Social Influence Based Clustering of Heterogeneous Information Networks

Yang Zhou  
College of Computing  
Georgia Institute of Technology  
Atlanta, GA 30332  
yzhou@gatech.edu

Ling Liu  
College of Computing  
Georgia Institute of Technology  
Atlanta, GA 30332  
lingliu@cc.gatech.edu

ABSTRACT
Social networks continue to grow in size and the type of information hosted. We witness a growing interest in clustering a social network of people based on both their social relationships and their participations in activity based information networks. In this paper, we present a social influence based clustering framework for analyzing heterogeneous information networks with three unique features. First, we introduce a novel social influence based vertex similarity metric in terms of both self-influence similarity and co-influence similarity. We compute self-influence and co-influence based similarity based on social graph and its associated activity graphs and influence graphs respectively. Second, we compute the combined social influence based similarity between each pair of vertices by unifying the self-similarity and multiple co-influence similarity scores through a weight function with an iterative update method. Third, we design an iterative learning algorithm, SI-CLUSTER, to dynamically refine the K clusters by continuously quantifying and adjusting the weights on self-influence similarity and on multiple co-influence similarity scores toward the clustering convergence. To make SI-CLUSTER converge fast, we transformed a sophisticated nonlinear fractional programming problem of multiple weights into a straightforward nonlinear parametric programming problem of single variable. Our experiment results show that SI-CLUSTER not only achieves a better balance between self-influence and co-influence similarities but also scales extremely well for large graph clustering.

Categories and Subject Descriptors
H.2.8 [Database Applications]: Data Mining

General Terms
Algorithms, Experimentation, Performance

Keywords
Graph Clustering, Heterogeneous Network, Social Influence

1. INTRODUCTION
Social influence studies the impact of a group of people on an individual member of the group by their opinions or actions. Social influence analysis has great potential for understanding the ways in which information, ideas, experiences and innovations are spread across social networks. As more and more people are engaged in social networks, we witness many forms of heterogeneous social networks in which entities are of different types and are interconnected through heterogeneous types of links, representing different kinds of semantic relations. Analyzing and mining heterogeneous social networks can provide new insights about how people interact with and influence each other and why ideas and opinions on different subjects propagate differently on social networks.

Clustering a heterogeneous social network with multiple types of links, entities, static attributes and dynamic and inter-connected activities demands for new clustering models and distance functions to address the following new challenges.

• The large scale heterogeneous social network analysis often displays features of social complexity and involves substantial non-trivial computational cost. For example, a full version of the DBLP bibliography data contains 964, 166 authors, 6, 992 conferences, 363, 352 keywords and 31, 962, 786 heterogeneous links.

• Each type of entity usually associates to one primary social world but participates in many other social worlds, each with domain-specific semantics. How to make good use of the information from various social worlds to provide more informative views of how people influence one another in a given social network? For instance, we may want to utilize the original facebook people network as well as the associated activity networks in the facebook dataset to generate a better clustering of people based on their social influence in terms of both their circle of friends (i.e., self-influence) and their participations in multiple domain specific activity networks (i.e., multiple types of co-influence).

• The information flow between two social worlds may be bidirectional so that we should be careful in differentiating them when we integrate the results from different information networks. For example, Bob may influence his circle of friends (direct or indirect) by his blogs on certain subject and his participation in some tennis tournaments. On the other hand, direct links from a blog (or a tournament) to other blogs (or tournaments) can serve as a recommendation by Bob to its circle of friends.

• As multiple social networks may be from arbitrary domains, it is challenging to efficiently integrate the multiple types of influences from multiple information networks into a unified distance space simultaneously. Moreover, social network clustering can be more meaningful if it is context aware and only the activity networks that are relevant to the context of interest will be utilized to perform the social influence based clustering analysis. With these new challenges in mind, in this paper we develop an innovative social influence based graph clustering approach for heterogeneous information networks, SI-CLUSTER. It captures not only the complex attributes of people (vertices) in the social collaboration network but also the nested and complex relationships between people and other types of entities in different information networks in terms of their participations in different activities of interest.
We perform extensive evaluation on real datasets to demonstrate a dynamic weight tuning method is provided to combine various works by dynamically combining self-influence from social graph and multiple types of co-influence from activity graphs.

2. RELATED WORK

The most closely related work to this research falls into three areas: social influence analysis, heterogeneous social network analysis and graph clustering. Social influence analysis is gaining attention in recent years. [1] proposed the first provable approximation algorithm for maximizing the spread of influence in a social network. [2] proposed a cascading viral marketing algorithm. [3] proposed a heat-diffusion based viral marketing model with top K most influential nodes. [4] used a user’s implicit social graph to generate a friend cluster, given a small seed set of contacts. [5] presented a model in which information can reach a node via the links of the social network or through the influence of external sources.

Graph clustering has attracted active research in the last decade. Most of existing graph clustering techniques have focused on the topological structure based on various criteria, including normalized cuts [11], modularity [12], structural density [13], stochastic flows [14] or clique [15]. K-SNAP [16] and CANAL [17] presented OLAP-style aggregation approaches to summarize large graphs by grouping nodes based on the user-selected attributes. [18] exploited an information-theoretic model for clustering by growing a random seed in a manner that minimizes graph entropy. [19] presented a clustering method which integrates numerical vectors with modularity into a spectral relaxation problem. SA-Cluster [20] and BAGC [21] perform clustering based on both structural and attribute similarities by incorporating attributes as augmented edges to its vertices, transforming attribute similarity to vertex closeness. PathSelClus [22] utilizes limited guidance from users in the form of seeds in some of the clusters and automatically learn the best weights for each meta-path in the clustering process. GenClus [23] proposed a model-based method for clustering heterogeneous networks with different link types and different attribute types.

3. PROBLEM STATEMENT

We consider three types of information networks in defining a social influence based graph clustering method: (1) the social collaboration network, which is the target of graph clustering and typically a social network of people, such as friend network, co-author network, to name a few; (2) the associated activity networks, such as product purchasing activity network, sport activity network or conference activity network; (3) the influence networks representing bipartite graphs connecting social network and activity networks. We formally define the three types of networks as follows.

A social graph is denoted as \( SG = (U, E) \), where \( U \) is the set of vertices representing the members of the collaboration network, such as customers or authors, and \( E \) is the set of edges denoting the collaborative relationships between members of the collaboration network. We use \( N_{SG} \) to represent the size of \( U \), i.e., \( N_{SG} = |U| \).

An activity graph is defined by \( AG_v = (V, S_v) \), where \( v \in V \) denotes an activity vertex in the \( v \)-th associated activity network \( AG_v \), and \( s \in S_v \) is a weighted edge representing the similarity between two activity vertices, such as functional or manufacture similarity. We denote the size of each activity vertex set as \( N_{AG_v} = |V_v| \).
4. INFLUENCE-BASED SIMILARITY

This section describes how to measure the vertex closeness in terms of self-influence and co-influence models. We first utilize heat diffusion model to capture self-influence based similarity between member vertices in the social graph. Then we use heat diffusion model to construct one co-influence model for each influence graph using a probabilistic classification method to compute co-influence similarities of two vertices in the social graph. Finally, we compute pairwise vertex similarities based on the influence similarity matrix and generate an influence-based pairwise similarity matrix on the social graph for each of its N influence graphs.

4.1 Heat Diffusion on Social Graph

Heat diffusion is a physical phenomenon that heat always flows from an object with high temperature to an object with low temperature. In a large social graph SG, experts with many publications often influence other late authors. Consumers purchasing many products may influence other consumers with little purchasing. Thus the spread of influence resembles the heat diffusion phenomenon. Early adopters of a product with many friends or experts on a subject with many coauthors may act as heat sources, transfer their heat to others and diffuse their influence to other majority.

To effectively measure vertex closeness in the social graph in terms of heat diffusion model, we first define the non-propagating heat diffusion kernel on social graph.

**Definition 1.** [Non-propagating Heat Diffusion Kernel on Social Graph] Let $SG = (U; E)$ denote a social graph where $U$ is the set of member vertices and $E$ is the edge set denoting the collaborative relationships between members. Let $\alpha$ be the thermal conductivity (the heat diffusion coefficient) of $SG$. The heat change at vertex $u_i \in U$ between time $t + \Delta t$ and time $t$ is defined by the sum of the heat that it receives from all its neighbors, deducted by what it diffuses.

$$f(t + \Delta t) - f(t) = \alpha \sum_{j: (u_i, u_j) \in E} p_{ij} f(t) - f(0),$$

where $f(t)$ is the vertex $u_i$’s temperature at time $t$, $p_{ij}$ denotes the probability of heat diffusion from $u_i$ to $u_j$, $n_i$ denotes the weight on edge $(u_i, u_j)$, e.g., the number of co-authored publications, and $n_i$ (or $n_j$) denotes the amount of heat/influence that $u_i$ (or $u_j$) has within the social graph, e.g., the number of authored publications. We express the above expression in a diffusion formula matrix in a form.

$$\frac{f(t+\Delta t) - f(t)}{\Delta t} = aHf(t),$$

where $H$ is a $N \times N$ matrix, called a non-propagating heat diffusion kernel on $SG$, as the heat diffusion process is defined in terms of one-hop neighbors of heat source.

$$H_{ij} = \begin{cases} p_{ij}, & (u_i, u_j) \in E, i \neq j, \\ -\tau_i, & i = j, \\ 0, & \text{otherwise.} \end{cases}$$

where $\tau_i = \sum_{j: (u_i, u_j) \in E} p_{ij}$. $\tau_i$ denotes the amount of heat diffused from $u_i$ to all its neighbors.

If we use $H$ to define self-influence similarity between vertices, then the similarity is based on one-hop or direct influence. For those authors who have no joint publications, they are considered to have zero influence on one another, which is unrealistic.

This motivates us to utilize both direct and indirect influence paths between two vertices in computing their vertex similarity. Thus, we define the self-influence similarity using the propagating heat diffusion kernel, where the heat diffusion process continues until vertices’ temperatures converge or the system-defined convergence condition is met. Concretely, by Eq (2), we have the following differential equation when $\Delta t \to 0$.

$$\frac{df(t)}{dt} = aHf(t)$$

Solving this differential equation, we obtain the following Eq (5).

**Definition 2.** [Propagating Heat Diffusion Kernel on Social Graph] Let $\alpha$ denote the thermal conductivity, $H$ be the non-propagating diffusion kernel of $SG$ and $f(0)$ denote an initial heat (influence) column vector at time $0$, which defines the initial heat distribution on $SG$. The vertex’s thermal capacity at time $t$, denoted by $f(t)$, is an exponential function with variable $t$ for constant $f(0)$.

We call $e^{\alpha t}$ as the propagating heat diffusion kernel. It can be expanded as a Taylor series, where $I$ is an identity matrix:

$$e^{\alpha t} = 1 + \alpha t + \frac{\alpha^2 t^2}{2!} + \frac{\alpha^3 t^3}{3!} + \cdots$$
where the heat diffusion reaches convergence, i.e., thermal equilibrium, at time $t$. Since $\alpha^{nt}$ captures both direct and indirect relationships between objects, it reflects the vertex closeness on social graph. We treat it as the self-similarity matrix $W_0$, i.e., $W_0 = \alpha^{nt}$. Here, the thermal conductivity $\alpha$ is a user specific parameter. We use it as a weight factor for the self-influence similarity in the unified similarity. Figure 3 follows the example of Figure 1. In Figure 3 (a), ochre dashed lines and associated blue numbers represent the self-influence similarity by setting $\alpha$ and $t$ equal to 1.

### 4.2 Heat Diffusion on Influence Graphs

We have presented the use of propagating heat diffusion kernel to measure the self-influence vertex closeness on social graph. In this section we describe how we compute pairwise co-influence similarity for vertices in $SG$ based on one of $N$ associated influence graphs.

Similarly, we first need to define the non-propagating heat kernel on an influence graph. By the definition of influence graph in Section 3, we should consider four types of one-hop influence diffusion path in defining the non-propagating heat kernel $H_t$.

**Definition 3.** [Non-propagating Heat Diffusion Kernel on Influence Graphs] We formulate $H_t$ on the influence graph $IG_i$ associated to the social graph $SG$ and the activity graph $AG_i$, by splitting it into four blocks.

$$H_t = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where $B = [B_1, \ldots, B_{NAG_i}]^T$ is a $NAG_i \times AG_i$ matrix representing the social influence of vertices in $AG_i$ on members in $SG$, defined by Eq.(8); $C = [C_1, \ldots, C_{NAG_i}]^T$ is a $NAG_i \times AG_i$ matrix denoting the social influence of members in $SG$ on vertices in $AG_i$, defined by Eq.(9); $A$ is a $NSG \times NSG$ matrix representing the activity similarities, defined by Eq.(10); and $D$ is a $NSG \times NSG$ diagonal matrix.

$$B_{jk} = \begin{cases} \sum_{n} a_{jk}^{nt}, & (u_j, v_k) \in T, \\ 0, & \text{otherwise.} \end{cases}$$

where $n_{jk}$ is the weight on edge $(u_j, v_k)$ and $B_{jk}$ computes the influence of $v_k$ on $v_j$ through $u_j$ and is defined by $n_{jk}$ normalized by the sum of weights on $(u_j, v_k)$ for any $v_k$ in $AG_i$. For example, the influence of a conference vertex on the social graph through an author, say Philip S. Yu, is defined by the number of papers he published in $v_i$ normalized by the total number of papers authored by him and published in any conference of the influence graph.

$$C_{jk} = \begin{cases} \sum_{n} n_{jk}^{nt}, & (v_j, u_k) \in T, \\ 0, & \text{otherwise.} \end{cases}$$

where $n_{jk}$ is the weight on edge $(u_k, v_j)$ and $C_{jk}$ computes the influence of $u_k$ on $AG_i$ through $v_j$ and is defined by $n_{jk}$ (the amount of papers $u_k$ published in $v_j$) normalized by the sum of the weights on $(u_k, v_j)$ for any $u_k$.

In the diagonal matrix $D$, the diagonal entry $D_{jj}$ in each row is equal to $-\tau_j$, where $\tau_j = \sum_{n} c_{jk}^{nt}$, $C_{jk}$, and $\tau_j$ summarizes the influence of vertex $v_j$ on other activity vertices and associated member vertices.

**Definition 4.** [Propagating Heat Diffusion Kernel on Influence Graphs] Let $IG_i$ denote the $i^{th}$ influence graph associated to $SG$ and $AG_i$, $\alpha$ denote the thermal conductivity, $H_t$ denote the non-propagating diffusion kernel of $IG_i$ and $f(0)$ be an initial heat distribution on these activity clusters, denoted by $f_{ic}(0)$ for constant $i$. For each activity vertex $v_k$, there exists one and only one $c_j$ cluster among the $M$ disjoint activity clusters, to which vertex $v_k$ belongs. We have $p_{ik} > 0$ in $f_{ic}(0)$. The last $N_{ICD}$ entries in $f_{ic}(0)$ represent the initial heats of member vertices in $SG$ with all 0s. Thus, the initial heat distribution matrix $f(0)$ is defined as

$$f(0) = [f_{10}(0), f_{20}(0), \ldots, f_{M0}(0)]$$

We argue that two members are similar if both of them participate in many activities in the same clusters. We propose a probability based co-influence classification method to classify members into the activity-based clusters and generate the co-influence similarity between members based on the member in each class. We first use $f_{ij}(0)$ for $1 \leq i, j \leq M$ as the training data and the $\alpha^{nt}$ as the classifier to execute influence propagation to generate member’s probability in each activity-based class. The heat distribution $f_i(t)$ at time $t$ is then given as follow.

$$f_i(t) = f_{ic}(0) f_{ic}(t) \ldots f_{ic}(M) t = \alpha^{nt} f_{ic}(0) f_{ic}(t) \ldots f_{ic}(M)$$

Consider conference classes $DM$ and $DB$ in Figure 3 (c), we have the initial conference influence distribution matrix $f_{conf}(0)$ below.

$$f_{conf}(0) = [f_{DM}(0), f_{DB}(0)]^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}$$

where 2 columns represent the conference classes $DM$ and $DB$ and 11 rows represent six conference vertices ($ICDM$, $KDD$, $SDM$, SIGMOD, VLDDB and ICDE), and five author vertices (Philip S. Yu, Jiawei Han, Charu C. Aggarwal, Kun-Lung Wu and Haixun Wang). By Eq.(14) with $\alpha$ and time $t$ set to 1, we can generate the final heat distribution vectors $f_{conf}(t)$ for them, which serve as their influence-based probabilities of belonging to each of $DM$ and $DB$.

We can further reduce the influence propagation matrix $f_i(t)$ with the size of $NAG_i \times NAG_i \times M$ to a $NAG_i \times M$ matrix $f_i(t)$ by removing the activity rows without loss of quality. Figure 3 (d) shows the
influence distribution $f_{\text{self}}(t)$ represented by blue numbers in the two different conference classes. The larger the number is, the more influence author has on the conference class.

The pairwise vertex closeness is an important measure of clustering quality. Let $W_i$ denote the co-influence vertex similarity matrix for influence graph $IG_i$, $M_i$ be the number of activity classes in $IG_i$, and $f_{\text{self}}(j)(k)$ denote the row-wise normalized influence distribution of member $i_j \in U$ on $IG_i$ at time $t$, i.e., the probability of $i_j$ to $i_k$ in the $m^{th}$ class of $AG_i$. $W_{ij}(k)$ representing the co-influence similarity between members $i_j$ and $i_k$ is defined below.

$$W_{ij}(k) = W_{ij}(k) = 1 - \frac{\sum_{m=1}^{M} (f_{\text{self}}(j)(k) - f_{\text{self}}(k)(j))^2}{\sum_{m=1}^{M} (f_{\text{self}}(j)(k) + f_{\text{self}}(k)(j))^2}$$

The green numbers in Figure 3 (e) represents the co-influence based similarity from the conference influence graph.

### 4.4 Unified Influence-based Similarity Measure

The problem of integrating the influence-based similarities on both social graph and multiple influence graphs into a cohesive and unified similarity measure is quite challenging. In this paper, we propose to use a unified influence-based similarity measure together with an iterative learning algorithm to address this problem.

Let $W_i$ denote the self-influence similarity from the social graph $SG$ with the weight factor $\alpha$. $W_i$ denote the co-influence similarity from the influence graph $IG_i$. Let $W_0 = e^{\alpha M}$, $\omega_i = \sum_{j=1}^{N} \alpha_i > 0$, $\omega_i > 0$, $i = 1, \ldots, N$.

The unified similarity measure between two member vertices in $SG$ is defined based on the set of $N + 1$ influence-based similarities.

$$d(u_i, u_j) = d(u_i, u_j) = \frac{1}{\sum_{k=1}^{N} \omega_k} \sum_{k=1}^{N} \omega_k W_k(i, j)$$

### 5. CLUSTERING ALGORITHM

This section presents our clustering framework, SI-CLUSTER, that partitions a social graph $SG$ based on both self-influence and co-influence similarities through a unified similarity model among $SG$, the activity graphs $AG_i$, and the influence graphs $IG_i$. SI-CLUSTER follows the $K$-Medoids clustering method [25] by using the unified influence-based similarity with the initial weights as an input. At each iteration, we select the most centrally located point in a cluster as a centroid, and assign the rest of points to their closest centroids. The weight update method computes the weighted contributions of each influence-based similarity to both clustering convergence and clustering objective, and updates $N + 1$ weights accordingly after each iteration. This process is repeated until convergence.

#### 5.1 Initialization

We will address two main issues in the initialization step: (1) initial weight setup and (2) cluster centroid initialization.

Choosing a weight assignment randomly often results in incorrect clustering results. In fact, we will prove that there exists one and only one optimal weight assignment to maximize the clustering objective. According to Definition 7 and Theorems 4-7 in Section 5.4, we choose parameter $\beta = 0$ and weights $\alpha = \omega_1 = \ldots = \omega_N = 1$ as an initial input. Thus, the dynamic weight update scheme continuously increases weights to important influence-based similarities and decreases weights or assign zero weights to trivial influence-based similarities at each iteration.

Good initial centroids are essential for the success of partitioning clustering algorithms. A member vertex which has a local maximum of the number of neighbors often can diffuse its heat to many vertices along multiple paths. A centroid-based cluster is thus formed when heat is diffused to the margin of the social graph. Thus, we select such $K$ members as the initial centroids $c_1, \ldots, c_k$.

#### 5.2 Vertex Assignment and Centroid Update

With $K$ centroids in the $i^{th}$ iteration, we assign each vertex $i_j \in U$ to its closest centroid $c^i = \arg\max_{c_k} d(u_j, c_k)$, i.e., a centroid $c^i \in [c_1, \ldots, c_k]$ with the largest unified similarity from $u_j$. When all vertices are assigned to some cluster, the centroid will be updated with the most centrally located vertex in each cluster. To find such a vertex, we first compute the “average point” $\mu_i$ of a cluster $U_i$ in terms of the unified similarity matrix as

$$d(\mu_i, u_j) = \frac{1}{|U_i|} \sum_{u_k \in U_i} d(u_k, u_j), \forall u_j \in U_i$$

Thus, $d(\mu_i, \cdot)$ is the unified average similarity vector for cluster $U_i$. Then we find the new centroid $c_i^{i+1}$ in cluster $U_i$ as

$$c_i^{i+1} = \arg\max_{c_k} |\mu_k| - d(\mu_k, \cdot) - d(\mu_i, \cdot).$$

Therefore, we find the new centroid $c_i^{i+1}$ in the $(i+1)^{th}$ iteration whose unified similarity vector is the closest to the cluster average.

#### 5.3 Clustering Objective Function

The objective of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity. We first define the inter-cluster similarity.

**Definition 5.** [Inter-cluster Similarity] Let $SG = (U, E)$ denote the social graph, $W(i, j)$ denote the unified influence-based similarity between $u_i$ and $u_j$, and $U_i$ and $U_j$ be two clusters of $U$. The inter-cluster similarity between $U_i$ and $U_j$ is defined as follow.

$$d(U_i, U_j) = \sum_{u_i \in U_i} \sum_{u_j \in U_j} W(i, j)$$

This inter-cluster similarity measure is designed to quantitatively measure the extent of similarity between two clusters of $U$.

**Definition 6.** [Graph Clustering Objective Function] Let $SG = (U, E)$ denote a social graph with the weight $\alpha$ and $IG_1, IG_2, \ldots, IG_N$ denote $N$ influence graphs with the weights $\omega_1, \ldots, \omega_N$ where $\omega_i$ is the weight for $IG_i$ and $K$ be a number of clusters. The goal of SI-CLUSTER is to find $K$ partitions $U_i$, $i = 1, \ldots, K$ such that $U = \bigcup_{i=1}^{K} U_i$ and $U_i \cap U_j = \phi$ for $\forall i, j \leq K, i \neq j$, and the following objective function $O(U_i)$ is maximized.

$$O(U_i) = \sum_{\alpha \in U_i} d(u_i, u_j) = \sum_{\alpha \in U_i} \sum_{\beta \in U_i} \sum_{\gamma \in U_i} \sum_{\delta \in U_i} d(\alpha, \beta)$$

subject to $\alpha + \sum_{\gamma \in U_i} \omega_i = |N| + 1$, $\alpha \geq 0$, $\omega_i \geq 0$, $i = 1, \ldots, N$.
Thus the graph clustering problem can be reduced to three subproblems: (1) cluster assignment, (2) centroid update and (3) weight adjustment, each with the goal of maximizing the objective function. The first two problems are common to all partitioning clustering algorithms. Thus we focus on the third subproblem, weight adjustment, in the next subsection.

5.4 Parameter-based Optimization

The objective function of our clustering algorithm is to maximize intra-cluster similarity and minimize inter-cluster similarity. Theorems 1 and 2 prove that our clustering objective is equivalent to maximizing a quotient of two convex functions of multiple variables. It is very hard to perform function trend identification and estimation to determine the existence and uniqueness of solutions. Therefore, we can not directly solve this sophisticated nonlinear fractional programming problem.

Definition 7. Suppose that \(f(\omega_1, \omega_2, \ldots, \omega_N) = \sum_{i=1}^{N} \omega_i \phi(i, j) + \sum_{i=1}^{N} a \omega_i \beta(i, j) \) and \(g(\omega_1, \omega_2, \ldots, \omega_N) = \sum_{i=1}^{N} \omega_i \phi(i, j) + \sum_{i=1}^{N} a \omega_i \beta(i, j) \), the original clustering goal is rewritten as the following optimization problem (NFP).

\[
\max \Omega(U)^{\alpha, \omega_1, \omega_2, \ldots, \omega_N} = f(\omega_1, \omega_2, \ldots, \omega_N) - g(\omega_1, \omega_2, \ldots, \omega_N) \tag{23}
\]

subject to \(a + \sum_{i=1}^{N} \omega_i = N + 1, a \geq 0, \omega_i > 0, i = 1, \ldots, N \).

Lemma 1. Let \(f \) be a function of a single variable on \(R \). Then (1) \(f \) is concave if \(f(x) \) is negative semidefinite for \(x \in R \). (2) \(f \) is convex for \(x \in R \) if \(f \) is strictly concave. (3) \(f \) is convex if \(f(x) \) is positive semidefinite for \(x \in R \). (4) If \(f(x) \) is positive definite for \(x \in S \), \(f \) is strictly convex.

Theorem 1. \(f(\omega_1, \omega_2, \ldots, \omega_N) \) is convex on the set \(S = \{ (\omega_1, \omega_2, \ldots, \omega_N) | \sum_{i=1}^{N} \omega_i = N + 1, 0 \leq \omega_i \leq 1 \} \).

Proof. We first prove that the set \(S \) is a convex set. Suppose that two arbitrary \((n+1)-\)vectors \(x = (\mu_1, \mu_2, \ldots, \mu_{n+1}) \) and \(x' = (v_1, v_2, \ldots, v_{n+1}) \) satisfy the following two constraints: \(\sum_{i=1}^{n+1} \mu_i = N + 1, \mu_i \geq 0, \forall i = 1, \ldots, N + 1 \). For an arbitrary \(\lambda \in [0, 1] \), the \((n+1)-\)vector \(1 - \lambda x + \lambda x' \) is also a feasible solution of this constraint. Thus, \(1 - \lambda x + \lambda x' \) is still in \(S \) and \(S \) is a convex set.

We can then write the Hessian matrix of \(f \) as follows.

\[
\Omega(x_1, x_2, \ldots, x_N) = D \Omega(x_1, x_2, \ldots, x_N) \tag{24}
\]

where \(D \) is the differentiation operator with respect to the \(i^{th} \) argument.

The Hessian becomes \(\Omega(x) = [\frac{\partial^2 f}{\partial x_1^2}, \frac{\partial^2 f}{\partial x_1 \partial x_2}, \ldots, \frac{\partial^2 f}{\partial x_1 \partial x_N} ; \frac{\partial^2 f}{\partial x_2 \partial x_1}, \frac{\partial^2 f}{\partial x_2^2}, \ldots, \frac{\partial^2 f}{\partial x_2 \partial x_N} ; \ldots ; \frac{\partial^2 f}{\partial x_N \partial x_1}, \frac{\partial^2 f}{\partial x_N \partial x_2}, \ldots, \frac{\partial^2 f}{\partial x_N^2}] \). Since \(f(\omega_1, \omega_2, \ldots, \omega_N) \) has only one non-linear term \(\sum_{i=1}^{N} \omega_i \phi(i, j) + \sum_{i=1}^{N} a \omega_i \beta(i, j) \), there is one non-zero term \(\frac{\partial^2 f}{\partial \omega_i \partial \omega_j} = \sum_{i=1}^{N} \omega_i \phi(i, j) + \sum_{i=1}^{N} a \omega_i \beta(i, j) \) in the Hessian matrix. We can easily prove that all of its eigenvalues are non-negative. Thus, it is positive-semidefinite for \(\omega_1, \omega_2, \ldots, \omega_N \in S \) and \(f(\omega_1, \omega_2, \ldots, \omega_N) \) is convex on the set \(S \).

Theorem 2. \(g(\omega_1, \omega_2, \ldots, \omega_N) \) is convex on \(S \) since its Hessian matrix \(\Omega(g) \) is positive-semidefinite for \(\omega_1, \omega_2, \ldots, \omega_N \in S \). The detailed proof is omitted due to space limit. This theorem can be tested by using the above-mentioned similar method.

Theorem 3. The NFP problem is equivalent to a polynomial programming problem with polynomial constraints (PFP).

\[
\max g(\omega_1, \omega_2, \ldots, \omega_N) \quad \text{subject to } 0 \leq \gamma < 1/\beta(\omega_1, \omega_2, \ldots, \omega_N), \quad a + \sum_{i=1}^{N} \omega_i = N + 1, \quad a > 0, \omega_i > 0, i = 1, \ldots, N.
\]

Proof. If \((\omega_1, \omega_2, \ldots, \omega_N) \) is a feasible solution of NFP, then \(f(\omega_1, \omega_2, \ldots, \omega_N) \) is a maximum value of NPPP, i.e., \(f(\omega_1, \omega_2, \ldots, \omega_N) < f(\omega_1, \omega_2, \ldots, \omega_N) \) if \(f(\omega_1, \omega_2, \ldots, \omega_N) \) is not a feasible solution of NFP. Conversely, if \((\omega_1, \omega_2, \ldots, \omega_N) \) solves NFP, then \(f(\omega_1, \omega_2, \ldots, \omega_N) \) is a feasible solution of NFP. Thus the graph clustering problem can be reduced to three subproblems, weight adjustment, each with the goal of maximizing the objective function.

5.5 Adaptive Weight Adjustment

The procedure of solving this NFP optimization problem includes two parts: (1) find such a reasonable parameter \(\beta \) \((F(\beta) = 0)\), making NFP equivalent to NFP; (2) given the parameter \(\beta \), solve...
Algorithm 1 Social Influence-based Graph Clustering

Input: a social graph \( SG \), multiple influence graphs \( IG_i \), a cluster number \( K \), initial weights \( \alpha = \beta_1 = \ldots = \beta_K = 1 \) and a parameter \( \beta = 0 \).
Output: \( K \) clusters \( U_1, \ldots, U_K \).
1: Calculate \( W_0, W_1, W_2, \ldots, W_N \), and \( W_i \);
2: Select \( K \) initial centroids with a local maximum of #neighbors;
3: Repeat until the objective function \( F(\beta) \) converges:
   4: Assign each vertex \( u_i \) to a cluster \( C^* \) with a centroid \( c^* \) where \( c^* = \arg \max_{c} d(a(c), c^*) \);
   5: Update the cluster centroids with the most centrally located point in each cluster;
   6: Solve the NPPP of \( F(\beta) \);
   7: Update \( \alpha, \omega_1, \ldots, \omega_K \);
   8: Refine \( \beta = \frac{1}{N} (\frac{1}{N} \sum_{i=1}^{N} d(a(c), c^*) \big) \); \( \beta \) helps the algorithm enter the next round. The algorithm repeats the above-mentioned iterative procedure until \( F(\beta) \) converges to 0.

5.6 Clustering Algorithm

By assembling different pieces together, we provide the pseudo code of our clustering algorithm - SI-Cluster in Algorithm 1.

**Theorem 8.** The objective function in Algorithm 1 converges to a local maximum in a finite number of iterations.

Proof. Existing work has studied the convergence properties of the partitioning approach to clustering, such as K-Means [27]. Our clustering follows a similar approach. So the cluster assignment and centroid update steps improve the objective function. In addition, we have explained that nonlinear parametric programming optimization also fast converges a local maximum value. Therefore, the objective function keeps increasing (but \( F(\beta) \) keeps decreasing) and converges to a local maximum in a finite number of iterations.

6. EXPERIMENTAL EVALUATION

We have performed extensive experiments to evaluate the performance of SI-Cluster on real graph datasets.

6.1 Experimental Datasets

We use a full version of the DBLP bibliography data with 96, 166 authors (dblp.xml, 836MB, 05/21/2011). We build a social graph where vertices represent authors and edges represent their collaboration relationships, and two associated activity graphs: conference graph and keyword graph. We make use of a multityped clustering framework. RankClus [24], to partition both conferences and keywords into clusters respectively. According to the conference’s or keyword’s clustering distribution and ranking in each cluster, we calculate the similarities between conferences or keywords. The two associated influence graphs capture how authors in the social graph interact with the activity networks. We also use a smaller DBLP collaboration network with 100,000 highly prolific authors. The third dataset is the Amazon product co-purchasing network with 20,000 products. The two activity networks are product category graph and customer review graph.

6.2 Comparison Methods and Evaluation

We compare SI-Cluster with three recently developed representative graph clustering algorithms, BAGC [21], SA-Cluster [20] and Inc-Cluster [28], and one baseline clustering algorithm, W-Cluster. The last three algorithms integrate entity, link and static attribute information into a unified model. SI-Cluster is our proposed algorithm which incorporates not only links, entities, static attributes but also multiple types of dynamic and inter-connected activities into a unified influence-based model. BAGC constructs a Bayesian probabilistic model to capture both structural and attribute aspects. Both SA-Cluster and Inc-Cluster combine both structural and attribute similarities in the clustering decisions by estimating the importance of attributes. W-Cluster combines structural and attribute similarities using the equal weighting factors.

**Evaluation Measures**

We use three measures to evaluate the quality of clusters \( \{U_i\}_{i=1}^{K} \) generated by different methods. The definitions of the metrics are given as follows.

\[
\text{density}(U_i) = \frac{\sum_{(u_i, u_j) \in E} |U_i|}{|E|} \quad \text{(26)}
\]
\[
\text{entropy}(U_i) = \frac{1}{|U_i|} \sum_{a \in U_i} \omega_a \sum_{c \in U} \text{entropy}(a, U_i) \quad \text{(27)}
\]
\[
\text{DBI}(U_i) = \frac{1}{K} \sum_{i=1}^{K} \frac{d(c_i, c_j)}{\sigma_i + \sigma_j} \quad \text{(28)}
\]

where \( \omega_a \) is the weight of influence graph \( IG_i \), \( \text{entropy}(a, U_i) \) is the percentage of vertices in cluster \( U_i \) which participate in the \( i \)-th activity in \( IG_i \) (or have value \( a \) on \( a \)). \( \text{entropy}(U_i) \) measures the weighted entropy from all influence graphs (or attributes) over \( K \) clusters.

**Davies-Bouldin Index (DBI)** measures the uniqueness of clusters with respect to the unified similarity measure.

\[
\text{DBI}(U_i) = \frac{1}{K} \sum_{i=1}^{K} \frac{d(c_i, c_j)}{\sigma_i + \sigma_j}
\]

where \( c_i \) is the centroid of \( U_i \), \( d(c_i, c_j) \) is the similarity between \( c_i \) and \( c_j \), \( \sigma_i \) the average similarity of vertices in \( U_i \) to \( c_i \).

6.3 Cluster Quality Evaluation

Figure 4 (a) shows the density comparison on Amazon 20,000 Products by varying the number of clusters \( K = 40, 60, 80, 100 \). The density values by SI-Cluster, BAGC, Inc-Cluster and SA-Cluster remains 0.89 or higher even when \( k \) is increasing. This demonstrates that these methods can find densely connected components. The density values of W-Cluster is relatively lower, in the range of 0.72-0.85 with increasing \( K \), showing that the generated clusters are a very loose intra-cluster structure. Figure 4 (b) shows the entropy comparison on Amazon 20,000 Products with \( K = 40, 60, 80, 100 \). SI-Cluster has the lowest entropy, while other four algorithms have a much higher entropy than SI-Cluster, since SI-Cluster considers not only static attributes but also multiple types of dynamic and inter-connected activities during the clustering process. Other methods can not handle dynamic activities and only treat them as static and isolated attributes. Figures 4 (c) shows the DBI comparison on Amazon 20,000 Products with different \( K \) values. SI-Cluster has the lowest DBI of around 0.0000008, while other methods have a much higher DBI than SI-Cluster. This demonstrates that SI-Cluster can achieve both high
intra-cluster similarity and low inter-cluster similarity. This is because SI-Cluster integrates self-influence similarity as well as co-influence similarity with the optimal weights assignment by parameter-based optimization. It fully utilizes the connections between activities and the interactions between members and activities so that the generated clusters have not only similar collaborative patterns but also similar interaction patterns with activities.

Figures 5 (a), (b) and (c) show density, entropy and DBI on DBLP with 100,000 authors when we set $K = 400, 600, 800, 1000$. These three figures have similar trends with Figures 4 (a), (b) and (c) respectively. As shown in the figures, SI-Cluster achieves high density values ($>0.63$), which is slightly lower than that of BAGC since the probabilistic clustering method partitions vertices into each possible cluster so that the density value by it often increases with $K$. SI-Cluster achieves a very low entropy around 2.86-3.04, which is obviously better than the other methods ($>6.35$). As $K$ increases, the entropy by SI-Cluster remains stable, while the density of SI-Cluster decreases. In addition, SI-Cluster achieves the lowest DBI ($<0.00005$) among different methods, while the DBI values by other methods are obviously larger than $>0.00005$.

Figures 6 (a), (b) and (c) show density, entropy and DBI comparisons on DBLP with 964,166 authors by varying $K = 4000, 6000, 8000, 10000$. Other four methods except SI-Cluster do not work on this large dataset due to the “out of memory” problem with our 8G main memory machine. However, SI-Cluster still shows good performance with varying $K$. It achieves similar high density values ($>0.55$), much lower entropy of about 2.45, and very low DBI ($<0$) for different $K$.

### 6.4 Clustering Efficiency Evaluation

Figures 7 (a), (b) and (c) show the clustering time on Amazon 20,000 Products, DBLP 100,000, and 964,166 authors respectively. SI-Cluster outperforms all other algorithms in all experiments. When facing with an extremely large dataset, such as DBLP964,166, other algorithms cannot work due to the “out of memory” error, while SI-Cluster scales well with large graphs and shows good performance with varying $K$. We make the following observations on the runtime costs of different methods. First, SA-Cluster is obviously worse than other methods since it needs to perform the repeating random walk distance calculation during each iteration of the clustering process and the distance computation takes more than 80% of the total clustering time. Second, Inc-Cluster, an optimized version of SA-Cluster, is much slower than SI-Cluster, BAGC and W-Cluster since it still needs to incrementally calculate the random walk distance. Third, although W-Cluster compute the random walk distance only once, it still runs on a large scale matrix.

### 6.5 Clustering Convergence

Figure 8 (a) shows the trend of clustering convergence in terms of the $F(\beta)$ value on DBLP 964, 166 Authors. The $F(\beta)$ value keeps decreasing and has a convex curve when we iteratively perform the tasks of vertex assignment, centroid update and weight adjustment during the clustering process. $F(\beta)$ converges very quickly, usually in three iterations. These are consistent with Theorems 4-7. Figure 8 (b) shows the trend of weight updates on DBLP 964, 166 Authors with different $K$ values: the social graph (red curve), the conference influence graph (green curve) and the keyword influence graph (blue curve). We observe that the graph weights converge as the clustering process converges. An interesting phenomenon is that both the social weight and the keyword weight are increasing but the conference weight is decreasing with more iterations. A reasonable explanation is that people who have many publications in the same conferences may have different research topics but people who have many papers with the same keywords usually have the same research topics, and thus have a higher collaboration probability as co-authors.

### 6.6 Case Study

We examine some details of the experiment results on DBLP 964, 166 Authors when we set $k = 100$ for both conferences and keywords. Table 1 (a) shows author’s influence score based on the social influence propagation between authors and keyword partitions. We only present most prolific DBLP experts in the area of data mining or database. When social influence propagation converges, each row represents the influence distribution of an author in each keyword category. We can look upon this influence distribution as a probability based clustering result. On the other hand, each column specifies the influence distribution of different authors in the same keyword category. This influence distribution is considered as a local ranking result.
(a) Influence Scores Based on All Keywords

<table>
<thead>
<tr>
<th>Author</th>
<th>Cluster 1 (DB)</th>
<th>Cluster 2 (DM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Christos Faloutsos</td>
<td>0.0465</td>
<td>0.0746</td>
</tr>
<tr>
<td>Jiawei Han</td>
<td>0.0585</td>
<td>0.0960</td>
</tr>
<tr>
<td>Vipin Kumar</td>
<td>0.0146</td>
<td>0.0545</td>
</tr>
<tr>
<td>Bing Liu</td>
<td>0.0153</td>
<td>0.0511</td>
</tr>
<tr>
<td>David Maier</td>
<td>0.0474</td>
<td>0.0079</td>
</tr>
<tr>
<td>Hector Garcia-Molina</td>
<td>0.0603</td>
<td>0.0847</td>
</tr>
<tr>
<td>M. Tamer Ozu</td>
<td>0.0408</td>
<td>0.0111</td>
</tr>
<tr>
<td>Jian Pei</td>
<td>0.0386</td>
<td>0.0653</td>
</tr>
<tr>
<td>Philip S. Yu</td>
<td>0.0606</td>
<td>0.0991</td>
</tr>
</tbody>
</table>

(b) Influence Scores Based on Selected Top Conferences

<table>
<thead>
<tr>
<th>Author</th>
<th>AI Cluster</th>
<th>DB Cluster</th>
<th>DM Cluster</th>
<th>IR Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Christos Faloutsos</td>
<td>0.0120</td>
<td>0.4267</td>
<td>0.3950</td>
<td>0.1771</td>
</tr>
<tr>
<td>Jiawei Han</td>
<td>0.0885</td>
<td>0.3724</td>
<td>0.3766</td>
<td>0.1628</td>
</tr>
<tr>
<td>Vipin Kumar</td>
<td>0.2511</td>
<td>0.1342</td>
<td>0.5198</td>
<td>0.0949</td>
</tr>
<tr>
<td>Bing Liu</td>
<td>0.2648</td>
<td>0.1001</td>
<td>0.4004</td>
<td>0.2347</td>
</tr>
<tr>
<td>David Maier</td>
<td>0.1570</td>
<td>0.8290</td>
<td>0.0117</td>
<td>0.0023</td>
</tr>
<tr>
<td>Hector Garcia-Molina</td>
<td>0.0031</td>
<td>0.8217</td>
<td>0.0055</td>
<td>0.0167</td>
</tr>
<tr>
<td>M. Tamer Ozu</td>
<td>0.0917</td>
<td>0.5506</td>
<td>0.1080</td>
<td>0.3979</td>
</tr>
<tr>
<td>Jian Pei</td>
<td>0.0876</td>
<td>0.3768</td>
<td>0.3717</td>
<td>0.1639</td>
</tr>
<tr>
<td>Philip S. Yu</td>
<td>0.0972</td>
<td>0.3504</td>
<td>0.3783</td>
<td>0.1761</td>
</tr>
</tbody>
</table>

Table 1: Influence Scores of Authors Based on Conference and Keyword Partitions

(a) shows author’s influence score normalized by conference partitions for each author, i.e., a better probability based clustering result.

7. CONCLUSIONS

In this paper, we present a social influence based clustering framework for heterogeneous information networks. First, we integrate different types of links, entities, static attributes and dynamic activities from different networks into a unifying influence-based model. Second, an iterative learning algorithm is proposed to dynamically refine the $K$ clusters by continuously quantifying and adjusting the weights on multiple influence-based similarity scores toward the clustering convergence. Third, we transform a sophisticated nonlinear fractional programming problem of multiple weights in a straightforward nonlinear parametric programming problem of single variable to speed up the clustering process.

Acknowledgement. This work is partially funded by grants from NSF CISE NetSE program and SaTC program and Intel Science and Technology Center on Cloud Computing.

8. REFERENCES