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 towards 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

Keywords

Graph Clustering, Heterogeneous Network, Social Influence

1. INTRODUCTION

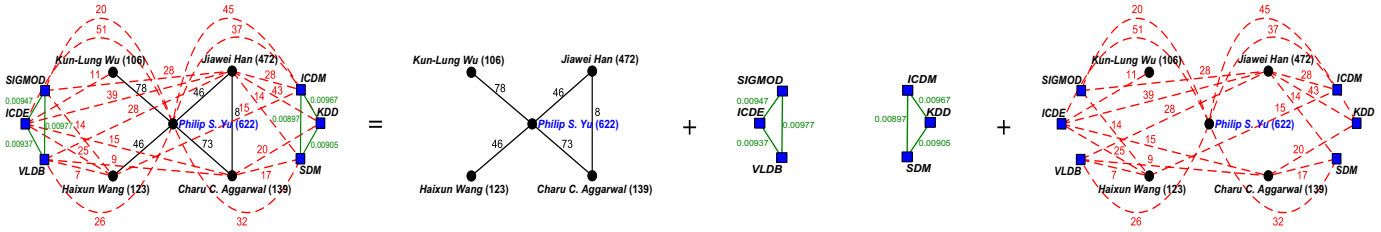
Social influence studies the impact of a group of people on an individual member of the group by their opinions or actions. Social influence analysis has great potential for understanding the ways in which information, ideas, experiences and innovations are spread across social networks. As more and 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 entities 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.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.



(a) Heterogeneous Graph (b) Social Graph (c) Activity Graph (d) Influence Flow

Figure 1: A Heterogeneous Network Example from DBLP

Concretely, we categorize the social influence based graph model into three categories: (1) the topological structure of the social network or the activity networks, (2) the single-valued or multi-valued vertex properties that represent relatively stable and static states of vertices in the social network (such as name, sex, age, and multiple education degrees a person may achieve), (3) the nested and complex relationships between the social network and the activity networks (such as multiple activities one may have participated). We show that the social influence based graph clustering for heterogeneous networks demands for a dynamic graph clustering method in contrast to conventional graph clustering algorithms. SI-Cluster is designed to cluster large social network with two new criteria: (1) it takes into account both the complex vertex properties and the topological structure to define the initial influence of a vertex and the weights of its influence propagation to its circle of friends; (2) it computes pairwise vertex closeness by considering not only the social influence patterns (influence-based similarities) based on both direct and indirect social connections existing in the relevant social and activity networks but also the potentially new interactions that have high propagation probabilities based on the existing interactions. A unique characteristics of SI-Cluster is its ability of integrating the self-influence and multiple types of co-influences into a unified influence-based similarity measure through iteratively clustering and dynamic weight tuning mechanism.

This paper makes the following original contributions.

- We integrate different types of links, entities, static attributes and dynamic activities from different networks into a unified influence-based model through the intra-network or inter-network social influences.
- We compute influence-based vertex similarity in terms of heat diffusion based influence propagation on both social graph (self-influence) and each of activity graphs (co-influence).
- A dynamic weight tuning method is provided to combine various influence-based similarities through an iterative learning algorithm, SI-Cluster, for social influence based graph clustering. To make the clustering process converge fast, a sophisticated nonlinear fractional programming problem with multiple weights is transformed to a straightforward parametric programming problem of a single variable.
- We perform extensive evaluation on real datasets to demonstrate that SI-Cluster can partition the graph into high-quality clusters with cohesive structures and homogeneous social influences.

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.

Recent works on heterogeneous social network analysis [6–10] combine links and content into heterogeneous information networks to improve the quality of querying, ranking and clustering. [6] proposed a method to model a relational database containing both attributes and links. [7] proposed to learn an optimal linear combination of different relations on heterogeneous social networks in terms of their importance on a certain query. [9] groups objects into pre-specified classes, while generating the ranking information for each type of object in a heterogeneous information network. [10] presented a query-driven discovery system for finding semantically similar substructures in heterogeneous networks.

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.

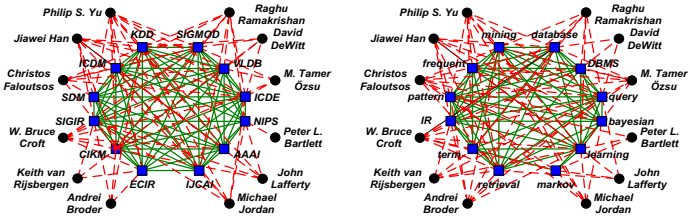
To our knowledge, this work is the first one to address the problem of social influence based clustering over heterogeneous networks by dynamically combining self-influence from social graph and multiple types of co-influence from activity graphs.

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 $AG_i = (V_i, S_i)$, where $v \in V_i$ denotes an activity vertex in the i^{th} associated activity network AG_i, and $s \in S_i$ 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 $N_{AG_i} = |V_i|$.



(a) Conference Influence Graph (b) Keyword Influence Graph

Figure 2: An Illustrating Example of Influence Graphs

An *influence graph* is denoted as $IG_i = (U, V_i, S_i, T_i)$, where U, V_i and S_i have the same definitions in the social graph SG and the activity graph AG_i respectively. Every edge $t \in T_i$, denoted by (u, v), connecting a member vertex $u \in U$ to an activity vertex $v \in V_i$, representing an influence flow between SG and AG_i, such as a purchasing or publishing activity. Thus, IG_i is a bipartite graph.

Given a social graph SG, multiple activity graphs AG_i and various influence graphs IG_i ($1 \le i \le N$), the problem of **Social Influence-based graph Clustering** (SI-Cluster) is to partition the member vertices U into K disjoint clusters U_i, where $U = \bigcup_{i=1}^{K} U_i$ and $U_i \bigcap U_j = \phi$ for $\forall 1 \le i, j \le K, i \ne j$, to ensure the clustering results in densely connected groups and each has vertices with similar activity behaviors. A desired clustering result should achieve a good balance between the following two properties: (1) vertices within one cluster should have similar collaborative patterns among themselves and similar interaction patterns with activity networks; (2) vertices in different clusters should have dissimilar collaborative patterns and dissimilar interaction patterns with activities.

Figure 1 (a) provides an illustrating example of a heterogeneous information network extracted from the DBLP dataset. It consists of two types of entities: authors and conferences and three types of links: co-authorship, author-conference, conference similarity. In our SI-Cluster framework, we reorganize a heterogeneous information network into a social graph, multiple activity graphs and multiple influence graphs without loss of information. The heterogeneous network in Figure 1 (a) is divided into three subgraphs: a social collaboration graph of authors, a conference activity graph, and an influence graph about author's publishing activity in conferences, as shown in Figures 1 (b), (c) and (d), respectively. A red number associated with a red dashed edge quantifies the number of publications that an author published in a conference. A green number on a green edge measures the similarity score between conferences. For ease of presentation, we removed the conference similarities with less than 0.005. A number of mechanisms can be used to compute similarity of conferences. We use RankClus [24] to partition activities into clusters. According to activity's clustering distribution and ranking in each cluster, we calculate the similarities between activities in activity graph. Black numbers in the bracket represent the total amount of publications of an author. Other black numbers on co-author edges denote the number of co-authored papers. A more complex example of influence graph with 12 authors and 12 conferences (or keywords) is presented in Figure 2.

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 α 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.

$$\frac{f_i(t+\Delta t) - f_i(t)}{\Delta t} = \alpha \sum_{j:(u_i,u_j)\in E} p_{ij}(f_j(t) - f_i(t)),\ p_{ij} = \begin{cases} \frac{n_{ij}}{\sqrt{n_i n_j}}, & (u_i,u_j) \in E_0, \\ 0, & otherwise. \end{cases} \tag{1}$$

where $f_i(t)$ 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_{ij} 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 heat diffusion formulation in a matrix form.

$$\frac{\mathrm{f}(t+\Delta t) - \mathrm{f}(t)}{\Delta t} = \alpha \mathrm{H} \mathrm{f}(t) \tag{2}$$

where H is a $N_{SG} \times N_{SG}$ 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.

$$H_{ij} = \begin{cases} p_{ij}, & (u_i,u_j) \in E, i \ne j, \\ -\tau_i, & i = j, \\ 0, & otherwise. \end{cases} \tag{3}$$

where $\tau_i = \sum_{(u_i,u_j)\in E, j\ne i} p_{ij}$. τ_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 $\Delta t \to 0$.

$$\frac{d\mathrm{f}(t)}{dt} = \alpha \mathrm{H} \mathrm{f}(t) \tag{4}$$

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

Definition 2. [Propagating Heat Diffusion Kernel on Social Graph] Let α 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).

$$\mathrm{f}(t) = e^{\alpha t \mathrm{H}} \mathrm{f}(0) \tag{5}$$

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

$$e^{\alpha t \mathrm{H}} = \mathrm{I} + \alpha t \mathrm{H} + \frac{\alpha^2 t^2}{2!}\mathrm{H}^2 + \frac{\alpha^3 t^3}{3!}\mathrm{H}^3 + \cdots \tag{6}$$

where the heat diffusion reaches convergence, i.e., thermal equilibrium, at time t. Since $e^{\alpha t \mathrm{H}}$ 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., $\mathrm{W}_0 = e^{\alpha t \mathrm{H}}$. Here, the thermal conductivity α 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 α 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 to 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 $\mathrm{H_i}$.

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

$$\mathrm{H_i} = \begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} \tag{7}$$

where $\mathrm{B} = [\mathrm{B}_1, \cdots, \mathrm{B}_{\mathrm{N_{AG_i}}}]^T$ is a $N_{AG_i} \times N_{SG}$ matrix representing the social influence of vertices in AG_i on members in SG, defined by Eq.(8); $\mathrm{C} = [\mathrm{C}_1, \cdots, \mathrm{C}_{\mathrm{N_{SG}}}]^T$ is a $N_{SG} \times N_{AG_i}$ matrix denoting the social influence of members in SG on vertices in AG_i, defined by Eq.(9); A is an $N_{AG_i} \times N_{AG_i}$ matrix representing the activity similarities, defined by Eq.(10); and D is a $N_{SG} \times N_{SG}$ diagonal matrix.

$$B_{jk} = \begin{cases} \dfrac{n_{jk}}{\sum_{l=1}^{N_{AG_i}} n_{lk}}, & (u_k, v_j) \in T_i, \\ 0, & otherwise. \end{cases} \tag{8}$$

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

$$C_{jk} = \begin{cases} \dfrac{n_{jk}}{\sum_{l=1}^{N_{SG}} n_{lk}}, & (u_j, v_k) \in T_i, \\ 0, & otherwise. \end{cases} \tag{9}$$

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

$$A_{jk} = \begin{cases} n_{jk}, & (v_j, v_k) \in S_i, \\ -\tau_j, & j = k, \\ 0, & otherwise. \end{cases} \tag{10}$$

where n_{jk} represents the similarity between two activity vertices v_j and v_k in the activity graph. $\tau_j = \sum_{(v_j, v_l) \in S_i, l \neq j} A_{jl} + \sum_{(u_l, v_j) \in T_i} B_{jl}$ where τ_j summarizes the influence of activity vertex v_j on other activity vertices and associated member vertices.

In the diagonal matrix D, the diagonal entry D_{jj} in each row is equal to $-\tau_j$ where $\tau_j = \sum_{(u_j, v_l) \in T_i} C_{jl}$. τ_j summarizes the influence of member vertex u_j on all activity 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, α denote the thermal conductivity, $\mathrm{H_i}$ denote the non-propagating diffusion kernel of IG_i and f(0) be an initial heat distribution on IG_i. The vertex's thermal capacity at time t is defined by an exponential function f(t) with variable t for constant f(0).

$$\mathrm{f_i}(t) = e^{\alpha t \mathrm{H_i}} \mathrm{f_i}(0) \tag{11}$$

where i represents the i^{th} influence graph. $e^{\alpha t \mathrm{H_i}}$ can be expanded as a Taylor series.

$$e^{\alpha t \mathrm{H_i}} = \mathrm{I} + \alpha t \mathrm{H_i} + \frac{\alpha^2 t^2}{2!} \mathrm{H_i^2} + \frac{\alpha^3 t^3}{3!} \mathrm{H_i^3} + \cdots \tag{12}$$

where I is a $(N_{AG_i} + N_{SG}) \times (N_{AG_i} + N_{SG})$ identity matrix.

Figure 3 (b) shows the propagating heat diffusion kernel $e^{\alpha t \mathrm{H_{conf}}}$ for the conference influence graph in our running example, where both α and t are set to 1. For presentation clarity, we only show the bidirectional influence flow between authors and conferences with value less than 0.02 in $e^{\alpha t \mathrm{H_{conf}}}$. Associated blue numbers and green numbers quantify the influence flows from author to conference and the influence flows from conference to author respectively.

4.3 Co-influence Model

We have defined the propagating heat diffusion kernel $e^{\alpha t \mathrm{H_i}}$ for the influence graph IG_i $(1 \leq i \leq N)$. According to Eq.11, in order to conduct heat diffusion on an influence graph and compute pairwise co-influence similarity, we need both $e^{\alpha t \mathrm{H_i}}$ and $\mathrm{f_i}(0)$ on IG_i. $\mathrm{f_i}(0)$ defines the heat sources from which the propagating heat kernel starts its diffusion process.

We observe that the co-influence between a pair of member vertices in the social graph can only be established through their interactions with activity vertices in one of the activity graphs. To make good use of the topological information of AG_i, find good heat sources from AG_i and reduce the commotional cost for large-scale activity graph, we propose to start by partitioning AG_i into M_i disjoint activity clusters, denoted by c_{i1}, c_{i2}, ..., c_{iM_i}. Based on these activity clusters, the initial heat distribution column vector with the size of $(N_{AG_i} + N_{SG}) \times 1$ is defined as follow.

$$\mathrm{f_{ij}}(0) = (p_{ij1}, p_{ij2}, \cdots, p_{ijN_{AG_i}}, 0, 0, \cdots, 0)^T \tag{13}$$

where p_{ijk} is the probability of activity vertex v_k belonging to cluster c_{ij} $(1 \leq k \leq N_{AG_i},\ 1 \leq j \leq M_i)$. If $p_{ijk} > 0$, then the activity vertex v_k in cluster c_{ij} is chosen as an initial heat source. Note that for each activity vertex v_k, there exists one and only one c_{ij} cluster among the M_i disjoint activity clusters, to which vertex v_k belongs. Thus we have $p_{ijk} = 1$ in $\mathrm{f_{ij}}(0)$. The last N_{SG} entries in $\mathrm{f_{ij}}(0)$ represent the initial heats of member vertices in SG with all 0s. Thus, the initial heat distribution matrix $\mathrm{f_i}(0)$ is defined as $[\mathrm{f_i}(0) = [\mathrm{f_{i1}}(0), \mathrm{f_{i2}}(0), \cdots, \mathrm{f_{iM_i}}(0)]$.

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 distribution in each class. We first use $\mathrm{f_{ij}}(0)$ $(1 \leq j \leq M_i)$ as the training data and the $e^{\alpha t \mathrm{H_i}}$ as the classifier to execute influence propagation to generate member's probability in each activity-based class. The heat distribution $\mathrm{f_i}(t)$ at time t is then given as follow.

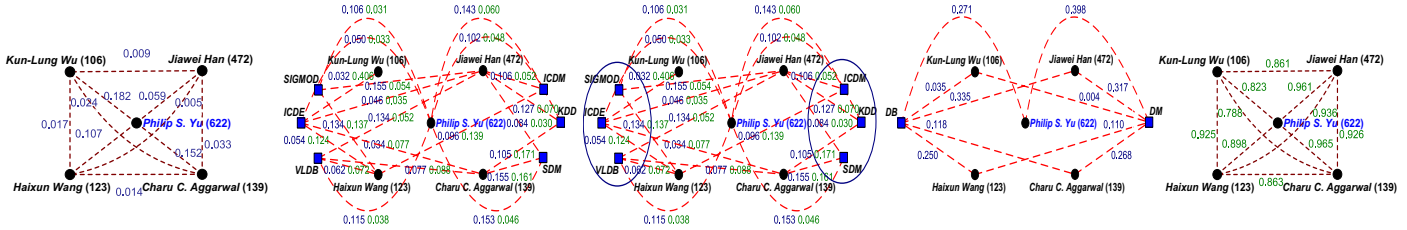
$$\mathrm{f_i}(t) = [\mathrm{f_{i1}}(t), \mathrm{f_{i2}}(t), \cdots, \mathrm{f_{iM_i}}(t)] = e^{\alpha t \mathrm{H_i}} [\mathrm{f_{i1}}(0), \mathrm{f_{i2}}(0), \cdots, \mathrm{f_{iM_i}}(0)] \tag{14}$$

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

$$\mathrm{f_{conf}}(0) = [\mathrm{f_{DM}}(0), \mathrm{f_{DB}}(0)] = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T \tag{15}$$

where 2 columns represent the conference classes *DM* and *DB* and 11 rows represent six conference vertices (*ICDM*, *KDD*, *SDM*, *SIGMOD*, *VLDB* and *ICDE*), and five author vertices (*Philip S. Yu*, *Jiawei Han*, *Charu C. Aggarwal*, *Kun-Lung Wu* and *Haixun Wang*). By Eq.(14) with α and time t set to 1, we can generate the final heat distribution vectors $\mathrm{f_{conf}}(t)$ 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 $\mathrm{f_i}(t)$ with the size of $(N_{AG_i} + N_{SG}) \times M_i$ to a $N_{SG} \times M_i$ matrix $\mathrm{f_i'}(t)$ by removing the activity rows without loss of quality. Figure 3 (d) shows the



(a) Self-influence Similarity (b) Kernel on Influence Graph (c) Activity Partition (d) Influence Propagation (e) Co-influence Similarity

Figure 3: Co-influence Model

influence distribution $f'_{conf}(t)$ 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 $f'_{im}(t)(j)$ denote the row-wise normalized influence distribution of member $u_j \in U$ on IG_i at time t, i.e., the probability of u_j in the m^{th} class of AG_i. $W_i(j,k)$ representing the co-influence similarity between members u_j and u_k is defined below.

$$W_i(j,k) = W_i(k,j) = 1 - \frac{\sqrt{\sum_{m_i=1}^{M}(f'_{im}(t)(j) - f'_{im}(t)(k))^2}}{\sum_{m_i=1}^{M} f'_{im}(t)(j) + f'_{im}(t)(k)} = 1 - \frac{\sqrt{\sum_{m_i=1}^{M}(p_{im(N_{AG_i}+j)} - p_{im(N_{AG_i}+k)})^2}}{\sum_{m_i=1}^{M} p_{im(N_{AG_i}+j)} + p_{im(N_{AG_i}+k)}} \quad (16)$$

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_0 denote the self-influence similarity from the social graph SG with the weight factor α, W_i denote the co-influence similarity from the influence graph IG_i ($1 \le i \le N$) with the weight ω_i. The unified similarity function W is defined as follow.

$$W = W_0 + \omega_1 W_1 + \cdots + \omega_N W_N \quad (17)$$

where $W_0 = e^{\alpha t H}$, $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$.

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

$$d(u_i,u_j) = W(i,j) = e^{\alpha t H}(i,j) + \omega_1 W_1(i,j) + \cdots + \omega_N W_N(i,j) = \sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j) \quad (18)$$

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 $\{c_1^0, ..., c_K^0\}$.

5.2 Vertex Assignment and Centroid Update

With K centroids in the t^{th} iteration, we assign each vertex $u_i \in U$ to its closest centroid $c^* = argmax_{c_j^t} d(u_i, c_j^t)$, *i.e.*, a centroid $c^* \in \{c_1^t, ..., c_K^t\}$ with the largest unified similarity from u_i. 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" $\overline{u_i}$ of a cluster U_i in terms of the unified similarity matrix as

$$d(\overline{u_i}, u_j) = \frac{1}{|U_i|} \sum_{u_k \in U_i} d(u_k, u_j), \forall u_j \in U_i \quad (19)$$

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

$$c_i^{t+1} = argmin_{u_j \in U_i} \|d(u_j, :) - d(\overline{u_i}, :)\| \quad (20)$$

Therefore, we find the new centroid c_i^{t+1} in the $(t+1)^{th}$ 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 $SG = (U, E)$ be the social graph, $W(i,j)$ denote the unified influence-based similarity between u_i and u_j, and U_p and U_q be two clusters of U. The inter-cluster similarity between U_p and U_q is defined as follow.

$$d(U_p, U_q) = \sum_{u_i \in U_p, u_j \in U_q} d(u_i, u_j) = \sum_{u_i \in U_p, u_j \in U_q} W(i,j) \quad (21)$$

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 α and $IG_1, IG_2, \ldots, IG_N$ denote N influence graphs with the weights $\omega_1, \ldots, \omega_N$ where ω_i is the weight for IG_i, and K be a number of clusters. The goal of SI-Cluster is to find K partitions $\{U_i\}_{i=1}^K$ such that $U = \bigcup_{i=1}^K U_i$ and $U_i \bigcap U_j = \phi$ for $\forall 1 \le i, j \le K, i \ne j$, and the following objective function $O(\{U_l\}_{l=1}^K, \alpha, \omega_1, \ldots, \omega_N)$ is maximized.

$$O(\{U_l\}_{l=1}^K, \alpha, \omega_1, \ldots, \omega_N) = \frac{\sum_{p=q=1}^{K} d(U_p, U_q)}{\sum_{p=1}^{K} \sum_{q=1, q \ne p}^{K} d(U_p, U_q)} = \frac{\sum_{p=q=1}^{K} \sum_{u_i \in U_p, u_j \in U_q} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j))}{\sum_{p=1}^{K} \sum_{q=1, q \ne p}^{K} \sum_{u_i \in U_p, u_j \in U_q} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j))}$$

subject to $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$. (22)

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 maximize 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 $f(\alpha, \omega_1, \ldots, \omega_N) = \sum_{p=q=1}^{K} \sum_{u_i \in U_p, u_j \in U_q} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j))$ and $g(\alpha, \omega_1, \ldots, \omega_N) = \sum_{p=1}^{K} \sum_{q=1, q\neq p}^{K} \sum_{u_i \in U_p, u_j \in U_q} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j))$, the original clustering goal is rewritten as the following optimization problem (NFPP).

$$Max\ O(\{U_l\}_{l=1}^{K}, \alpha, \omega_1, \ldots, \omega_N) = \frac{f(\alpha, \omega_1, \ldots, \omega_N)}{g(\alpha, \omega_1, \ldots, \omega_N)} \tag{23}$$

subject to $\alpha + \sum_{i=1}^{N} \omega_i = N+1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$.

Lemma 1. *Let f be a function of a single variable on $\mathbb{R}$. Then*

(1) f is concave iff for $\forall x_1, x_2 \in \mathbb{R}$ and $\forall \lambda \in (0,1)$ we have $f((1-\lambda)x_1 + \lambda x_2) \geqslant (1-\lambda)f(x_1) + \lambda f(x_2)$.

(2) f is convex iff for $\forall x_1, x_2 \in \mathbb{R}$ and $\forall \lambda \in (0,1)$ we have $f((1-\lambda)x_1 + \lambda x_2) \leqslant (1-\lambda)f(x_1) + \lambda f(x_2)$.

Definition 8. A set S of n-vectors is convex if $(1-\lambda)x + \lambda x' \in S$ whenever $x, x' \in S$, and $\lambda \in [0,1]$.

Lemma 2. *Let f be a function of multiple variables with continuous partial derivatives of first and second order on the convex set S and denote the Hessian of f at the point x by $\Pi(x)$. Then*

(1) f is concave iff $\Pi(x)$ is negative semidefinite for $\forall x \in S$.

(2) if $\Pi(x)$ is negative definite for $\forall x \in S$, f is strictly concave.

(3) f is convex iff $\Pi(x)$ is positive semidefinite for $\forall x \in S$.

(4) if $\Pi(x)$ is positive definite for $\forall x \in S$, f is strictly convex.

Lemmas 1, 2 and the detailed proof can be found in [26].

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

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 $x' = (\nu_1, \nu_2, \ldots, \nu_{N+1})$ satisfy the following two constraints: $\sum_{i=1}^{N+1} \mu_i = N+1$, $\mu_i \geqslant 0$, $\sum_{i=1}^{N+1} \nu_i = N+1$, $\nu_i \geqslant 0$, $i = 1, \cdots, N+1$.

For an arbitrary $\lambda \in [0,1]$, the $(n+1)$-vector $(1-\lambda)x + \lambda x' = ((1-\lambda)\mu_1 + \lambda\nu_1, (1-\lambda)\mu_2 + \lambda\nu_2, \cdots, (1-\lambda)\mu_{N+1} + \lambda\nu_{N+1})$. The sum of each dimension for this $(n+1)$-vector is equal to $(1-\lambda)\sum_{i=1}^{N+1} \mu_i + \lambda \sum_{i=1}^{N+1} \nu_i = (1-\lambda)(N+1) + \lambda(N+1) = N+1$. Thus, $(1-\lambda)x + \lambda x'$ is still in S and S is a convex set.

We then calculate the Hessian matrix of f as follows.

$$\Pi(f)_{ij}(\alpha, \omega_1, \ldots, \omega_N) = D_i D_j f(\alpha, \omega_1, \ldots, \omega_N) \tag{24}$$

where D_i is the differentiation operator with respect to the i^{th} argument.

The Hessian becomes $\Pi(f) = [\frac{\partial^2 f}{\partial \alpha^2}\ \frac{\partial^2 f}{\partial \alpha \partial \omega_1}\ \cdots\ \frac{\partial^2 f}{\partial \alpha \partial \omega_N}; \frac{\partial^2 f}{\partial \omega_1 \partial \alpha}\ \frac{\partial^2 f}{\partial \omega_1^2}\ \cdots\ \frac{\partial^2 f}{\partial \omega_1 \partial \omega_N}; \ldots; \frac{\partial^2 f}{\partial \omega_N \partial \alpha}\ \frac{\partial^2 f}{\partial \omega_N \partial \omega_1}\ \cdots\ \frac{\partial^2 f}{\partial \omega_N^2}]$. Since $f(\alpha, \omega_1, \ldots, \omega_N)$ has only one non-linear term $\sum_{p=q=1}^{K} \sum_{u_i \in U_p, u_j \in U_q} \sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j)$, there is one non-zero term $\frac{\partial^2 f}{\partial \alpha^2} = \sum_{p=q=1}^{K} \sum_{u_i \in U_p, u_j \in U_q} \sum_{k=1}^{\infty} k(k-1) \frac{\alpha^{k-2} t^k}{k!} H^k(i,j)$ in the Hessian matrix. We can easily prove that all of its eigenvalues are non-negative. Thus, it is positive-semidefinite for $\forall \alpha$, ω_1, $\ldots$, $\omega_N \in S$, and $f(\alpha, \omega_1, \ldots, \omega_N)$ is convex on the set S.

Theorem 2. *$g(\alpha, \omega_1, \ldots, \omega_N)$ is convex on S since its Hessian matrix $\Pi(g)$ is positive-semidefinite for $\forall \alpha$, ω_1, $\ldots$, $\omega_N \in S$.*

The detailed proof is omitted due to space limit. This theorem can be testified by using the above-mentioned similar method.

Theorem 3. *The NFPP problem is equivalent to a polynomial programming problem with polynomial constraints (PPPPC).*

$$Max\ \gamma f(\alpha, \omega_1, \ldots, \omega_N) \tag{25}$$

subject to $0 \leqslant \gamma \leqslant 1/g(\alpha, \omega_1, \ldots, \omega_N)$, $\alpha + \sum_{i=1}^{N} \omega_i = N+1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$.

Proof. If $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N, \overline{\gamma})$ is a possible solution of PPPPC, then $\overline{\gamma} = 1/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$. Thus $\overline{\gamma} f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) / g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$. For any feasible solution $(\alpha, \omega_1, \ldots, \omega_N)$ of NFPP, the constraints of PPPPC are satisfied by setting $\gamma = 1/g(\alpha, \omega_1, \ldots, \omega_N)$, so $\gamma f(\alpha, \omega_1, \ldots, \omega_N) \leqslant \overline{\gamma} f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$, i.e. $f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N) \leqslant f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$.

Conversely, if $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ solves NFPP, then for any feasible solution $(\alpha, \omega_1, \ldots, \omega_N, \gamma)$ of PPPPC we have $\gamma f(\alpha, \omega_1, \ldots, \omega_N) \leqslant f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N) \leqslant f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = \overline{\gamma} f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ with $\overline{\gamma} = 1/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$.

Although PPPPC is a polynomial programming problem, the polynomial constraints make it very hard to solve. We further simplify it as an nonlinear parametric programming problem (NPPP).

Theorem 4. *A nonlinear parametric programming problem (NPPP) is defined as $F(\beta) = Max\ \{f(\alpha, \omega_1, \ldots, \omega_N) - \beta g(\alpha, \omega_1, \ldots, \omega_N)\}$ subject to $\alpha + \sum_{i=1}^{N} \omega_i = N+1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$. The NFPP problem of Eq.(23) is equivalent to this NPPP, i.e., β is a maximum value of NFPP iff $F(\beta) = 0$.*

Proof. If $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ is a possible solution of $F(\beta) = 0$, then $f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = 0$. Thus $f(\alpha, \omega_1, \ldots, \omega_N) - \beta g(\alpha, \omega_1, \ldots, \omega_N) \leqslant f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = 0$. We have $\beta = f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) \geqslant f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$. Therefore, β is a maximum value of NFPP and $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ is a feasible solution of NFPP.

Conversely, if $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ solves NFPP, then we have $\beta = f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)/g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) \geqslant f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$. Thus $f(\alpha, \omega_1, \ldots, \omega_N) - \beta g(\alpha, \omega_1, \ldots, \omega_N) \leqslant f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = 0$. We have $F(\beta) = 0$ and the maximum is taken at $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$.

Now we have successfully transformed the original NFPP in Eq.(23) into the straightforward NPPP. This transformation can help the algorithm converge in a finite number of iterations. Although it is not clear whether the original objective is concave or convex, the objective $F(\beta)$ of NPPP has the following properties.

Theorem 5. *$F(\beta)$ is a convex function.*

Proof: Suppose that $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ is a possible solution of $F((1-\lambda)\beta_1 + \lambda\beta_2)$ with $\beta_1 \neq \beta_2$ and $0 \leqslant \lambda \leqslant 1$. $F((1-\lambda)\beta_1 + \lambda\beta_2) = f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - ((1-\lambda)\beta_1 + \lambda\beta_2) g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) = \lambda(f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_2 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)) + (1-\lambda)(f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_1 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)) \leqslant \lambda \cdot max(f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_2 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)) + (1-\lambda) \cdot max(f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_1 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)) = \lambda F(\beta_2) + (1-\lambda) F(\beta_1)$. According to Lemma 1, we know that $F(\beta)$ is convex.

Theorem 6. *$F(\beta)$ is a monotonic decreasing function.*

Proof: Suppose that $\beta_1 > \beta_2$ and $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$ is a possible solution of $F(\beta_1)$. Thus, $F(\beta_1) = f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_1 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) < f(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) - \beta_2 g(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N) \leqslant F(\beta_2)$.

Theorem 7. *$F(\beta) = 0$ has a unique solution.*

Proof: Based on the above-mentioned theorems, we know $F(\beta)$ is continuous as well as decreasing. In addition, $lim_{\beta \to +\infty} F(\beta) = -\infty$ and $lim_{\beta \to -\infty} F(\beta) = +\infty$.

5.5 Adaptive Weight Adjustment

The procedure of solving this NPPP optimization problem includes two parts: (1) find such a reasonable parameter β ($F(\beta) = 0$), making NPPP equivalent to NFPP; (2) given the parameter β, solve

Algorithm 1 Social Influence-based Graph Clustering

Input: a social graph SG, multiple influence graphs IG_i, a cluster number K, initial weights $\alpha = \omega_1 = \ldots = \omega_N = 1$ and a parameter $\beta = 0$.
Output: K clusters $U_1, ..., U_K$.

1: Calculate $W_0, W_1, W_2, \cdots, W_N$, and W;
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 $c^* = argmax_{c_j} d(u_i, c_j)$;
5: Update the cluster centroids with the most centrally located point in each cluster;
6: Solve the NPPP of $F(\beta)$;
7: Update $\alpha, \omega_1, ..., \omega_N$;
8: Refine $\beta = f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$;
9: Update W;
10: Return K clusters $U_1, ..., U_K$.

a polynomial programming problem about the original variables. Our weight adjustment mechanism is an iterative procedure to find the solution of $F(\beta) = 0$ and the corresponding weights α, ω_1, ..., ω_N after each iteration of the clustering process. We first generate an initial unified similarity matrix W with equal weights to initialize cluster centroids and partition the social graph. Since $F(\beta)$ is a monotonic decreasing function and $F(0) = Max\ \{f(\alpha, \omega_1, \ldots, \omega_N)\}$ is obviously non-negative, we start with an initial $\beta = 0$ and solve the subproblem $F(0)$ by using existing fast polynomial programming model to update the weights α, ω_1, ..., ω_N. The updated parameter by $\beta = f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$ 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.*

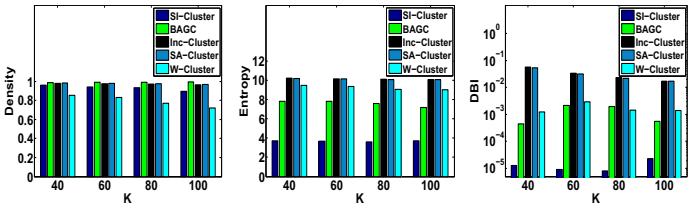
Proof. Existing work has studied the convergence properties of the partitioning approach to clustering, such as K-Means [27]. Our clustering follows a similar approach. So the cluster assignment and centroid update steps improve the objective function. In addition, we have explained that nonlinear parametric 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 964,166 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.



(a) density (b) entropy (c) DBI

Figure 4: Cluster Quality on Amazon 20,000 Products

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 of to evaluate the quality of clusters $\{U_l\}_{l=1}^K$ generated by different methods. The definitions of the metrics are given as follows.

$$density(\{U_l\}_{l=1}^K) = \sum_{i=1}^{K} \frac{|\{(u_p, u_q)|u_p, u_q \in U_i, (u_p, u_q) \in E\}|}{|E|} \quad (26)$$

$$entropy(\{U_l\}_{l=1}^K) = \sum_{i=1}^{N} \frac{\omega_i}{\sum_{p=1}^{N} \omega_p} \sum_{j=1}^{K} \frac{|U_j|}{|U|} entropy(a_i, U_j) \quad (27)$$

where ω_i is the weight of influence graph IG_i, $entropy(a_i, U_j) = -\sum_{n=1}^{n_i} p_{ijn} log_2 p_{ijn}$, n_i (or attribute a_i) is the number of IG_i's activities (or the number of a_i's values) and p_{ijn} is the percentage of vertices in cluster U_j which participate in the n^{th} activity in IG_i (or have value a_{in} on a_i). $entropy(\{U_l\}_{l=1}^K)$ measures the weighted entropy from all influence graphs (or attributes) over K clusters.

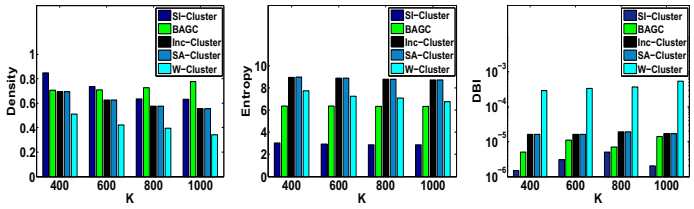
Davies-Bouldin Index (DBI) measures the uniqueness of clusters with respect to the unified similarity measure.

$$DBI(\{U_l\}_{l=1}^K) = \frac{1}{K} \sum_{i=1}^{K} max_{j \neq i} \frac{d(c_i, c_j)}{\sigma_i + \sigma_j} \quad (28)$$

where c_x is the centroid of U_x, $d(c_i, c_j)$ is the similarity between c_i and c_j, σ_x is the average similarity of vertices in U_x to c_x.

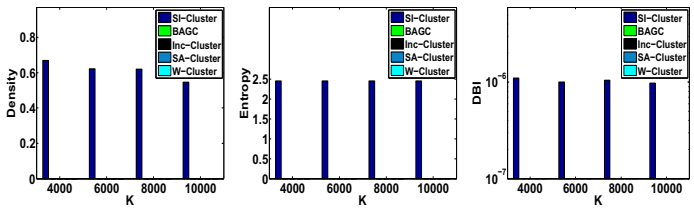
6.3 Cluster Quality Evaluation

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 have 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 0.000008 – 0.000023, while other methods have a much higher DBI than SI-Cluster. This demonstrates that SI-Cluster can achieve both high



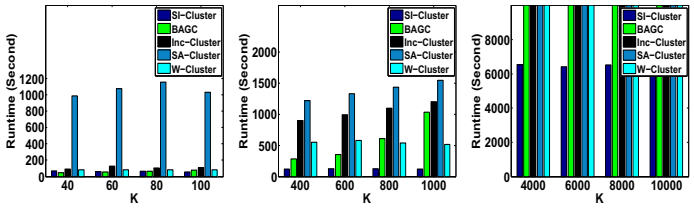
(a) density (b) entropy (c) DBI

Figure 5: Cluster Quality on DBLP 100,000 Authors



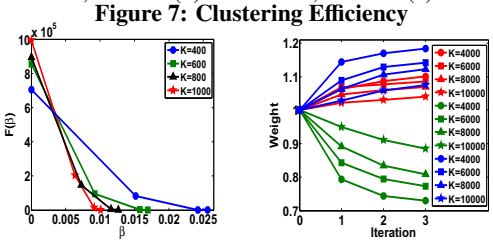
(a) density (b) entropy (c) DBI

Figure 6: Cluster Quality on DBLP 964,166 Authors



(a) Amazon 20,000 (b) DBLP 100,000 (c) DBLP 964,166

Figure 7: Clustering Efficiency



(a) $F(\beta)$ (b) Weight Update

Figure 8: Clustering Convergence on DBLP 964,166 Authors

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 (< 0.000005) among different methods, while the DBI values by other methods are obviously larger than > 0.000005.

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 worst 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. Fourth, the performance by BAGC is better than other approaches except SI-Cluster. Although it does not need to repeatedly compute the distance matrix, it needs to iteratively update lots of temporary matrices or interim variables and its computational cost is proportional to K^2 so that it may not work well when facing large K value. In comparison, SI-Cluster reorganizes a large scale heterogeneous network into multiple small scale subgraphs. It reduces the cost by partitioning activities with the topological information of the activity graph. Furthermore, SI-Cluster calculates influence-based similarity matrices only once. According to Theorems 4-7, solving $F(\beta)$ for a given β is a polynomial programming problem which can be sped up by existing fast polynomial programming model.

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

Author	Cluster 1 (DB)	Cluster 2 (DM)
Elisa Bertino	0.0568	0.0249
Christos Faloutsos	0.0465	0.0746
Jiawei Han	0.0585	0.0960
Vipin Kumar	0.0146	0.0545
Bing Liu	0.0153	0.0511
David Maier	0.0474	0.0079
Hector Garcia-Molina	0.0603	0.0047
M. Tamer Özsu	0.0408	0.0111
Jian Pei	0.0386	0.0653
Philip S. Yu	0.0606	0.0991

(b) Influence Scores Based on Selected Top Conferences

Author	AI Cluster	DB Cluster	DM Cluster	IR Cluster
Elisa Bertino	0.0047	0.7135	0.0055	0.2763
Christos Faloutsos	0.0012	0.4267	0.3950	0.1771
Jiawei Han	0.0883	0.3724	0.3766	0.1628
Vipin Kumar	0.2511	0.1342	0.5198	0.0949
Bing Liu	0.2648	0.1001	0.4004	0.2347
David Maier	0.1570	0.8290	0.0117	0.0023
Hector Garcia-Molina	0.0031	0.8217	0.0075	0.1677
M. Tamer Özsu	0.0017	0.5506	0.1080	0.3397
Jian Pei	0.0876	0.3768	0.3717	0.1639
Philip S. Yu	0.0972	0.3504	0.3763	0.1761

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

Table 1 (a) actually presents an unbalanced result since the influence propagation process is based on the full DBLP dataset. We know that academic research in the area of database has a longer history and there are more academic conferences or forums focusing on database research. Thus, we choose the same number of top conferences for each research area to better evaluate the quality of our co-influence model. Here, we choose three top conferences from four research areas of database, data mining, information retrieval and artificial intelligence, respectively. The detailed conference list is, DB: VLDB, SIGMOD, ICDE; DM: KDD, ICDM, SDM; IR: SIGIR, CIKM, ECIR; AI: IJCAI, AAAI, ECAI. Table 1 (b) 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 towards the clustering convergence. Third, we transform a sophisticated nonlinear fractional programming problem of multiple weights into 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

[1] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[2] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[3] H. Ma, H. Yang, M. R. Lyu, and I. King. Mining social networks using heat diffusion processes for marketing candidates selection. In *CIKM*, pages 233–242, 2008.

[4] M. Roth, A. Ben-David, D. Deutscher, G. Flysher, I. Horn, A. Leichtberg, N. Leiser, Y. Matias, and R. Merom. Suggesting friends using the implicit social graph. In *KDD*, pages 233–242, 2010.

[5] S. Myers, C. Zhu, J. Leskovec. Information Diffusion and External Influence in Networks. In *KDD*, pages 33-41, 2012.

[6] B. Taskar, E. Segal, D. Koller. Probabilistic Classification and Clustering in Relational Data. In *IJCAI*, 2001.

[7] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *PKDD*, 2005.

[8] 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.

[9] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *KDD*, pages 1298–1306, 2011.

[10] X. Yu, Y. Sun, P. Zhao, and J. Han. Query-driven discovery of semantically similar substructures in heterogeneous networks. In *KDD*, pages 1500–1503, 2012.

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

[12] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. In *Phys. Rev. E 69, 026113*, 2004.

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

[14] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: Applications to community discovery. In *KDD*, 2009.

[15] K. Macropol and A. Singh. Scalable discovery of best clusters on large graphs. In *PVLDB*, 3(1), 693–702, 2010.

[16] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, 567–580, 2008.

[17] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.

[18] E. C. Kenley and Y.-R. Cho. Entropy-based graph clustering: Application to biological and social networks. In *ICDM'11*.

[19] M. Shiga, I. Takigawa, H. Mamitsuka. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *KDD*, 2007.

[20] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. In *VLDB*, 718–729, 2009.

[21] 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.

[22] Y. Sun, B. Norick, J. Han, X. Yan, P. Yu, X. Yu. Integrating Meta-Path Selection with User-Guided Object Clustering in Heterogeneous Information Networks. In *KDD*, 2012.

[23] 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.

[24] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogenous information network analysis. In *EDBT*, 2009.

[25] L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis based on the L1 Norm*, pages 405–416, 1987.

[26] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

[27] L. Botton and Y. Bengio. Convergence properties of the k-means algorithms. In *NIPS*, pages 585–592, 1994.

[28] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, 2010.