Semi-Supervised Embedding of Attributed Multiplex Networks

Ylli Sadikaj
ylli.sadikaj@univie.ac.at
Faculty of Computer Science and UniVie Doctoral School
Computer Science, University of Vienna
Vienna, Austria

Justus Rass
justus.rass@univie.ac.at
Faculty of Computer Science, University of Vienna
Vienna, Austria

Yllka Velaj
yllka.velaj@univie.ac.at
Faculty of Computer Science, University of Vienna
Vienna, Austria

Claudia Plant
claudia.plant@univie.ac.at
Faculty of Computer Science and ds:Univie
University of Vienna
Vienna, Austria

ABSTRACT

Complex information can be represented as networks (graphs) characterized by a large number of nodes, multiple types of nodes, and multiple types of relationships between them, i.e. multiplex networks. Additionally, these networks are enriched with different types of node features.

We propose a Semi-supervised Embedding approach for Attributed Multiplex Networks (SSAMN), to jointly embed nodes, node attributes, and node labels of multiplex networks in a low dimensional space. Network embedding techniques have garnered research attention for real-world applications. However, most existing techniques solely focus on learning the node embeddings, and only a few learn class label embeddings. Our method assumes that we have different classes of nodes and that we know the class label of some, very few nodes for every class. Guided by this type of supervision, SSAMN learns a low-dimensional representation incorporating all information in a large labeled multiplex network. SSAMN integrates techniques from Spectral Embedding and Homogeneity Analysis to improve the embedding of nodes, node attributes, and node labels. Our experiments demonstrate that we only need very few labels per class in order to have a final embedding that preservers the information of the graph. To evaluate the performance of SSAMN, we run experiments on four real-world datasets. The results show that our approach outperforms state-of-the-art methods for downstream tasks such as semi-supervised node classification and node clustering.

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**; • **Networks → Network properties**.

KEYWORDS

Network Embedding, Multiplex Networks, Attributed Networks.

WWW '23, April 30–May 04, 2023, Austin, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583485

ACM Reference Format:
Ylli Sadikaj, Justus Rass, Yllka Velaj, and Claudia Plant. 2023. Semi-Supervised Embedding of Attributed Multiplex Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583485

1 INTRODUCTION

Complex data from different domains can be represented by networks (graphs). There are many examples of these networks in various fields: social networks [6], collaboration or citation networks [29], biological networks or brain networks [38], and many more. Through networks, we represent different entities by nodes and different relations between two nodes by edges. In social networks, nodes represent users who are friends or follow each other; in collaboration networks, nodes represent authors who worked together; in brain networks, nodes represent brain regions and their communication. Additionally, networks are often enriched with node attributes (features). They represent different characteristics of nodes, e.g., in social networks, node attributes can represent education, gender, or occupancy, or in collaboration networks, the number of citations, number of publications, or h-index. When there exist different types of relations between nodes, we can use multiplex networks to represent such complex settings, and an example is shown in Figure 1.

Attributed multiplex networks (AMNs) have received great attention from the data mining community as it is becoming more evident that they are a powerful tool to model real-world scenarios. However, performing data mining tasks, such as node clustering, on AMNs poses challenges. Although the concept of community is relatively intuitive, there has been no formal definition of community on which there is general consensus [4]. Also, existing computational problems, such as finding the best communities of AMNs, are computationally very expensive.

Given the graph size, embedding techniques are often applied to obtain a compressed data representation without losing important information. However, many works focus on designing simple graph embedding methods for graphs with a single type of edges [12, 17, 31, 36, 39, 46]. Existing multiplex network methods learn node embeddings using solely structural information [19, 23, 24, 45], and only some of them encode node attribute information [13, 15, 26, 30, 35, 41, 44].

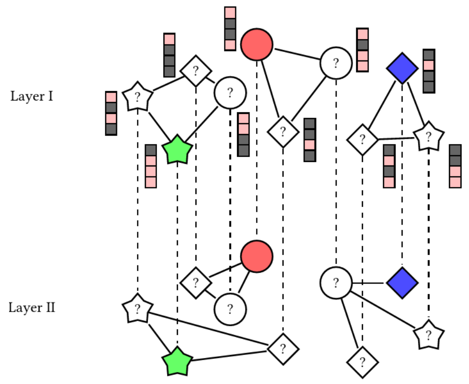


Figure 1: An attributed multiplex network. Different types of nodes are represented by different shapes (circle, star, and diamond). Layers I and II represent different types of relationships. Colored nodes represent labeled nodes, and question-marked nodes (?) represent unlabeled nodes. Node attributes are illustrated by boxes colored according to their values. The same nodes in two different layers are connected by a dashed line.

Nevertheless, to our knowledge, no approach is dedicated explicitly to the embedding of different types of nodes, different types of categorical attributes, and node labels from AMNs in the same dimensional space. Thus, in this work, we propose a Semi-supervised Embedding method for Attributed Multiplex Networks called SSAMN to embed nodes, node attributes, and node labels of AMNs to a joint low dimensional space with a focus on semi-supervised node classification and node clustering tasks.

Recently, semi-supervised learning has been gaining much traction for its practical benefits. There is a lack of labeled data in many tasks, and it may be challenging to obtain the labels because they require human annotators and special and expensive devices. Lately, semi-supervised methods have been often applied to different domains mainly related to drug discovery, classification of new drugs as toxic or not toxic, or drug repurposing [32, 34]. In these scenarios, neither supervised nor unsupervised learning algorithms can effectively use a few labeled data and a large number of unlabeled data. Our main contributions are:

- We propose a semi-supervised joint embedding approach for nodes, node attributes, and node labels of attributed multiplex networks. Going beyond existing methods, our framework handles different relation types, considering all types of information in order to obtain embeddings with the same dimensionality in the same vector space.
- We show how to exploit Homogeneity Analysis [9] and Laplacian Eigenmaps [3] to have the joint embedding of nodes, node features and node labels in the same vector space for multiplex networks with or without node attributes.
- We evaluate the proposed approach on real-world datasets with various evaluation metrics to demonstrate the effectiveness of the proposed method.
- We highlight the expressivity of our method in providing the interpretability of the results through a visualization task using a few dimensions of our embeddings.
- We design a new approach, SSAMN, that has only one tuning parameter, the dimensionality parameter.

2 RELATED WORKS

Several works focus on semi-supervised learning on networks. We can classify them based on different categories.

Attributed Networks (ANs). To the best of our knowledge, one of the most powerful methods for embedding attributed simple networks is Deep Graph Infomax (DGI) [39]. DGI [39] exploits deep learning and uses a graph convolutional network architecture to maximize the mutual information between patch representations and corresponding high-level summaries of networks. Other interesting approaches are [12, 17, 31, 36, 46].

Multiplex Networks (MNs). Different methods [19, 23, 24, 45] have been developed to perform different tasks on multiplex networks. A limitation of these methods is that they are not designed to utilize the information from node attributes. The best performer in this category, as shown in [15, 24, 30], is Deep Multi-Graph Clustering (DMGC) [24]. DMGC is an unsupervised deep learning method, and it performs two tasks: network clustering and cross-network cluster association.

Attributed Multiplex Networks (AMNs). Recently, AMNs have started to receive more attention, and one of the first approaches which emphasized the importance of multiplex networks is a Heterogeneous Graph Attention Network (HAN) [41]. HAN [41] is a semi-supervised graph neural network approach based on two attention-level mechanisms, hierarchical node-level, and metapath-level attentions. Representation Learning for Attributed Multiplex Heterogeneous Networks (GATNE) [7] is an approach that supports transductive and inductive embedding learning for attributed multiplex networks. Another method is presented in [30]: Unsupervised Attributed Multiplex Network Embedding (DMGI), which is an extension to DGI [39] and employs the InfoMax principle for multiplex networks. Two other approaches that consider mutual information are HDMI: High-order Deep Multiplex Infomax (HDMI) [15], and Semi-Supervised Deep Learning for Multiplex Networks (SSDCM) [26]. The HDMI approach is a self-supervised framework that extends the previous works, DGI and DMGI, but performs a supervised node classification task. SSDCM is a semi-supervised approach that employs the mutual information between the local node and global label representation. More recently, two approaches designed for heterogeneous networks have been published: Implicit Graph Neural Networks (IGNN) [13] and Network Schema Preserving Heterogeneous Information Network Embedding (NSHE) [47]. The IGNN [13] approach is a graph learning framework that captures long-range dependencies in networks. Whereas the NSHE [47] approach uses subgraphs and multi-task learning tasks to sustain the heterogeneous structure of a network. Presented results on both original papers show that IGNN [13] outperforms NSHE [47].

Most of the above-mentioned methods apply semi-supervised or supervised learning techniques when considering node classification or node clustering tasks, or both, and we focus on comparing our work with those baselines. Although in the meantime, unsupervised methods designed for MNs that do not use node label information have been published, such as [2, 20, 27, 35]. For a comprehensive review, please refer to [8, 40, 48].

Our new approach, SSAMN, differs from the previous methods in the following aspects: (i) they either ignore different relation types, different node attributes, or both types of information (ii) they do not provide a joint embedding of nodes, node attributes, and class labels and, as shown in Section 5, (iii) they exhibit good performances only in one task or for a single category of graphs, and (iv) they have many tuning parameters, up to thousands of parameters. Thus, if we consider the same learning process, semi-supervised learning, our approach uses information from different types of relations and different categorical node attributes, with few labeled nodes and only one tuning parameter. It can outperform all baselines in different tasks for different categories of networks and is more expressive even with fewer dimensions.

3 NOTATION AND PROBLEM DEFINITION

Notation. We define an attributed multiplex network $\mathcal{G}$ as a set of simple graphs $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2 \dots, \mathcal{G}_\mathcal{R}\}$, one for each relation type $r = 1, 2, \dots, \mathcal{R}$. We assume that each layer in the multiplex graph represents a relation type. We denote by $\mathcal{V}$ the vertex set $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$, and with n the total number of nodes. Each relation type r is characterized by an adjacency matrix denoted by $\mathcal{A}_r$, where $r = 1, 2, \dots, \mathcal{R}$. Thus, a layer in a multiplex network can be denoted as $\mathcal{G}_r = (\mathcal{V}, \mathcal{A}_r)$. With $|\mathcal{A}_r|$, we denote the number of edges of graph $\mathcal{G}_r$.

Each node $i = 1, 2, \dots, n$ has a set of attributes (features) denoted by $\mathcal{F}_i = \{F_1, F_2, \dots, F_c\}$ where c is the dimensionality of the attribute space. Moreover, each node attribute F_j with $j = 1, 2, \dots, c$, can have k_j distinct values. For binary attributes such as the appearance of a keyword in an abstract of a paper, we have $k_j = 2$. We denote the categorical value of a node attribute for node v_i by $F_j(v_i)$. We denote by C the total number of categorical values which is $C = \sum_{j=1}^{c} k_j$.

For our semi-supervised method, we denote a set of labeled nodes by $\mathcal{L}$, a subset of $\mathcal{V}$. Thus, $\mathcal{L} = \{1, 2, \dots, L\}$, with $n >> L$. Each label can have a value $y \in \{1, \dots, Y\}$, as Y is the total number of classes. Each labeled node of an attributed multiplex network is embedded into a low dimensional vector space with other nodes and each category of node attributes.

Problem definition. Many approaches learn representation on multiplex networks by embedding the nodes for each layer r into a low dimensional vector space, and then they aggregate embeddings of each layer into a joint embedding. Such methods exploit a Graph Convolutional Networks (GCNs) architecture [30, 39, 49]. Differently, our approach is designed to utilize information from all r layers and generate a joint node embedding which is updated for a few iterations using all available information from the network structure, node attributes, and node labels.

Thus, our approach defines a mapping $\mathcal{M} : \mathcal{G} \rightarrow \mathbb{R}^d$, where d is the dimensionality, and the goal is that nodes with similar characteristics are close to each other in the embedding space. SSAMN provides us with two matrices, the first one denoted by $\mathcal{Z}^{n \times d}$ with the node embeddings, and the second one, $\mathcal{Y}^{(C+Y)\times d}$ with node feature and node label embeddings. The coordinate for a node v_i in a dimension $x = 1, \dots, d$ is defined by $z_{i,x}$, that of an attribute F_j with categorical value $F_j(v_i)$ is defined by $y_{F_{j_i},x}$, and the coordinate for class $y = 1, \dots, Y$ is defined by $y_{C+y,x}$.

Our algorithm SSAMN is sketched in Algorithm 1 and it takes as input a graph $\mathcal{G}$, which consists of $\mathcal{R}$ adjacency matrices of order $n \times n$, an attribute matrix of order $n \times C$, a matrix of labeled nodes of order $l \times Y$, and dimensionality d. The output of our method is a d-dimensional vector space representation of nodes denoted by matrix $\mathcal{Z}$, and a d-dimensional vector space representation of node attributes and labels. Also, our method can be applied to a single network with or without node attributes or multiplex networks without node attributes. The final embeddings of SSAMN are suitable for tasks such as semi-supervised node classification and node clustering. Moreover, visualization is possible and it helps to obtain a better understanding of the datasets by showing embeddings of similar nodes, node attributes, and class labels and highlighting their impact on each other.

4 PROPOSED METHODOLOGY

Our method is designed for mixed-type data of several modalities.

In our problem setting we have a set of nodes that are linked by various relations and are characterized by many node attributes, and also, we have few nodes for which we have class information. The aim of our algorithm is to generate a joint vector space representation that integrates all of the distinct modalities. Our objective function combines notions of spectral embedding of graphs and homogeneity analysis of categorical data. Additional details about our approach are given in Figure 3.

4.1 Embedding Attributed Multiplex Networks

First and foremost, spectral embedding is a well-known technique for representing information such as networks into a low dimensional space [28, 42, 43]. We adapt this technique for our problem setting, and as in [35], we adjust it for multiplex networks. Given a multiplex network, we aim at minimizing the distance in low dimensional space between two nodes v_i and v_j, which are connected in any layer of the multiplex graph, and we minimize the euclidean distance between them in dimension x:

$$||z_{i,x} - z_{j,x}||_2 \tag{1}$$

Our approach outputs a d-dimensional embedding and we minimize the distance of connected nodes in all d-dimensions. Each edge (v_i, v_j) in the graph $\mathcal{G}_r$ of relation type r can be weighted or unweighted; we denote the weight by $w_r(i, j)$. Additionally, we introduce a normalization term, $d_{i,j}$, which denotes the euclidean distance of two connected nodes v_i and v_j in low-dimensional space over all dimensions d. To consider that into our approach we extend (1) as follows:

$$\frac{1}{2} \sum_{e=(v_i,v_j)\in E_r} w_r(i,j) \cdot \sum_{x=1}^{d} \frac{||z_{i,x} - z_{j,x}||_2}{d_{i,j}}. \tag{2}$$

While considering each relation type, we need to define a weighting factor for each layer, denoted by α_r. Differently from the spectral embedding technique, where only one relation type is considered, here, our approach considers all different relation types. Thus we extend equation (2) as follows:

$$\frac{1}{2}\sum_{r=1}^{\mathcal{R}}\alpha_r\cdot\left(\sum_{e=(v_i,v_j)\in E_r}w_r(i,j)\cdot\sum_{x=1}^{d}\frac{||z_{i,x}-z_{j,x}||_2}{d_{i,j}}\right) \tag{3}$$

Equation (3) represents the impact of network structure over the node embedding. Also, it is possible to consider all $\mathcal{R}$ relation types with the same weight, setting $\alpha_r = 1$. That case would represent the unweighted impact of network structure on node embeddings.

Besides the structure impact, we use the available information on node attributes to improve the node embedding. In order to use node attribute information, we adapt the Homogeneity Analysis method [9], a technique from statistics for PCA of discrete data [16]. By applying homogeneity analysis, we are able to construct a bipartite graph between nodes of multiplex networks and their categorical attributes. Each categorical node attribute, F_j, constructs its own bipartite graph, and each category of a node attribute F_{j_i} is connected to the corresponding node v_i. An illustration of a bipartite graph is shown in Figure 2.

SSAMN is a semi-supervised joint embedding approach, and we assume that a few labeled nodes for the multiplex networks are given. Then, for the given information, we apply Homogeneity Analysis for node labels, similarly as we showed for node attributes. Thus, we construct a bipartite graph for node labels $y = 1, \ldots, Y$ of the ground truth data, and each node v_i is connected with its corresponding class y. In this case, we have only a few edges in the bipartite graph as we have only a few labeled nodes. If for a node v_j we are not given any information about its label, then node v_j is not connected to any class label y in the bipartite graph.

We acknowledge the impact of embeddings of any node attribute $\mathcal{F}_j$ with a categorical value k, on any node embedding $z_{i,x}$ by minimizing the distance between node v_i and its corresponding categorical values as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{c}\alpha_{\mathcal{F}_j}\cdot\sum_{x=1}^{d}||z_{i,x}-y_{\mathcal{F}_{j_i},x}||_2 \tag{4}$$

Similarly, we include the information from node labels for the labeled nodes, and we minimize the distance in the low dimensional space between labeled nodes and their corresponding class label l as follows:

$$\sum_{i=1}^{L}\sum_{l=1}^{Y}\alpha_l\cdot\sum_{x=1}^{d}||z_{i,x}-y_{C+l,x}||_2 \tag{5}$$

The coefficients $\alpha_{\mathcal{F}_j}$ and α_l represent the weighting factors for node attributes and node labels, respectively. To set the value of all these coefficients, including α_r, we introduce the parameter m, and it represents the maximum between two values: the number of edges of the relation type with higher density in the multiplex network or the number of edges in the bipartite graphs generated by the attributes and the labels. Then, each coefficient is computed as $\alpha_r = \frac{m}{|\mathcal{A}_r|}$, where $|\mathcal{A}_r|$ denotes the total number of edges in the adjacency matrix corresponding to the relation type r; $\alpha_{\mathcal{F}_j} = \frac{m}{|\mathcal{A}_{\mathcal{F}_j}|}$, where $|\mathcal{A}_{\mathcal{F}_j}|$ represents the number of edges in the bipartite graph generated by all categorical values of the node attribute $\mathcal{F}_j$. For the node labels, the weighting coefficient depends on the importance we assign to the given labels; we set $\alpha_l = \frac{L}{L_y}$, by L we denote the total number of labeled nodes, and by L_y we denote the total number of nodes labeled with a class label y.

4.2 Objective Function

Here we show that in order to obtain the final low-dimensional representation of nodes, node attributes, and node labels, we minimize the following objective function defined by combining ideas represented in equations (3), (4), and (5):

$$\begin{aligned}\min \frac{1}{2}\sum_{r=1}^{\mathcal{R}}\alpha_r\cdot&\left(\sum_{e=(v_i,v_j)\in E_r}w_r(i,j)\cdot\sum_{x=1}^{d}\frac{||z_{i,x}-z_{j,x}||_2}{d_{i,j}}\right)\\ &+\sum_{i=1}^{n}\sum_{j=1}^{c}\alpha_{\mathcal{F}_j}\cdot\sum_{x=1}^{d}||z_{i,x}-y_{\mathcal{F}_{j_i},x}||_2\\ &+\sum_{i=1}^{L}\sum_{l=1}^{Y}\alpha_l\cdot\sum_{x=1}^{d}||z_{i,x}-y_{C+l,x}||_2\\ &\text{subject to } \mathcal{Z}^T\mathcal{Z}=I_n.\end{aligned} \tag{6}$$

We apply the Gram-Schmidt orthonormalization algorithm [33] to node embeddings such that the matrix $\mathcal{Z}$ is column-orthonormal, ensuring we avoid trivial solutions.

In order to obtain embeddings for nodes, node attributes, and node labels, we randomly initialize matrices $\mathcal{Z}$ and $\mathcal{Y}$ for all instances n and $C + Y$, respectively (*line* 1 in Algorithm 1). Dimensionality for both matrices is d, given as the input parameter. In each run of Algorithm 1, these matrices are updated. To update node embeddings in matrix $\mathcal{Z}$, we iterate through all nodes v_i, all the relation types r, and all the neighbors v_p of node v_i. We update, in dimension x, the coordinate z_{ix} by adding the product of the weighting factor of relation type r, with the edge weight $w_r(i, p)$, and the embedding value of node v_p in dimension x. Finally, this product is divided by $sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$ (*line* 3 – 7). This value is computed by summation of the product of the degree of node v_i in relation type r with the weighting factor of that relation type α_r. Then we add the weighting factor of each categorical node attribute $\alpha_{\mathcal{F}_j}$ and node label α_l to that sum.

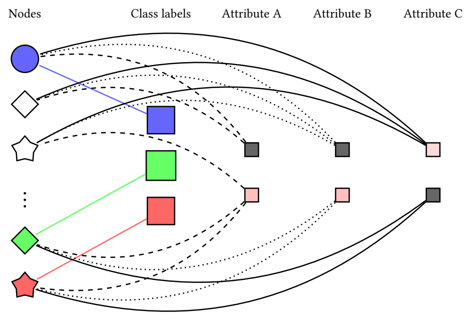


Figure 2: Bipartite graphs between nodes, class labels, and node attributes. Labeled nodes are connected to class labels and are colored. All nodes are linked with attributes according to their categorical value.

Algorithm 1: SSAMN

```
   Input: A multiplex graph $\mathcal{G}$ ($\mathcal{R}$ adjacency matrices $\mathcal{A}_r$, feature matrix $\mathcal{F}$,
          node labels $\mathcal{L}$), dimensionality $d$
   Output: $d$-dimensional feature space representation of data objects $\mathcal{Z}$,
          categories and classes $\mathcal{Y}$
1  Initialize matrices $\mathcal{Z}$ and $\mathcal{Y}$ randomly;
2  repeat
       /* update node embeddings */
3      for $i = 1, \dots, n$ do
4          for $r = 1, \dots, \mathcal{R}$ do
5              for $p = 1, \dots |N_i|$ with $N_i = \{v_p | (v_i, v_p) \in \mathcal{G}_r\}$ do
6                  for $x = 1, \dots, d$ do
7                      $z_{ix} \leftarrow z_{ix} + \alpha_r \cdot w_r(i,p) \cdot z_{px} / sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$;
8          for $j = 1, \dots, c$ do
9              for $x = 1, \dots, d$ do
10                 $z_{ix} \leftarrow z_{ix} + \alpha_{\mathcal{F}_j} \cdot y_{\mathcal{F}_{j_i}x} / sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$;
11         if node $i$ is labeled then
12             if node $i$ has label $l$ then
13                 for $x = 1, \dots, d$ do
14                     $z_{ix} \leftarrow z_{ix} + \alpha_l \cdot y_{(C+l)x} / sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$;
15             else
16                 for $x = 1, \dots, d$ do
17                     $z_{ix} \leftarrow z_{ix} - \alpha_l \cdot y_{(C+l)x} / sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$;
18     Apply Gram-Schmidt orthonormalization algo. to ensure $\mathcal{Z}^T \mathcal{Z} = I_n$;
       /* update category and class embeddings */
19     for $i = 1, \dots, n$ do
20         for $j = 1, \dots, c$ do
21             for $x = 1, \dots, d$ do
22                 $y_{\mathcal{F}_{j_i}x} \leftarrow y_{\mathcal{F}_{j_i}x} + z_{ix} / deg_{\mathcal{F}_j}$;
23         for $l = 1, \dots, Y$ do
24             for $x = 1, \dots, d$ do
25                 $y_{(C+l)x} \leftarrow y_{(C+l)x} + z_{ix} / deg_l$;
26 until convergence;
27 return $\mathcal{Z}, \mathcal{Y}$;
```

Similarly, we consider the impact of node attributes and node labels on node embeddings by updating the embedding of node v_i (*line 8-17*). To update the embedding of node v_i, in dimension x, we add the product of the weighting factor of the categorical node attribute $\alpha_{\mathcal{F}_j}$, and the embedding of the corresponding categorical value of node attribute $\mathcal{F}_{j_i}$, in dimension x, denoted by $\mathcal{Y}_{\mathcal{F}_{j_i}x}$, which we divide by the weighted sum of node v_i. We repeat the same steps for updating the node embedding using the node label information when the label is available for the node. If node v_i has label l then we update the embedding of that node, in dimension x, by adding the product of the weighting factor for the node label α_l, and its embedding in dimension x, denoted by $y_{(C+l)x}$, which we divide by the weighted sum of node v_i. Otherwise, if class label for node v_i is not the label l, then we subtract the product of the weighting factor for the class label α_l, and its embedding in dimension x, which we divide by the weighted sum of node v_i. Thus, our approach forces the embedding of nodes in low-dimensional space to be closer to their corresponding class label and further apart from other class labels.

In order to have the matrix $\mathcal{Z}$ column-orthonormal, we apply the Gram-Schmidt orthonormalization algorithm [33] (*line* 18).

To update embeddings of the categorical node attributes, $y_{\mathcal{F}_{j_i},x}$, and the node labels, $y_{C+l,x}$, we iterate through all nodes, all categorical values of node attributes c, and node labels, Y, for each dimension x (*line* 19 − 25). For the categorical attribute $y_{\mathcal{F}_{j_i}}$, we update its embedding, in dimension x, by adding the division of the node embedding v_i, which is connected to that categorical attribute in the bipartite graph, and the degree of the categorical attribute in the bipartite graph. Similarly, for the class label l, we add to its embedding, $y_{(C+l)}$, the division of the node embedding v_i, represented by z_{ix}, and the degree that the class label has in the bipartite graph.

Complexity analysis. Lines 4 − 7 in Algorithm 1 compute the contribution provided by the relational types and network structure to the final embedding. Its time complexity is $O(\mathcal{R} \cdot m \cdot d)$, where m is the number of edges. Lines 8 − 17 compute the contribution given by the categorical attributes and the class labels. Here the time complexity is $O(C \cdot d)$. In line 3, we iterate over all nodes; thus, the time complexity of lines 3 − 10 for updating the node coordinates is $O(n(\mathcal{R} \cdot m \cdot d + C \cdot d))$. In line 18, we apply the Gram-Schmidt orthonormalization algorithm with complexity $O(n \cdot d^2)$. In lines 19 − 25, we update the coordinates of the categorical attributes and node labels represented as nodes in the bipartite graphs. It requires $O(n \cdot d(C + Y))$. The total time complexity of Algorithm 1 is $O(I \cdot n \cdot d(\mathcal{R} \cdot m + d + C + Y))$ where I is the number of iterations the algorithm needs in order to converge.

Details about the run-time of SSAMN and other baselines are provided in the Appendix.

5 EXPERIMENTS

We evaluate our approach SSAMN [1] on common setup and on the same datasets that most of the baselines have used in their original papers [13, 15, 26, 30, 41]. For comparison methods, we use the source code published with the papers and set the values for tuning parameters and hyper-parameters as they are set in the original experiments. More details are provided in the Appendix.

Datasets. To compare our algorithm SSAMN with other methods, we use four datasets, which are categorized as AMNs, ACM [30] and IMDB [30], and MNs, FLICKR [24] and DBLP [24]. More information about the datasets is provided in the Appendix.

Baselines. We compare SSAMN against various state-of-the-art methods described in Section 2, namely, node2vec [12], GCN [17], DGI [39], HAN [41], GATNE [7], DMGI [30], DMGC [24], HDMI [15], IGNN [13] and SSDCM [26]. We compare our proposed approach with those baselines that have shown the best performances for each network category; therefore, some related works are not included in the conducted experiments.

Evaluation. We use a random sampling strategy to split the nodes into the train, validation, and test sets. For the ACM and IMDB datasets we use the same number of samples for each set as in [13, 15, 26, 30, 41]. Similarly, for the other two datasets, FLICKR and DBLP, we set 20% of labeled nodes for training, 10% for validation, and the rest for testing. For fair comparisons, we randomly split our datasets 5 times each. Our proposed approach, SSAMN, for the node classification task applies the logistic regression classifier with 10 cross-fold validation and subsequently is evaluated on the test set. As an evaluation metric for the classification task, we compute Micro-F1 and Macro-F1 scores. Results for the node classification task are reported in Table 1. To evaluate our algorithm for the node clustering task, we apply K-Means on the final node embeddings of the test set by setting K to the number of clusters and K-Means++ for initialization. We run it 100 times and report the average results. We apply the same method for all baselines too. For the clustering

[1]We provide our code here: https://gitlab.cs.univie.ac.at/yllis19cs/ssamn/

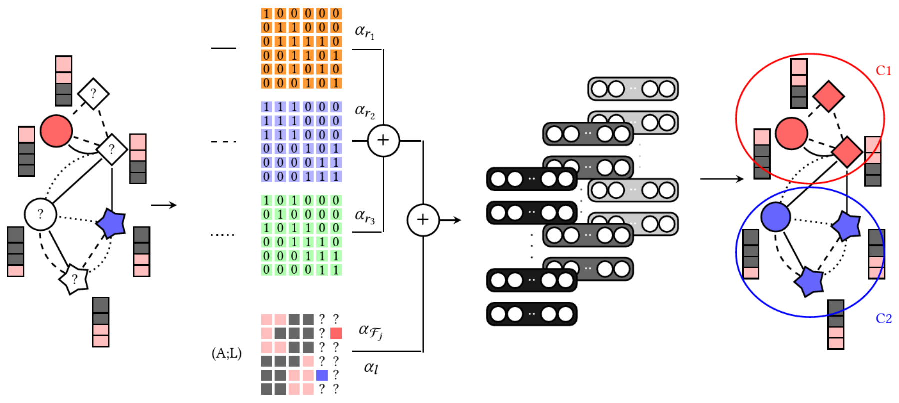


Figure 3: The framework of SSAMN. The input network consists of different types of nodes (circles, stars, diamonds), different relation types (bold, dashed, and dotted lines), node attributes (boxes next to nodes), and node labels. The data is represented as adjacency matrices corresponding to relation types and an attribute-label matrix that holds node attributes and class label information. Each relation type is characterized by a weighting factor, and two additional weighting factors correspond to categorical node attributes and class labels. Then SSAMN embeds nodes, node attributes, and node labels to the same vector space and outputs the final embeddings. These embeddings can be used to classify or cluster the nodes into similar groups (C_1 and C_2) or visualize them.

task, we compute Normalized Mutual Information (NMI) [14] and Adjusted Rand Index (ARI) [14], results are reported in Table 1. Additionally, for an extensive evaluation of the node clustering task, we use the NR-KMeans algorithm [25]; more information is provided in the dimensionality representation paragraph. For both tasks, node classification, and node clustering, we report the average results on the test set over 5 random sets for each dataset.

Our parameter setting. For our method, regarding our only parameter, dimensionality, we set it $d = 32$ for AMNs and $d = 128$ for MNs. As for the MNs, we do not have additional information provided by node attributes; therefore, we use more dimensions to capture a more enhanced representation of the data in the low-dimensional space. One of the benefits of our method is that it has only one parameter, dimensionality, compared to baseline methods such as [17, 41], as mentioned in [21], which need to train a large number of parameters as well as a long training time. We use early stopping criteria with the patience of 20 (extra-iterations), i.e., we stop training if the objective function value does not decrease for 20 consecutive iterations.

Performance Analysis. Results in Table 1 show that methods designed for AMNs do not perform with the same efficiency when they are applied at least to one of MNs, FLICKR, or DBLP, compared to when they are applied to AMNs, ACM, and DBLP. Therefore, we perform the node clustering task only on AMNs, and the results are shown in Table 1. Overall, we observe that our proposed approach, SSAMN, outperforms all baselines for all datasets for both tasks.

Regarding the node classification task, node2vec, GCN, DGI, and DMGC do not perform well, and their performance is not as good as the other baselines due to the fact that they disregard multi-relational edge types, node attributes, or both. DGI can preserve the cluster structure better and therefore is one of the best performers for the clustering task. We observe that methods, which lack the mechanism to handle node attributes, are better performers for MNs than AMNs. Their performance emphasizes the importance of node attributes in the final embeddings.

Most AMN embedding methods (HAN, DMGI, DMGI attn, HDMI, and SSDCM) show very competitive performance and achieve good results on all AMNs, but their performance drops for at least one of the MNs. This is due to the fact that the weight or attention mechanisms constructed by these methods for node attributes are, by default giving high importance (weight) to them and can not be adjusted if node attributes are missing. Surprisingly, GATNE is among the poorest performers regarding both tasks, even though it uses base embeddings and edge embeddings to capture the influential factors between different edge types. We observe that the best performers from baselines for both AMNs, for both tasks, are the methods that are designed to maximize the Mutual Information, DMGI, and HDMI. The difference between these two approaches is that the DMGI approach applies a regularization strategy that jointly integrates the embeddings from different relation types by reusing the negative node representations used for learning the discriminator weights, which shows to be more helpful when the training set is smaller, in case of the DBLP dataset. On the other hand, the HDMI approach emphasizes the dependence between node embedding and node attributes in multiplex networks by using a joint supervision signal that employs high-order mutual information. On the other hand, the performance of the IGNN approach is very consistent and competitive, as it achieves good performance for both categories of networks and tasks. The reason is that IGNN captures long-range dependencies in networks based on a fixed-point equilibrium equation facilitated using the Perron-Frobenius theory to formulate well-posedness conditions. Interestingly, the SSDCM approach observes a well-defined structure and more competitive performance only for the FLICKR dataset, even though it has a joint node and cluster representation for multiplex networks. The lack of node attribute embeddings in the joint representation affects its performance.

Overall, our approach, SSAMN, is the best performer for all datasets, showing that it can perform well on small AMN, such as IMDB, and larger datasets, such as ACM, DBLP, and FLICKR.

Table 1: Node classification performance on test data for AMNs and MNs. Node clustering performance on test data for AMNs. (Bold indicates the best result, while underline indicates the second best.)

Dataset	ACM				IMDB				FLICKR		DBLP	
Metric	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	Macro-F1	Micro-F1
node2vec	0.721	0.723	0.375	0.376	0.446	0.48	0.06	0.05	0.96	0.961	0.75	0.751
GCN	0.727	0.733	0.484	0.441	0.61	0.615	0.125	0.127	0.95	0.951	0.816	0.819
DGI	0.784	0.803	0.654	0.695	0.37	0.465	0.166	0.14	0.292	0.421	0.442	0.442
DMGC	0.614	0.691	0.502	0.452	0.659	0.66	0.078	0.071	0.335	0.335	0.745	0.756
HAN	0.917	0.916	0.673	0.706	0.64	0.645	0.193	0.172	0.891	0.894	0.605	0.604
GATNE	0.622	0.702	0.54	0.52	0.497	0.5641	0.05	0.051	0.808	0.811	0.696	0.745
DMGI	0.879	0.889	0.598	0.597	0.678	0.68	0.202	0.201	0.926	0.929	0.578	0.58
DMGI attn	0.898	0.898	0.611	0.643	0.655	0.659	0.191	0.195	0.914	0.917	0.465	0.47
HDMI	0.935	0.933	0.671	0.707	0.651	0.649	0.176	0.157	0.874	0.88	0.605	0.6
IGNN	0.895	0.894	0.651	0.703	0.643	0.644	0.196	**0.213**	0.973	0.974	0.822	0.825
SSDCM	0.915	0.914	0.524	0.475	0.671	0.672	0.193	0.198	0.959	0.96	0.338	0.351
SSAMN	**0.936**	**0.935**	**0.682**	**0.74**	**0.695**	**0.694**	**0.209**	0.162	**0.978**	**0.978**	**0.847**	**0.845**

Thus, we can also note that our method outperforms other baselines for two categories of networks: AMNs and MNs. The main reason behind its performance is that embedding node attributes and class labels in the same vector space with node embeddings exposes the power of joint embedding by improving the final representation after each iteration until convergence. An important fact to mention is that our proposed approach outperforms all baselines in both tasks; this implies that the adaptation of laplacian eigenmaps for our embeddings is not beneficial only for unsupervised learning, as it was designed. The idea to combine it with the power of homogeneity analysis enables SSAMN to be applicable in a semi-supervised learning fashion, too. Thus, our method outputs an embedding that pushes similar objects, nodes, node attributes, and class labels closer to each other. However, if objects are dissimilar, such a case can be for a labeled node and other class labels to which it does not belong, then our method pushes them apart, as noted in Algorithm 1. Also, an essential aspect of SSAMN is the computation of the weighting factors, which enables it to consider each relation type equally important and correctly measure the importance of each node attribute and class label in the constructed bipartite graphs. Thus, we assign appropriate weights to different layers, even with layers that have distinct connectivity patterns, such as the layers of ACM and DBLP datasets. The weighting schema applied by our algorithm enables us to use laplacian eigenmaps and homogeneity analysis techniques for AMNs and MNs. Therefore, we emphasize the vital role of features on node embeddings, as noted in Table 2 and Figure 5. We provide more details in the Appendix.

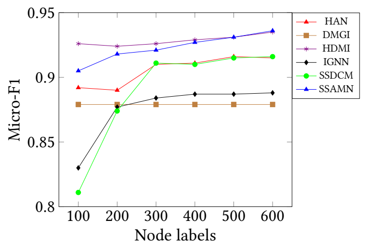
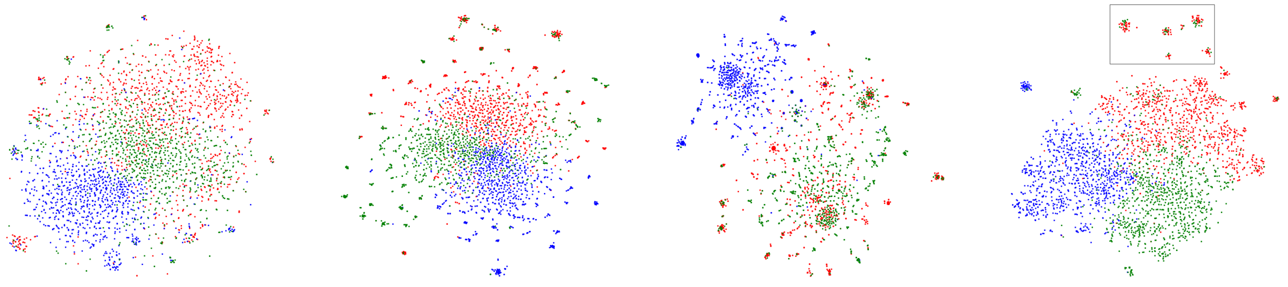


Figure 4: Analysis of the effect of node labels on the ACM dataset.

Effect of node labels. In Figure 4, we show the impact of node labels used for training on our proposed approach and baseline approaches performance for the ACM dataset. We consider the best semi-supervised performers designed for AMNs. The HDMI approach has an advantage for the first two runs when we use up to 400 node labels, and we observe that the performance does not improve reciprocally to the increasing size of the training set. A similar case is reported for the DMGI approach. The SSDCM, IGNN, and HAN performance improves only on the first three runs, but it stays almost the same, even though the training set size increases. On the other hand, our proposed approach has an advantage over semi-supervised baselines for the first four runs, and then when the training set size is increased, we outperform the HDMI approach too. One of the main features of semi-supervised methods is that performance improves as the size of the training set increases; this applies to our proposed approach, and at the same time, it outperforms alternative methods.

Dimensionality representation. To measure the expressivity of our method, we compare it with the best performer from baselines, HDMI [15]. We use the NR-Kmeans algorithm [25] to analyze the embeddings obtained by SSAMN and HDMI for the ACM dataset. The NR-Kmeans algorithm finds $k - 1$ dimensions as a subspace of the high-dimensional space for alternative clusterings, where k is the number of classes. Therefore, by applying NR-Kmeans, we can understand if a method is able to capture more information even when the embedding is represented in very low-dimensional space, i.e., in two dimensions. In Figure 6, we show the visualization of the most representative subspaces for the ACM dataset, which has three class labels; therefore, NR-Kmeans selects two most representative dimensions for that clustering. Thus, we observe that by using node embeddings with two dimensions, we are able to have a better representation than our best competitor for this dataset, HDMI. The accuracy measured by NMI [14] using only two dimensions shows that the score for the HDMI approach is 0.37 and for our proposed approach is 0.64. Thus, the result confirms that we are able to capture more information in fewer dimensions. Therefore, embeddings obtained by our proposed approach are more powerful and expressive. Here we note that for our proposed approach, the most enriched dimensions are very few first dimensions, and



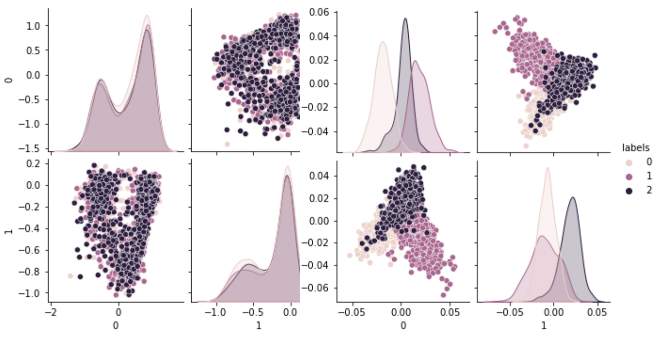
(a) PSP + Attributes. (b) PAP + Attributes. (c) PAP + PSP. (d) PAP + PSP + Attributes.

Figure 5: Visualisation of the node embedding on the ACM dataset, which consist of two layers: PSP and PAP. The three different colors represent the classes of the nodes. Nodes in the black box represent outliers (papers written by the same authors or have the same subject).

this correlates to the embeddings obtained by the Laplacian Eigenmaps for single graphs, which inspired our approach. Thus, for the node clustering task, we evaluate SSAMN using only the first 8 dimensions for the ACM dataset and 2 dimensions for the IMDB dataset.

Ablation study. In Section 4, we stated that all layers are equally important; Table 2 shows the benefits of considering all layers equally important. We can note that the accuracy of our approach on each layer, when considered separately, is similar. However, when we combine information from all layers and use them simultaneously, then the overall performance improves. Also, it illustrates that the most important aspect is to include the information that node attributes provide for the dataset, as the performance diminishes the most in terms of Macro-F1 and Micro-F1 scores when node attributes are not considered in the embeddings obtained by our method. Therefore, the embedding of node attributes inspired by Homogeneity Analysis embraces major information within the final node embeddings.

In Figure 5, the node embeddings are visualized by using the t-sne [37] approach, which validates the importance of all available information in the final node embeddings of SSAMN. Figures 5a and 5b show that the structure of node embeddings is not well defined. Slightly better embedding is preserved when the PSP layer with node attributes is considered. In Figure 5c, we use the information from both layers, but we do not consider node attributes, then the embedding degrades. In the final visualization, in Figure 5d, when we include all available information, we have a more defined cluster structure.



(a) HDMI approach. (b) Our approach, SSAMN.

Figure 6: Visualisation of the most representative subspaces for the clustering of the ACM dataset.

Table 2: Ablation study on the node classification task for attributed multiplex networks. MaF1 and MiF1 denote Macro-F1 and Micro-F1 scores. Sum denotes the sum weight for each node ($sum_i(\mathcal{A}, \mathcal{F}, \mathcal{L})$). A denotes node attributes. SSMN denotes the SSAMN version, which does not consider node attributes.

Dataset	ACM				IMDB			
Layer	PSP		PAP		MAM		MDM	
Metric	MaF1	MiF1	MaF1	MiF1	MaF1	MiF1	MaF1	MiF1
-	0.609	0.601	0.574	0.585	0.32	0.321	0.28	0.304
Sum	0.639	0.661	0.585	0.592	0.347	0.351	0.312	0.364
A	0.865	0.864	0.825	0.8425	0.558	0.554	0.539	0.538
A + Sum	**0.877**	**0.876**	**0.845**	**0.845**	**0.568**	**0.565**	**0.549**	**0.546**

Metric	MaF1	MiF1	MaFi	MiF1
SSMN	0.729	0.728	0.471	0.518
$\alpha_{r_1} = \alpha_{r_2}$	0.894	0.893	0.599	0.595
SSAMN	**0.936**	**0.935**	**0.695**	**0.694**

6 CONCLUSION AND FUTURE WORK

In this work, we propose SSAMN, a Semi-supervised Embedding method for Attributed Multiplex Networks that provides us with a low-dimensional space representation. SSAMN exploits techniques from Spectral Embedding [3] and Homogeneity Analysis [9] to obtain embeddings of nodes, node attributes, and node labels. Results of conducted experiments show that SSAMN outperforms state-of-the-art methods for tasks such as semi-supervised node classification and node clustering.

As future work, we plan to exploit different initialization methods such as eigenvectors of the supra-adjacency matrix [10] inspired by the authors of [18] that apply eigenvectors as positional encodings for graph transformers with spectral attention. It would also be interesting to investigate the p-Laplacian eigenvectors [1, 5, 11, 22]. Moreover, SSAMN can be extended to be applied to hypergraphs where a hyperedge can be considered as an attribute of a node.

REFERENCES

[1] Saïd Amghibech. 2003. Eigenvalues of the Discrete p-Laplacian for Graphs. *Ars Comb.* 67 (2003).

[2] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Unsupervised Inductive Graph-Level Representation Learning via Graph-Graph Proximity. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 1988–1994.

[3] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*. MIT Press, 585–591.

[4] Ginestra Bianconi. 2018. *Multilayer Networks: Structure and Function.* Oxford University Press. https://doi.org/10.1093/oso/9780198753919.001.0001

[5] Thomas Bühler and Matthias Hein. 2009. Spectral Clustering Based on the Graph P-Laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 81–88. https://doi.org/10.1145/1553374.1553385

[6] Leonid A. Bunimovich, Chi-Jen Wang, Seokjoo Chae, and Benjamin Z. Webb. 2018. Uncovering Hierarchical Structure in Social Networks Using Isospectral Reductions. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 1199–1206. https://doi.org/10.1109/ASONAM.2018.8508506

[7] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network *(KDD '19)*. 1358–1368.

[8] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A Survey on Network Embedding. *IEEE Transactions on Knowledge and Data Engineering* 31, 5 (2019), 833–852. https://doi.org/10.1109/TKDE.2018.2849727

[9] Jan de Leeuw and Patrick Mair. 2009. Gifi Methods for Optimal Scaling in R: The Package homals. *Journal of Statistical Software, Articles* 31, 4 (2009), 1–21.

[10] Francisco Aparecido Rodrigues Yamir Moreno Emanuele Cozzo, Guilherme Ferraz de Arruda. 2018. *Multiplex Networks.* Springer Cham. https://doi.org/10.1007/978-3-319-92255-3

[11] Guoji Fu, Peilin Zhao, and Yatao Bian. 2022. p-Laplacian Based Graph Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 6878–6917. https://proceedings.mlr.press/v162/fu22e.html

[12] 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* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[13] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. 2020. Implicit Graph Neural 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., 11984–11995.

[14] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification* 2 (12 1985), 193–218. Issue 1. https://doi.org/10.1007/BF01908075

[15] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. HDMI: High-order Deep Multiplex Infomax. *Proceedings of the Web Conference 2021* (Apr 2021). https://doi.org/10.1145/3442381.3449971

[16] I. T. Jolliffe. 1986. *Principal Components in Regression Analysis.* Springer New York, New York, NY, 129–155.

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

[18] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=huAdB-Tj4yG

[19] 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)*. 134–141. https://doi.org/10.1109/ICDMW.2017.23

[20] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*. 1392–1403.

[21] Zhijun Liu, Chao Huang, Yanwei Yu, Baode Fan, and Junyu Dong. 2020. Fast Attributed Multiplex Heterogeneous Network Embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 995–1004. https://doi.org/10.1145/3340531.3411944

[22] Dijun Luo, Heng Huang, Chris Ding, and Feiping Nie. 2010. On the eigenvectors of p-Laplacian. *Machine Learning* 81 (2010). https://doi.org/10.1007/s10994-010-5201-z

[23] Dongsheng Luo, Jingchao Ni, Suhang Wang, Yuchen Bian, Xiong Yu, and Xiang Zhang. 2020. Deep Multi-Graph Clustering via Attentive Cross-Graph Association. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 393–401. https://doi.org/10.1145/3336191.3371806

[24] Dongsheng Luo, Jingchao Ni, Suhang Wang, Yuchen Bian, Xiong Yu, and Xiang Zhang. 2020. Deep Multi-Graph Clustering via Attentive Cross-Graph Association *(WSDM '20)*. Association for Computing Machinery, 393–401.

[25] Dominik Mautz, Wei Ye, Claudia Plant, and Christian Böhm. 2020. Non-Redundant Subspace Clusterings with Nr-Kmeans and Nr-DipMeans. *ACM Trans. Knowl. Discov. Data* 14, 5, Article 55 (jun 2020), 24 pages. https://doi.org/10.1145/3385652

[26] Anasua Mitra, Priyesh Vijayan, Ranbir Sanasam, Diganta Goswami, Srinivasan Parthasarathy, and Balaraman Ravindran. 2021. *Semi-Supervised Deep Learning for Multiplex Networks.* Association for Computing Machinery, 1234–1244.

[27] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. 2022. Simple Unsupervised Graph Representation Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 7 (Jun. 2022), 7797–7805. https://doi.org/10.1609/aaai.v36i7.20748

[28] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *NIPS'01*. 849–856.

[29] Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, and Xiang Zhang. 2018. Co-Regularized Deep Multi-Network Embedding. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 469–478. https://doi.org/10.1145/3178876.3186113

[30] Chanyoung Park, D. Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*.

[31] 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* (New York, New York, USA) *(KDD '14)*. ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[32] Sudeep Pushpakom, Francesco Iorio, Patrick A. Eyers, K. Jane Escott, Shirley Hopper, Andrew Wells, Andrew Doig, Tim Guilliams, Joanna Latimer, Christine McNamee, Alan Norris, Philippe Sanseau, David Cavalla, and Munir Pirmohamed. 2019. Drug repurposing: progress, challenges and recommendations. *Nature Reviews Drug Discovery* 18 (2019). https://doi.org/10.1038/nrd.2018.168

[33] David Hilbert (auth.) Richard Courant. 1968. *Methoden der Mathematischen Physik I.* Springer Berlin Heidelberg.

[34] Camilo Ruiz, Marinka Zitnik, and Jure Leskovec. 2021. Identification of disease treatment mechanisms through the multiscale interactome. *Nature Communications* 12 (2021). https://doi.org/10.1038/s41467-021-21770-8

[35] 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 & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1431–1440. https://doi.org/10.1145/3447548.3467381

[36] 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* (Florence, Italy) *(WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1067–1077. https://doi.org/10.1145/2736277.2741093

[37] L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[38] Barch DM Behrens TE Yacoub E Ugurbil K Van Essen DC, Smith SM. 2013. The WU-Minn Human Connectome Project: an overview. *Neuroimage* (2013). https://doi.org/10.1016/j.neuroimage.2013.05.041

[39] Petar Veličković, Wiliam Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.

[40] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. 2022. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *IEEE Transactions on Big Data* (2022), 1–1. https://doi.org/10.1109/TBDATA.2022.3177455

[41] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network *(WWW '19)*. 2022–2032.

[42] Wei Ye, Sebastian Goebl, Claudia Plant, and Christian Böhm. 2016. FUSE: Full Spectral Clustering. In *KDD*. ACM, 1985–1994.

[43] Wei Ye, Linfei Zhou, Xin Sun, Claudia Plant, and Christian Böhm. 2017. Attributed Graph Clustering with Unimodal Normalized Cut. In *ECML/PKDD*.

[44] Lorenzo Zangari, Roberto Interdonato, Antonio Calió, and Andrea Tagarelli. 2021. Graph convolutional and attention models for entity classification in multilayer networks. *Applied Network Science* 6 (2021). https://doi.org/10.1007/s41109-021-00420-4

[45] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.* International Joint Conferences on Artificial Intelligence Organization, 3082–3088. https:

//doi.org/10.24963/ijcai.2018/428

[46] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In *Proc. of IJCAI'18*. ijcai.org, 3155–3161.

[47] Jianan Zhao, Xiao Wang, Chuan Shi, Zekuan Liu, and Yanfang Ye. 2020. Network Schema Preserving Heterogeneous Information Network Embedding. In *IJCAI'20*, Christian Bessiere (Ed.). 1366–1372. https://doi.org/10.24963/ijcai.2020/190 Main track.

[48] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

[49] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 34, 13 (06 2018), i457–i466.