Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 32(4): 2772–2788.
DOI: 10.3934/era.2024125
Received: 29 December 2023
Revised: 29 February 2024
Accepted: 04 March 2024
Published: 03 April 2024

Research article

A one-step graph clustering method on heterogeneous graphs via variational graph embedding

Chuang Ma[*] **and Helong Xia**

School of Software Engineering, Chongqing University of Posts and Telecommunications, No. 2 Chongwen Road, Nanshan Street, Chongqing, China

* **Correspondence:** Email: s211201026@stu.cqupt.edu.cn; Tel: +8613258185529.

Abstract: Graph clustering is one of the fundamental tasks in graph data mining, with significant applications in social networks and recommendation systems. Traditional methods for clustering heterogeneous graphs typically involve obtaining node representations as a preliminary step, followed by the application of clustering algorithms to achieve the final clustering results. However, this two-step approach leads to a disconnection between the optimization of node representation and the clustering process, making it challenging to achieve optimal results. In this paper, we propose a graph clustering approach specifically designed for heterogeneous graphs that unifies the optimization of node representation and the clustering process for nodes in a heterogeneous graph. We assume that the relationships between different meta-paths in the heterogeneous graph are mutually independent. By maximizing the joint probability of meta-paths and nodes, we derive the optimization objective through variational methods. Finally, we employ backpropagation and reparameterization techniques to optimize this objective and thereby achieve the desired clustering results. Experiments conducted on multiple real heterogeneous datasets demonstrate that the proposed method is competitive with existing methods.

Keywords: heterogeneous graph; graph cluster; variational graph autoencoder; graph neural network

1. Introduction

Graph clustering is a fundamental task in graph data mining, with the aim of grouping all nodes in a graph into clusters. Clustering usually results in a high similarity of nodes within clusters and a low similarity between clusters. It is often applied in recommendation systems and social network analysis. By performing graph clustering on social networks, potential associations and similarities between users can be discovered. These associations can be utilized to improve the performance of recommendation systems, increasing the accuracy of understanding of user interests and thus enhancing the personalization of recommendations. However, many existing graph clustering algorithms suffer from shortcomings, since they typically follow a two-step process. They first obtain node embeddings in a graph, and then use algorithms such as K-means and Gaussian mixture models for clustering. However, such a two-step approach often results in suboptimal performance, mainly because of node embeddings not being cluster-oriented, i.e., designed for a specific clustering task. There exists a gap between node embeddings and clustering [1–3].

Heterogeneous graph structures are widely used to model complex relationships between entities in different domains, such as social networks [4–6], and other domains [7–10]. Figure 1 illustrates a heterogeneous graph composed of three node types (author, paper, conference), three types of edges (red = author wrote a paper, green = conference included the paper, blue = two papers cited each other), and two types of meta-paths (Author–Paper–Author = two authors who wrote are co-authors, Paper–Conference–Paper = two papers included in the same conference). Owing to the diverse types of nodes and edges, complex semantic relationships, etc., attempts to apply many existing graph clustering algorithms directly to heterogeneous graphs face challenges [11–15].

Addressing the complex characteristics of heterogeneous graphs and the shortcomings of two-step clustering, we propose a one-step clustering method for heterogeneous graphs. First, this method utilizes meta-paths to extract semantic relationships in heterogeneous graphs and assumes independence between each meta-path to decompose the complexity of the heterogeneous graph. Second, it maximizes the joint probability of meta-paths and nodes and derives an optimization objective, namely, the evidence lower bound (ELBO), using variational methods. Finally, it continuously optimizes its own parameters using backpropagation and reparameterization techniques to minimize the optimization objective and achieve clustering effects. Experiments on multiple real heterogeneous datasets demonstrate that the proposed method is competitive with existing methods.

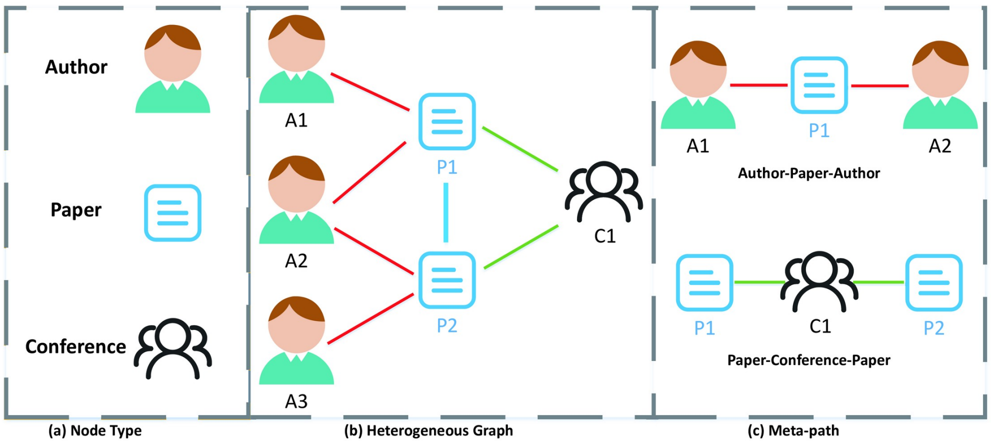

Figure 1. Academic heterogeneous network containing three types of nodes and three types of edges. Different combinations of nodes and edges constitute different meta-paths on the right side, each representing a different meaning.

2. Related work

2.1. Graph clustering method based on homogeneous graphs

Homogeneous graphs contain only one type of nodes and edges, and most early graph clustering methods were developed for homogeneous graphs. For example, early walk-based methods [16, 17] captured structural similarities by walking and then learning node embedding by language models. They had scalability, but could not exploit the properties of nodes. With the development of deep learning in recent years, graph neural networks have also developed rapidly [18–20]. There have been many studies of graph neural networks with a framework of auto-encoders. For example, graph auto-encoders are widely used in various fields, since they extend auto-encoders to graph structures, which achieve graph embedding by reconstructing the graph structure. Variational graph auto-encoders represent a further improvement by introducing hidden embedding to obtain node embedding by optimizing the ELBO [21]. Salehi and Davulcu [22] reconstructed the node features and graph structure by stacking multiple encoders with attention while reversing the encoders in the decoding phase. Graph contrastive learning is a typical approach to learning graph embedding by maximizing the mutual information between the graph-level representation and the node representation [23]. However, all the above methods first obtain the node embedding and then perform clustering using traditional algorithms such as K-means, and thus they are not clustering-oriented. Many current studies have been concerned with joint optimization of the learning graph embedding and graph clustering [2,24,25] by reducing the distance between the constructed auxiliary distributions. However, in a recent study [26], it was found that with such a self-supervised strategy, the problem of distribution drift arose, leading to performance degradation.

2.2. Graph clustering method based on heterogeneous graphs

Heterogeneous graphs have multiple types of nodes and edges, containing rich structural and semantic information. Early approaches to heterogeneous graph clustering captured semantic information through meta-path walking [11, 27, 28]. In the context of graph neural networks, many heterogeneous graph clustering studies have been based on meta-paths [29–31]. Fu et al. [30] added virtual edges based on meta paths directly to the original graph to capture latent semantic information. Zheng et al. [31] showed that previous studies on heterogeneous graphs, in which the direct fusion of individual meta-paths ignored the incompatibility of different meta-paths, made it difficult to reflect the true relationship between nodes. However, meta-paths are a priori knowledge and need to be selected by experts, which will consume a considerable amount of human resources. With the development of graph contrastive learning, there have been a number of attempts to apply this to heterogeneous graphs [32–34], with the node embedding being obtained by maximizing the mutual information between the graph-level representation and the node representation. Again, most of these methods require clustering using classical algorithms. Perossi et al. [16] combined variational and contrastive modules to learn clustering-oriented node embedding from heterogeneous graphs. However, this method is parameter-sensitive and requires time-consuming pretraining.

2.3. Limitations of previous methods and the contributions of our proposed method

Graph clustering methods have been extensively studied in both homogeneous and heterogeneous graph settings, each of which has its own unique challenges and limitations. The most significant limitation is the lack of direct optimization methods for the clustering process, particularly in the case of heterogeneous graphs with complex heterogeneous relationships. In our proposed method, we are able to jointly learn the representations of nodes and optimize the clustering process directly for heterogeneous graphs, rather than splitting them into two separate steps. This approach eliminates the gap between the two processes.

3. Preliminaries

This section introduces some definitions and notation used in the rest of the paper.

Definition 1. Heterogeneous graph. A heterogeneous graph is defined as $G = (V, E, A, R, \phi, \beta)$, where V and E are the sets of nodes and edges, respectively, A and R are the sets of node-types and edge-types, respectively, $\phi : V \to A$ is a node-type mapping function, $\beta : E \to R$ is a edge-type mapping function, and the inequality $|A| + |R| > 2$ is satisfied.

Definition 2. Meta-path. A meta-path of length m is usually defined as the following sequence of forms: $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{m-1}} A_m$, where $A_i \in A$ and $R_i \in R$. It describes the combinatorial relation $R_1 \circ R_2 \circ \cdots \circ R_{m-1}$ from A_1 to A_m, where $\circ$ denotes the constituent operation or relation.

For example, Figure 1(c) contains two kinds of meta-paths

$$\text{Author} \xrightarrow{\text{Write}} \text{Paper} \xrightarrow{\text{Write}^{-1}} \text{Author} \quad \text{(APA)},$$
$$\text{Paper} \xrightarrow{\text{Publish}} \text{Conference} \xrightarrow{\text{Publish}^{-1}} \text{Paper} \quad \text{(PCP)}.$$

Different meta-paths contain different semantic relations, and in our later sections we will use these meta-paths to encode node embedding. In this paper, M^m denotes the mth meta-path, and M^m_{ij} represents the connection relationship between node i and node j of the mth meta-path. X and E^m denote the input features of the nodes and the edge sets of the mth meta-path.

4. Our method

This section details our proposed method. In Section 4.1, we derive the objective function (4.12) by maximizing $\log p(M, X)$, for which several assumptions will be utilized. In Sections 4.2 and 4.3, we focus on presenting the specific implementations of each component of the objective function (4.12). For instance, we employ graph neural networks to realize the expression $z^m_i \sim q(z^m_i|X, M)$, and provide a concrete implementation of the use of a discrete uniform distribution as the distribution of $p(c_i)$.

4.1. Derivation of the objective function

The random variables z_i and c_i denote the latent embedding and cluster assignment, respectively, of node x_i, and z_i is constituted by z^m_i in each meta-path. A node x_i corresponds to a continuous latent embedding z_i and a discrete cluster c_i, where the latent embedding z_i is related to the latent embedding in the m meta-paths, i.e., $z_i = [z_i^1, z_i^2, \dots, z_i^m]$. The maximum log-likelihood of M and X is given by

$$\log p(M, X) = \log \int \sum_c p(X, M, Z, c)\, dZ, \tag{4.1}$$

where $Z = [z_1, z_2, \dots, z_N]$, $c = [c_1, c_2, \dots, c_N]$. Equation (4.1) indicates that the data distribution in higher dimensions can be handled by Z and c. To solve for this log-likelihood, we use the variational posterior $q(Z, c \mid X, M)$ to automatically learn the latent embeddings Z and c. Then, we use Jason's inequality:

$$\begin{aligned}\log p(M, X) &= \log \int \sum_c \frac{p(X, M, Z, c)}{q(Z, c \mid X, M)} q(Z, c \mid X, M)\, dZ \\ &= \log \left[E_{q(Z,c|X,M)} \frac{p(X, M, Z, c)}{q(Z, c \mid X, M)} \right] \\ &\geq E_{q(Z,c|X,M)} \log \left[\frac{p(X, M, Z, c)}{q(Z, c \mid X, M)} \right] \\ &= L_{\text{low}}(M, X). \end{aligned} \tag{4.2}$$

After the ELBO has been obtained, a Bayesian approach will be used to factorize it:

$$p(X, M, Z, c) = p(c)p(Z \mid c)p(X, M \mid Z, c). \tag{4.3}$$

This represents a generative process, picking a clustering c from the distribution $P(c)$, sampling the latent embedding Z by c, and then reconstructing the node features and the connection relationship between nodes. Meanwhile, $q(Z, c \mid X, M)$ is decomposed into $q(Z \mid X, M)q(c \mid Z)$ to obtain the ELBO:

$$\begin{aligned} L_{\text{low}}(M, X) = & -E_{q(Z|M,X)}[D_{KL}(q(c \mid Z) \parallel p(c))] \\ & -E_{q(c|Z)}[D_{KL}(q(Z \mid M, X) \parallel p(Z \mid c))] \\ & +E_{q(Z,c|M,X)}[\log p(X, M \mid Z, c)]. \end{aligned} \tag{4.4}$$

To decompose this equation, we make several assumptions:

1) The variables $z_i^m \mid c_i$ are i.i.d.
2) The variables c_i are also i.i.d.
3) $p(X, M \mid Z, c)$ satisfies a mean field distribution and can therefore be decomposed into $p(X, M \mid Z, c) = p(M \mid Z, c)p(X \mid Z, c)$, where $p(X \mid Z, c) = p(X \mid Z)$.
4) Each meta-path M^m is independent of the others, and the generation of M_{ij}^m is related to z_i^m, z_j^m, c_i, c_j. That is, the reconstruction of the mth meta-path is related to the latent embedding and the clustering under the meta-path.

Under these four assumptions and the decomposition process corresponding to Eq (4.3) is as follows:

$$p(c) = \prod_{i=1}^{N} p(c_i), \tag{4.5}$$

$$p(Z \mid c) = \prod_{i=1}^{N} p(z_i^1, z_i^2, \ldots, z_i^m \mid c_i) = \prod_{i=1}^{N} \prod_{m} p(z_i^m \mid c_i), \tag{4.6}$$

$$p(M \mid Z, c) = \prod_{m} \prod_{(i,j)\in E^m} p(M_{ij}^m \mid z_i^m, z_j^m, c_i, c_j), \tag{4.7}$$

$$p(X \mid Z) = \prod_{i=1}^{N} p(x_i \mid z_i). \tag{4.8}$$

The nodes in different meta-paths are independent, and the latent embedding z_i^m of each meta-path satisfies a Gaussian distribution. We need to determine the parameters of the posterior distribution $q(z_i^m \mid X, M)$:

$$q(Z \mid X, M) = \prod_{i=1}^{N} q(z_i \mid X, M), \tag{4.9}$$

$$q(z_i \mid X, M) = \prod_{m} q(z_i^m \mid X, M), \tag{4.10}$$

$$q(z_i^m \mid X, M) = N(z_i^m \mid \mu_i^m, \mathrm{diag}(\delta_i^{m,2})). \tag{4.11}$$

The parameters μ_i^m and δ_i^m are the mean and variance of the Gaussian distribution of the node v_i in the mth meta-path. They can be obtained by using any suitable graph neural network, such as RGCN [35] or HGAT [36].

By decomposing Eq (4.4) using Eqs (4.5)–(4.11), we finally arrive at the following objective function, which is the ELBO:

$$\begin{aligned} L_{\text{low}}(M, X) \approx\ & -\sum_{i=1}^{N} D_{KL}(q(c_i \mid z_i) \parallel p(c_i)) \\ & -\sum_{i=1}^{N} \sum_{m} D_{KL}(q(z_i^m \mid X, M) \parallel p(z_i^m \mid c_i)) \\ & +\sum_{m} \sum_{(i,j)\in E^m} \log p(M_{ij}^m \mid z_i^m, z_j^m, c_i, c_j) \\ & +\sum_{i=1}^{N} \log p(x_i \mid z_i). \end{aligned} \tag{4.12}$$

The ELBO contains four parts. The first part ensures that the predicted distribution approximates the prior distribution $p(c)$, which can be specified by a human. The second part ensures that z_i^m aligns with the distribution of c_i, and has a clustering effect. The third part ensures that the reconstruction of the connection relationship of meta-paths by nodes and clusters preserves as much information about the graph as possible. The fourth part corresponds to the reconstruction of the initial nodes' features through the nodes.

4.2. Implementation of $z_i^m \sim q(z_i^m \mid X, M)$

In this subsection, we provide a suitable graph neural network for specific implementation. Given the connection relation M_{ij}^m between nodes v_i and v_j of the meta-path, to use this information more efficiently, we construct a series of neural networks $f^m(X, M) \in f(X, M)$ for each meta-path.

4.2.1. The graph neural network $f^m(X, M)$ corresponding to $q(z_i^m \mid X, M)$

The neural network $f^m(X, M)$ embeds high-dimensional data into low-dimensional data. Inspired by [29], for $f^m(X, M)$, we use node-level encoders taking account of the heterogeneous graph G. We suppose that the input features $x_i^m, x_j^m \in R^{d_{\text{in}}}$ of node v_i and v_j and the output features $h_i^m, h_j^m \in R^{d_{\text{out}}}$. The correlation between nodes v_i and v_j at the mth path M^m is given by

$$e_{ij}^m = \sigma(V_s^m W^m x_i^m + V_r^m W^m x_j^m), \tag{4.13}$$

where $W^m \in R^{d_{\text{out}} \times d_{\text{in}}}$, $V_r^m \in R^{d_{\text{out}}}$, and $V_s^m \in R^{d_{\text{out}}}$ are the trainable parameters at the node-level encoder. Note that the correlation between node v_j and node v_i is different, and the node-level attention preserves the asymmetry of the heterogeneous graph. To normalize e_{ij}^m, we use the softmax function as the normalization function:

$$\alpha_{ij}^m = \frac{\exp(e_{ij}^m)}{\sum_{l \in N_i^m} \exp(e_{il}^m)}, \tag{4.14}$$

where N_i^m denotes the neighboring nodes of node v_i in the meta-path M^m, including itself. For the meta-path M^m, the latent embedding of node v_i generated by the encoder $f^m(X, M)$ is as follows:

$$h_i^m = \sum_{j \in N_i^m} \alpha_{ij}^m W^m x_j^m. \tag{4.15}$$

This encoder is able to capture the semantic information of the corresponding meta-path. So far, we have used a node attention-based graph neural network structure to embed the high-dimensional node information of each meta-path into the continuous feature space.

4.2.2. Obtaining z_i^m with $f^m(X, M)$

With this graph neural network $f^m(X, M)$, the mean μ_i^m and variance δ_i^m of the Gaussian distribution can be calculated. In other words, we utilize $f^m(X, M)$ to obtain μ_i^m and δ_i^m. We then obtain z_i^m using the following reparameterization trick:

$$z_i^m = \mu_i^m + \delta_i^m * \epsilon, \qquad \mu_i^m, \delta_i^{m,2} \sim f^m(X, M), \tag{4.16}$$

where ϵ is sampled from the standard Gaussian distribution.

4.3. Determining the distribution

For each term in $L_{\text{low}}(X, M)$, we need to determine the distribution in order to calculate the value.

4.3.1. Distribution of $p(c_i), q(c_i \mid z_i)$

We choose the discrete uniform distribution as the distribution of $p(c_i)$. For the posterior distribution $q(c_i \mid z_i)$, we first sum $z_i^1, z_i^2, \ldots, z_i^m$ to obtain z_i, and we then use the softmax function to obtain the distribution. In inference, we use $q(c_i \mid z_i)$ to obtain the cluster assignment of node v_i:

$$q(c_i \mid z_i) = \text{softmax}(z_i^T W + b), \tag{4.17}$$

$$z_i = \text{mean}(z_i^1, z_i^2, \ldots, z_i^m), \tag{4.18}$$

where $W \in R^{d_{\text{out}} \times K}$ and $b \in R^K$ are the learnable parameters, and K is the size of the clustering.

4.3.2. Distribution of $p(z_i^m \mid c_i)$

Given the number of clusters K, we introduce the clustering embedding $\{g_1^m, g_2^m, \ldots, g_K^m\}, g_K^m \in R^{d_{\text{out}}}$. We choose $p(z_i^m \mid c_i = k)$ to be a Gaussian distribution with mean g_k^m and variance 1:

$$p(z^m \mid c_i = k) = N(z_i^m \mid g_k^m, 1). \tag{4.19}$$

4.3.3. Distribution of $p(M_{ij}^m \mid z_i^m, z_j^m, c_i, c_j)$

$p(M_{ij}^m \mid z_i^m, z_j^m, c_i, c_j)$ is a Bernoulli distribution that is modeled to maximize the probability of reconstructing the mth meta-path connectivity relationship. It is related to the latent embedding of the current node z and the cluster c:

$$p(M_{ij}^m \mid z_i^m, z_j^m, c_i = L, c_j = R) = \text{Ber}(\mu_{ij}^m), \tag{4.20}$$

$$\mu_{ij}^m = \frac{\sigma((g_L^m)^T z_j^m) + \sigma((g_R^m)^T z_j^m)}{2}. \tag{4.21}$$

The reconstruction function makes similar nodes more similar, and σ is the sigmoid function.

4.3.4. Distribution of $p(x_i \mid z_i)$

The variable $x_i \mid z_i$ satisfies a Gaussian distribution whose mean is related to z_i and whose variance is a constant. The $z_i = [z_i^1, z_i^2, \ldots, z_i^m]$ and z_i^m of different meta-paths are independent of each other, and so

$$p(x_i \mid z_i) = N(x_i \mid \mu_i, \text{diag}(\delta_i^2)), \tag{4.22}$$

$$\mu_i = \sum_m \sum_{j \in N_i^m} \alpha_{ij}^m (W^m)^T z_i^m, \tag{4.23}$$

$$\delta_i^2 = C, \tag{4.24}$$

where C is a constant, and α_{ij}^m and W^m are the attention score and parameter matrix, respectively, of $f^m(X, M)$.

Finally, in order to make our proposed method more accessible to readers, we have added the Pseudocode Algorithm 1 for Eq (4.12).

5. Experiments

In this section, we validate our model with three public datasets and compare it with other methods. The experimental results show the superiority of our method.

5.1. Datasets

The following datasets are used:

- **ACM**: This dataset collects papers (P), authors (A), and subjects (S) from KDD, SIGMOD, SIGCOMM, MobiCOMM, and VLDB as nodes. The keywords of the paper are taken as its feature and the research area as its label, such as data-mining, database, and wireless. We use meta-paths PAP and PSP in the model.

Algorithm 1 Algorithm to implement Eq (4.12)

```
Input: node features matrix $\mathbf{X}$, set of adjacency matrices corresponding to $m$ meta-paths $\mathbf{M} = [M^1, M^2, \dots, M^m]$, set of edges of $m$ meta-paths $E = [E^1, E^2, \dots, E^m]$
Output: loss used for training Loss
 1: Initialize $z \leftarrow [\mathbf{0}^1, \dots, \mathbf{0}^m]$    ▷ initialize $m$ $\mathbf{0}$ to store the result of $q(z^m \mid M, X)$.
 2: Initialize $temp\_kls \leftarrow [\mathbf{0}^1, \dots, \mathbf{0}^m]$
 3: Initialize $\alpha s \leftarrow [\mathbf{0}^1, \dots, \mathbf{0}^m]$    ▷ store the attention scores of $f^m(M, X)$.
 4: Initialize graph encoders $f^1(X, M), f^2(X, M), \dots, f^m(X, M)$
 5: Initialize $structure\_loss \leftarrow 0$
 6: Initialize trainable parameters $W_f$, $W_c$, $b_c$, $uy$ ▷ $uy$ is the randomly initialized embedding of cluster centers, which will be optimized during training.
 7: for $k$ in $1, 2, \dots, m$ do
 8:     $\mu^k, \delta^{k,2}, \alpha^k \leftarrow f^k(\mathbf{M}, \mathbf{X})$
 9:     $z^k \leftarrow \mu^k * \epsilon + \delta^k, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
10:     $temp\_kl \leftarrow -\frac{1}{2}\log\delta^{k,2} + \frac{1}{2}||z^k - uy||^2$
11:     update the $\mathbf{0}^k$ in $z$ to $z^k$
12:     update the $\mathbf{0}^k$ in $temp\_kls$ to $temp\_kl$
13:     update the $\mathbf{0}^k$ in $\alpha s$ to $\alpha^k$
14: end for
15: $feature\_loss \leftarrow \text{MSE}\Big(\sum_{k=1}^{m}\sum_{j\in N_i^k}\alpha_{ij}^k(W_f^k)^T z_i^k, \mathbf{X}\Big)$    ▷ to calculate $\log p(x_i \mid z_i)$ in Eq (4.12)
16: $y \leftarrow \text{softmax}\left((W_c)^T z + b_c\right)$    ▷ implement $q(c_i \mid z_i)$ in Eq (4.17)
17: $cat\_loss \leftarrow reduce\_mean(y * \log(y))$    ▷ to calculate $D_{KL}(q(c_i \mid z_i) \mid\mid p(c_i))$ in Eq (4.12)
18: $kl\_loss \leftarrow reduce\_mean(y * temp\_kls)$    ▷ to calculate $D_{KL}(q(z_i^m \mid X, M) \mid\mid p(z_i^m \mid c_i))$ in Eq (4.12)
19: for $k$ in $1, 2, \dots, m$ do
20:     for connected nodes $(i,j)$ in $E^k$ do
21:         obtain prediction results $L$ and $R$ for $i$ and $j$ through $y$.
22:         obtain clustering center embeddings $g_L^k$ and $g_R^k$ for $L$ and $R$ through $uy$.
23:         obtain embedding representations $z_i^k$ and $z_j^k$ for $i$ and $j$ through $z$.
24:         $structure\_loss += \dfrac{\sigma((g_L^k)^T z_j^k) + \sigma((g_R^k)^T z_j^k)}{2}$    ▷ to calculate $\log p(M_{ij}^m \mid z_i^m, z_j^m, c_i, c_j)$ in Eq (4.12)
25:     end for
26: end for
27: Loss = $cat\_loss + kl\_loss - structure\_loss + feature\_loss$
28: Back-propagate Loss to update parameters in graph encoders, $W_f$, $W_c$, $b_c$ and $uy$.
```

- **DBLP**: This dataset contains papers (P), conferences (C), and authors (A) in several fields of computer science. Researchers are labeled into four fields: database, data mining, machine learning, and information retrieval. Authors are featured as their keywords. We use meta-paths APC and APCPA in the model.
- **FreeBase**:This dataset contains four types of nodes: movie (M), actor (A), director (D), and producer (P). The movie genres are classified as Action, Comedy, and Drama, which are also

used as the labels. We use meta-paths MAM and MDM in the model.

5.2. Baselines

We compare common cluster methods on graphs:

- **KMeans** [35]: This is awidely used clustering algorithm. In this paper, we use node features as its input.
- **MetaPath2Vec** [11]: This method obtains the embedding of a node by randomly walking the specified meta-path on a heterogeneous graph and then making the neighboring nodes more similar by using a language model.
- **GAE** [21]: This method is an auto-encoder on the graph, which obtains node embedding by reconstructing the adjacency matrix of the graph.
- **VGAE** [21]: The method is a variational auto-encoder on the graph, where the embedding of the nodes is obtained by introducing the maximum log-likelihood of the latent embedding optimization.
- **HGATE** [29]: This method models the relationship between meta-paths and nodes by introducing an attention mechanism and reconstructing node features to obtain embeddings of nodes.
- **HDGI** [32]: This is an extension of contrastive learning on heterogeneous graphs, which decreases the similarity of negative sample pairs and increases that of positive sample pairs to obtain the embedding of nodes.
- **HeCo** [36]: This extracts node embedding from the network structure and meta-path perspectives on the basis of traditional graph contrastive learning.
- **VACA-HINE** [3]: This also involves joint learning of clustering with contrastive learning. However, it mainly focuses on learning embeddings.

5.3. Experimental setup and metrics

For the models in the baseline, we perform several experiments and take the best results on each dataset. Our model sets the hidden layer dimension 100 on the DBLP dataset and 128 on the ACM and FreeBase datasets, while setting the learning rate to 0.01, 0.03, and 0.01 respectively. The Adam optimizer is used for optimization. Similar to VAE and VGAE, our method also suffers from posterior collapse. Therefore, to avoid this, we add batch normalization to the model to increase its stability [37]. Compared with other one-step heterogeneous graph clustering methods [16, 38], our method does not require any pretraining, which increases its range of applicability. The overall code is implemented using the DGL framework.

We use several commonly used clustering metrics to validate our results on the datasets: accuracy (ACC), normalized mutual information (NMI), F-score (F1), and adjusted Rand index (ARI).

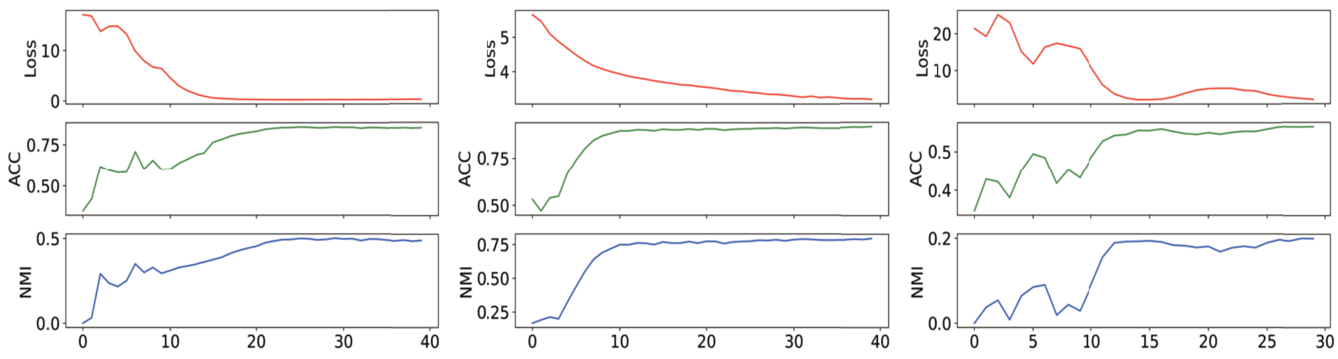

Figure 2. Iterations of objective function, accuracy, and NMI values of our model on three datasets: (a) ACM; (b) DBLP; (c) FreeBase. At the beginning, the values fluctuate, but as the training proceeds, they become smoother until convergence is attained.

Table 1. Experimental results.

		K-means	MP2Vec	GAE	VGAE	HGATE	HDGI	HeCo	VaCA	Our
DBLP	ACC	0.381	0.920	0.794	0.878	0.874	0.860	0.892	0.923	**0.925**
	NMI	0.111	0.758	0.523	0.680	0.676	0.650	0.701	0.764	**0.765**
	ARI	0.670	0.806	0.529	0.714	0.714	0.691	0.751	0.806	**0.820**
	F1	0.313	0.915	0.788	0.870	0.838	0.846	0.882	0.916	**0.920**
ACM	ACC	0.562	0.648	0.847	0.730	0.830	0.767	0.856	0.855	**0.857**
	NMI	0.281	0.378	0.588	0.475	0.600	0.480	0.597	**0.604**	0.577
	ARI	0.219	0.302	0.585	0.389	0.547	0.433	0.606	0.608	**0.610**
	F1	0.582	0.663	0.854	0.754	0.841	0.781	0.851	0.855	**0.860**
FreeBase	ACC	0.443	0.539	0.523	0.479	0.540	0.518	0.544	0.629	**0.631**
	NMI	0.001	0.185	0.141	0.114	0.188	0.126	0.185	0.188	**0.190**
	ARI	0.001	0.149	0.135	0.105	0.144	0.136	0.201	0.215	**0.218**
	F1	0.205	0.424	0.505	0.453	0.407	0.495	0.449	**0.507**	0.497

5.4. Experimental results

Figure 2 shows the iterations of the objective function, accuracy, and NMI values during the training process. As the number of iterations increases, our method positively influences the embedding space to enable embedding and clustering to be closely linked. Table 1 gives the experimental results of the different model methods on three public heterogeneous graph datasets. For both two-step methods, we use the K-means algorithm as the final clustering method. From the table, we can see that our method is superior in most of the metrics, while noting that the heterogeneous graph algorithm is generally better than the homogeneous graph algorithm. For the ACM dataset, MetaPath2Vec performs poorly, with drops of 20.9%, 19.9%, 30.8%, and 19.7% in the metrics compared with our model. This is because node features have a stronger influence in the ACM dataset, while metapath2vec only utilizes the structural features of the graph. At the same time, our method considers the structural features of the graph and the node features, thus giving it an advantage. The results on the FreeBase dataset show that our method significantly outperforms the suboptimal methods in terms of ACC, NMI, and ARI by 8.7%,0.2%, and 6.9%, respectively, which confirms the effectiveness of our method. Compared with the GAE, VGAE, and HGATE models, which aim at reconstructing features, our method outperforms GAE and VGAE on several datasets because (1) it uses the semantic information of meta-paths and (2) it uses an attention mechanism to capture the correlation between nodes. Our method performs better than HGATE with respect to most metrics, mainly because we jointly optimize the cluster process and representation learning, which improves the clustering performance. From an analysis of HeCo and HDGI based on contrastive learning, it is found that the main difference is that HeCo completes contrastive learning from two perspectives, namely, that of the heterogeneous graph network structure and that of the meta-path, whereas HDGI only uses the meta-path perspective. The results show that HeCo outperforms HDGI on all datasets, which proves that the network structure information of the heterogeneous graph is beneficial to obtaining the representation of nodes to complete the clustering task. All of the above methods are two-step clustering methods, whereas our proposed method is able to jointly learn node representation and clustering. The benefit of this is that our model learns the node representation in a clustering-oriented way and thus gives better results than the other methods in clustering tasks. Figure 2 demonstrates the ability of our method to converge in the clustering task.

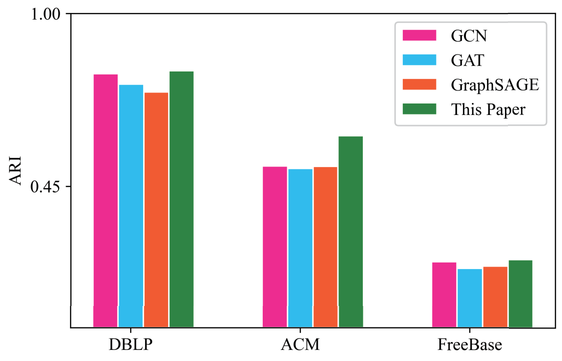

Figure 3. ARI values of different encoders.

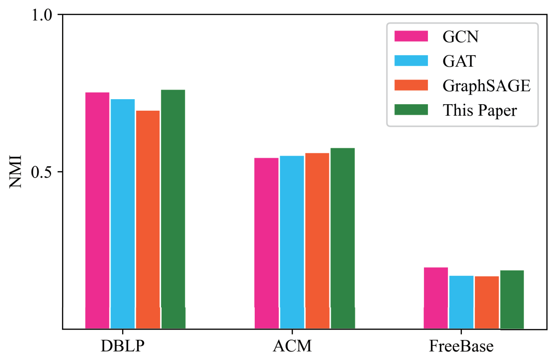

Figure 4. NMI values of different encoders.

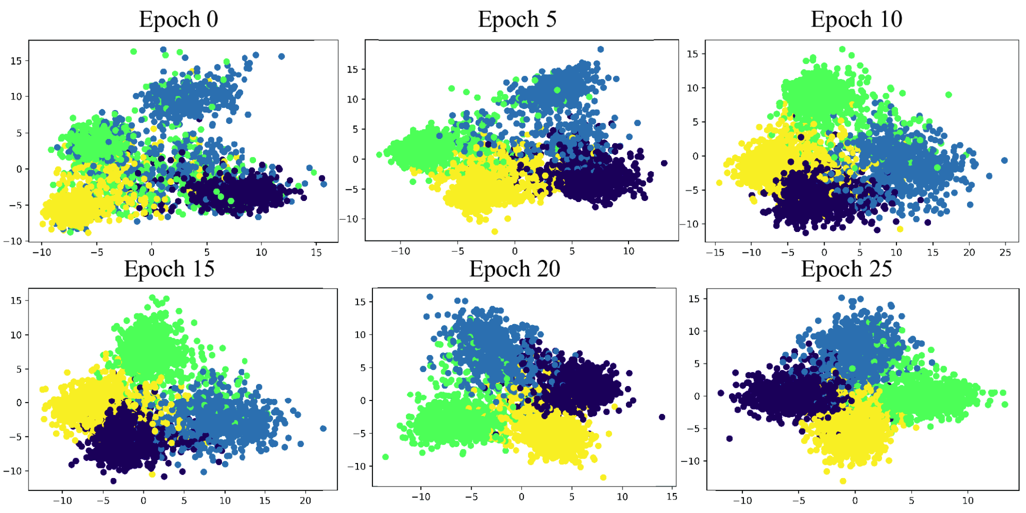

Figure 5. DBLP data clustering process.

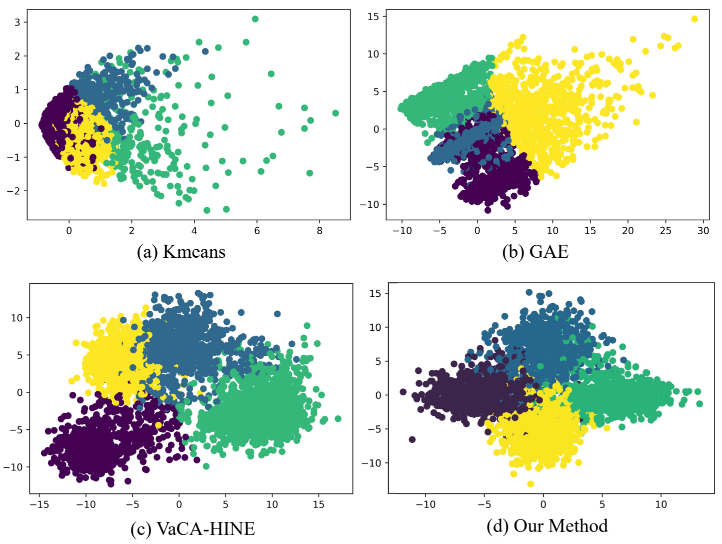

Figure 6. Visualization of nodes using different methods.

5.5. *Performance of different encoders*

In this paper, z_i^m is obtained through $f^m(X, M)$, and different models can be chosen for implementing $f^m(X, M)$. We compare GCN, GAT, GraphSAGE, and our method. From Figures 3 and 4, it can be observed that in terms of the ARI and NMI clustering metrics, the scores of GCN, GAT, and GraphSAGE are all lower than those of our method, owing to its ability to preserve the asymmetry of heterogeneous graphs. The performance of the GCN model is only slightly lower than that of our method, which may be because the distribution of node features in this dataset may be more suitable for GCN's linear feature aggregation approach. If the relationships between node features are linear, GCN can more effectively capture these relationships, leading to better performance. However, if the relationships between node features are nonlinear, GAT might be more suitable, owing to its more flexible mechanism for nonlinear feature aggregation. GraphSAGE is designed specifically for large-scale graph structures, incorporating node sampling. However, when dealing with smaller graphs, node sampling can lead to information loss. Therefore, on the three datasets, the performance of the GraphSAGE model is comparatively inferior.

5.6. Visualization

To obtain an intuitive understanding of how our method correctly groups nodes into the appropriate clusters, we visualize the clustering process of our method. Figure 5 shows a scatter plot on the DBLP dataset after we have used principal component analysis (PCA) to reduce the dimensionality, with different colors representing different clusters. From Epoch 0, it can be seen that the data points are cluttered when the clustering process has not yet been run. With training, the data points in different colors are gradually separated. From Epoch 20, data of different clusters have separated and moved far apart. Data points of the same color are denser, indicating that the clustering process clusters the data to a good state. Figure 6 illustrates the final clustering results from several methods. It can be observed that our method yields the best clustering results, with distinct separation between different clusters and closer proximity within the same cluster. On the other hand, the results of the other methods exhibit overlapping nodes from different clusters, a phenomenon less prevalent in our method. This visualization further validates the superiority of our approach.

6. Conclusions

In this paper, we have proposed a one-step clustering method for heterogeneous graphs to address their complex features and the shortcomings of two-step clustering. First, this method extracts semantic relationships from a heterogeneous graph using meta-paths and assumes independence between each meta-path to decompose the complexity of the graph. Second, it maximizes the joint probability of meta-paths and nodes and derives an optimization objective, namely, the evidence lower bound (ELBO), using variational methods. Finally, it continuously optimizes its parameters using backpropagation and reparameterization techniques to minimize the optimization objective and achieve clustering effects. Experiments on multiple real heterogeneous datasets demonstrate the competitiveness of the proposed method compared with existing methods. In the future, we will explore a one-step heterogeneous graph clustering method that does not require meta-paths.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflicts of interest

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. N. Park, R. Rossi, E. Koh, I. A. Burhanuddin, S. Kim, F. Du, et al., CGC: Contrastive graph clustering for community detection and tracking, in *Proceedings of the ACM Web Conference 2022*, (2022), 1115–1126. https://doi.org/10.1145/3485447.3512160
2. L. Guo, Q. Dai, Graph clustering via variational graph embedding, *Pattern Recognit.*, **122** (2022), 108334. https://doi.org/10.1016/j.patcog.2021.108334
3. R. A. Khan, M. Kleinsteuber, Cluster-aware heterogeneous information network embedding, in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, (2022), 476–486. https://doi.org/10.1145/3488560.3498385
4. C. Song, Y. Teng, Y. Zhu, S. Wei, B. Wu, Dynamic graph neural network for fake news detection, *Neurocomputing*, **505** (2022), 362–374. https://doi.org/10.1016/j.neucom.2022.07.057
5. H. Bo, R. McConville, J. Hong, W. Liu, Social influence prediction with train and test time augmentation for graph neural networks, in *Proceedings of 2021 International Joint Conference on Neural Networks (IJCNN)*, (2021), 1–8. https://doi.org/10.1109/IJCNN52387.2021.9533437
6. H. Bo, R. McConville, J. Hong, W. Liu, Social network influence ranking via embedding network interactions for user recommendation, in *Proceedings of WWW'20: The Web Conference*, (2020), 379–384. https://doi.org/10.1145/3366424.3383299
7. W. Peng, J. Wang, Z. Zhang, F. X. Wu, Applications of random walk model on biological networks, *Curr. Bioinf.*, **11** (2016), 2111–220. https://doi.org/10.2174/1574893611666160223200823
8. W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, et al., Graph neural networks for social recommendation, in *Proceedings of WWW'19: The World Wide Web Conference*, (2019), 417–426. https://doi.org/10.1145/3308558.3313488
9. C. Huang, H. Xu, Y. Xu, P. Dai, L. Xia, M. Lu, et al., Knowledge-aware coupled graph neural network for social recommendation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2021), 4115–4122. https://doi.org/10.1609/aaai.v35i5.16533
10. C. Huang, J. Chen, L. Xia, Y. Xu, P. Dai, Y. Chen, et al., Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2021), 4123–4130. https://doi.org/10.1609/aaai.v35i5.16534
11. D. Yuxiao, N. Chawla, A. Swami, Metapath2vec: Scalable representation learning for heterogeneous networks, in *KDD'17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2017), 135–144. https://doi.org/10.1145/3097983.3098036
12. X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, et al., Heterogeneous graph attention network, in *Proceedings of WWW '19: The World Wide Web Conference*, (2019), 2022–2032. https://doi.org/10.1145/3308558.3313562
13. X. Fu, J. Zhang, Z. Meng, I. King, MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding, in *Proceedings of WWW '20: The Web Conference*, (2020), 2331–2341. https://doi.org/10.1145/3366423.3380297

14. C. Zhang, D. Song, C. Huang, A. Swami, N. Chawla, Heterogeneous graph neural network, in *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2019), 793–803. https://doi.org/10.1145/3292500.3330961

15. C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, et al., Do transformers really perform bad for graph representation?, *Adv. Neural Inf. Process. Syst.*, **34** (2021), 28877–28888.

16. B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online learning of social representations, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2014), 701–710. https://doi.org/10.1145/2623330.2623732

17. A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in *KDD: Proceedings of International Conference on Knowledge Discovery & Data Mining*, (2016), 855–864. https://doi.org/10.1145/2939672.2939754

18. M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in*Proceedings of the 30th International Conference on Neural Information Processing Systems*, (2016), 3844–3852. https://doi.org/10.5555/3157382.3157527

19. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Inductive representation learning on large graphs, in *Proceedings of International Conference on Learning Representations*, (2017).

20. W. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (2017), 1025–1035. https://doi.org/10.5555/3294771.3294869

21. T. Kipf, M. Welling, Variational graph auto-encoders, preprint, arXiv: 1611.07308.

22. A. Salehi, H. Davulcu, Graph attention auto-encoders, in *Proceedings of IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, (2020), 989–996. https://doi.org/10.1109/ICTAI50040.2020.00154

23. P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep Graph Infomax, in *Proceedings of International Conference on Learning Representations*, (2019).

24. C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, (2019), 3670–3676. https://doi.org/10.24963/ijcai.2019/509

25. D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in *Web Conference 2020: Proceedings of the World Wide Web Conference (WWW 2020)*, (2020), 1400–1410. https://doi.org/10.1145/3366423.3380214

26. H. Liu, B. Hu, X. Wang, C. Shi, Z. Zhang, J. Zhou, Confidence may cheat: Self-training on graph neural networks under distribution shift, in *Proceedings of the ACM Web Conference*, (2022), 1248–1258. https://doi.org/10.1145/3485447.3512172

27. C. Shi, B. Hu, W. Zhao, P. Yu, Heterogeneous information network embedding for recommendation, *IEEE Trans. Knowl. Data Eng.*, **31** (2017), 357–370. https://doi.org/10.1109/TKDE.2018.2833443

28. T. Fu, W. C. Lee, Z. Lei, HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning, in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, (2017), 1797–1806. https://doi.org/10.1145/3132847.3132953

29. W. Wang, X. Wei, X. Suo, B. Wang, H. Wang, H. N. Dai, et al., HGATE: heterogeneous graph attention auto-encoders, *IEEE Trans. Knowl. Data Eng.*, **35** (2021), 3938–3951. https://doi.org/10.1109/TKDE.2021.3138788

30. X. Fu, J. Zhang, Z. Meng, I. King, MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding, in *Proceedings of The Web Conference 2020*, (2020), 2331–2341. https://doi.org/10.1145/3366423.3380297

31. S. Zheng, D. Guan, W. Yuan, Semantic-aware heterogeneous information network embedding with incompatible meta-paths, *World Wide Web*, **25** (2022), 1–21. https://doi.org/10.1007/s11280-021-00903-5

32. Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, J. Zhang, Heterogeneous deep graph infomax, preprint, arXiv: 1911.08538.

33. C. Park, D. Kim, J. Han, H. Yu, Unsupervised attributed multiplex network embedding, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2020), 5371–5378.

34. C. Mavromatis, G. Karypis, HeMI: Multi-view embedding in heterogeneous graphs, preprint, arXiv: 2109.07008.

35. J. Macqueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of Symposium on Mathematical Statistics and Probability*, (1967).

36. X. Wang, N. Liu, H. Han, C. Shi, Self-supervised heterogeneous graph neural network with co-contrastive learning, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, (2021), 1726–1736. https://doi.org/10.1145/3447548.3467415

37. Q. Zhu, W. Bi, X. Liu, X. Ma, X. Li, D. Wu, A batch normalized inference network keeps the KL vanishing away, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (2020), 2636–2649. https://doi.org/10.18653/v1/2020.acl-main.235

38. Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: An unsupervised and generative approach to clustering, in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, (2017), 1965–1972. https://doi.org/10.24963/ijcai.2017/273