructing $\mathbf{g}_i\mathbf{F}\mathbf{W}_v^\top$. A proof of this is available in [33].

3.3 Paying Attention to the Individual Views

Not all views have the same importance, and for this reason it is not optimal to directly add the losses from each view without applying some kind of importance weighting scheme. To address this issue we introduce a new set of parameters $\{\alpha_v\}_{v=1}^{v=V}$ such that $\sum_v \alpha_v = 1$ where α_v represents the relative importance of each view v. With this, we obtain the final formulation of our problem

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}_1,\ldots,\mathbf{W}_V} \quad & \sum_v \alpha_v\|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|^2 \\ s.t. \quad & \forall v \quad \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \end{aligned} \tag{9}$$

Note, however, that we do not introduce α_v as a parameter, since this would cause a solution to pay attention to only a single view, i.e., the view with the smallest individual view loss. With this in mind, we define α as the softmax α of the negative inertia of each view, and we add a temperature parameter for greater flexibility. The formula for each α is then

$$\alpha_v \leftarrow \frac{e^{\frac{-I_v}{\tau}}}{\sum_{w=1}^{w=V} e^{\frac{-I_w}{\tau}}} \tag{10}$$

where I stands for inertia. For the v-th view I_v is computed as follows: $I_v = \|\mathbf{H}_v - \mathbf{G}_v\mathbf{F}_v\|$ such that $\mathbf{W}_v$ is obtained through a truncated singular value decomposition (SVD) on $\mathbf{X}_v$, and $\mathbf{G}_v$ and $\mathbf{F}_v$ are the results of a k-means applied on the embeddings of the v-th view $\mathbf{X}_v\mathbf{W}_v$. When the temperature τ is sufficiently high, only the best view in terms of inertia is selected, and when it is sufficiently low, all the views have the same weight.

4 OPTIMIZATION AND COMPLEXITY

Even though solving LMGEC exactly may be NP-hard, a solution can be computed reasonably efficiently via the use of heuristics. To this end, we propose using a Block Coordinate Descent (BCD) scheme that boils down to iteratively solving sub-problems where we alternately solve for one of $\mathbf{W}_1, \ldots, \mathbf{W}_V, \mathbf{G}, \mathbf{F}$ while keeping the others fixed. All optimizations are described below.

4.1 Optimizing for G

When solving for $\mathbf{G}$ and fixing the other matrices, we obtain the following problem

$$\min_{\mathbf{G}} \quad \sum_v \alpha_v \|\mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F}\|^2 \quad s.t \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \tag{11}$$

This problem is hard to solve, so instead we propose solving the following relaxation obtained using the Cauchy-Schwarz inequality:

$$\min_{\mathbf{G}} \quad \left\| \sum_v \alpha_v \mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F} \right\|^2 \quad s.t. \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \tag{12}$$

In this way we can efficiently minimize the objective of this problem with the assignment step

$$g_{ij} \leftarrow \begin{cases} 1 & \text{if } j = \arg\min_l \| \left(\sum_v \alpha_v \mathbf{H}_v\mathbf{W}_v\right)_i - \mathbf{f}_l \|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

4.2 Optimizing for F

When optimizing for $\mathbf{F}$ we retrieve the same criterion as for $\mathbf{G}$, the difference lying in the constraints

$$\min_{\mathbf{F}} \quad \left\| \sum_v \alpha_v \mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F} \right\|^2. \tag{14}$$

This problem is an instance of an ordinary least-squares problem whose exact solution is easy to find and is given as

$$\mathbf{F} = (\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{G}^\top \left(\sum_v \alpha_v \mathbf{H}_v\mathbf{W}_v \right). \tag{15}$$

4.3 Optimizing for $\mathbf{W}_1, \ldots, \mathbf{W}_V$

Optimizing for the different $\mathbf{W}_{v=1}^{v=V}$ matrices results in V sub-problems. For a specific $\mathbf{W}_v$, the resulting problem is $\min_{\mathbf{W}_v} \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|^2$ which can be solved using a singular value decomposition as follows:

$$\mathbf{W}_v = \mathbf{U}\mathbf{V}^\top \quad s.t. \quad \mathbf{U}, \Sigma, \mathbf{V} = \mathtt{SVD}(\mathbf{X}_v^\top\mathbf{G}\mathbf{F}) \tag{16}$$

For more details on the derivation we refer the reader to [5, 33].

4.4 Optimization Algorithm

The main steps are summarized in Algorithm 1. The loss function might not be strictly decreasing due to the use of relaxations for the sub-problems, but it should nevertheless have an overall decreasing trend, as shown in figure 2 where we observe that our algorithm does not need many iterations to converge.

Algorithm 1: Block Coordinate Descent (BCD) for LMGEC

Input : - Sequence of views $\{(\mathbf{A}_v, \mathbf{X}_v)\}_{v=1,\ldots,V}$
- Number of clusters k
- Embedding dimension f
- Temperature τ
- Tolerance ϵ
- Maximum number of iterations max_iter

Output: - Consensus membership indicator $\mathbf{G} \in \{0,1\}^{n\times k}$
- Consensus embedding centers $\mathbf{F} \in \mathbf{R}^{k\times f}$
- Consensus embedding matrices $\mathbf{W} \in \mathbf{R}^{d\times f}$

$\forall v$ $\mathbf{H}_v \leftarrow \mathbf{A}_v\mathbf{X}_v$;
$\forall v$ Initialize $\mathbf{W}_v$ through a truncated SVD on $\mathbf{H}_v$;
$\forall v$ Compute α_v using formula (10);
Initialize $\mathbf{G}$ and $\mathbf{F}$ through a k-means on $\sum_v \alpha_v\mathbf{H}_v\mathbf{W}_v$;
while *change in loss* $> \epsilon$ **and** max_iter *not reached* **do**
  $\forall v$ Update $\mathbf{W}_v$ using formula (16);
  Update $\mathbf{G}$ using formula (13);
  Update $\mathbf{F}$ using formula (15);
  $loss \leftarrow \sum_v \alpha_v \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|$;
end

4.5 Complexity Analysis

For simplicity, we suppose that $f \in O(k)$, that $d_1, \ldots, d_V \in O(d)$ and that $|\mathcal{E}_1|, \ldots, |\mathcal{E}_V| \in O(|\mathcal{E}|)$. We also suppose that $k << n, d$, which is almost always the case in real-world attributed graph datasets. Note that in what follows, the multiplication of matrix $\mathbf{G}$ with another matrix amounts only to a re-indexing of this other matrix, because of the structure of $\mathbf{G}$.

- **First-order Neighborhood Propagation.** This step consumes roughly $O(v|\mathcal{E}|d)$ operations.
- **Initializing** $\{\mathbf{W}_v\}_{v=1}^{v=V}$**.** This step takes $O(vnd\log(k))$ when using a randomized SVD algorithm.
- **Computing** $\{\alpha_v\}_{v=1}^{v=V}$**.** Here it is necessary to compute the inertia as well as the sets $\{\mathbf{G}_v\}_{v=1}^{v=V}$ and $\{\mathbf{F}_v\}_{v=1}^{v=V}$ for each view, totalling around $O(vndk)$ operations.
- **Initializing G and F.** Computing the summation $\sum_v \alpha_v\mathbf{H}_v\mathbf{W}_v$ and the application of k-means on it amounts to $O(ndk)$ operations, where the number of iterations of k-means is held constant.

- **Updating** $\{\mathbf{W}_v\}_{v=1}^{v=V}$**.** Like their initialization steps, this step takes roughly $O(vnd\log(k))$.
- **Updating G.** The rule associated with this step takes $O(nk^2)$ computations.
- **Updating F.** This rule is computed in $O(vndk)$, as a result of computing the embeddings for the different views $\mathbf{H}_v\mathbf{W}_v$.
- **Objective Value Calculation.** The computation here can be performed in $O(nd)$.
- **Overall Complexity.** Altogether, the total computation time for our algorithm is $O(v|\mathcal{E}|d + tvndk)$, where t is the number of iterations of LMGEC.

5 EXPERIMENTATION

In this section we present the experimental setup and results. We start by introducing the datasets and the evaluation metrics used in relation to clustering, the methods used for comparison with LMGEC, and the experimental settings. We then present the results in terms of quality of clustering, followed by analysis where we consider embedding, complexity, sensitivity, and robustness in the face of noisy views.

5.1 Datasets and Metrics

In order to demonstrate the generic nature of our model we looked at the three possible types of multi-view attributed graph datasets:

(1) **Datasets with Heterogeneous Graph Topology.** These datasets have the same set of features $\mathbf{X}$ but multiple graph topologies, i.e. multiple adjacency matrices $\{\mathbf{A}_v\}_{v=1}^{v=V}$. They include ACM, DBLP and IMDB.
(2) **Datasets with Heterogeneous Features.** These datasets have the same graph structure $\mathbf{A}$ but multiple sets of features $\{\mathbf{X}_v\}_{v=1}^{v=V}$. The example we chose was Amazon Photos.
(3) **Datasets with Both.** These datasets have multiple graph structures $\{\mathbf{A}_v\}_{v=1}^{v=V}$ as well as multiple sets of features $\{\mathbf{X}_v\}_{v=1}^{v=V}$, The only such dataset is Wiki for which we create the additional views from the initial data, we initially have a single topology and features, we then generate a second topology using a nearest neighbor graph based on the cosine distance and a second set of features by using a log-scale of the original ones.

The characteristics of these datasets are given in table 1. In quantifying the quality of a clustering we use four metrics: clustering accuracy (CA), clustering F1-score (F1) [27], normalized mutual information (NMI) [25] and adjusted rand index (ARI) [8].

5.2 Baselines

Below we list all the methods evaluated in our proposal.

- **LINE** [26]: A single-view graph embedding method. It is applied on each view and the best results are reported.
- **GAE** [9]: Another single-view graph embedding method based on the autoencoder.
- **X-avg**: To utilize multiple views of a network we apply the X method to learn node representations on each single view, then average all learned representations.
- **LINE-avg** and **GAE-avg**: By this we mean that the node representations learned for each view using LINE and GAE are averaged and clustered as such.
- **MNE** [36]: A scalable multi-view network embedding model. Only the graph structure information (adjacency matrix) of each view is input into this model.
- **PMNE** [13]: Encompasses three multi-view graph embedding methods, including network aggregation PMNE (n), results aggregation PMNE (r) and layer co-analysis PMNE (c).
- **RMSC** [30]: A multi-view spectral clustering method that uses Markov chains and low-rank decomposition.
- **PwMC** and **SwMC** [14]: PwMC is a parameter-weighted multi-view graph clustering method, while SwMC is a self-weighted multi-view graph clustering method.
- **O2MA** and **O2MAC** [3]: O2MAC is also an autoencoder-based multi-view graph clustering method. O2MA is a simplified version of O2MAC with the clustering loss removed from the objective function.
- **DMGI** [16]: This is an unsupervised embedding method for attributed multiplex network embedding. This approach is only applicable to datasets with one set of features and multiple graphs.
- **MvAGC** [11]: Performs graph filtering to do multi-view attributed graph clustering.
- **MAGC** [12]: A multi-view graph clustering method that utilizes both node attributes and graphs.

5.3 Experimental Settings

We use the clustering results reported in the original papers where possible. When performing our own tests we tried to follow the setups prescribed by the authors of the different models as faithfully as possible. For our model, we set the maximum number of iterations to 30, the tolerance to 0 and $f = k + 1$ in all experiments. We performed experiments with different hyper-parameter values for β and τ. We did a tf-idf normalization of the inputs to our model, and we also centered the data after neighborhood propagation. For the values of β and τ, we try $\beta \in \{.2, 1, 2\}$ and $\tau \in \{1, 10, 100\}$ and we report the best results. The different experiments were run on the same machine with two Intel(R) Xeon(R) CPU @ 2.20GHz and 13GB RAM. Most of the source codes in the official repositories of the baselines were not optimized for GPU. Note that the results reported for our method are the averages of five runs.

5.4 Experimental Results

Below we study in detail the results from our comparisons and highlight the interest of **LMGEC**.

Clustering Results. Tables 2 and 3 show the results of our experiments for the clustering task. Some of the results for ACM, DBLP, IMDB and Amazon Photos are taken from [3, 11, 12], while results for Wiki are those that we obtained in our experiments. The pattern that emerges is that methods combining multi-view information tend to outperform those using single-view information. Our model LMGEC consistently achieves competitive performances, outperforming other models on ACM, DBLP, IMDB, Amazon Photos and Wiki on most metrics, and being competitive on IMDB, where it has the second best results on two out of four performance metrics. Overall, our model offers the best results in 15 out of 20 cases and has the best or second best results in 18 out of 20 cases, showing that it is competitive despite its simple nature. Note that

Table 1: Characteristics of the Datasets. For wiki, there are two topologies and two features matrices leading to four possible combinations/views.

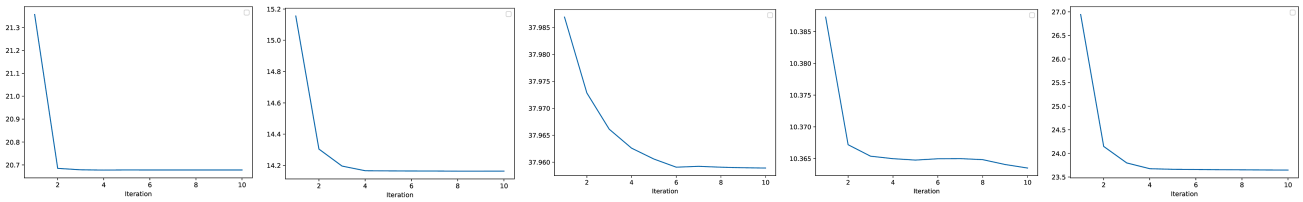
Dataset	Multi-view type	#Views	#Nodes	#Features	#Edges	#Clusters
ACM [28]	Topology	2	3,025	1,830	29,281 2,210,761	3
DBLP [28]	Topology	3	4,057	334	11,113 5,000,495 6,776,335	4
IMDB [28]	Topology	2	4,780	1,232	98,010 21,018	3
Amazon Photos [23]	Features	2	7,487	745 7,487	119,043	8
Wiki [34]	Both	4	2405 2405	4973 4973	24,357 12,025	17

Table 2: Clustering results on ACM, DBLP and IMDB. Best results are highlighted in bold and the second best results in italic.

Model	ACM				DBLP				IMDB			
	CA	F1	NMI	ARI	CA	F1	NMI	ARI	CA	F1	NMI	ARI
LINE-avg	0.6479	0.6594	0.3941	0.3433	0.875	0.866	0.6681	0.7056	0.4719	0.2985	0.0063	-0.009
GAE	0.8216	0.8225	0.4914	0.5444	0.8859	0.8743	0.6925	0.741	0.4298	0.4062	0.0402	0.0473
GAE-avg	0.699	0.7025	0.4771	0.4378	0.5558	0.5418	0.3072	0.2577	0.4442	0.4172	0.0413	0.0491
MNE	0.637	0.6479	0.2999	0.2486	out of memory error				0.3958	0.3316	0.0017	0.0008
PMNE(n)	0.6936	0.6955	0.4648	0.4302	0.7925	0.7966	0.5914	0.5265	0.4958	0.3906	0.0359	0.0366
PMNE(r)	0.6492	0.6618	0.4063	0.3453	0.3835	0.3688	0.0872	0.0689	0.4697	0.3183	0.0014	0.0115
PMNE(c)	0.6998	0.7003	0.4775	0.4431	out of memory error				0.4719	0.3882	0.0285	0.0284
RMSC	0.6315	0.5746	0.3973	0.3312	0.8994	0.8248	0.7111	0.7647	0.2702	0.3775	0.0054	0.0018
PwMC	0.4162	0.3783	0.0332	0.0395	0.3253	0.2808	0.019	0.0159	0.2453	0.3164	0.0023	0.0017
SwMC	0.3831	0.4709	0.0838	0.018	0.6538	0.5602	0.376	0.38	0.2671	0.3714	0.0056	0.0004
O2MA	0.888	0.8894	0.6515	0.6987	0.904	0.8976	0.7257	0.7705	0.4697	0.4229	0.0524	0.0753
O2MAC	0.9042	0.9053	0.6923	0.7394	0.9074	0.9013	0.7287	0.778	0.4502	0.4159	0.0421	0.0564
DMGI	0.8973	0.8985	0.6974	0.7296	0.8722	0.8691	0.6931	0.7034	0.5827	0.4253	**0.1317**	0.1457
MvAGC	0.8975	0.8986	0.6735	0.7212	0.9277	0.9225	0.7727	0.8276	0.5633	0.3783	0.0371	0.0940
MAGC	0.8806	0.8835	0.6180	0.6808	0.9282	0.9237	**0.7768**	0.8267	**0.6125**	**0.4551**	0.1167	**0.1806**
LMGEC	**0.9302**	**0.9311**	**0.7513**	**0.8031**	**0.9285**	**0.9241**	0.7739	**0.8284**	0.5893	0.4267	0.0632	0.1294

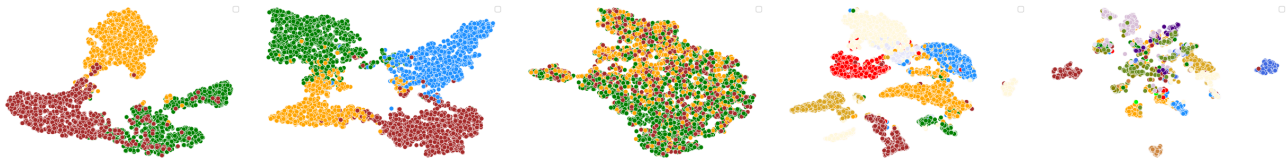
Table 3: Clustering results on Amazon Photos and Wiki. Additionally, we report the performance of LMGEC on each individual view (for the other datasets see figure 4). Note that Amazon Photos has only two views, while Wiki has four.

Model	Amazon Photos				Wiki			
	CA	F1	NMI	ARI	CA	F1	NMI	ARI
LMGEC (view 1)	0.6726	0.6451	0.5903	0.4865	0.4757	0.4154	0.4772	0.2944
LMGEC (view 2)	0.6835	0.6164	0.5971	0.4896	0.5181	*0.4463*	0.5079	0.3226
LMGEC (view 3)	-	-	-	-	0.5202	0.4333	0.5383	0.3401
LMGEC (view 4)	-	-	-	-	0.5264	0.4384	0.5362	0.3455
MAGC	0.4511	0.3359	0.4297	0.1127	0.4972	0.4084	0.5139	0.2707
MvAGC	0.6775	0.6397	0.5237	0.3968	0.3297	0.2432	0.3531	0.0864
LMGEC	**0.7117**	**0.6500**	**0.6114**	**0.5123**	**0.5333**	**0.4501**	**0.5408**	**0.3496**



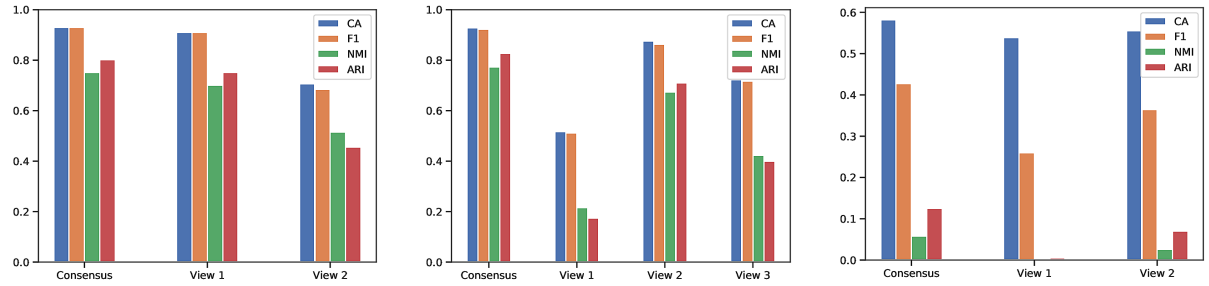
(a) ACM (b) DBLP (c) IMDB (d) Photos (e) Wiki

Figure 2: Evolution of the loss value across iterations using BCD for LMGEC.



(a) ACM (b) DBLP (c) IMDB (d) Photos (e) Wiki

Figure 3: Two-dimensional projections of the LMGEC embeddings using t-SNE colored according to the real class labels.



(a) ACM (b) DBLP (c) IMDB

Figure 4: Performance of LMGEC on each individual view vs. its consensus performance when considering all views on ACM, DBLP and IMDB (for the other datasets see table 3).

Table 4: Training times. Best results are in bold, second best results are in italic. DMGI is only applicable to datasets with one set of features.

Model	ACM	DBLP	IMDB	Wiki	Amazon Photos
DMGI	943.19	3117.87	843.96	-	-
MvAGC	*14.48*	*26.28*	*32.97*	*34.73*	*139.14*
MAGC	139.93	242.99	395.45	150.93	1661.64
LMGEC	**3.49**	**3.07**	**4.96**	**18.06**	**19.42**

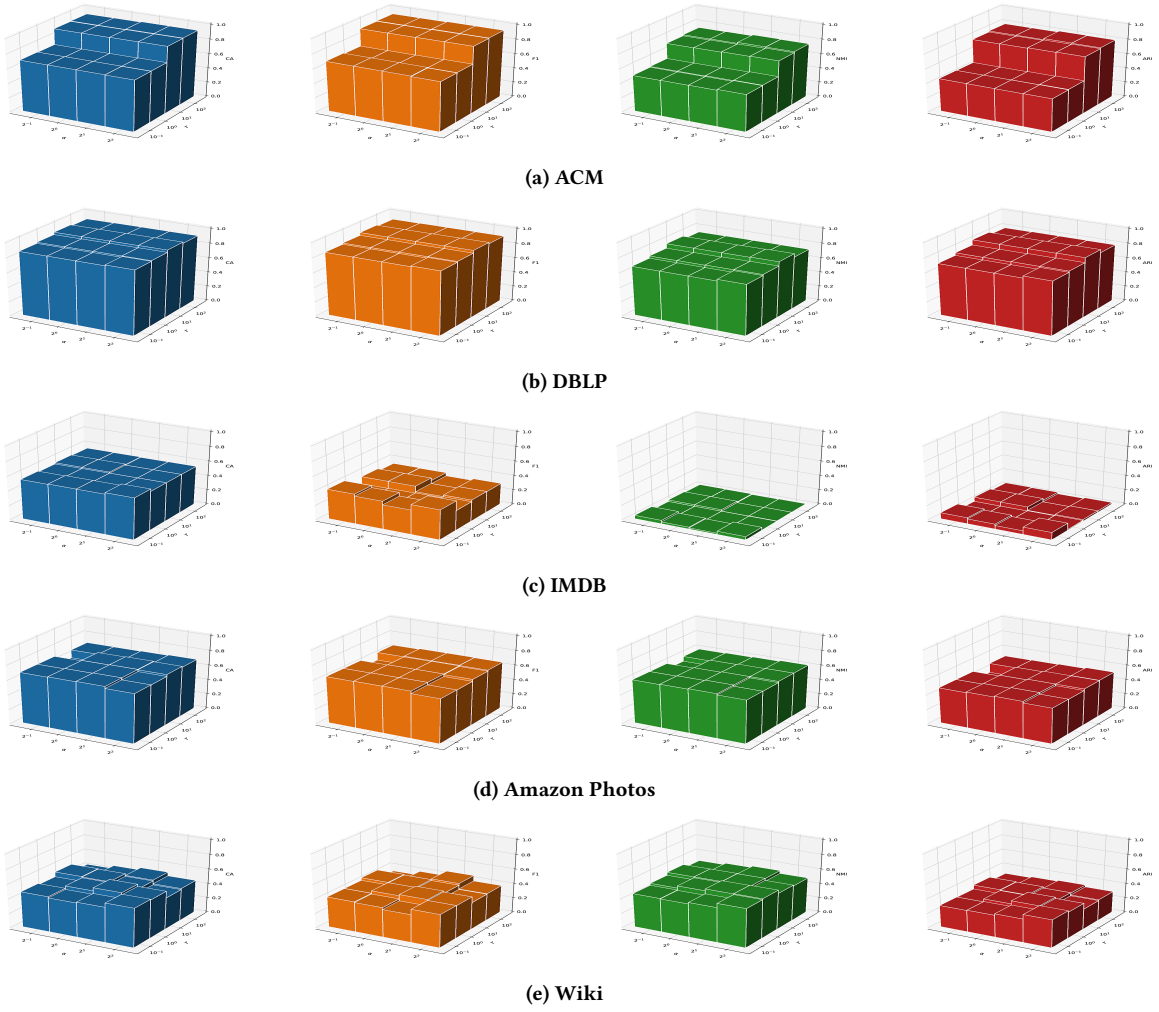
for datasets with multiple features, in our experiments we used only the baselines that were the best performing (on average).

We can see the benefits of our model from tables 2 and 3 and from figure 4, where the performance of LMGEC on individual views is shown against the consensus performance; the consensus performance is consistently better than for the individual views. In some instances LMGEC applied to the individual views even outperforms state-of-the-art models, for example, on ACM, Amazon Photos and Wiki.

Embedding Results. Figure 3 depicts the embeddings produced by LMGEC on the different datasets by projecting them onto a 2d-plane using t-SNE. A clustering structure is visible on all datasets apart from IMDB, where the embeddings are not very well separated.

Efficiency Results. We report the training times of our method in table 4 as well as those of the best performing (on average) baselines in our experiments. Our model is consistently much faster than other models, improving on the training time of the second fastest model (MvAGC) by 75%, 89%, 85%, 48% and 86% on ACM, DBLP, IMDB, Wiki and Amazon Photos respectively.

Sensitivity Analysis. In our experiments we tried various values for the β and τ hyperparameters. Figure 5 illustrates the performance of LMGEC for different pairs of these parameter values on the different datasets and for the different clustering metrics. We see that the on most datasets the performance remains fairly constant, which is an indication of LMGEC's robustness. The exception is ACM, where LMGEC is sensitive to the temperature parameter τ



(a) ACM

(b) DBLP

(c) IMDB

(d) Amazon Photos

(e) Wiki

Figure 5: Sensitivity analysis of the parameters of LMGEC on the graph topology heterogeneous datasets.

because of the presence of uninformative views. We discuss this in greater detail in the following paragraph. As a rule of thumb, we suggest taking $\tau = 10$ and $\beta = 1$.

Robustness in the face of noisy views. In real applications noisy data sources are not uncommon and can impair the performance of multi-view models. Since LMGEC takes into account the importance of the different views via a weighting scheme based each view's inertia, it is able to filter out noisy views by increasing the temperature τ of the softmax function used in computing $\{\alpha_v\}_{v=1}^{v=V}$. Tables 2 and 3 and figure 4 report the clustering performance of LMGEC on each individual view for the different datasets, as well as the consensus performance. In the case of DBLP we may consider that the first view is noisy, since LMGEC performs considerably less well on this view than on the second and third views. We remark, however, that the consensus performance is not influenced by the presence of this noisy view, which shows the robustness of LMGEC in the face of noise. The same is true of Wiki as regards the first view, albeit less significantly than for DBLP, since the performance gap is not as flagrant w.r.t the other views.

6 CONCLUSION

In this paper we proposed a simple linear additive model that addresses the dual tasks of multi-view attributed graph representation learning and multi-view attributed clustering in a unified framework. This model is more generic than most state-of-the-art approaches in the sense that it can deal with any number of graph structures and/or any number of feature sets. Experiments showed that our model is competitive with more complex state-of-the-art models, outperforming these models on most benchmarks in terms of both performance and computation time.

ACKNOWLEDGMENTS

This work has been funded by Informatique Caisse des Dépôts et Consignations (ICDC), Association Nationale de la Recherche et de la Technologie (ANRT), and Idex-Spectrans of UP Cité.

REFERENCES

[1] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2021. Implicit consensus clustering from multiple graphs. *Data Mining and Knowledge Discovery* 35, 6 (2021), 2313–2340.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[3] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. 2020. One2multi graph autoencoder for multi-view graph clustering. In *Proceedings of The Web Conference 2020*. 3070–3076.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/3308558.3313488

[5] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2022. Efficient Graph Convolution for Joint Node Representation Learning and Clustering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 289–297.

[6] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2022. Subspace Co-clustering with Two-Way Graph Convolution. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3938–3942.

[7] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2023. Scalable Attributed-Graph Subspace Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37.

[8] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.

[9] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

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

[11] Zhiping Lin and Zhao Kang. 2021. Graph Filter-based Multi-view Attributed Graph Clustering.. In *IJCAI*. 2723–2729.

[12] Zhiping Lin, Zhao Kang, Lizong Zhang, and Ling Tian. 2021. Multi-view attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[13] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 134–141.

[14] Feiping Nie, Jing Li, Xuelong Li, et al. 2017. Self-weighted Multiview Clustering with Multiple Graphs.. In *IJCAI*. 2564–2570.

[15] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. 2018. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE* 106, 5 (2018), 808–828.

[16] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised attributed multiplex network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5371–5378.

[17] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 2017. 3D Graph Neural Networks for RGBD Semantic Segmentation. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 5209–5218.

[18] Paul Riverain, Simon Fossier, and Mohamed Nadif. 2022. Semi-supervised Latent Block Model with pairwise constraints. *Machine Learning* 111, 5 (2022), 1739–1764.

[19] Aghiles Salah and Mohamed Nadif. 2017. Social regularized von Mises–Fisher mixture model for item recommendation. *Data Mining and Knowledge Discovery* 31, 5 (2017), 1218–1241.

[20] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis. 2020. Simple and effective graph autoencoders with one-hop linear models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 319–334.

[21] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. 2018. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*. 4470–4479.

[22] Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-Shot Learning with Graph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJj6qGbRW

[23] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[24] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre 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.

[25] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3, Dec (2002), 583–617.

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[27] van Rijsbergen (CJ). 1979. *Information retrieval*. Butterworth.

[28] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. 2022–2032.

[29] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[30] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. 2014. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.

[31] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634* (2013).

[32] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. 2017. Scene Graph Generation by Iterative Message Passing. In *Computer Vision and Pattern Recognition (CVPR)*.

[33] Michio Yamamoto and Heungsun Hwang. 2014. A general formulation of cluster analysis with dimension reduction and subspace separation. *Behaviormetrika* 41, 1 (2014), 115–129.

[34] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *IJCAI*.

[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[36] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable multiplex network embedding.. In *IJCAI*, Vol. 18. 3082–3088.

[37] Dengyong Zhou and Bernhard Schölkopf. 2004. A regularization framework for learning from graph data. In *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields (SRL 2004)*. 132–137.