HKU CS Tech Report TR-2020-02

SCHAIN-IRAM: An Efficient and Effective Semi-supervised Clustering Algorithm for Attributed Heterogeneous Information Networks

Xiang Li, Yao Wu, Martin Ester, Ben Kao, Xin Wang, Yudian Zheng

Abstract—A heterogeneous information network (HIN) is one whose nodes model objects of different types and whose links model objects' relationships. To enrich its information, objects in an HIN are typically associated with additional attributes. We call such an HIN an *Attributed HIN* or AHIN. We study the problem of clustering objects in an AHIN, taking into account objects' similarities with respect to both object attribute values and their structural connectedness in the network. We show how supervision signal, expressed in the form of a *must-link set* and a *cannot-link set*, can be leveraged to improve clustering results. We put forward the SCHAIN algorithm to solve the clustering problem, and two highly efficient variants, SCHAIN-PI and SCHAIN-IRAM, which employ the *power iteration based method* and the *implicitly restarted Arnoldi method* respectively to compute eigenvectors of a matrix. We conduct extensive experiments comparing SCHAIN-based algorithms with other state-of-the-art clustering algorithms. Our results show that SCHAIN-IRAM outperforms other competitors in terms of clustering effectiveness and is highly efficient.

Index Terms—Semi-supervised clustering, attributed heterogeneous information network, object attributes, network structure

1 Introduction

Networks model real world entities and their relationships by objects and links. A heterogeneous information network (HIN) is a network whose objects are of different types and whose links represent different kinds of relationships between objects. Compared with homogeneous information networks (in which all objects/links are of one single type), an HIN is much more expressive in capturing complex real-world knowledge. For example, the Facebook Open Graph contains objects that represent Facebook users and other non-human entities, such as photos, events and pages. To enrich the information content of an HIN, objects are often associated with various attributes. For example, on Facebook, a "user" object is associated with attributes like *age*, *gender*, *school*, and *workplace*, while a "photo" object has attributes like *lat-long* and *date/time* that record where and when the photo was taken. We call an HIN with object attributes an *attributed HIN* or AHIN for short.

Cluster analysis is a fundamental task in data analytics. Given a set of objects, the goal is to partition them into clusters such that objects in the same cluster are similar among themselves, while objects from different clusters are dissimilar. Clustering finds many interesting applications in AHINs. For example, it can be applied to a social network to identify user communities, based on which target marketing can be effectively done. The key to effective clustering is the formulation of a similarity measure between objects that well matches the clustering objective. In some cases, such similarity measure cannot be intuitively derived and needs to be discovered, typically via a learning process.

The challenges of clustering in large AHINs are twofold. (1) Objects similarity can be *attribute-based* or *link-based*. The former refers to the similarity of two objects' attribute values, while the latter refers to how well two objects are connected in the network. For AHINs, link-based similarity can be measured by simple network distance measures or by meta-path relations. A meta-path is a sequence of node types that expresses a relation between two objects in an AHIN. For example, if U and P represent "user" and "product page" object types on Facebook, respectively, and that an edge between a user and a product page in the network represents a fact that the user "likes" a product page, then the meta-path U-P-U represents a relation between two users who have *liked* the same product page. Meta-paths have been shown to be very useful in many data mining tasks in expressing the structural relations between objects in HINs [1], [2], [3], [4]. An interesting issue is how the various types of similarities, be they attribute-based or link-based, be aggregated to measure the overall similarities of objects. (2) For complex AHINs, there could be a large number of object attributes and theoretically an unlimited number of possible meta-paths to be considered in the formulation of a similarity measure. In most cases, only certain attributes and meta-paths are relevant to a clustering task. The complexity necessitates an automatic process of selecting the best set of attributes/meta-paths and evaluating their importance (often captured by a weighting scheme) for deriving the best similarity formula. One practical approach to guide such a process is for a data analyst to provide supervision, typically made available via examples such as a *must-link set* (object

- *Xiang Li and Ben Kao are with the Department of Computer Science, The University of Hong Kong, Hong Kong.*
  E-mail: {xli2, kao}@cs.hku.hk
- *Yao Wu and Yudian Zheng are with Twitter, San Fransisco, USA.*
  E-mail: {yawwu51, zhydhkcws}@gmail.com
- *Martin Ester is with the School of Computing Science, Simon Fraser University, Burnaby, Canada.*
  E-mail: ester@cs.sfu.ca
- *Xin Wang is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China.*
  E-mail: xin_wang@tsinghua.edu.cn

pairs that should be put into the same clusters) and a *cannot-link set* (object pairs that should not be put into the same clusters).

In this paper we study the problem of semi-supervised clustering on AHINs. Our main contributions include:

• We show how attribute-based similarities and link-based similarities can be effectively aggregated via a weighting scheme. Given a supervision constraint expressed via a must-link set and a cannot-link set, we show how the weighting scheme can be theoretically optimized with respect to the constraint. Our approach is to solve the optimization problem using an iterative mutual update process.

• We show how the mutual update process is reducible to a *trace maximization problem* and a *non-linear parametric programming (NPP) problem*. We prove some properties of the NPP problem that allow us to solve it computationally. Based on the iterative update process, we put forward the SCHAIN [5] algorithm for clustering objects in an AHIN.

• We propose two variants of SCHAIN, namely, SCHAIN-PI and SCHAIN-IRAM, which respectively use the power iteration method and the implicitly restarted Arnoldi method to compute eigenvectors to improve clustering efficiency.

2 Related Work

Cluster analysis is a fundamental task in data mining. For a survey on clustering algorithms for traditional relational data, see [6]. Our goal is to cluster objects in an AHIN given a supervision constraint expressed via a must-link set and a cannot-link set, which are also adopted in [7]. The clustering algorithm we seek should consist of the following elements: (1) It considers both object attribute values and object connectedness in measuring object similarity. (2) It applies to networks that are heterogeneous, i.e., objects and links can be of different types. (3) It is a semi-supervised process which takes into account supervision constraints. There are quite a few algorithms previously proposed to cluster networked objects, but most of these algorithms miss one or more elements we mentioned above. In this section we summarize and categorize these previous algorithms. We also briefly describe five algorithms, namely, PathSelClus [1], GNetMine [8], SemiRPClus [9], FocusCO [10] and HAN [11], and show how they could be adapted to solving our clustering problem. The performances of the five algorithms are evaluated and compared with our SCHAIN-based algorithms in Section 5.

[Link-based clustering] There are algorithms that cluster objects in a network based on object linkage. While the works presented in [12], [13], [14], [15], [16] focus on homogeneous information networks, *RankClus* [17], *NetClus* [18], *SI-Cluster* [19] and matrix-factorization-based methods [20] focus on heterogeneous networks. These methods are unsupervised methods and they do not consider object attributes.

[Embedding-based clustering] Network embedding has attracted much attention lately. The idea is to embed network objects into low dimensional vectors, which facilitate downstream data mining tasks, such as classification and clustering. While embedding techniques like DeepWalk [21], LINE [16] and node2vec [22] are based on the network structure only, other techniques such as GCN [23], SNE [24] and LANE [25] consider both network links and object attributes. Moreover, metapath2vec [26] and HIN2Vec [27] are two representative methods for embedding objects in HINs, while HNE [28] and HAN [11] are methods especially designed for AHINs. In addition, [29] proposes a semi-supervised graph embedding method. However, the method is restricted to homogeneous networks.

[Unsupervised clustering] In recent years, a number of algorithms have been proposed to cluster network objects considering both attribute values and network links. Most of these works apply to homogeneous networks only [30], [31], [32], [33], [34], [35], [36]. Other more elaborate methods that apply to HINs include [37], [38], [39], [40], [41]. All of these algorithms are unsupervised ones.

[Semi-supervised clustering] Semi-supervised clustering algorithms on networked data include [42], [43], [44], [45], [46], [47]. Here, we briefly describe a few representative ones and discuss how they could be applied on AHINs.

PathSelClus [1] is a meta-path-based clustering algorithm on HINs. Supervision is given by users providing seed objects for some clusters. Given two objects, the number of instances of a certain meta-path P connecting them reflects how strongly the two objects are "connected" via the meta-path relation P. Objects' similarities via a meta-path relation are captured by a *relation matrix*, which is regarded as *observations*. A probabilistic model of the hidden clusters is employed to evaluate the probabilities of the observations (i.e., relation matrix). Each meta-path is assigned a weight. These weights are learned by an iterative strategy that maximizes the consistency between the *weighted relation matrix* and the clustering results as given by the seed objects. Since PathSelClus does not consider object attribute values, when we apply it to AHINs, the attribute values are ignored.

GNetMine [8] is a graph regularized transductive classification method for HINs. It first constructs a predictive function $f(l_j|x)$ for each object x and object label l_j. Then, it minimizes an objective function that consists of two values: (1) for any two linked objects x_p and x_q, the difference between their predictive values $f(l_j|x_p)$ and $f(l_j|x_q)$, and (2) for any labeled object, x_r, the difference between its predictive value $f(l_j|x_r)$ and its true label-induced value, which is 1 if x_r's label is l_j; 0 otherwise. The predictive functions $f(l_j|x)$'s are trained by optimizing the objective function via an iterative method. Finally, labels are predicted based on the $f(l_j|x)$'s. Even though GNetMine is a classification algorithm, we can apply it to our clustering problem by regarding cluster id's as object labels. Moreover, by assigning objects that "must-link" with the same label and objects that "cannot-link" with different labels, we obtain labeled objects as training data. Like PathSelClus, GNetMine does not consider attribute values.

SemiRPClus [9] is a semi-supervised algorithm for clustering objects in an HIN. Based on *relation-paths* (which are subsets of meta-paths), the method derives several measures, which are linearly combined to evaluate the similarities of objects in the network. An objective function is defined to learn the weights of different measures with the goal of maximizing intra-cluster similarity and minimizing inter-cluster similarity. A logistic model is used to learn the weights of the relation-paths. After the weights are learned and a weighted similarity matrix is derived, the algorithm resorts to traditional clustering algorithms to cluster objects.

SemiPRClus does not consider object attribute values.

In [10], a user-oriented clustering approach FocusCO for homogeneous network is proposed. Given a set of user-provided exemplar nodes, the algorithm first infers the object attributes (and their weights) that are the most relevant in making the exemplar nodes similar among themselves. Then, the algorithm assigns a weight to each link in the network based on the weighted similarity of its end-nodes' attribute values. Next, edges with large weights are retained and each connected component in the resulting graph forms a *core set*. The core sets are then adjusted by adding or removing members with the goal of decreasing the *conductance* of the cores, which essentially measures how well objects in a core are isolated from those outside the core. The resulting cores are then regarded as clusters of interest. FocusCO considers both object attributes and link information. However, it only applies to homogeneous networks. When we apply FocusCO to our clustering problem, we ignore object and link types, and regard an AHIN as a simple graph.

HAN [11] is a state-of-the-art graph neural network model for representing AHINs. It generates node embeddings based on a hierarchical attention mechanism, which includes node-level and semantic-level attentions. Specifically, the node-level attention is used to learn the relative importance of meta-path-based neighbors of a given node while the semantic-level attention is used to learn the importance of meta-paths. For each meta-path, HAN generates node embeddings by iteratively aggregating the embeddings of meta-path-based neighbors. The final node embedding vectors are derived by aggregating the embeddings generated based on different meta-paths with the learned meta-path weights taken into account. Similar to GNetMine, we apply HAN to our semi-supervised clustering problem by regarding cluster id's as object labels and training it as a classification model. The predicted labels of objects are taken as the objects' cluster id's.

3 Definitions

In this section we give a formal problem definition.

Definition 1. **Attributed Heterogeneous Information Network (AHIN)**. Let $\mathcal{T} = \{T_1, ..., T_m\}$ be a set of $m > 1$ object types. For each type T_i, let $\mathcal{X}_i$ be the set of objects of type T_i and A_i be the set of attributes defined for objects of type T_i. An object x_j of type T_i is associated with an attribute vector $\boldsymbol{f}_j = (f_{j1}, f_{j2}, ..., f_{j|A_i|})$. An AHIN is a graph $G = (V, E, \mathcal{A})$, where $V = \bigcup_{i=1}^{m} \mathcal{X}_i$ is a set of nodes, E is a set of links (each represents a binary relation between two objects in V), and $\mathcal{A} = \bigcup_{i=1}^{m} A_i$. □

Definition 2. **Network schema**. A network schema is the meta template of an AHIN $G = (V, E, \mathcal{A})$. Let (1) $\phi : V \rightarrow \mathcal{T}$ be an object-type mapping that maps an object in V into its type, and (2) $\psi : E \rightarrow \mathcal{R}$ be a link-relation mapping that maps a link in E into a relation in a set of relations $\mathcal{R}$. The network schema of an AHIN G, denoted by $T_G = (\mathcal{T}, \mathcal{R})$, shows how objects of different types are related by the relations in $\mathcal{R}$. T_G can be represented by a *schematic graph* with $\mathcal{T}$ and $\mathcal{R}$ being the node set and the edge set, respectively. Specifically, there is an edge (T_i, T_j) in the schematic graph iff there is a relation in $\mathcal{R}$ that relates objects of type T_i to objects of type T_j. □

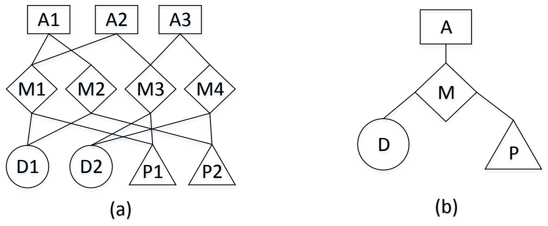

Fig. 1: An AHIN (a) and its schematic graph (b)

Figure 1(a) shows an example AHIN that models movie information (attribute information is not shown). The AHIN consists of four object types: $\mathcal{T}$ = { movie (◇), actor(□), director(○), producer(△) }. There are also three relations in $\mathcal{R}$, which are illustrated by the three edges in the schematic graph (Figure 1(b)). For example, the relation between *actor* and *movie* carries the information of which actor has acted in which movie. Actors, directors and producers have attributes like age, gender, birthplace, while movies are associated with attributes like release date, box office, etc.

Definition 3. **Meta-path**. A meta-path $\mathcal{P}$ is a path defined on a schematic graph. A meta-path $\mathcal{P}$: $T_1 \xrightarrow{R_1} \cdots \xrightarrow{R_l} T_{l+1}$ defines a composite relation $R = R_1 \circ \cdots \circ R_l$ that relates objects of type T_1 to objects of type T_{l+1}. If two objects x_u and x_v are related by the composite relation R, then there is a path, denoted by $p_{x_u \rightsquigarrow x_v}$, that connects x_u to x_v in G. Moreover, the sequence of links in $p_{x_u \rightsquigarrow x_v}$ matches the sequence of relations R_1, ..., R_l based on the link-relation mapping ψ. We say that $p_{x_u \rightsquigarrow x_v}$ is a *path instance* of $\mathcal{P}$, denoted by $p_{x_u \rightsquigarrow x_v} \vdash \mathcal{P}$. □

As an example, the path $p_{M1 \rightsquigarrow M3} = M1 \rightarrow A2 \rightarrow M3$ in Figure 1(a) is an instance of the meta-path Movie-Actor-Movie (abbrev. MAM).

Definition 4. **Supervision constraint**. The clustering process is supervised by a user through a constraint ($\mathcal{M}$, $\mathcal{C}$), where $\mathcal{M}$ and $\mathcal{C}$ are the *must-link set* and the *cannot-link set*, respectively. Each is a set of object pairs (x_a, x_b). An object pair in $\mathcal{M}$ represents that the two objects must belong to the same cluster, while a pair in $\mathcal{C}$ indicates that the pair should not be put into the same cluster. □

Definition 5. **Semi-supervised clustering in an AHIN**. Given an AHIN $G = (V, E, \mathcal{A})$, a supervision constraint ($\mathcal{M}$, $\mathcal{C}$), a target object type T_i, the number of clusters k, and a set of meta-paths $\mathcal{PS}$, the problem of semi-supervised clustering of type T_i objects in G is to (1) discover an object similarity measure S that is based on object attributes and meta-paths, and (2) partition the objects in $\mathcal{X}_i$ into k disjoint clusters $\mathfrak{C} = \{C_1, ..., C_k\}$ based on the similarity measure S such that the clustering results best agree with the constraint ($\mathcal{M}$, $\mathcal{C}$). □

4 Algorithm

In this section we present our algorithm SCHAIN (**S**emi-supervised **C**lustering in **H**eterogeneous **A**ttributed **I**nformation **N**etworks) and two more-efficient variants SCHAIN-PI and SCHAIN-IRAM. SCHAIN first composes a similarity matrix S that measures the similarity of every object pair based on the objects' attribute similarity and network connectedness. The latter is derived based on the meta-paths connecting the object pair. Since attributes and meta-paths vary in their relevancy to a clustering objective, SCHAIN assigns a weight to each object attribute and meta-path in composing S. To take into account the supervision constraint, SCHAIN derives a penalty function involving all the weightings as well as objects' cluster assignment. It then employs an iterative, staggered 2-step learning process to determine the optimal weights and cluster assignment as output. Sections 4.1 and 4.2 present the similarity matrix and the penalty function, respectively. Section 4.3 depicts the optimization technique. In Section 4.4, we analyze the computation cost of SCHAIN and put forward SCHAIN-PI and SCHAIN-IRAM to accelerate the computation.

4.1 Similarity Matrix

[Attribute-based] Given two objects x_u, x_v of type T_i, let $\boldsymbol{f}_u$ and $\boldsymbol{f}_v$ be their respective attribute vectors (see Definition 1). Recall that A_i is the set of attributes associated with type-T_i objects. We define an *attribute weight vector* $\boldsymbol{\omega} \in \mathbb{R}^{1\times|A_i|}$, whose j-th component, w_j, captures the importance of the j-th attribute in A_i for the clustering task. We define the attribute-based similarity matrix, denoted S_A, by

$$S_A(x_u, x_v) = \sum_{j=1}^{|A_i|} \left(\omega_j \cdot sim(f_{uj}, f_{vj})\right), \tag{1}$$

where $sim(f_{uj}, f_{vj})$ can be any standard similarity function defined over the j-th attribute of A_i [48].

[Link-based] We use meta-paths to measure the connectedness of objects in the network. Given a symmetric meta path $\mathcal{P}$, SCHAIN measures the similarity between two objects x_u and x_v w.r.t. $\mathcal{P}$ by *PathSim* [3]:

$$S_{\mathcal{P}}(x_u, x_v) = \frac{2 \times |\{p_{x_u \rightsquigarrow x_v} : p_{x_u \rightsquigarrow x_v} \vdash \mathcal{P}\}|}{|\{p_{x_u \rightsquigarrow x_u} : p_{x_u \rightsquigarrow x_u} \vdash \mathcal{P}\}| + |\{p_{x_v \rightsquigarrow x_v} : p_{x_v \rightsquigarrow x_v} \vdash \mathcal{P}\}|},$$

where $p_{x_u \rightsquigarrow x_v}$ denotes a path instance from object x_u to object x_v in the network, and $p_{x_u \rightsquigarrow x_v} \vdash \mathcal{P}$ denotes that the path is an instance of the meta-path $\mathcal{P}$. *PathSim* is shown to be a very effective measure of meta-path-based similarity. It compares favorably against other link-based similarity measures, such as random walk and SimRank [3].

Given a set of meta-paths $\mathcal{PS}$, each meta-path $\mathcal{P}_j \in \mathcal{PS}$ derives a similarity matrix $S_{\mathcal{P}_j}$ and is given a weight λ_j. We define the link-based similarity matrix, denoted S_L, by:

$$S_L = \sum_{j=1}^{|\mathcal{PS}|} \lambda_j S_{\mathcal{P}_j}. \tag{2}$$

Let $\boldsymbol{\lambda} \in \mathbb{R}^{1\times|\mathcal{PS}|}$ be the *meta-path weight vector*, whose j-th component is λ_j. Finally, the overall similarity matrix S is a weighted sum of S_L and S_A:

$$S = \alpha S_A + (1-\alpha) S_L, \tag{3}$$

where α is a weighting factor that controls the relative importance of the two similarity matrices.

4.2 Supervision Constraints

Given a clustering $\{C_r\}_{r=1}^k$ that partitions objects in $\mathcal{X}_i$ into k clusters, the quality of the clustering can be measured by how similar objects of different clusters are to each other — the larger is the inter-cluster similarity, the worse is the clustering quality. We measure the inter-cluster similarity based on *normalized cut* [12]. Specifically, for any two clusters C_p, C_q, define $links(C_p, C_q) = \sum_{x_u \in C_p, x_v \in C_q} S(x_u, x_v)$. The normalized cut of a clustering $\{C_r\}_{r=1}^k$ is given by $NC = \sum_{r=1}^k \frac{links(C_r, \mathcal{X}_i \backslash C_r)}{links(C_r, \mathcal{X}_i)}$. Note that NC is dependent on S. Hence, it is a function of $\boldsymbol{\omega}$, $\boldsymbol{\lambda}$, and $\{C_r\}_{r=1}^k$.

Another way to evaluate the clustering quality is to check how well the clustering agrees with the supervision constraint. Specifically, consider an object pair (x_u, x_v) in a must-link set $\mathcal{M}$. If a clustering assigns the objects to the same cluster, the clustering agrees with the constraint, which is an indication of good clustering quality. On the contrary, if the object pair is in the cannot-link set $\mathcal{C}$, then having the objects in the same cluster indicates poor clustering quality. Taking supervision constraint into consideration, we modify NC into the following penalty function:

$$\mathcal{J}(\boldsymbol{\lambda}, \boldsymbol{\omega}, \{C_r\}_{r=1}^k) = \sum_{r=1}^k \frac{links(C_r, \mathcal{X}_i \backslash C_r)}{links(C_r, \mathcal{X}_i)} - \sum_{r=1}^k \sum_{\substack{(x_u,x_v)\in\mathcal{M} \\ L(x_u)=L(x_v)=r}} \frac{S(x_u, x_v)}{links(C_r, \mathcal{X}_i)} + \sum_{r=1}^k \sum_{\substack{(x_u,x_v)\in\mathcal{C} \\ L(x_u)=L(x_v)=r}} \frac{S(x_u, x_v)}{links(C_r, \mathcal{X}_i)}, \tag{4}$$

where $L(x)$ denotes the assigned cluster for object x. For convenience, we encode a clustering $\{C_r\}_{r=1}^k$ by k indicator vectors $\boldsymbol{z}_r$'s. Each $\boldsymbol{z}_r$ consists of $n = |\mathcal{X}_i|$ bits, such that $\boldsymbol{z}_r(u) = 1$ if object $x_u \in \mathcal{X}_i$ is assigned to cluster C_r; 0 otherwise. We further encode the supervision constraint as a constraint matrix $\mathcal{W} \in \mathbb{R}^{n\times n}$, where $\mathcal{W}(u,v) = 1$ if $< x_u, x_v > \in \mathcal{M}$; -1 if $< x_u, x_v > \in \mathcal{C}$; and 0 otherwise. Let $D \in \mathbb{R}^{n\times n}$ be a diagonal matrix such that $d(i,i)$ is the sum of the entries in the i-th row of S. Eq. 4 can be rewritten as

$$\begin{aligned}\mathcal{J}(\boldsymbol{\lambda}, \boldsymbol{\omega}, \{\boldsymbol{z}_r\}_{r=1}^k) &= \sum_{r=1}^k \frac{\boldsymbol{z}_r^T(D-S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D \boldsymbol{z}_r} - \sum_{r=1}^k \frac{\boldsymbol{z}_r^T \mathcal{W}\circ S \boldsymbol{z}_r}{\boldsymbol{z}_r^T D \boldsymbol{z}_r} \\ &= \sum_{r=1}^k \frac{\boldsymbol{z}_r^T(D-S-\mathcal{W}\circ S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D \boldsymbol{z}_r},\end{aligned} \tag{5}$$

where $\circ$ is the Hadamard product for two matrices.

Furthermore, to prevent overfitting, we add a regularization term to Eq. 5 and get,

$$\mathcal{J}(\boldsymbol{\lambda}, \boldsymbol{\omega}, \{\boldsymbol{z}_r\}_{r=1}^k) = \sum_{r=1}^k \frac{\boldsymbol{z}_r^T(D-S-\mathcal{W}\circ S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D \boldsymbol{z}_r} + \gamma(||\boldsymbol{\lambda}||^2 + ||\boldsymbol{\omega}||^2). \tag{6}$$

Finally, to find the best clustering, we minimize the penalty function $\mathcal{J}()$ subject to the following constraints: (1) each object is assigned to one cluster: $\sum_{r=1}^k \boldsymbol{z}_r(u) = 1$; $\boldsymbol{z}_r(u) \in \{0,1\}$; (2) meta-path weights and attribute weights are non-negative with their respective sums equal to 1: $\sum_{j=1}^{|\mathcal{PS}|} \lambda_j = 1$; $\sum_{l=1}^{|A_i|} \omega_l = 1$; $\lambda_j \geq 0$ and $\omega_l \geq 0$.

4.3 Model optimization

Our objective is to find the best clustering, or equivalently, the indicator vectors $\{\boldsymbol{z}_r\}_{r=1}^k$ that minimizes the penalty function $\mathcal{J}()$. Note that $\mathcal{J}()$ is a function of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ (which are the weights of meta-paths and object attributes), whose values need to be learned as well. SCHAIN learns these parameters using an iterative mutual update approach. Each iteration consists of two steps. First, given $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$, we find the optimal clustering $\{\boldsymbol{z}_r\}_{r=1}^k$. Second, given $\{\boldsymbol{z}_r\}_{r=1}^k$, we find the optimal $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. SCHAIN iterates until the change in the penalty is smaller than a threshold ϵ or a fixed number of iterations have been executed. Next, we show how the two update steps are performed.

4.3.1 Finding the optimal $\{\boldsymbol{z}_r\}_{r=1}^k$ given $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$

Given $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$, $\mathcal{J}()$ is a function of $\{\boldsymbol{z}_r\}_{r=1}^k$. We define a matrix $\tilde{Z}$, whose r-th column $\tilde{Z}_{\cdot r}$ equals $D^{\frac{1}{2}}\boldsymbol{z}_r/(\boldsymbol{z}_r^T D\boldsymbol{z}_r)^{\frac{1}{2}}$. Note that since $\tilde{Z}^T\tilde{Z} = I_k$, where I_k is the $k \times k$ identity matrix, $\tilde{Z}$ is an orthonormal matrix. For fixed values of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$, minimizing $\mathcal{J}()$ is equivalent to minimizing:

$$\begin{aligned}\mathcal{J}'(\tilde{Z}) &= trace(\tilde{Z}^T D^{-\frac{1}{2}}(D - S - W \circ S)D^{-\frac{1}{2}}\tilde{Z}),\\ &= trace(I_k - \tilde{Z}^T D^{-\frac{1}{2}}(S + W \circ S)D^{-\frac{1}{2}}\tilde{Z}).\end{aligned} \tag{7}$$

Since $trace(I_k)$ is a constant, the above is equivalent to solving the following trace maximization problem:

$$\max_{\tilde{Z}^T\tilde{Z}=I_k} trace(\tilde{Z}^T D^{-\frac{1}{2}}(S + W \circ S)D^{-\frac{1}{2}}\tilde{Z}). \tag{8}$$

Since $\tilde{Z}$ is a rigorous cluster indicator matrix, the optimization problem is NP-hard [37]. To address this issue, we allow real relaxation to $\tilde{Z}$ so that its entries can assume real values. Then, according to the Ky-Fan theorem [49], the maximization problem (8) has a closed-form solution that corresponds to the subspace spanned by the top k eigenvectors of $K = D^{-\frac{1}{2}}(S + W \circ S)D^{-\frac{1}{2}}$. Since $\tilde{Z}_{\cdot r} = D^{\frac{1}{2}}\boldsymbol{z}_r/(\boldsymbol{z}_r^T D\boldsymbol{z}_r)^{\frac{1}{2}}$, we need to transform each $\tilde{Z}_{\cdot r}$ back to a real-relaxed $\boldsymbol{z}_r$. We first calculate $U = D^{-\frac{1}{2}}\tilde{Z}$ and then normalize it by column. Each column in U is a real-relaxed $\boldsymbol{z}_r$. Finally, with the real relaxation, entries in U take on fractional values, so the clustering is not definite. To derive a hard clustering, we treat each row in U as a feature vector of an object. After row normalization on U, we adopt k-means to cluster objects.

4.3.2 Finding the optimal $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ given $\{\boldsymbol{z}_r\}_{r=1}^k$

For fixed $\{\boldsymbol{z}_r\}_{r=1}^k$, $\mathcal{J}()$ is a function of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. We rewrite Eq. 6 as:

$$\begin{aligned}\mathcal{J}(\boldsymbol{\lambda}, \boldsymbol{\omega}) &= \sum_{r=1}^{k} \frac{\boldsymbol{z}_r^T D\boldsymbol{z}_r - \boldsymbol{z}_r^T(S + \mathcal{W} \circ S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D\boldsymbol{z}_r} + \gamma(||\boldsymbol{\lambda}||^2 + ||\boldsymbol{\omega}||^2),\\ &= k - \sum_{r=1}^{k} \frac{\boldsymbol{z}_r^T(S + \mathcal{W} \circ S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D\boldsymbol{z}_r} + \gamma(||\boldsymbol{\lambda}||^2 + ||\boldsymbol{\omega}||^2).\end{aligned} \tag{9}$$

Minimizing $\mathcal{J}(\boldsymbol{\lambda}, \boldsymbol{\omega})$ is equivalent to maximizing:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\omega}} \sum_{r=1}^{k} \frac{\boldsymbol{z}_r^T(S + \mathcal{W} \circ S)\boldsymbol{z}_r}{\boldsymbol{z}_r^T D\boldsymbol{z}_r} - \gamma(||\boldsymbol{\lambda}||^2 + ||\boldsymbol{\omega}||^2). \tag{10}$$

Note that the entries of matrices S and D are linear functions of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. Therefore, the numerator and the denominator of each term in the summation are both linear functions of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. Hence, (10) can be rewritten as:

$$\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{\omega}) = \max_{\boldsymbol{\lambda},\boldsymbol{\omega}} \frac{f(\boldsymbol{\lambda}, \boldsymbol{\omega})}{g(\boldsymbol{\lambda}, \boldsymbol{\omega})}, \tag{11}$$

where $f(\boldsymbol{\lambda}, \boldsymbol{\omega})$ and $g(\boldsymbol{\lambda}, \boldsymbol{\omega})$ are two nonlinear multivariate polynomial functions.

It is shown in [50] that the maximization problem with the form shown in Eq. 11 can be solved by solving the following related *non-linear parametric programming* problem:

Definition 6. **[Non-linear parametric programming (NPP)]** Let $f(\boldsymbol{\lambda}, \boldsymbol{\omega})$ and $g(\boldsymbol{\lambda}, \boldsymbol{\omega})$ be two multivariate polynomial functions. For a given μ, find

$$F(\mu) = \max_{\boldsymbol{\lambda},\boldsymbol{\omega}} \left(f(\boldsymbol{\lambda}, \boldsymbol{\omega}) - \mu g(\boldsymbol{\lambda}, \boldsymbol{\omega})\right). \tag{12}$$

In our context, the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ are subject to the constraints listed at the end of Section 4.2. □

In [50], the following theorem is proved.

Theorem 1. Given a fixed μ, let $(\boldsymbol{\lambda}^*, \boldsymbol{\omega}^*)$ be the optimal solution to $F(\mu)$ (Eq. 12). $(\boldsymbol{\lambda}^*, \boldsymbol{\omega}^*)$ is also an optimal solution to $\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{\omega})$ (Eq. 11) if and only if $F(\mu) = 0$. □

Besides Theorem 1, a few lemmas are also proved in [50]:

Lemma 1. $F(\mu)$ is convex. □

Lemma 2. $F(\mu)$ is continuous. □

Lemma 3. $F(\mu)$ is strictly monotonically decreasing, i.e., if $\mu_1 < \mu_2$, $F(\mu_1) > F(\mu_2)$. □

Lemma 4. $F(\mu) = 0$ has a unique solution. □

Due to space limit, readers are referred to [50], [51] for the proofs of the theorem and lemmas.

From Theorem 1, we need to find a μ^* and its corresponding $(\boldsymbol{\lambda}^*, \boldsymbol{\omega}^*)$ such that $F(\mu^*) = 0$. SCHAIN does so by an iterative numerical method. In each iteration, SCHAIN computes a μ and $(\boldsymbol{\lambda}, \boldsymbol{\omega})$. Let μ_i, $(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)$ be those computed in the i-th iteration. SCHAIN first sets $\mu_1 = 0$ and in each iteration, performs two steps: (Step 1:) Solve the NPP problem (Eq. 12) for $\mu = \mu_i$ and set $(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)$ to be the solution found. (Step 2:) Set $\mu_{i+1} = f(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)/g(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)$. Next, we show theoretical properties of this update process.

Property 1: $F(\mu_1) > 0$. Without loss of generality, we assume $f(\boldsymbol{\lambda}, \boldsymbol{\omega}) > 0$ and $g(\boldsymbol{\lambda}, \boldsymbol{\omega}) > 0$.[1] Now, $F(\mu_1) = F(0) = \max_{\boldsymbol{\lambda},\boldsymbol{\omega}} f(\boldsymbol{\lambda}, \boldsymbol{\omega}) > 0$.

Property 2: if $F(\mu_i) > 0$ then $0 \leq F(\mu_{i+1}) < F(\mu_i)$. Since $(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)$ is the solution of the NPP problem for $\mu = \mu_i$ (Eq. 12), we have $f(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i) - \mu_i g(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i) = F(\mu_i) > 0$. Hence, $\mu_{i+1} = f(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i)/g(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i) > \mu_i$. By Lemma 3, $F(\mu_{i+1}) < F(\mu_i)$. Also, we have $F(\mu_{i+1}) = \max_{\boldsymbol{\lambda},\boldsymbol{\omega}}(f(\boldsymbol{\lambda}, \boldsymbol{\omega}) - \mu_{i+1}g(\boldsymbol{\lambda}, \boldsymbol{\omega})) \geq f(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i) - \mu_{i+1}g(\boldsymbol{\lambda}_i, \boldsymbol{\omega}_i) = 0$.

From the properties, we see that SCHAIN starts with a positive $F(\mu)$, whose value stays positive and decreases across iterations until it reaches 0. The update procedure thus converges to the optimal values. The SCHAIN algorithm is summarized in Algorithm 1.

1. One can show that the quantity (10) is bounded below by -2γ. We can add an arbitrary large constant to (10) to make it, and thus $f(\boldsymbol{\lambda}, \boldsymbol{\omega})$ and $g(\boldsymbol{\lambda}, \boldsymbol{\omega})$, positive.

Algorithm 1 SCHAIN

```
Input: $G$, $\mathcal{M}$, $\mathcal{C}$, $T_i$, $k$, $\mathcal{PS}$.
Output: $\mathfrak{C} = \{C_1, ..., C_k\}$
 1: Compute similarity matrices $S_A$, $S_L$, and $S$
 2: $t = 0$, $\Delta\mathcal{J} = \infty$, $\boldsymbol{\lambda} = (\frac{1}{|\mathcal{PS}|}, ..., \frac{1}{|\mathcal{PS}|})$, $\boldsymbol{\omega} = (\frac{1}{|A_i|}, ..., \frac{1}{|A_i|})$
 3: while $\Delta\mathcal{J} > \epsilon$ or $t <$ max_iter do
 4:     ▷ Step 1: Optimize $\{\boldsymbol{z}_r\}_{r=1}^k$ given $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$
 5:     Solve Eq. 8 to obtain real-relaxed $\tilde{Z}$
 6:     Calculate $U = D^{-1/2}\tilde{Z}$ and normalize it
 7:     Derive $\{\boldsymbol{z}_r\}_{r=1}^k$ from $U$ by k-means
 8:     ▷ Step 2: Optimize $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ given $\{\boldsymbol{z}_r\}_{r=1}^k$
 9:     $j = 1$; $\mu_j = 0$
10:     repeat
11:         Solve Eq. 12 with $\mu = \mu_j$ to obtain $\boldsymbol{\lambda}_j$, $\boldsymbol{\omega}_j$
12:         $\mu_{j+1} = f(\boldsymbol{\lambda}_j, \boldsymbol{\omega}_j)/g(\boldsymbol{\lambda}_j, \boldsymbol{\omega}_j)$; $j$++
13:     until $F(\mu_{j+1})$ converges to 0
14:     $\Delta\mathcal{J}$ = change in $\mathcal{J}$ with the updated $\{\boldsymbol{z}_r\}_{r=1}^k$, $\boldsymbol{\lambda}$, $\boldsymbol{\omega}$
15:     $t$++
16: end while
17: Decode $\{C_r\}_{r=1}^k$ from $\{\boldsymbol{z}_r\}_{r=1}^k$
18: return $\mathfrak{C} = \{C_1, ..., C_k\}$
```

4.4 Improving Efficiency

The most computationally expensive component of SCHAIN is the eigen decomposition in solving Eq. 8, which has a time complexity of $O(n^3)$ where n is the number of objects to be clustered ($|\mathcal{X}_i|$). To speed up, we propose two methods to compute the eigenvectors.

4.4.1 Power iteration based method

Power iteration (PI) is an efficient technique to compute the dominant eigenvector of a matrix. Given a matrix W, PI starts with a random vector $\boldsymbol{v}_0 \neq \mathbf{0}$ and iterates:

$$\boldsymbol{v}_{t+1} = \frac{W\boldsymbol{v}_t}{||W\boldsymbol{v}_t||_1}, \quad t \geq 0.$$

Suppose W has eigenvalues $\tau_1 > \tau_2 > ... > \tau_n$ with associated eigenvectors $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, ..., \boldsymbol{\eta}_n$. We express $\boldsymbol{v}_0$ as

$$\boldsymbol{v}_0 = c_1\boldsymbol{\eta}_1 + c_2\boldsymbol{\eta}_2 + ... + c_n\boldsymbol{\eta}_n,$$

where the c_1, ..., c_n are coefficients. Let $R = \prod_{i=0}^{t-1}||W\boldsymbol{v}_i||_1$. We have,

$$\begin{aligned}\boldsymbol{v}_t &= W^t\boldsymbol{v}_0/R \\ &= (c_1W^t\boldsymbol{\eta}_1 + c_2W^t\boldsymbol{\eta}_2 + ... + c_nW^t\boldsymbol{\eta}_n)/R \\ &= (c_1\tau_1^t\boldsymbol{\eta}_1 + c_2\tau_2^t\boldsymbol{\eta}_2 + ... + c_n\tau_n^t\boldsymbol{\eta}_n)/R \\ &= \frac{c_1\tau_1^t}{R}\left[\boldsymbol{\eta}_1 + \frac{c_2}{c_1}\left(\frac{\tau_2}{\tau_1}\right)^t\boldsymbol{\eta}_2 + ... + \frac{c_n}{c_1}\left(\frac{\tau_n}{\tau_1}\right)^t\boldsymbol{\eta}_n\right].\end{aligned}$$

$\boldsymbol{v}_t$ is a linear combination of the eigenvectors. If $c_1 \neq 0$, $\boldsymbol{v}_t$ converges to a scaled version of the dominant eigenvector $\boldsymbol{\eta}_1$.

It has been shown in [52] that one can truncate the iteration process to obtain an intermediate pseudo-eigenvector $\boldsymbol{v}_t$, based on which the *Power Iteration Clustering* (PIC) method is proposed. PIC takes the j-th component of $\boldsymbol{v}_t$ as the feature of object $\boldsymbol{x}_j$ and applies k-means to the feature to compute object clusters. When the number of clusters is large, a single pseudo-eigenvector is generally not sufficient to produce high-quality cluster. In [53], the PIC-k method is proposed. The method runs truncated PI multiple times to derive multiple pseudo-eigenvectors. Compared with the standard spectral clustering methods, PIC-k uses the k pseudo-eigenvectors to obtain an improved clustering result. Inspired by PIC-k, we approximate the top k eigenvectors of $K = D^{-\frac{1}{2}}(S + W \circ S)D^{-\frac{1}{2}}$ by running truncated PI k times. The truncated PI algorithm is summarized in Algorithm 2. When the matrix K is sparse, the computational cost of PI is reduced to $O(dn)$, where $d \ll n$ is the average number of nonzero entries per row in K. The complexity of this PI-based approximation method is $O(kdn)$, which is substantially more efficient than the basic $O(n^3)$ method. We call this method SCHAIN-PI.

Algorithm 2 Truncated power iteration

```
Input: $K$, $k$, $\boldsymbol{v}_0$.
Output: $\boldsymbol{v}_{t+1}$
 1: repeat
 2:     $\boldsymbol{v}_{t+1} \leftarrow \frac{K\boldsymbol{v}_t}{||K\boldsymbol{v}_t||_1}$; $\boldsymbol{\delta}_{t+1} \leftarrow |\boldsymbol{v}_{t+1} - \boldsymbol{v}_t|$; $t$++
 3: until $|\boldsymbol{\delta}_{t+1} - \boldsymbol{\delta}_t| \rightarrow 0$
 4: return $\boldsymbol{v}_{t+1}$
```

4.4.2 Implicitly restarted Arnoldi method

The power iteration method computes the dominant eigenvector of a matrix W. During the process, a squence of intermediate vectors $\{\boldsymbol{v}_0, W\boldsymbol{v}_0, W^2\boldsymbol{v}_0, ...\}$ are generated. This sequence is called a *Krylov sequence*. Although PI discards this sequence, interestingly, the sequence is useful in deriving a good approximation of the eigenvectors. Based on the Krylov sequence, one can define $\mathcal{K}_n = span\{\boldsymbol{v}_0, W\boldsymbol{v}_0, ..., W^{n-1}\boldsymbol{v}_0\}$ as the *n-th order Krylov subspace*. *Arnoldi Factorization* derives an orthogonal projection of W onto the Krylov subspace that can be used to approximate the eigenpairs of W. Specifically, given $W \in R^{n\times n}$, a *k-step Arnoldi factorization* of W has the form $WV_k = V_kH_k + \boldsymbol{f}_k\boldsymbol{e}_k^T$, where (1) $V_k \in R^{n\times k}$ has orthonormal columns, (2) $V_k^T\boldsymbol{f}_k = \mathbf{0}$, (3) $H_k \in R^{k\times k}$ is an upper Hessenberg matrix with non-negative subdiagonal elements, and (4) $\boldsymbol{e}_k$ is a unit vector whose k-th entry is 1. Note that H_k represents an orthogonal projection of W. An alternative form of the factorization is

$$WV_k = (V_k, \boldsymbol{v}_{k+1})\begin{pmatrix} H_k \\ \beta_k\boldsymbol{e}_k^T \end{pmatrix},$$

where $\beta_k = ||\boldsymbol{f}_k||_2$ and $\boldsymbol{v}_{k+1} = \frac{1}{\beta_k}\boldsymbol{f}_k$.

Arnoldi factorization can be used to compute the eigenvalues and eigenvectors of W from those of the small matrix H_k. Given an eigenpair $(\theta, \boldsymbol{s})$ of H_k, i.e., $H_k\boldsymbol{s} = \boldsymbol{s}\theta$, the vector $\boldsymbol{x} = V_k\boldsymbol{s}$ satisfies

$$||W\boldsymbol{x} - \boldsymbol{x}\theta||_2 = ||(WV_k - V_kH_k)\boldsymbol{s}||_2 = |\beta_k\boldsymbol{e}_k^T\boldsymbol{s}|.$$

A small value of $|\beta_k\boldsymbol{e}_k^T\boldsymbol{s}|$ indicates that $(\theta, \boldsymbol{s})$ well approximates an eigenpair of W. In particular, when $||\boldsymbol{f}_k||_2 = 0$, $|\beta_k\boldsymbol{e}_k^T\boldsymbol{s}| = 0$ and $(\theta, \boldsymbol{x})$ becomes an exact eigenpair of W.

In practice, the number of steps needed to obtain accurate approximations to the eigenvectors of interest may be large, and which is determined by the initial vector $\boldsymbol{v}_0$. A desired $\boldsymbol{v}_0$ should contain nonzero values in the directions of the eigenvectors to be computed without containing any irrelevant components. In the Arnoldi factorizarion process, we would like to adaptively refine $\boldsymbol{v}_0$ and restart the Arnoldi factorization with a new $\boldsymbol{v}_0$. *Implicitly restarted Arnoldi method* (IRAM) [54] employs an implicit way to update $\boldsymbol{v}_0$. Consider an m-step Arnoldi factorization $WV_m = V_m H_m + \boldsymbol{f}_m \boldsymbol{e}_m^T$, where $m = k + p$. It will be repeatedly compressed to a k-step factorization that retains the desired eigen-information through implicitly shifted QR decomposition. Let σ_j be a shift and $H_m - \sigma_j I = Q_j Y_j$, where Q_j is orthogonal and Y_j is upper triangular. We have

$$\begin{aligned} WV_m - V_m H_m &= (W - \sigma_j I)V_m - V_m(H_m - \sigma_j I), \\ &= (W - \sigma_j I)V_m - V_m Q_j Y_j, \\ &= \boldsymbol{f}_m \boldsymbol{e}_m^T. \end{aligned} \tag{13}$$

Hence,

$$W(V_m Q_j) - (V_m Q_j)(Y_j Q_j + \sigma_j I) = \boldsymbol{f}_m \boldsymbol{e}_m^T Q_j. \tag{14}$$

This can be easily extended with up to p shifts applied:

$$WV_m^+ = V_m^+ H_m^+ + \boldsymbol{f}_m \boldsymbol{e}_m^T \hat{Q}, \tag{15}$$

where $V_m^+ = V_m \hat{Q}$, $H_m^+ = \hat{Q}^T H_m \hat{Q}$ and $\hat{Q} = Q_1 Q_2 ... Q_p$. Due to the Hessenberg structure of the matrices Q_j, the first $k - 1$ entries of the vector $\boldsymbol{e}_m^T \hat{Q}$ are zero and an updated k-step Arnoldi factorization can be derived by equating the first k columns on both sides of Eq. 15. From Eq. 13, each shift σ_j will replace the starting vector $\boldsymbol{v}_0$ with $(W - \sigma_j I)\boldsymbol{v}_0$, so an appropriate selection of σ_j will filter the unwanted eigenvector information from $\boldsymbol{v}_0$. Based on the updated k-step Arnoldi factorization, p additional steps will be performed to return to m-step Arnoldi factorization. The whole process will be repeated (restarted Arnoldi) until convergence. For more details, see [55].

We use IRAM to compute the top k eigenvectors of K in Eq. 8. We call this method SCHAIN-IRAM. We assume an m-step Arnoldi factorization, where $m = k + p$ and $k < m \ll n$. In each iteration, an eigen-decomposition is performed on H_m with a time complexity of $O(m^3)$. Moreover, each iteration takes p steps to update matrices V_m^+, H_m^+ and $\hat{Q}$, with the computation cost of each step being $O(m^3 + nm^2)$. The overall cost of SCHAIN-IRAM is $O(h(m^3 + p(m^3 + nm^2)))$, where h is the number of iterations to convergence. Our experiments show that h is practically very small.

5 Experiments

In this section we evaluate the performance of SCHAIN, SCHAIN-PI, and SCHAIN-IRAM[2] and compare them against 9 other algorithms by applying them to three example clustering tasks on real data. We have also generated synthetic datasets to study the various aspects of the algorithms. We will illustrate the importance of integrating attribute-based similarity and link-based similarity in clustering objects in AHINs and show the effectiveness of SCHAIN in determining the relative weights ($\boldsymbol{\omega}$ and $\boldsymbol{\lambda}$) of attributes and meta-paths. We show that the weight-learning process of SCHAIN converges quickly under the example clustering tasks. We also perform an efficiency study and show that SCHAIN-PI and SCHAIN-IRAM are much more efficient than SCHAIN. In particular, our results show that SCHAIN-IRAM achieves high efficiency without sacrificing clustering quality.

5.1 Algorithms for comparison

We compare 12 algorithms, which can be categorized into four groups:

• **Attribute-only:** The first group of clustering algorithms consider only object attributes. These are traditional methods which ignore the network structure of an AHIN. We chose Spectral-Learning [46] and a semi-supervised version of normalized cuts [44] as representatives, which are denoted *SL* and *SNcuts*, respectively. Both methods are spectral clustering approaches of semi-supervised clustering. The difference is that SL uses additive normalization while SNcuts adopts row normalization. Since SL and SNcuts do not learn attribute weights, we give all attributes equal weights in constructing an attribute similarity matrix.

• **Link-only:** These methods utilize only the link information of the network and they ignore object attribute values. We chose *GNetMine* [8], *PathSelClus* [1] and *SemiRPClus* [9] as representative methods of this category. These algorithms were described in Section 2.

• **Attribute+Link:** Methods of this category use both attribute and link information. We consider both *FocusCO* and *HAN*, which were described in Section 2. In particular, HAN is a state-of-the-art deep neural network model for AHINs.

• **SCHAIN and its variants:** We evaluate SCHAIN and four variants: (1) SCHAIN uses meta-paths to derive the link-based similarity matrix. An alternative measure is *random walk with restart* (RWR) [56]. Specifically, for the link-based similarity matrix S_L, we set its (i, j) entry to be the steady-state probability from object i in the network to object j. We call this variant SCHAIN-RWR. By comparing SCHAIN with this variant, we will learn about the importance of meta-paths in solving the clustering problem. (2) SCHAIN uses an iterative learning process to determine the optimal weights of attributes and meta-paths. To study the effectiveness of weight learning, we modify SCHAIN such that it assumes certain initial values of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$ (see Line 3 of Algorithm 1), finds the optimal $\{\boldsymbol{z}_r\}_{r=1}^k$ once, and reports that as the final clustering. In other words, we take away the iteration of the while-loop and retain only Lines 6-7. We call this variant SCHAIN-NL (**N**o weight-**L**earning). Moreover, we evaluate SCHAIN-PI and SCHAIN-IRAM, which are two variants that efficiently compute approximate top-k eigenvectors of the matrix K (see Eq. 8).

5.2 Clustering tasks

We use two real datasets, namely, Yelp and DBLP in our experiments. Yelp[3] contains information of businesses, their users, locations, reviews, etc. DBLP[4] is a bibliographic network dataset which captures authors/venues/keywords information of academic publications. From these datasets, we define three clustering tasks:

• **Yelp-Business**. We extracted businesses located in three states of the US: North Carolina (NC), Wisconsin (WI),

2. The codes of SCHAIN, SCHAIN-PI, and SCHAIN-IRAM are publicly available at https://github.com/lixiang3776/SCHAIN.

3. http://www.yelp.com/academic_dataset
4. http://dblp.uni-trier.de

TABLE 1: NMI comparison on Yelp-Business

	Attribute-only		Link-only			Attribute+Link		SCHAIN Variants				
% seeds	SL	SNcuts	GNM	PSC	SemiRPClus	FocusCO	HAN	S-RWR	S-NL	S-PI	S-IRAM	SCHAIN
5%	0.001	0.783	0.996	0.687	0.232	0.088	0.802	**1.000**	0.909	**1.000**	**1.000**	**1.000**
10%	0.016	0.764	0.996	0.697	0.312	0.084	0.807	**1.000**	0.920	**1.000**	**1.000**	**1.000**
15%	0.011	0.672	0.996	0.730	0.356	0.084	0.827	**1.000**	0.968	**1.000**	**1.000**	**1.000**
20%	0.004	0.630	0.996	0.757	0.371	0.085	0.840	**1.000**	0.969	**1.000**	**1.000**	**1.000**
25%	0.004	0.565	0.996	0.787	0.587	0.087	0.857	**1.000**	0.970	**1.000**	**1.000**	**1.000**

TABLE 2: NMI comparison on Yelp-Restaurant

	Attribute-only		Link-only			Attribute+Link		SCHAIN Variants				
% seeds	SL	SNcuts	GNM	PSC	SemiRPClus	FocusCO	HAN	S-RWR	S-NL	S-PI	S-IRAM	SCHAIN
5%	0.225	0.185	0.284	0.564	0.142	0.088	0.214	0.427	0.628	0.480	**0.716**	0.689
10%	0.258	0.188	0.332	0.610	0.134	0.087	0.214	0.429	0.635	0.486	**0.722**	0.707
15%	0.416	0.192	0.367	0.627	0.136	0.095	0.250	0.433	0.655	0.481	**0.734**	0.725
20%	0.425	0.198	0.379	0.635	0.132	0.087	0.243	0.426	0.678	0.485	**0.740**	0.738
25%	0.437	0.251	0.392	0.637	0.136	0.090	0.248	0.436	0.689	0.498	**0.747**	0.744

TABLE 3: NMI comparison on DBLP

	Attribute-only		Link-only			Attribute+Link		SCHAIN Variants				
% seeds	SL	SNcuts	GNM	PSC	SemiRPClus	FocusCO	HAN	S-RWR	S-NL	S-PI	S-IRAM	SCHAIN
5%	0.551	0.576	0.183	0.137	0.113	0.057	0.475	0.601	0.613	0.503	0.632	**0.634**
10%	0.554	0.554	0.241	0.170	0.090	0.058	0.511	0.598	0.611	0.506	0.629	**0.639**
15%	0.558	0.540	0.284	0.216	0.084	0.059	0.526	0.595	0.614	0.511	**0.633**	**0.633**
20%	0.560	0.531	0.314	0.251	0.080	0.061	0.521	0.599	0.615	0.508	**0.631**	**0.631**
25%	0.563	0.524	0.333	0.265	0.077	0.055	0.514	0.603	0.616	0.513	0.629	**0.637**

Pennsylvania (PA); and in Edinburgh (EDH) of the UK. We constructed an AHIN that comprises 10,133 business objects (B); 73,366 user objects (U); 100 city objects (C); and 472 business sector objects (T) (such as "restaurant" and "shopping"). Each business object is associated with several attributes including lat-long, review count, quality star, and parking lot (whether parking facility is provided). Links include B-T (business and its category), U-B (customer of a business), B-C (business located in a city). We consider the meta-path set {BCB, BUB, BTB}. The clustering task is to cluster business objects by state. We use the state information provided in the dataset as the ground truth.

This clustering task is a very simple one. In particular, either attributes or links provide reliable sources to allow perfect clustering to be obtained. All the clustering algorithms, whether they are attribute-based only, link-based only, or both, are expected to perform well.

• **Yelp-Restaurant**. We extracted information related to restaurant business objects of three sub-categories: "Fast Food", "Sushi Bars" and "American (New) Food". We constructed an AHIN of 2,614 business objects (B); 33,360 review objects (R); 1,286 user objects (U) and 82 food relevant keyword objects (K). Each restaurant has 3 categorical attributes: reservation (whether reservation is required), service (waiter service or self service) and parking; 1 numerical attribute: review count; and 1 ordinal attribute: quality star. Links include B-R (business receives a review), R-U (review written by a customer), R-K (review contains a keyword). We consider the meta-path set {BRURB, BRKRB}. The clustering task is to cluster restaurants by category.

This clustering task is slightly more difficult than Yelp-Business because it is not totally obvious which attributes/meta-paths are relevant to the task. It is thus more interesting to see how the various algorithms fair against each other, particularly in their ability to identify the most relevant features and their appropriate weights.

• **DBLP**. CIKM is a conference focusing on three research areas: Information Retrieval (IR), Data Mining (DM) and Databases (DB). We extracted a subset of the DBLP network that comprises 387 authors (A), 2,044 papers (P), and 2,171 key terms (T). Each of the 387 authors has published in CIKM and has published in at least one of the conferences SIGIR, KDD, and VLDB. For each author object, the numbers of his/her publications in the four conferences serve as the object's attribute values (i.e., 4 numerical attributes). Links include A-P (author publishes a paper), P-T (paper contains a key term). We consider the meta-path set: {APA, APAPA, APTPA}. The clustering task is to cluster authors by their research areas (IR, DM, DB). We obtained the ground truth from the dataset *dblp-4area* [18], which labels each author by his/her primary research area.

This task is the most difficult among the three tasks because the research areas somewhat overlap. Cluster memberships are therefore not as clear cut as in the other tasks.

5.3 Results

For each clustering task, we construct a supervision constraint ($\mathcal{M}$, $\mathcal{C}$) in the following way. We randomly pick a certain percentage of the objects (to be clustered) as *seeds*. Since we know the true labels of objects (the ground truth), for each pair of seed objects x_u, x_v, we put (x_u, x_v) in $\mathcal{M}$ if x_u and x_v share the same label; we put (x_u, x_v) in $\mathcal{C}$ otherwise. We use *Normalized Mutual Information (NMI)* [57] between a clustering result $\mathfrak{C}$ and the clustering based on the true objects' labels to measure the quality of the clustering $\mathfrak{C}$. NMI ranges from 0 to 1; the higher the NMI is, the more $\mathfrak{C}$ resembles the true clustering. NMI = 1 if $\mathfrak{C}$ perfectly agrees with the true clustering. Each reported NMI is an average of 10 runs and each run uses a different set of seed objects to construct the supervision constraint. Since HAN is a deep neural network model, in addition to a training set, it further needs a validation set to prevent overfitting during training.

In our experiments, we use an extra 10% seeds to construct a validation set for HAN. We measure the attribute similarity of two Yelp-Business objects x_u and x_v in the following way: For a (normalized) numerical attribute j, we use the similarity function $sim(f_{uj}, f_{vj}) = 1 - |f_{uj} - f_{vj}|$ (see Eq 1); For a categorical attribute, $sim(f_{uj}, f_{vj}) = 1$ if $f_{uj} = f_{vj}$; 0 otherwise. Likewise for Yelp-Restaurant objects. For DBLP, the feature vector of an author object x_u is a normalized vector $\boldsymbol{f_u}$ of four conference paper counts. The attribute similarity of two authors x_u and x_v is given by the dot-product $\boldsymbol{f_u} \cdot \boldsymbol{f_v}$.

5.3.1 Clustering quality

Tables 1, 2 and 3 compare the clustering qualities of the various algorithms on the three tasks. We use GNM and PSC as shorthands for GNetMine and PathSelClus, respectively. Moreover, for all SCHAIN variants, we use S as shorthand for SCHAIN due to space limitation. The first column (% seeds) of each table shows the percentage of objects taken as seed objects to construct $\mathcal{M}$ and $\mathcal{C}$. In each row, the NMI of the best algorithm is highlighted. From the tables, we make the following observations.

• As we have explained previously, Yelp-Business is a relatively simple clustering task. In particular, there are attributes (e.g., lat-long) and meta-paths (e.g., BCB) that individually provide good similarity measures for the clustering task. We therefore see algorithms that give very good quality results. These include SNcuts (attribute-based), GNetMine (link-based), and particularly all SCHAIN variants, which produce perfect or almost perfect clusterings.

• As we move from Yelp-Business to Yelp-Restaurant and then to DBLP, the clustering tasks become more challenging. For link-based methods and the SCHAIN family, clustering quality drops. The drop in quality for the link-based methods is very pronounced. For example, the NMI of GNetMine (at 5% seeds) drops from 0.996 on Yelp-Business to 0.183 on DBLP. This shows that the latter tasks require attribute information to achieve good clustering quality. From this discussion, we see that both attribute and link information can play important roles in object clustering, particularly for more complex clustering tasks.

• The performance of spectral learning algorithms (SL, SNcuts) is not very "stable". For example, for Yelp-Business, SL performs very poorly while SNcuts does very well. On the other hand, SL performs better than SNcuts for Yelp-Restaurant. As explained in [46], additive normalization can lead to very poor performance in the presence of distant outliers. This explains the very low NMIs of SL (which employs additive normalization) on Yelp-Business, which happens to contain distant outliers. SNcuts, which employs row normalization, does not suffer from such problems. We also see that the performance of SNcuts on Yelp-Business and DBLP gets worse with more seeds. This is due to the following *scaling problem*. We observe from our experiment that the values in the similarity matrices derived with SNcuts for Yelp-Business and DBLP are generally fairly small numbers (even for similar objects). The similarity matrices are then modified by considering the given supervision constraint. In particular, a pair of seed objects (x_u, x_v) given in a must link set will have their similarity value set to 1, which is relatively large in the matrix. These large values cause SNcuts to incorrectly identify some similar object pairs as cuts, leading to poorer clustering. For SCHAIN, a must-link set exerts its influence via Eq. 6, which essentially doubles (x_u, x_v)'s similarity (instead of setting that to 1). This avoids the scaling problem.

• For DBLP, the NMI of SemiRPClus decreases when the % of seeds increases. The reason for this observation is that SemiRPClus identifies the meta-path APA (co-authorship) as the most important meta-path by giving it the highest weight among all meta-paths. However, APA is a *sparse* relation. This is because a typical author only co-authors with a handful of other researchers. That means APA is a weak relation in measuring object (author) similarity. As the percentage of seeds increases, we find that SemiRPClus gives APA even larger weights. This adversely affects the performance of SemiRPClus.

• Our adaptation of FocusCO performs poorly in all cases of our experiments. Even though FocusCO is a semi-supervised clustering algorithm that utilizes both attribute and link information, it is designed for homogeneous networks. By converting an AHIN to a homogeneous one, information about object and link types is removed. This significantly weakens the effectiveness of FocusCO.

• Although HAN performs much better than FocusCO, it compares unfavorably against SCHAIN-based methods. HAN uses node-level attention and semantic-level attention to distinguish meta-path based neighbors and to learn weights of meta-paths. However, it does not learn the relative importance of object attributes.

• SCHAIN performs better than SCHAIN-RWR. This shows that meta-paths are more effective than random walk in deriving objects' link-based similarity. This observation is consistent with other works related to mining on heterogeneous information networks, such as [3].

• SCHAIN performs better than SCHAIN-NL. This shows that SCHAIN's ability in learning and distinguishing the weights of different attributes and meta-paths is important in achieving good clustering quality.

• SCHAIN, SCHAIN-PI, and SCHAIN-IRAM generally perform the best among all algorithms. In particular, all three registered perfect scores for Yelp-Business. The techniques we employed in SCHAIN, namely, attribute-based similarity, meta-path-based similarity, as well as weight learning, contribute effectively to the clustering tasks. For Yelp-Restaurant and DBLP, SCHAIN and SCHAIN-IRAM significantly outperform the others. We observe that for these two tasks SCHAIN-PI does not perform as well as SCHAIN-IRAM. We remark that SCHAIN-PI uses power iteration to generate k pseudo-eigenvectors to approximate the top-k eigenvectors. Each pseudo-eigenvector is a weighted linear combination of all the eigenvectors. In such a process, the generated pseudo-eigenvectors could contain redundant information and may contain noise. In comparison, SCHAIN-IRAM generates more accurate approximation to the top-k eigenvectors. Therefore, SCHAIN-IRAM's performance is more stable than SCHAIN-PI.

5.3.2 Weight learning

An interesting feature of SCHAIN is its ability to learn the weights of attributes and meta-paths. In this section we take a closer look at the effectiveness of SCHAIN's iterative weight-learning process. We will use the three clustering tasks as examples for illustration. We observe that SCHAIN-IRAM has a similar weight-learning process as SCHAIN, so we omit the details due to page limitation. In the following discussion, we assume 5% seed objects.

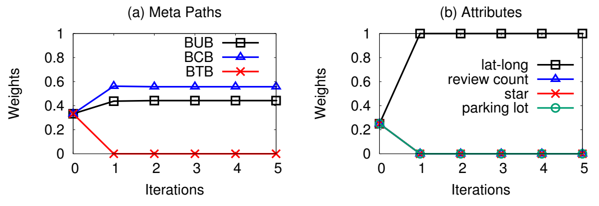

Fig. 2: Weight learning on Yelp-Business

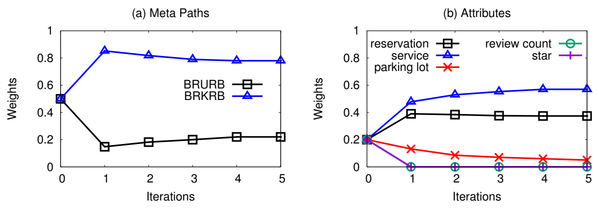

Fig. 3: Weight learning on Yelp-Restaurant

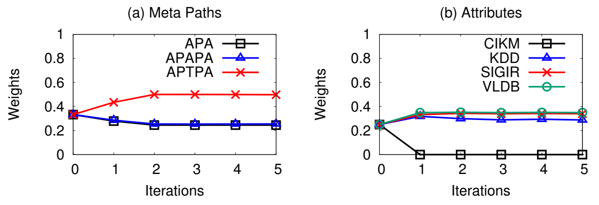

Fig. 4: Weight learning on DBLP

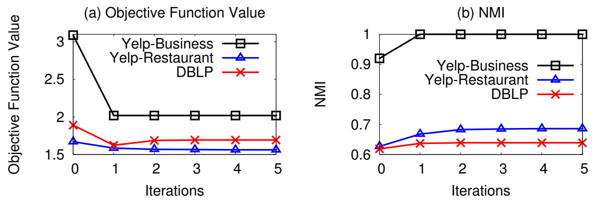

Fig. 5: Convergence analysis

Figures 2(a) and (b) show the weights learned across iterations for attributes and meta-paths, respectively, on Yelp-Business. Recall that the task is to cluster business objects by their geographical locations. From Figure 2(a), we see that SCHAIN correctly identifies that the meta-paths BCB (businesses that are co-located in the same city) and BUB (businesses that serve the same customer) give the most relevant relations in the locality of the businesses. It also correctly gives a 0 weight to the meta-path BTB (businesses of the same sector), which is irrelevant to the businesses' locations. Moreover, from Figure 2(b), we see that SCHAIN correctly identifies lat-long to be the only relevant attribute (which is given a weight of 1.0), and considers other attributes irrelevant (which are given 0 weights).

Figure 3 shows the weight learning for Yelp-Restaurant. Recall that the task is to cluster restaurant objects by the kind of food served. The figure shows that SCHAIN gives a larger weight to the meta-path BRKRB (restaurants whose reviews share the same keyword, such as dishes) than to the meta-path BRURB (restaurants visited by the same customer). This is reasonable because the same customers can visit restaurants serving different categories of foods. Interestingly, SCHAIN also finds that whether a restaurant requires reservation and provides wait services are relevant to predicting the restaurant's category. This is because those that do are likely higher-end restaurants, which serve more expensive foods (such as Japanese Sushi).

Figure 4 shows the results for DBLP. We see that SCHAIN finds all three meta-paths relevant to the clustering task, and they are given similar weights. Interestingly, SCHAIN gives the attribute CIKM (the number of papers one published in CIKM) a 0 weight. This is because for the dataset we extracted, all authors have CIKM publications. So the attribute has no discerning power for the task. Also, SCHAIN gives more or less equal weights to the other 3 attributes because they are equally relevant in determining the research areas of authors. From this discussion, we see that SCHAIN is highly effective in learning the appropriate weights of meta-paths and attributes.

TABLE 4: Eigenvectors computation time (in seconds)

	Yelp-business	Yelp-restaurant	DBLP
SCHAIN-PI	7.89	0.19	0.02
SCHAIN-IRAM	10.04	0.21	0.04
SCHAIN	945.36	16.12	0.10

5.3.3 *Convergence analysis*

From Figures 2, 3 and 4, we see that the weights reach their optimal values in two to three iterations. Figures 5(a) and (b) further show the convergence of the objective function $\mathcal{J}$ and the NMI of the resulting clusterings, respectively. From the figures, we see that SCHAIN converges very quickly.

5.3.4 *Efficiency study*

The major computation cost of SCHAIN is due to the computation of the top-k eigenvectors of the matrix K (see Eq. 8), which has a complexity of $O(n^3)$, where n is the number of objects to be clustered. SCHAIN-PI and SCHAIN-IRAM provide approximate solutions to speed up the eigenvectors computation. Table 4 shows the computation costs of the three methods when they are applied to the three clustering tasks. From the table, we see both SCHAIN-PI and SCHAIN-IRAM are much more efficient than SCHAIN. In particular, for Yelp-Business, which has a much larger dataset than the other tasks, SCHAIN-PI is about 120 times faster than SCHAIN. Although SCHAIN-IRAM is slightly slower than SCHAIN-PI, as we have discussed in Section 5.3.1, SCHAIN-IRAM provides a better approximation of the eigenvectors and thus it gives similar clustering quality as that of SCHAIN. In conclusion, SCHAIN-IRAM is an effective and efficient solution for clustering objects in AHINs.

5.3.5 *Parameter analysis*

We study the sensitivity of SCHAIN-IRAM w.r.t. parameters α and γ. Recall that α controls the relative importance between attribute-based similarity and link-based similarity (see Eq. 3) and γ is the regularization hyper-parameter. We conducted an experiment with 25% seed objects. Fig. 6(a) and 6(b) show SCHAIN-IRAM's NMIs as α and γ vary, respectively. When $\alpha = 0$ (1), only link (attribute) information is used. From Fig. 6(a), we see that SCHAIN-IRAM provides very stable performance over the range of α values except when α is very small. The reason behind this stable performance is that our SCHAIN-based methods are able to scale the importance of attribute-based and link-based information via the weight-learning process. The mild variations at the small-α end of the figure is due to an extreme focus on link information. From Fig. 6(b), we see that SCHAIN-IRAM is generally not very sensitive to γ. In practice, one can tune the parameters for the best performance by cross-validation with the must-link and the cannot-link sets as supervision information.

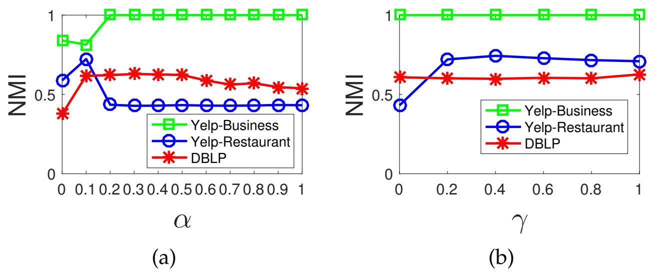

Fig. 6: The performance of SCHAIN-IRAM as α and γ vary

5.3.6 *Synthetic datasets*

To further evaluate the algorithms, we conduct experiments on synthetic datasets. We generate our synthetic datasets following [37]. Specifically, given the number of object types, the generator generates three things: (1) A number of objects of each object type. (2) Relation matrices that represent the edges between objects of two given object types. (3) Object's attribute values.

As an example, here are the details of one such synthetic dataset, which we call SYN. We then scale SYN to obtain AHINs of various sizes (up to an order of 10^5 nodes) and of various edge densities. SYN consists of three types of nodes, namely, T_a, T_b and T_c. We generate 1,000 T_a objects, 2 T_b objects and 2 T_c objects. Each Type-T_a object is linked with one Type-T_b and one Type-T_c object. We generate two relation matrices, R^{ab} and R^{bc}, which represent edges between T_a and T_b objects, and those between T_b and T_c objects, respectively. We randomly partition Type-T_a objects into two clusters. The relation R^{ab} is derived from a block structure $\left[\begin{smallmatrix}1 & 0\\ 0.3 & 0.7\end{smallmatrix}\right]$. The i-th row in the block structure shows the probability distribution of a Cluster-i T_a object being connected to the two T_b objects. For example, the 2nd row of the block structure shows that T_a objects in Cluster 2 are connected to the first and the second T_b objects with probabilities 0.3 and 0.7, respectively. (For other synthetic datasets, if there are g clusters of T_a objects and h T_b objects, the block structure would be a g-by-h matrix.) Furthermore, the block structure for deriving R^{bc} is $\left[\begin{smallmatrix}0.6 & 0.4\\ 0.4 & 0.6\end{smallmatrix}\right]$.

Objects of type T_a are associated with two binary attributes, A_1 and A_2. Objects' values of each attribute are generated by a block structure. Specifically, the block structure that derives the first attribute values is $\left[\begin{smallmatrix}0.9 & 0.1\\ 0 & 1\end{smallmatrix}\right]$. The i-th row in the block structure shows the probability distribution of a Cluster i T_a object's attribute A_1 value. For example, the first row of the block structure shows that a Cluster 1 T_a object's A_1 value is 0 or 1 with probabilities 0.9 and 0.1, respectively. The block structure for generating attribute A_2's values is $\left[\begin{smallmatrix}0.8 & 0.2\\ 0.3 & 0.7\end{smallmatrix}\right]$. In general, with d attributes, each with f distinct values, there are d attribute-generating g-by-f block structures.

Table 5 shows the NMI of the 12 algorithms when they are applied to dataset SYN with 10% seed objects for constructing the must-link and the cannot-link sets. (For SCHAIN-variants, we shortened their names with the prefix "S" instead of "SCHAIN" to fit the table within the width of page.) Recall that SCHAIN-PI and SCHAIN-IRAM are the two efficient algorithms we put forward in this paper to speed up the processing of SCHAIN. Also, SCHAIN-RWR and SCHAIN-NL are two baseline SCHAIN variants with which we evaluate different components of SCHAIN. From the table, we see that SCHAIN-PI and SCHAIN-IRAM achieve comparable clustering quality against SCHAIN and all three methods greatly outperform other competitors. Moreover, in the experiment, SCHAIN-PI and SCHAIN-IRAM are 38.5 and 28.2 times faster than SCHAIN, respectively. The results show that SCHAIN-PI and SCHAIN-IRAM are much more efficient than SCHAIN, while they are able to provide very comparable clustering quality.

We scale SYN to obtain larger datasets and observe similar conclusions. In particular, for the dataset with 10^5 nodes, the runtimes of SCHAIN-PI and SCHAIN-IRAM are 34.71s and 39.06s, respectively. SCHAIN, however, does not terminate in an hour. This shows that SCHAIN-PI and SCHAIN-IRAM are very practical for large datasets.

To investigate how the network density affects the performance of our proposed methods, we generate synthetic datasets to vary the network density by adding edges to the network. In SYN, each type-T_a object connects with one Type-T_b object and one Type-T_c object. We vary the network density by randomly selecting a fraction of Type-T_a objects and adding edges between them and all T_b and T_c objects. In our experiment, we gradually increase this fraction of T_a objects from 10% to 60%. At 10%, the NMI of SCHAIN-IRAM is 0.736, which is very close to the result given in Table 5. At 60%, the NMI decreases to 0.674. The drop in NMI is due to the noise introduced by the added edges. However, SCHAIN-IRAM's performance in terms of NMI remains close to that of SCHAIN. In terms of runtime, SCHAIN-IRAM remains efficient; Its execution time is around 0.03s over the range of tested values. However, since our SCHAIN-based methods employ meta-paths to capture the link-based similarities between objects, the network density could affect the sparsity of the derived similarity matrices, which could in turn lower the efficiency of our methods, especially for very large datasets.

6 Conclusions

In this paper we studied semi-supervised clustering in attributed heterogeneous information networks. We put forward a novel algorithm SCHAIN, which integrates object attributes and meta-paths with a weighting scheme in formulating a similarity matrix for object clustering. SCHAIN takes a supervision constraint in the form of a must-link set and a cannot-link set, and through an iterative update

TABLE 5: Algorithms' NMIs on SYN

	Attribute-only		Link-only			Attribute+Link		SCHAIN Variants				
% seeds	SL	SNcuts	GNM	PSC	SemiRPClus	FocusCO	HAN	S-RWR	S-NL	S-PI	S-IRAM	SCHAIN
10%	0.497	0.478	0.483	0.339	0.5425	0.033	0.654	0.505	0.678	0.733	0.734	0.735

process, optimizes the weighting scheme. To further speed up the model, we proposed two methods SCHAIN-PI and SCHAIN-IRAM, which respectively use power iteration based method and implicitly restarted Arnoldi method to compute the eigenvectors of a matrix. We conducted extensive experiments to show the effectiveness of SCHAIN and illustrated its ability in assigning the most appropriate weights to attributes and meta-paths. We also showed that even though both SCHAIN-PI and SCHAIN-IRAM are efficient, SCHAIN-IRAM achieves as good performance as SCHAIN while the performance of SCHAIN-PI is unstable.

ACKNOWLEDGMENTS

This research is supported by Hong Kong Research Grants Council GRF HKU 17254016.

REFERENCES

[1] Y. Sun *et al.*, "Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," in *KDD*, 2012.
[2] C. Wan *et al.*, "Classification with active learning and meta-paths in heterogeneous information networks," in *CIKM*, 2015.
[3] Y. Sun *et al.*, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, 2011.
[4] X. Li *et al.*, "On transductive classification in heterogeneous information networks," in *CIKM*, 2016.
[5] X. Li *et al.*, "Semi-supervised clustering in attributed heterogeneous information networks," in *WWW*, 2017.
[6] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, 2006.
[7] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *KDD*, 2010, pp. 563–572.
[8] M. Ji *et al.*, "Graph regularized transductive classification on heterogeneous information networks." in *ECML/PKDD*, 2010.
[9] C. Luo *et al.*, "Semi-supervised clustering on heterogeneous information networks," in *PAKDD*, 2014.
[10] B. Perozzi *et al.*, "Focused clustering and outlier detection in large attributed graphs," in *KDD*, 2014.
[11] X. Wang *et al.*, "Heterogeneous graph attention network," in *WWW*, 2019.
[12] J. Shi and J. Malik, "Normalized cuts and image segmentation," *TPAMI*, 2000.
[13] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, 2004.
[14] X. Xu *et al.*, "Scan: a structural clustering algorithm for networks." in *KDD*, 2007.
[15] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach." in *WSDM*, 2013.
[16] J. Tang *et al.*, "Line: Large-scale information network embedding," in *WWW*, 2015.
[17] Y. Sun *et al.*, "Rankclus: integrating clustering with ranking for heterogeneous information network analysis." in *EDBT*, 2009.
[18] Y. Sun *et al.*, "Ranking-based clustering of heterogeneous information networks with star network schema." in *KDD*, 2009.
[19] Y. Zhou and L. Liu, "Social influence based clustering of heterogeneous information networks," in *KDD*, 2013.
[20] F. Wang *et al.*, "Community discovery using nonnegative matrix factorization," *DMKD*, 2011.
[21] B. Perozzi *et al.*, "Deepwalk: Online learning of social representations," in *KDD*, 2014.
[22] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016.
[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[24] L. Liao *et al.*, "Attributed social network embedding," *TKDE*, 2018.
[25] X. Huang *et al.*, "Label informed attributed network embedding," in *WSDM*, 2017.
[26] Y. Dong *et al.*, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017.
[27] T.-y. Fu *et al.*, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017.
[28] S. Chang *et al.*, "Heterogeneous network embedding via deep architectures," in *KDD*, 2015.
[29] Z. Yang *et al.*, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016.
[30] J. Yang *et al.*, "Community detection in networks with node attributes." in *ICDM*, 2013.
[31] T. Yang *et al.*, "Combining link and content for community detection: a discriminative approach." in *KDD*, 2009.
[32] Z. Xu *et al.*, "A model-based approach to attributed graph clustering." in *SIGMOD*, 2012.
[33] Y. Zhou *et al.*, "Graph clustering based on structural/attribute similarities," *PVLDB*, 2009.
[34] S. Gnnemann *et al.*, "Subspace clustering meets dense subgraph mining: A synthesis of two paradigms." in *ICDM*, 2010.
[35] Y. Ruan *et al.*, "Efficient community detection in large networks using content and links." in *WWW*, 2013.
[36] J. J. McAuley *et al.*, "Learning to discover social circles in ego networks." in *NIPS*, 2012.
[37] B. Long *et al.*, "Spectral clustering for multi-type relational data." in *ICML*, 2006.
[38] B. Long *et al.*, "A probabilistic framework for relational clustering." in *KDD*, 2007.
[39] Y. Sun *et al.*, "Relation strength-aware clustering of heterogeneous information networks with incomplete attributes," *PVLDB*, 2012.
[40] B. Boden *et al.*, "Density-based subspace clustering in heterogeneous networks," in *ECML/PKDD*, 2014.
[41] X. Li *et al.*, "Spectral clustering in heterogeneous information networks," in *AAAI*, vol. 33, 2019, pp. 4221–4228.
[42] S. Basu *et al.*, "Semi-supervised clustering by seeding." in *ICML*, 2002.
[43] S. Basu *et al.*, "A probabilistic framework for semi-supervised clustering." in *KDD*, 2004.
[44] B. Kulis *et al.*, "Semi-supervised graph clustering: a kernel approach." in *ICML*, 2005.
[45] M. Bilenko *et al.*, "Integrating constraints and metric learning in semi-supervised clustering," in *ICML*, 2004.
[46] S. D. Kamvar *et al.*, "Spectral learning." in *IJCAI*, 2003.
[47] X. Zhu *et al.*, *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.
[48] J. Han *et al.*, *Data mining: concepts and techniques*. Elsevier, 2011.
[49] R. Bhatia, *Matrix analysis*. Springer-Verlag, New York, 1997.
[50] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, 1967.
[51] I. Stancu-Minasian, *Fractional programming: theory, methods and applications*, 2012, vol. 409.
[52] F. Lin and W. W. Cohen, "Power iteration clustering," in *ICML*, 2010.
[53] F. Lin, "Scalable methods for graph-based unsupervised and semi-supervised learning," Ph.D. dissertation, Carnegie Mellon University, 2012.
[54] R. B. Lehoucq *et al.*, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.
[55] D. C. Sorensen, "Implicit application of polynomial filters in ak-step arnoldi method," *Siam journal on matrix analysis and applications*, 1992.
[56] H. Tong *et al.*, "Fast random walk with restart and its applications," in *ICDM*, 2006.
[57] C. D. Manning *et al.*, *Introduction to Information Retrieval*. Cambridge University Press, 2008.