Semi-supervised Clustering in Attributed Heterogeneous Information Networks

Xiang Li[†], Yao Wu[‡], Martin Ester[‡], Ben Kao[†], Xin Wang[‡], Yudian Zheng[†]
[†] The University of Hong Kong, Pokfulam Road, Hong Kong
[‡] Simon Fraser University, Burnaby, BC, Canada
[†]{xli2, kao, ydzheng2}@cs.hku.hk [‡]{wuyaow, ester, xwa49}@sfu.ca

ABSTRACT

A heterogeneous information network (HIN) is one whose nodes model objects of different types and whose links model objects' relationships. In many applications, such as social networks and RDF-based knowledge bases, information can be modeled as HINs. To enrich its information content, objects (as represented by nodes) 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. We conduct extensive experiments comparing SCHAIN with other state-of-the-art clustering algorithms and show that SCHAIN outperforms the others in clustering quality.

Keywords

semi-supervised clustering; attributed heterogeneous information network; object attributes; network structure

1. INTRODUCTION

Networks (or graphs) 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 entities and their relationships. With the rapid development of Web technology, much information is collected and often the information can be modeled by HINs. These HINs vary in terms of their complexities – from relatively simple social networks to very complex knowledge bases. For example, the Facebook Open Graph[1] contains objects that represent Facebook users and other non-human entities, such as photos, events and pages. As another example, Yago[2] is a knowledge base that captures information derived from Wikipedia, WordNet and GeoNames. Yago is a repository of information on more than 10 million objects (such as persons, organizations, cities, etc.) and it records more than 120 million facts about these entities. Yago can be modeled as RDF graphs, which are examples of HINs.

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* at 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 clusters are similar among themselves, while objects from different clusters are dissimilar. Clustering finds many interesting applications in AHINs. For example, clustering 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 need 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 (such as shortest-path length and random-walk-based distances) 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, then the meta-path U-P-U could represent a relation between two users who have *liked* the same product page; and the meta-path U-U-U could represent the relation between two users who have a common friend. Meta-paths have been shown to be very useful in many data mining tasks in expressing the structural relations between objects in HINs [27, 31, 25,

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052576

[1]https://developers.facebook.com/docs/sharing/opengraph
[2] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago

13, 32]. 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. For example, the similarity of two Facebook users in their interests in different kinds of consumer products could involve factors like their age, gender, as well as how often they *like* the same product pages and whether they have common friends. 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 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 attributed heterogeneous information networks. 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 can be computationally done by proving that it is reducible to a *trace maximization problem* and a *non-linear parametric programming (NPP) problem*. We further prove some properties of the NPP problem, which allow us to solve it computationally. Based on the iterative update process, we put forward the SCHAIN algorithm for clustering objects in an AHIN.
- We perform extensive experiments on real data sets studying the various characteristics of SCHAIN. We compare the performance of SCHAIN against other state-of-the-art HIN clustering algorithms. Our results show that SCHAIN outperforms its competitors.

The rest of the paper is organized as follows. Section 2 mentions related works. Section 3 gives some basic definitions. In Section 4, we discuss how attribute-based similarities and link-based similarities are integrated into a single similarity measure via a weighting scheme. We show how to model supervision constraints as a penalty function of the weighting scheme that is subject to optimization, and describe the SCHAIN algorithm. Section 5 presents experimental results. Finally, Section 6 concludes the paper.

2. RELATED WORK

Cluster analysis is a fundamental task in data mining. For a survey of clustering algorithms on traditional relational data, see [3]. As we have explained in the introduction, our goal is to cluster objects in an AHIN given a supervision constraint expressed via a must-link set and a cannot-link set. The clustering algorithm we seek should consist of the following elements: (1) It considers both object attribute values as well as 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 number of 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 four algorithms, namely, PathSelClus [27], GNetMine [10], SemiRPClus [16] and FocusCO [20], and show how they could be adapted to solving our clustering problem. The performances of the four algorithms are evaluated and compared with SCHAIN in Section 5.

[Link-based clustering] There are algorithms that cluster objects in a network based on object linkage. While the works presented in [22, 19, 34, 36, 29] focus on homogeneous information networks, *RankClus* [26], *NetClus* [28], *SI-Cluster* [40] and matrix-factorization-based methods [33] focus on heterogeneous networks. These methods are unsupervised methods and they do not consider object attributes.

[Unsupervised clustering] In recent years, a number of algorithms have been put forward to cluster network objects considering both attribute values and network links. Most of these works apply to homogeneous networks only [37, 38, 35, 39, 8, 21, 18]. There also exist some more elaborate methods that apply to heterogeneous networks, such as [14, 15, 24, 6], however, they are unsupervised algorithms.

[Semi-supervised clustering] There are a number of semi-supervised clustering algorithms on networked data [2, 1, 12, 5, 11, 41]. However, they are applicable to homogeneous networks only.

PathSelClus [27] is a semi-supervised clustering algorithm on HINs that is based on meta-path selection. 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. For each meta-path P, objects' similarities via the meta-path relation are captured by a *relation matrix*, which is regarded as *observations*. A probabilistic model of the hidden clusters is then 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 of each cluster. Since PathSelClus does not consider object attribute values, when we apply it to AHINs, the attribute values are ignored.

GNetMine [10] is a graph regularized transductive classification method in heterogeneous information networks. It first constructs a predictive function $f(l_j|x)$ for each object x and object label l_j. Then it derives an objective function which aims to minimize 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" to the same label and objects that "cannot-link" to different

labels, we obtain labeled objects as the training data. Like PathSelClus, GNetMine does not consider attribute values.

SemiRPClus [16] 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. Also, 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 (e.g., hierarchical clustering) to cluster objects. SemiPRClus does not consider object attribute values.

In [20], a novel user-oriented clustering approach FocusCO on 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.

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 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 $\mathcal{A} = \bigcup_{i=1}^{m} A_i$. If $m = 1$ (i.e., there is only one object type), G reduces to a homogeneous information network. □

Definition 2. **Network schema**. A network schema is the meta template of an AHIN $G = (V, E, \mathcal{A})$. Let (1) $\phi : V \to \mathcal{T}$ be an object-type mapping that maps an object in V into its type, and (2) $\psi : E \to \mathcal{R}$ 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. □

Figure 1(a) shows an example AHIN that models movie information (attribute information is not shown). The AHIN

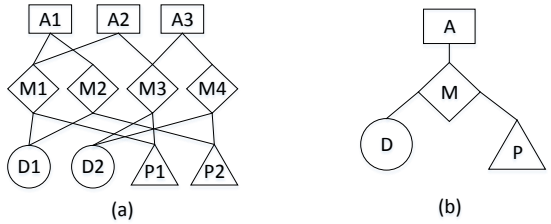

Figure 1: An AHIN (a) and its schematic graph (b)

consists of four object types: $\mathcal{T}$ = { movie ($\diamond$), actor($\square$), director($\bigcirc$), producer($\triangle$) }. 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 the schematic graph of a network schema. 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}. We say $\mathcal{P}$ is *symmetric* if the defined relation R is symmetric. 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 $p_{M1 \rightsquigarrow M3} = M1 \to A2 \to M3$ 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 of these sets 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 two objects 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). 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.

4.1 Similarity Matrix

[**Attribute-based**] Given two objects x_u and x_v of type T_i, let $\boldsymbol{f}_u$ and $\boldsymbol{f}_v$ be the attribute vectors of x_u and x_v, respectively (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 [9]. We require that $sim()$ be normalized to the range [0,1].

[**Link-based**] Object similarity can also be measured by the connectedness of objects in the network. As we have explained in the introduction, meta-paths have been shown to be very effective in capturing object relations in an AHIN. Given a symmetric meta path $\mathcal{P}$, SCHAIN measures the similarity between two objects of the same type x_u and x_v w.r.t. $\mathcal{P}$ by *PathSim* [25]:

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

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 [25].

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 the normalized cut [22]. 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 the clustering $\{C_r\}_{r=1}^k$ in terms of object similarity 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 the similarity matrix 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:

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

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: $\sum_{r=1}^k \boldsymbol{z}_r(u) = 1$; $\boldsymbol{z}_r(u) \in \{0,1\}$; $\sum_{j=1}^{|\mathcal{PS}|} \lambda_j = 1$; $\sum_{l=1}^{|A_i|} \omega_l = 1$; $\lambda_j \geq 0$ and $\omega_l \geq 0$. Note that α and γ are also parameters in the function $\mathcal{J}()$. In practice, since the must-link set and the cannot-link set are provided as supervision information, the two parameters can be tuned by cross-validation.

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 $\boldsymbol{\omega}$

For fixed values of $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$, $\mathcal{J}()$ is a function of $\{\boldsymbol{z}_r\}_{r=1}^k$. We define a matrix $\tilde{Z}$, where its 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 [14]. 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 [4], 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:

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

Since k is a constant, 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 [7] 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 [7], 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 [7]:

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 [7, 23] 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 $g(\boldsymbol{\lambda},\boldsymbol{\omega}) > 0$.[3] 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 $0 \le F(\mu_{i+1}) < F(\mu_i)$. 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 $F(\mu_{i+1}) = \max_{\boldsymbol{\lambda},\boldsymbol{\omega}}(f(\boldsymbol{\lambda},\boldsymbol{\omega}) - \mu_{i+1}g(\boldsymbol{\lambda},\boldsymbol{\omega})) \ge f(\boldsymbol{\lambda}_i,\boldsymbol{\omega}_i) - \mu_{i+1}g(\boldsymbol{\lambda}_i,\boldsymbol{\omega}_i) = 0$.

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.

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

5. EXPERIMENTS

In this section we evaluate the performance of SCHAIN and compare it against 8 other algorithms by applying them to three example clustering tasks on real data. 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. Finally, we show that the weight-learning process of SCHAIN converges quickly under the example clustering tasks. This shows that SCHAIN is practically efficient.

5.1 Algorithms for comparison

We consider 8 other algorithms, which can be categorized into four groups:

• **Attribute-only:** The first group of clustering algorithms considers only object attributes. These are traditional methods which ignore the network structure of an AHIN. We chose Spectral-Learning [11] and a semi-supervised version of normalized cuts [12] 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* [10], *PathSelClus* [27] and *SemiRP-Clus* [16] 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 *FocusCO*, which was described in Section 2.

• **SCHAIN Variants:** To analyze the performance of SCHAIN, we consider two variants: (1) SCHAIN uses meta-paths to derive the link-based similarity matrix. An alternative measure is *random walk with restart* (RWR) [30]. 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).

5.2 Clustering tasks

We use two datasets, namely, Yelp and DBLP in our experiments. Yelp[4] contains information of businesses, their users, locations, reviews, etc. DBLP[5] 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), Pennsylvania (PA); and in Edinburgh (EDH) of the UK. From the extracted information, 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 net-

[4]http://www.yelp.com/academic_dataset
[5]http://dblp.uni-trier.de

Table 1: NMI comparison on Yelp-Business

	Attribute-only		Link-only			Attribute+Link	SCHAIN Variants		
% seeds	SL	SNcuts	GNetMine	PathSelClus	SemiRPClus	FocusCO	SCHAIN-RWR	SCHAIN-NL	SCHAIN
5%	0.001	0.783	0.996	0.687	0.232	0.088	**1.000**	0.909	**1.000**
10%	0.016	0.764	0.996	0.697	0.312	0.084	**1.000**	0.920	**1.000**
15%	0.011	0.672	0.996	0.730	0.356	0.084	**1.000**	0.968	**1.000**
20%	0.004	0.630	0.996	0.757	0.371	0.085	**1.000**	0.969	**1.000**
25%	0.004	0.565	0.996	0.787	0.587	0.087	**1.000**	0.970	**1.000**

Table 2: NMI comparison on Yelp-Restaurant

	Attribute-only		Link-only			Attribute+Link	SCHAIN Variants		
% seeds	SL	SNcuts	GNetMine	PathSelClus	SemiRPClus	FocusCO	SCHAIN-RWR	SCHAIN-NL	SCHAIN
5%	0.225	0.185	0.284	0.564	0.142	0.088	0.427	0.628	**0.689**
10%	0.258	0.188	0.332	0.610	0.134	0.087	0.429	0.635	**0.707**
15%	0.416	0.192	0.367	0.627	0.136	0.095	0.433	0.655	**0.725**
20%	0.425	0.198	0.379	0.635	0.132	0.087	0.426	0.678	**0.738**
25%	0.437	0.251	0.392	0.637	0.136	0.090	0.436	0.689	**0.744**

Table 3: NMI comparison on DBLP

	Attribute-only		Link-only			Attribute+Link	SCHAIN Variants		
% seeds	SL	SNcuts	GNetMine	PathSelClus	SemiRPClus	FocusCO	SCHAIN-RWR	SCHAIN-NL	SCHAIN
5%	0.551	0.576	0.183	0.137	0.113	0.057	0.601	0.613	**0.634**
10%	0.554	0.554	0.241	0.170	0.090	0.058	0.598	0.611	**0.639**
15%	0.558	0.540	0.284	0.216	0.084	0.059	0.595	0.614	**0.633**
20%	0.560	0.531	0.314	0.251	0.080	0.061	0.599	0.615	**0.631**
25%	0.563	0.524	0.333	0.265	0.077	0.055	0.603	0.616	**0.637**

work that comprises 387 authors (A), 2044 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* [28], 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)* [17] 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. In Yelp-related tasks, for numerical and ordinal attributes[6], we first normalize them to [0,1] and then $sim(f_{uj}, f_{vj}) = 1 - |f_{uj} - f_{vj}|$; for categorical attributes, $sim(f_{uj}, f_{vj}) = 1$, if $f_{uj} = f_{vj}$; 0, otherwise. For DBLP, since attributes are sparse, we first normalized them to [0,1] and then choose $sim(f_{uj}, f_{vj}) = f_{uj} \cdot f_{vj}$.

[6]Specially, with regard to attribute lat-long, we first compute the distance between two objects by Euclidean distance and then calculate $sim() = \frac{1}{1+distance}$.

5.3.1 Clustering quality

Tables 1, 2 and 3 compare the clustering qualities of the various algorithms on the three tasks. 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 for the more difficult clustering problems, we need both attribute and link information.
- The performance of spectral learning algorithms (SL, SNcuts) is more unpredictable. 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 [11], 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.
- 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.

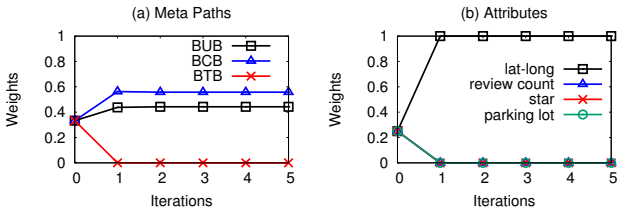

Figure 2: Weight learning on Yelp-Business

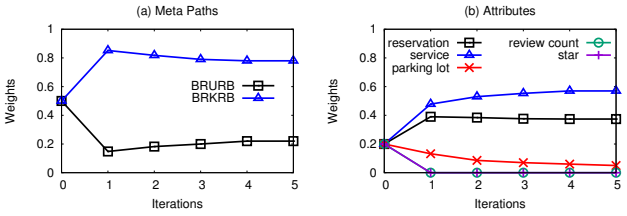

Figure 3: Weight learning on Yelp-Restaurant

• 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 [25].
• 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.
• Finally, SCHAIN gives the best performance under all the cases in our experiment. This shows that each one of the elements SCHAIN makes use of, namely, attribute values, meta-paths and weight-learning, contributes additively to high clustering quality.

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. In the following discussion, we assume 5% seed objects.

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

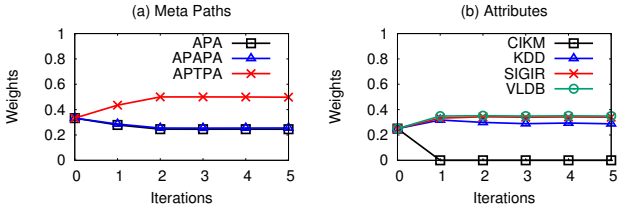

Figure 4: Weight learning on DBLP

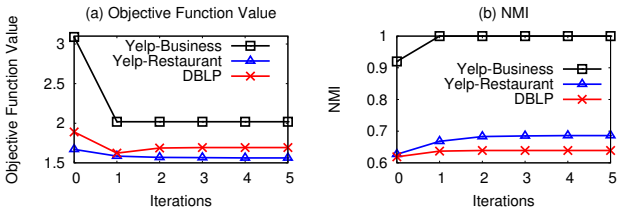

Figure 5: Convergence analysis

(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 waiter 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. Recall that the task is to cluster authors by research area. 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.

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. Again, for the cases we studied, SCHAIN converges quickly and is therefore practically efficient.

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 process, optimizes the weighting scheme. 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.

Acknowledgements

This research is supported by Hong Kong Research Grants Council GRF HKU 17254016 and partly supported by NSERC Discovery Grant in Canada.

7. REFERENCES

[1] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, pages 59–68, 2004.

[2] S. Basu et al. Semi-supervised clustering by seeding. In *ICML*, pages 27–34, 2002.

[3] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[4] R. Bhatia. *Matrix analysis*. Springer-Verlag, New York, 1997.

[5] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, pages 81–88, 2004.

[6] B. Boden, M. Ester, and T. Seidl. Density-based subspace clustering in heterogeneous networks. In *ECML/PKDD*, pages 149–164, 2014.

[7] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[8] S. Günnemann et al. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, pages 845–850, 2010.

[9] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[10] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECML/PKDD*, pages 570–586, 2010.

[11] S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *IJCAI*, pages 561–566, 2003.

[12] B. Kulis, S. Basu, I. S. Dhillon, and R. J. Mooney. Semi-supervised graph clustering: a kernel approach. In *ICML*, pages 457–464, 2005.

[13] X. Li, B. Kao, Y. Zheng, and Z. Huang. On transductive classification in heterogeneous information networks. In *CIKM*, 2016.

[14] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML*, pages 585–592, 2006.

[15] B. Long, Z. M. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In *KDD*, pages 470–479, 2007.

[16] C. Luo, W. Pang, and Z. Wang. Semi-supervised clustering on heterogeneous information networks. In *PAKDD*, pages 548–559, 2014.

[17] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] J. J. McAuley et al. Learning to discover social circles in ego networks. In *NIPS*, pages 548–556, 2012.

[19] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[20] B. Perozzi et al. Focused clustering and outlier detection in large attributed graphs. In *KDD*, pages 1346–1355, 2014.

[21] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *WWW*, pages 1089–1098, 2013.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000.

[23] I. Stancu-Minasian. *Fractional programming: theory, methods and applications*, volume 409. Springer Science & Business Media, 2012.

[24] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB*, 5(5):394–405, 2012.

[25] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[26] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT*, pages 565–576, 2009.

[27] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD*, pages 1348–1356, 2012.

[28] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, pages 797–806, 2009.

[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[30] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[31] C. Wan et al. Classification with active learning and meta-paths in heterogeneous information networks. In *CIKM*, pages 443–452, 2015.

[32] M. Wan, Y. Ouyang, L. Kaplan, and J. Han. Graph regularized meta-path based transductive regression in heterogeneous information network. In *SDM*, pages 918–926, 2015.

[33] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding. Community discovery using nonnegative matrix factorization. *DMKD*, 22(3):493–521, 2011.

[34] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD*, pages 824–833, 2007.

[35] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *SIGMOD*, pages 505–516, 2012.

[36] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.

[37] J. Yang, J. J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, pages 1151–1156, 2013.

[38] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *KDD*, pages 927–936, 2009.

[39] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

[40] Y. Zhou and L. Liu. Social influence based clustering of heterogeneous information networks. In *KDD*, pages 338–346. ACM, 2013.

[41] X. Zhu, J. Lafferty, and R. Rosenfeld. *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.