Social Influence Based Clustering and Optimization over Heterogeneous Information Networks

YANG ZHOU, College of Computing, Georgia Institute of Technology
LING LIU, College of Computing, Georgia Institute of Technology

Social influence analysis has shown great potential for strategic marketing decision. It is well known that people influence one another based on both their social connections and the social activities that they have engaged in the past. In this paper we develop an innovative and high-performance social influence based graph clustering framework with four unique features. First, we explicitly distinguish social connection based influence (self-influence) and social activity based influence (co-influence). We compute the self-influence similarity between two members based on their social connections within a single collaboration network, and compute the co-influence similarity by taking into account not only the set of activities that people participate but also the semantic association between these activities. Second, we define the concept of influence-based similarity by introducing a unified influence-based similarity matrix that employs an iterative weight update method to integrate self-influence and co-influence similarities. Third, we design a dynamic learning algorithm, called SI-CLUSTER, for social influence based graph clustering. It iteratively partitions a large social collaboration network into K clusters based on both the social network itself and the multiple associated activity information networks, each representing a category of activities that people have engaged. To make the SI-CLUSTER algorithm converge fast, we transform sophisticated nonlinear fractional programming problem with respect to multiple weights into a straightforward nonlinear parametric programming problem of single variable. Finally, we develop an optimization technique of diagonalizable-matrix approximation to speed up the computation of self-influence similarity and co-influence similarities. Our SI-Cluster-Opt significantly improves the efficiency of SI-Cluster on large graphs while maintaining high quality of clustering results. Extensive experimental evaluation on three real-world graphs shows that, compared to existing representative graph clustering algorithms, our SI-CLUSTER-OPT approach not only achieves a very good balance between self-influence and co-influence similarities but also scales extremely well for clustering large graphs in terms of time complexity while meeting the guarantee of high density, low entropy and low DBI.

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

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Graph Clustering, Heterogeneous Information Network, Social Influence

ACM Reference Format:
Yang Zhou and Ling Liu. 2015. Social Influence Based Clustering and Optimization over Heterogeneous Information Networks. *ACM Trans. Knowl. Discov. Data.* 9, 4, Article 39 (March 2010), 53 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

A preliminary version of this paper appeared in the proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining [Zhou and Liu 2013].
This material is based upon work partially supported by the National Science Foundation under Grants IIS-0905493, CNS-1115375, IIP-1230740, and a grant from Intel ISTC on Cloud Computing.
Author's addresses: Y. Zhou and L. Liu, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332; email: yzhou@gatech.edu, lingliu@cc.gatech.edu.

DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Social networks are popularly used to model and analyze the relationships and interactions among individuals in many application domains, such as marketing networks, communication networks, web blogs, and so forth. Social influence analysis is to study how information, ideas, experiences and innovations are spread or diffused across social networks and influence their members. Social network analysis is gaining increasing and renewed attention from multiple disciplines. Most of existing social network analyses have focused on the topological structure of social networks for community detection, entity classification and ranking, link prediction, social influence inference, entity resolution, evolution analysis, to name a few.

Recently, social influence analysis has received a fair amount of attention [Domingos and Richardson 2001], [Kempe et al. 2003], [Yang et al. 2005], [Ma et al. 2008], [Anagnostopoulos et al. 2008], [Roth et al. 2010], [Chen et al. 2010] and [Myers et al. 2012]. Marketing campaign on social networks has shown great potential for success than traditional marketing techniques. According to eMarketer [ema], Facebook online display ad revenue growth will outperform Yahoo!, AOL, Microsoft and Google combined. Although social influence analysis has shown great potential for strategic marketing decision, we argue that social influence analysis should not only take into account the explicit relationships and interactions in social networks but also consider the implicit, deep and hidden correlations among members of a social network. Concretely, we argue that social network clustering method should contemplate both social interactions (such as friendships among people) within a single collaboration network (self-influence) and activity-based interactions between the collaboration network and other information networks (co-influence). The former reflects the collaboration patterns among people in the social collaboration network and the latter reveals the activity patterns of people in the social network based on their partitions in other information networks (such as sport events, entertainment activities, professional communities). For example, it is known that people influence one another in terms of not only their social friendships but also their purchasing behaviors about different categories of products or or participation behaviors about different categories of activities. Self-influence together with activity-based co-influences from purchasing or participating behaviors can be critical indicators for deeper understanding of how people are socially connected and better predicting of individual purchase or participation tendency. Thus, full fledged social influence analysis should take into account not only the topological structure and static attributes captured explicitly in social collaboration networks but also the multiple activity information networks that implicitly connect the members of the social collaboration networks.

Social influence based clustering analysis over heterogeneous information network involves clustering of a heterogeneous graph with various types of entities, links, static attributes, dynamic and inter-connected activities, which demands for new distance functions, clustering models and fast algorithms to address a number of new challenges.

— **[Graph Propagation and Computational Complexity]** The large scale clustering analysis over heterogeneous information networks often displays features of social complexity and involves substantial non-trivial computational cost. Concretely, social influence based graph clustering over heterogeneous information networks involves both self-propagation within a social collaboration network and the across-propagation between the social collaboration network and multiple activity networks. The computational cost for these graph propagation operations is often $O(n^3)$ where n is the number of vertices in a network. For instance, to calculate a social influence similarity score between any pair of authors on the DBLP bibliography dataset,

we not only need to check the direct connection (co-authorship) between them but also have to inspect up to n-hop indirect interactions through the circle of other co-authors. Thus, a highly efficient algorithm for computing pairwise self-influence and co-influence similarities is critical.

— **[Interaction over Heterogeneous Information Networks]** 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 example, 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).

— **[Integration and Context-Awareness]** 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. For instance, if the objective of clustering the facebook people graph is to find out the most influential people on particular subject, say sport, then instead of utilizing multiple social networks from arbitrary domains, we advocate to consider only the people network and the sport related information networks in the clustering analysis and exclude those activity graphs on music, cooking and other non-sport events.

— **[Bidirectional Information Flow]** 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.

Bearing the above problems in mind, we develop an innovative social influence based graph clustering approach for heterogeneous information networks, called SI-CLUSTER, which captures not only the static attributes of people (vertices) in the social collaboration network but also the nested and dynamic relationships between people and other types of entities in different information networks in terms of their participations in different activities of interest. To better position the SI-CLUSTER development, in comparison with conventional graph clustering approaches, we identify three types of social interactions for graph modeling that are vital for social influence based graph clustering: (i) social interactions (such as friendship among people) within a single collaboration network (self-influence), (ii) single-valued and multi-valued vertex attributes that represent relatively stable and static state of the vertices in the social network (such as name, sex, age, birth place, and multiple education degrees a person may achieve), and (iii) domain-specific network-wise interactions (such as multiple different activities one may have participated or are currently engaged) between the collaboration network and other information networks. We show that, in contrast to conventional graph clustering algorithms, SI-CLUSTER is by design a dynamic social influence based graph clustering method with two new criteria: First, we characterize social entities (vertices) by their stable attributes and social connections and model activities of different categories engaged by the social entities as different activity graphs. Thus, the activities engaged by social entities are represented as com-

plex relationships between the social collaboration graph and its multiple associated activity graphs. Second, we introduce the concept of self-influence similarity between two members to capture their direct and indirect social connection patterns within a single collaboration network and the concept of co-influence similarity to capture the social activity based influence similarity. To adequately model the social activity based influence, we take into account not only the set of activities that people participate but also the semantic association between these activities by considering multiple activity-based information networks (such as sport events, entertainment activities, professional communities). A unique characteristic of SI-CLUSTER is its ability of integrating the self-influence and multiple types of co-influences into a weighted unified influence-based similarity measure, ensuring fast clustering convergence through dynamic weight refinement at each of the clustering iterations.

This paper makes four unique contributions. First, we define the concept of influence-based vertex similarity using heat diffusion based influence propagation on both social collaboration graph and each of associated activity graphs. We compute a self-influence similarity matrix based on direct and indirect social connections in the social collaboration graph. For each influence graph, we propose to model and refine a co-influence similarity matrix based on the dynamic characteristics of social influence propagation patterns that evolve in the chosen influence graph over time. Second, we develop a dynamic weight tuning method to combine various influence-based similarities through an iterative learning algorithm, SI-CLUSTER, for social influence based graph clustering, such that each cluster produced by SI-CLUSTER contains a densely connected group with similar collaborative patterns among themselves and similar interaction patterns with activity networks. Third, SI-CLUSTER dynamically refines the K initial clusters by continuously quantifying and adjusting the weighted contributions from different kinds of similarities until reaching convergence. A theoretical analysis is provided to quantify the contributions of social graph and multiple influence graphs to the unified influence-based similarity for measuring vertex closeness in the social graph, and prove that the weights are adjusted towards the direction of clustering convergence. 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. Fourth but not the least, to scale the computation of the influence-based similarity matrices, we propose an optimization technique of diagonalizable-matrix approximation to speed up the computation of propagating heat kernel and influence-based similarity. We refer to this optimized SI-CLUSTER algorithm as SI-CLUSTER-OPT and our experimental results show that SI-CLUSTER-OPT is significantly faster than SI-CLUSTER and the state-of-the-art graph clustering mechanisms. We perform extensive evaluation by our proposed clustering approach on large-scale real graphs, demonstrating that SI-CLUSTER-OPT not only achieves a very good balance between self-influence and co-influence similarities but also scales extremely well for clustering large graphs in terms of time complexity while meeting the guarantee of high density, low entropy and low DBI.

The rest of the paper proceeds as follows. Section 2 briefly summarizes related work. Section 3 gives an overview of our social influence based clustering framework. Section 4 introduces a unified influence-based similarity measure to integrate self-influence similarity and co-influence similarities. Section 5 presents our baseline social influence based clustering algorithm SI-CLUSTER. Section 6 describes our optimized algorithm SI-Cluster-Opt, which employs diagonalizable-matrix approximation to speed up the computation of influence-based similarity in SI-CLUSTER. Section 7.3 makes complexity analysis for SI-Cluster-Opt and several state-of-the-art graph clus-

tering algorithms. We present our experimental evaluation in Section 7, and conclude the paper in Section 8.

2. RELATED WORK

The novelty of our social influence based graph clustering approach lies in social influence analysis, heat diffusion kernel, vertex similarity definition, fast matrix computation, heterogeneous network analysis and graph clustering. In this section we will briefly overview the related work in these areas.

Social influence analysis. Social influence analysis has received increasing attention over the last decade [Domingos and Richardson 2001], [Kempe et al. 2003], [Yang et al. 2005], [Ma et al. 2008], [Anagnostopoulos et al. 2008], [Roth et al. 2010], [Chen et al. 2010] and [Myers et al. 2012]. As marketing techniques, social influence analysis holds the potential to utilize social influence to increase brand or product awareness through word-of-mouth promotion. [Domingos and Richardson 2001] proposed a cascading viral marketing algorithm, which tries to find such a subset of individuals that if these individuals adopt a new product or innovation, then they will trigger a large cascade of further adoptions. [Kempe et al. 2003] pioneered the concept of social influence by modeling the selection of influential sets of individuals in a social graph as a discrete optimization problem. It utilizes the provable greedy approximation algorithm for maximizing the spread of influence in a social network. By imitating the way that heat flows in a medium with a geometric structure, [Yang et al. 2005] proposed two novel classification algorithms to employ the heat kernel to construct the kernel-based classifier directly. [Ma et al. 2008] proposed a heat-diffusion based viral marketing model with top k most influential nodes which utilizes the heat diffusion theory from Physics to describe the diffusion of innovations and help marketing companies divide their marketing strategies into several phases. [Anagnostopoulos et al. 2008] applied statistical analysis on the data from a large social system to identify and measure social influence as a source of correlation between the actions of individuals with social ties. A novel friend suggestion algorithm proposed in [Roth et al. 2010] uses a user's implicit social graph to generate a friend cluster, given a small seed set of contacts. [Chen et al. 2010] proposed a new heuristic influence maximization algorithm to maximize the spread of influence under certain influence cascade models. [Myers et al. 2012] presented a model in which information can reach a node via the links of social network or through the influence of external sources. None of existing work, to the best of our knowledge, has modeled social influence patterns based on iteratively combining self-influence and co-influence similarities throughout the clustering process.

Heat kernel. Recently, the idea of heat kernel has been successfully applied to multiple areas [Kondor and Lafferty 2002], [Belkin and Niyogi 2003], [Lafferty and Lebanon 2005], [Yang et al. 2007] and [Ma et al. 2008]. [Kondor and Lafferty 2002] presented a natural approach to constructing diffusion kernels on graphs, which are based on the heat equation on Riemannian manifolds and can be regarded as the discretization of the familiar Gaussian kernel of Euclidean space. [Belkin and Niyogi 2003] utilized a heat kernel to construct the weight of a neighborhood graph, and apply it to nonlinear dimensionality reduction that has locality-preserving properties. [Lafferty and Lebanon 2005] introduced a family of kernels that is intimately based on the heat equation on the Riemannian manifold defined by the Fisher information metric associated with a statistical family, and generalize the Gaussian kernel of Euclidean space. DiffusionRank [Yang et al. 2007] presented a ranking algorithm, which is a generalization of PageRank, using heat diffusion model. [Ma et al. 2008] proposed a heat-diffusion based viral marketing model with top k most influential nodes which utilizes the heat diffusion theory from Physics to describe the diffusion of innovations and help marketing companies divide their marketing strategies into several phases.

However, to the best of our knowledge, our SI-Cluster is the first to employ multiple heat diffusion kernels to learn different types of social influence similarities and combine multiple social influence based similarities through a weight function with iterative weight refinement during each of the clustering iterations.

Vertex similarity definition. The vertex similarity definition lies in the center of graph analysis. We can classify existing works into local topology based vertex similarity and global topology based vertex similarity . Several known local topology based measures, such as Jaccard coefficient [Tan et al. 2005], Salton Cosine coefficient [Salton 1989] and Dice coefficient [Dice 1945], are based on the intuition that two vertices are more similar if they share more common neighbors. The Adamic/Adar's inverse log-weighted coefficient [Adamic and Adar 2003] refined the measures by assigning more weights to the vertices with fewer degrees. However, these measures have significant shortcomings: two vertices may be highly similar even if they share no common neighbors. The HITS algorithm [Kleinberg 1999] motivated the importance and the significance of global topology based measures. [Blondel et al. 2004] provided a general HITS-based method to identify the vertex similarity in the directed graph. [Katz 1953] proposed a measure based on the total number of simple paths between vertices with lower weights to longer paths. Several global topology based measures, such as SimRank [Jeh and Widom 2002], Leicht-Holme-Newman (LHN) [Leicht et al. 2006], and P-Rank [Zhao et al. 2009] define the similarity measures recursively: two vertices are similar if their immediate neighbors in the network are themselves similar. Although global topology based measures offer a boarder perspective of how vertices are similar in the context of the whole network, they are known to be computationally very expensive. PathSim [Sun et al. 2011] presented a novel meta path-based similarity measure for heterogeneous information networks, which captures the subtle similarity semantics among peer objects in heterogeneous networks. To the best of our knowledge, SI-Cluster is the first work that introduces the social influence based vertex similarity measure over heterogeneous information networks, with progressive weight refinement throughout the iterative clustering process.

Efficient matrix multiplication. The computation of influence-based similarity is related to research in fast matrix multiplication and higher power computation. Strassen's algorithm [Strassen 1969] made the startling discovery that one can multiply two $n \times n$ matrices in only $O(n^{2.807})$ field operations, compared with $2n^3$ for the standard algorithm. It is based on a way of multiplying two 2×2-matrices which requires only 7 multiplications (instead of the usual 8), at the expense of several additional addition and subtraction operations. The Coppersmith-Winograd algorithm [Coppersmith and Winograd 1990] with an asymptotic complexity of $O(n^{2.3727})$ is similar to Strassen's algorithm: a way is devised for multiplying two $n \times n$-matrices with fewer than n^3 multiplications, and this technique is applied recursively. [Cohn et al. 2005] recently proposed a group-theoretic approach to bounding the exponent of matrix multiplication to at least $O(n^{2.41})$ if families of wreath products of Abelian groups with symmetric groups satisfying certain conditions exist. The Summation Formula of the Matrix Neumann Series [Strang 2005] proposed a fast solution of calculating the sum of the finite Neumann series of a square matrix A. [Larsen and McAllister 2001] proposed a fast and scalable technique for multiplying large matrices using the graphics hardware found in a PC. The method is an adaptation of the parallel computing technique by distributing the computation over a logically cube-shaped lattice of processors and performing a portion of the computation at each processor. [Bodrato 2010] presented a new sequence for Strassen-like matrix multiplication, which is not worse than the Winograd sequence for multiplications. When computing chain products or a power, the algorithm can further reduce the number of linear combinations, collating the post-combination sequence of partial results with the precombination needed for

the next multiplication. In our social influence based clustering framework, we develop diagonalizable-matrix approximation to effectively reduce the computational complexity of the baseline SI-Cluster algorithm [Zhou and Liu 2013] by enabling high quality approximation and fast computation of influence-based similarity.

Heterogeneous network analysis. Recent studies on heterogeneous social network analysis combine links and content into heterogeneous information networks to improve the quality of querying, ranking and clustering. [Taskar et al. 2001] proposed a general class of models for classification and clustering in relational domains that capture probabilistic dependencies between related instances in a relational database containing both attributes and links. [Cai et al. 2005] proposed to learn an optimal linear combination of different relations on heterogeneous social networks in terms of their importance on a certain query. [Yang et al. 2009] proposed a unified model to combine link and content analysis for community detection. The conditional link model and the discriminative content model are combined via a probabilistic framework through the shared variables of community memberships. [Ji et al. 2011] groups objects into pre-specified classes, while generating the ranking information for each type of object in a heterogeneous information network. It is therefore beneficial to integrate classification and ranking in a simultaneous, mutually enhancing framework. [Yu et al. 2012] presented a query-driven discovery system for finding semantically similar substructures in heterogeneous networks. A filter-and-verification search framework is proposed to generate promising subgraph candidates using off-line indices, and verify candidates with a recursive pruning matching process. [Zhou et al. 2013] proposed a unified random walk distance measure integrating various types of entities, links and attributes for a heterogeneous service network. A reinforcement algorithm is provided to tightly integrate ranking and clustering by mutually and simultaneously enhancing each other. [Zhou and Liu 2014] presented an activity-edge centric multi-label classification framework for analyzing heterogeneous information networks by doing multi-label classification of friendship multigraph based on activity-based edge classification. To the best of our knowledge, our social influence based clustering framework is the first one to perform social influence based clustering over heterogeneous networks by dynamically combining self-influence from social graph and multiple types of co-influences from activity graphs.

Graph clustering. 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 [Shi and Malik 2000], modularity [Newman and Girvan 2004], structural density [Xu et al. 2007], stochastic flows [Satuluri and Parthasarathy 2009] or clique [Macropol and Singh 2010]. The clustering results often contain densely connected components within clusters. However, such methods usually ignore vertex attributes in the clustering process. On the other hand, K-SNAP [Tian et al. 2008] and CANAL [Zhang et al. 2010] presented OLAP-style aggregation approaches to summarize large graphs by grouping nodes based on the user-selected attributes. [Kenley and Cho 2011] exploited an information-theoretic model for clustering by growing a random seed in a manner that minimizes graph entropy. This kind of methods achieve homogeneous attribute values within clusters, but ignore the intra-cluster topological structure. Recently, [Shiga et al. 2007] presented a clustering method which integrates numerical vectors with modularity into a spectral relaxation problem. SA-Cluster [Zhou et al. 2009] and Inc-Cluster [Zhou et al. 2010] perform clustering based on both structural and attribute similarities by incorporating attributes as augmented edges to its vertices, transforming attribute similarity to vertex closeness. BAGC [Xu et al. 2012] constructs a Bayesian probabilistic model to capture both structural and attribute aspects. GenClus [Sun et al. 2012] proposed a model-based method for clustering heterogeneous networks with different link types

and different attribute types. PathSelClus [Sun et al. 2012] utilizes limited guidance from users in the form of seeds in some of the clusters and automatically learns the best weights for each meta-path in the clustering process. Our experimental results over three real-world large graph datasets show that SI-Cluster-Opt is significantly faster and more effective compared to the representative state-of-the-art graph clustering algorithms.

3. OVERVIEW

In this section, we first introduce the problem formulation, followed by an overview of our social influence based graph clustering framework illustrated by a walkthrough example.

3.1. 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.

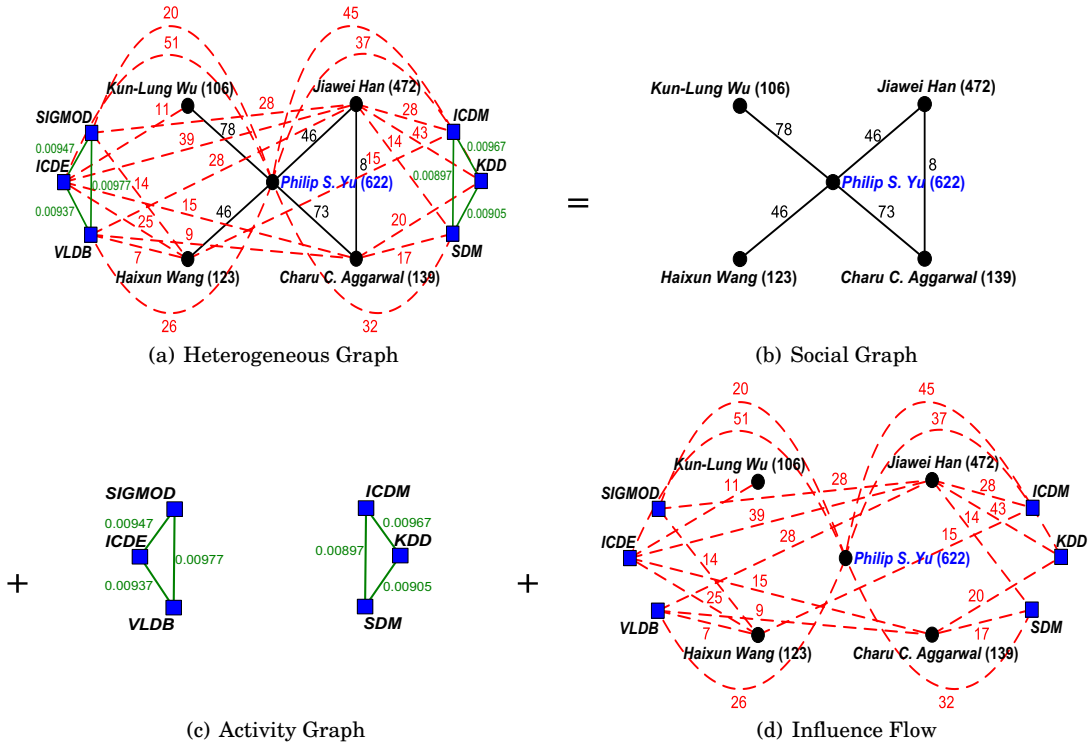
A *social graph* is denoted as $SG = (U, E)$, where U is the set of vertices representing the members of the collaboration network, such as customers or authors, and E is the set of edges denoting the collaborative relationships between members of the collaboration network. We use N_{SG} to represent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

An *influence graph* is denoted as $IG_i = (U, V_i, T_i)$, where U and V_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 $(1 \leq i \leq N)$, 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 $\forall 1 \leq i, j \leq K, i \neq j$, 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 social influence based clustering 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 con-



(a) Heterogeneous Graph (b) Social Graph

(c) Activity Graph (d) Influence Flow

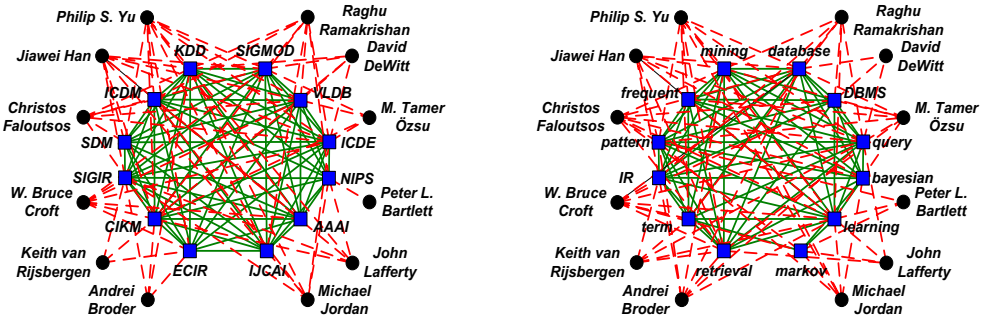
Fig. 1. A Heterogeneous Network Example from DBLP

ferences, 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 remove the conference similarities with less than 0.005. A number of mechanisms can be used to compute similarity of conferences. We use RankClus [Sun et al. 2009] 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. For ease of presentation, we also ignore the co-author edges representing the co-authored papers less than 5. A more complex example of influence graph with 12 authors and 12 conferences (or keywords) is presented in Figure 2.

3.2. Design Framework

The overall design of the social influence based graph clustering framework consists of three components: (a) Selection of the associated information networks and construction of the corresponding influence networks, (b) Defining the influence-based similarity and the co-influence model, (c) Designing an iterative learning based clustering algorithm that can integrate multiple influence networks and their co-influence.

The first component can be viewed as the preparation step for the social influence based graph clustering. The choice of associated information networks is usually domain-specific and clustering goal specific. For instance, if we want to partition facebook people based on their sport events, then all activity networks that are sport-



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

Fig. 2. An Illustrating Example of Influence Graphs

related are chosen, such as sport product network, sport news network, sport blog network. At the same time, sport vertices (sport news or blogs) should be connected based on their similarities in terms of sport categories or players of sport news or blogs. The other irrelevant information networks such as music, debate, cooking will not be selected.

In this paper, we aim to partition a social graph SG based on self influence similarity and co-influence similarity for each pair of vertices in SG. The second component is thus to perform the social influence similarity computation in three steps before executing the clustering algorithm on SG.

— The self-influence similarity is computed on SG itself by taking into account both topological structure and static attributes. The result is a $N_{SG} \times N_{SG}$ matrix of $\mathrm{W_0}$ with each entry representing the self-influence similarity score of a pair of vertices in SG.
— The co-influence similarity is computed on IG_i by utilizing SG, AG_i and the links between U and V_i. The result is a $N_{SG} \times N_{SG}$ matrix of $\mathrm{W_i}$ with each element specifying the co-influence similarity score of a pair of vertices in SG. If we have N number of AG_is, then we can generate N co-influence similarity matrices denoted as $\mathrm{W_1}, \cdots, \mathrm{W_N}$.
— The unified influence-based similarity between any pair of vertices in the social graph SG can be computed by combining the self-influence similarity weighted by α with the N weighted co-influence similarity scores, ω_i for $\mathrm{W_i}$ $(i = 1, \cdots, N)$. The weight for each influence-based similarity matrix is defined based on its contribution to the clustering convergence (overall clustering objective).

The third component is to perform the following tasks to complete the social influence based clustering process.

— *Centroid Initialization and Refinement.* Our social influence based clustering algorithm follows the *K-Medoids* clustering method [Kaufman and Rousseeuw 1987] by using the unified influence-based similarity with the initial weights assigned of α and ω_i $(i = 1, \cdots, N)$ as an input. In 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.
— *Dynamic Weight Update.* Although it is simple to integrate the self-influence similarity and the co-influence similarities into a unified similarity space with the static weight assignment, we argue that the static weighted similarity function often results in incorrect and rather biased clustering results. The weight function should capture the fact that different activity graphs may contribute to the clustering of the social graph differently, i.e., assigning large weights to important activities and smal-

l weights or zero weights to irrelevant activities. Furthermore, the weights for both the self-influence similarity and all N co-influence similarity scores should be dynamically tuned at each iteration of the clustering process to continuously improve the clustering quality and the clustering convergence.

— *Maximization of Clustering Objective.* The original clustering objective corresponds to a sophisticated nonlinear fractional programming problem with respect to multiple weights. It is very hard to perform function trend identification and estimation to determine the existence and uniqueness of solutions. Thus we need to transform the original objective function to a monotonic function with a unique solution to maximize the clustering objective.

Our social influence based graph clustering framework consists of two algorithms: SI-Cluster and SI-Cluster-Opt. The later improves the former by reducing the computational complexity of SI-Cluster baseline algorithm through two novel optimizations to speed up the computation of influence-based vertex similarity. In the subsequent sections, we will first describe the influence-based similarity (Section 4) and introduce the social influence based clustering framework (Section 5), followed by the optimization techniques for fast computation of propagating heat kernel (Section 6).

4. INFLUENCE-BASED SIMILARITY

In SI-Cluster, we define social influence-based similarity for each pair of vertices in the social graph SG based on the self-influence similarity scores computed on the social graph SG itself (Section 4.1) and N co-influence similarity scores computed on N activity graphs AG_i and N influence graphs IG_i (Section 4.2 and Section 4.3). Then we combine the $N + 1$ pairwise influence-based similarity scores by defining a weighted unified influence-based similarity measure (Section 4.4).

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. Heat diffusion kernel has received increasing attention in recent years [Kondor and Lafferty 2002], [Belkin and Niyogi 2003], [Lafferty and Lebanon 2005], [Yang et al. 2007], and [Ma et al. 2008]. In this paper, we define our heat diffusion based influence propagation model to capture multiple kinds of social influence based similarity scores between member vertices in the social graph. Based on the conventional heat diffusion kernel [Ma et al. 2008], we define two propagating heat diffusion kernels, one for computing self-influence similarity in the social graph in this section and one for computing co-influence similarity in each of the activity graphs in the next sections.

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 4.1. [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.

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

where $f_i(t)$ is the vertex u_i's temperature at time t and vertex u_j is an neighbor of u_i. 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. Fox example, in the social graph of DBLP, p_{ij} actually denotes the Geometric mean between the proportion of the number of co-authored publications by u_i and u_j to the number of authored publications by u_i and the proportion of the number of co-authored publications by u_i and u_j to the number of authored publications by u_j, i.e., $p_{ij}=\sqrt{\frac{n_{ij}}{n_i}\times\frac{n_{ij}}{n_j}}=\frac{n_{ij}}{\sqrt{n_in_j}}$. The non-negative heat diffusion coefficient α determines the conduction velocity of heat within the social graph, i.e., has an important effect on the convergence of heat diffusion process. If α is relatively large, heat will diffuse very quickly. Otherwise, heat will diffuse slowly. In the extreme case, if α is equal to zero, then heat will never diffuse among vertices. On the other hand, if α is infinite, then heat will diffuse from one vertex to other vertices immediately. We express the above heat diffusion formulation in a matrix form.

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

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.

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

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

For the social graph of Figure 1 (b), the 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. Let vertices u_1, u_2, u_3, u_4 and u_5 represent *Philip S. Yu*, *Jiawei Han*, *Charu C. Aggarwal*, *Kun-Lung Wu* and *Haixun Wang* respectively. Thus, $n_1=622$, $n_2=472$, $n_3=139$, $n_4=106$, $n_5=123$, $n_{12}=46$, $n_{13}=73$, $n_{14}=78$, $n_{15}=46$ and $n_{23}=8$. We have H for Figure 1 (b) defined as follow.

$$\mathbf{H}=\begin{pmatrix}-0.8033 & 0.0849 & 0.24827 & 0.30377 & 0.16631\\ 0.0849 & -0.1161 & 0.03123 & 0 & 0\\ 0.24827 & 0.03123 & -0.2795 & 0 & 0\\ 0.30377 & 0 & 0 & -0.30377 & 0\\ 0.16631 & 0 & 0 & 0 & -0.16631\end{pmatrix} \tag{4}$$

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 $\Delta t \to 0$.

$$\frac{d\mathrm{f}(t)}{dt} = \alpha \mathrm{H}\mathrm{f}(t) \tag{5}$$

Solving this differential equation, we obtain the following Eq.(6).

Definition 4.2. [Propagating Heat Diffusion Kernel on Social Graph] Let α denote the thermal conductivity, H be the non-propagating diffusion kernel of SG and $\mathrm{f}(0)$ 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 $\mathrm{f}(t)$, is an exponential function with variable t for constant $\mathrm{f}(0)$.

$$\mathrm{f}(t) = e^{\alpha t \mathrm{H}} \mathrm{f}(0) \tag{6}$$

We call $e^{\alpha t \mathrm{H}}$ as the propagating heat diffusion kernel. According to Chapter I in [Bhatia 1997], the matrix exponential can be expanded as a Taylor series. We thus rewrite $e^{\alpha t \mathrm{H}}$ as the following Taylor series, where I is an identity matrix:

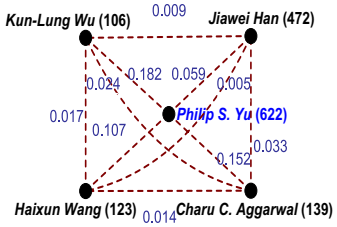
$$e^{\alpha t \mathrm{H}} = \mathrm{I} + \alpha t \mathrm{H} + \frac{\alpha^2 t^2}{2!}\mathrm{H}^2 + \frac{\alpha^3 t^3}{3!}\mathrm{H}^3 + \cdots \tag{7}$$

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. Consider the example in Figure 1(b), by setting α and time t equal to 1, we can compute the self-influence similarity, defined by $e^{\alpha t \mathrm{H}}$, on the social graph, as follow.

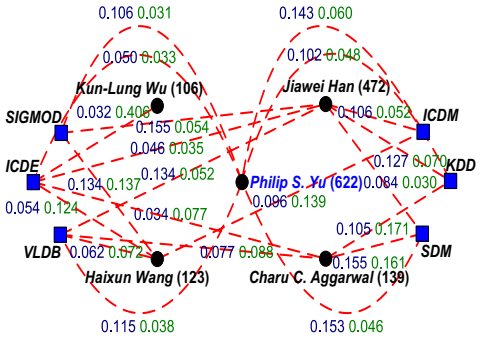
$$e^{\alpha t \mathrm{H}} = e^{\mathrm{H}} = \begin{pmatrix} 0.49977 & 0.05901 & 0.15172 & 0.18211 & 0.10736 \\ 0.05901 & 0.89358 & 0.03316 & 0.00907 & 0.00520 \\ 0.15172 & 0.03316 & 0.77673 & 0.02439 & 0.01400 \\ 0.18211 & 0.00907 & 0.02439 & 0.76750 & 0.01693 \\ 0.10736 & 0.00520 & 0.01400 & 0.01693 & 0.85650 \end{pmatrix} \tag{8}$$

Figure 3 follows the example of Figure 1. Figure 3 (a) presents the above-mentioned self-influence similarity matrix on the social graph in Figure 1(b) where ochre dashed lines and associated blue numbers represent the corresponding self-influence similarity scores. This example matrix shows that, in contrast to non-propagating heat kernel based similarity computation (see Eq.(4), the authors who have no joint publications now have small but non-zero self-influence similarity scores. This makes a lot of sense since even if *Charu C. Aggarwal* and *Kun-Lung Wu* have no joint publications, both are co-authors of *Philip S. Yu* and have published in the same conferences. Thus, they may have some indirect self-influence similarity scores on one another. It is not only inappropriate but also incorrect to model their self-influence similarity as zero.

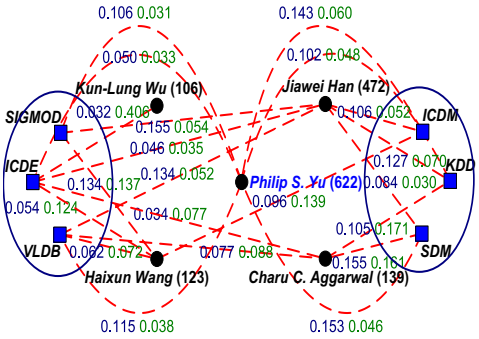
It is worth noting that when we use the publication vector of authors as the initial temperature to execute heat diffusion propagation on the social graph in Figure 1(b), we observe some interesting phenomenon. As shown in Table I, $\mathrm{f}(t)$ denotes the temperature of authors at time t. The heat vector $\mathrm{f}(0)$ is initialized with the total number of



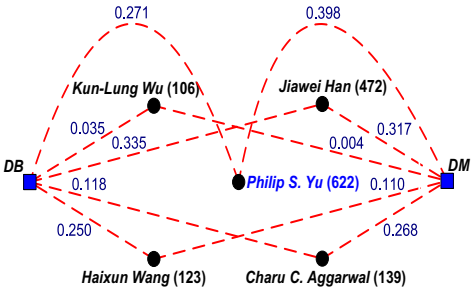
(a) Self-influence Similarity



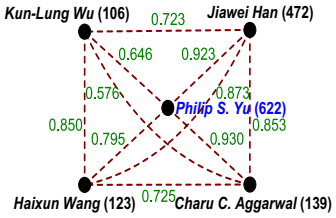
(b) Kernel on Influence Graph



(c) Activity Partition



(d) Influence Propagation



(e) Co-influence Similarity

Fig. 3. Co-influence Model

publications of each author on six conferences of *ICDM*, *KDD*, *SDM*, *SIGMOD*, *VLDB* and *ICDE*. f$(t)(1)$, f$(t)(2)$, f$(t)(3)$, f$(t)(4)$ and f$(t)(5)$ correspond to the temperature at time t for *Philip S. Yu*, *Jiawei Han*, *Charu C. Aggarwal*, *Kun-Lung Wu* and *Haixun Wang* respectively. Although both *Philip S. Yu* and *Jiawei Han* have high initial temperatures, *Philip S. Yu*'s temperature decreases quickly when t increases. This is because he has more coauthors compared to *Jiawei Han* and his heat diffuses more quickly. Similarly, although *Charu C. Aggarwal* and *Haixun Wang* have the same initial temperature, the former's temperature increases quickly when t increases. This is because *Charu C. Aggarwal* receives the heat diffused from both *Philip S. Yu* and *Jiawei Han* but *Haixun Wang* gains the heat diffusion from only *Philip S. Yu*. Although

Table I. Influence Propagation on Social Graph Based on Publications of Authors

t	0	1	2	20	200
f(t)(1)	211	133.88	107.95	103.88	104.71
f(t)(2)	180	175.74	168.67	110.86	104.74
f(t)(3)	61	86.48	95.63	104.47	104.72
f(t)(4)	11	51.02	68.54	103.08	104.71
f(t)(5)	61	76.88	83.21	101.68	104.71

the heat diffusion speed are quite different, the entire system can achieve a stable thermal equilibrium after enough time.

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 4.3. [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.

$$\mathrm{H_i} = \begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} \tag{9}$$

where $\mathrm{B} = [\mathrm{B_1}, \cdots, \mathrm{B_{N_{AG_i}}}]^T$ 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.(10); $\mathrm{C} = [\mathrm{C_1}, \cdots, \mathrm{C_{N_{SG}}}]^T$ 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.(11); A is an $N_{AG_i} \times N_{AG_i}$ matrix representing the activity similarities, defined by Eq.(12); and D is a $N_{SG} \times N_{SG}$ diagonal matrix.

$$B_{jk} = \begin{cases} \dfrac{n_{jk}}{\sum_{l=1}^{N_{AG_i}} n_{lk}}, & (u_k, v_j) \in T_i, \\ 0, & otherwise. \end{cases} \tag{10}$$

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.

$$C_{jk} = \begin{cases} \dfrac{n_{jk}}{\sum_{l=1}^{N_{SG}} n_{lk}}, & (u_j, v_k) \in T_i, \\ 0, & otherwise. \end{cases} \tag{11}$$

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.

$$A_{jk} = \begin{cases} n_{jk}, & (v_j, v_k) \in S_i, \\ -\tau_j, & j = k, \\ 0, & otherwise. \end{cases} \tag{12}$$

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.

$$\mathrm{H_{conf}} = \left(\begin{array}{cccccc|ccccc} -0.64606 & 0.00967 & 0.00897 & 0.00408 & 0.00397 & 0.00464 & 45/211 & 28/180 & 0 & 0 & 15/61 \\ 0.00967 & -0.77344 & 0.00905 & 0.00407 & 0.00393 & 0.00460 & 37/211 & 43/180 & 20/61 & 0 & 0 \\ 0.00897 & 0.00905 & -0.53857 & 0.00408 & 0.00392 & 0.00443 & 32/211 & 14/180 & 17/61 & 0 & 0 \\ 0.00408 & 0.00407 & 0.00408 & -0.51132 & 0.00977 & 0.00947 & 20/211 & 28/180 & 0 & 0 & 14/61 \\ 0.00397 & 0.00393 & 0.00392 & 0.00977 & -0.57204 & 0.00937 & 26/211 & 28/180 & 9/61 & 0 & 7/61 \\ 0.00464 & 0.00460 & 0.00443 & 0.00947 & 0.00937 & -2.14662 & 51/211 & 39/180 & 15/61 & 11/11 & 25/61 \\ \hline 45/88 & 37/100 & 32/63 & 20/62 & 26/70 & 51/141 & -2.44501 & 0 & 0 & 0 & 0 \\ 28/88 & 43/100 & 14/63 & 28/62 & 28/70 & 39/141 & 0 & -2.09861 & 0 & 0 & 0 \\ 0 & 20/100 & 17/63 & 0 & 9/70 & 15/141 & 0 & 0 & -0.70480 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11/141 & 0 & 0 & 0 & -0.07801 & 0 \\ 15/88 & 0 & 0 & 14/62 & 7/70 & 25/141 & 0 & 0 & 0 & 0 & -0.67357 \end{array}\right) \tag{13}$$

The non-propagating heat diffusion kernel $\mathrm{H_{conf}}$ on the conference influence graph in Figure 1(d) is given as Eq.(13) where each dimension in $\mathrm{H_{conf}}$ represents 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*, respectively. The top left block represents the conference similarities, the top right block specifies the influences of conferences on the social graph through authors, and the bottom left block denotes the influences of authors on the conference activity graph through conferences. For example, the sixth row represents the *ICDE* conference, it has large similarities with *SIGMOD* (0.00947) and *VLDB* (0.00937) but has small similarities with *ICDM* (0.00464), *KDD* (0.00460), and *SDM* (0.00443). $26/211$, $28/180$, $9/61$, 0 and $7/61$ in the fifth row represent the influence of *VLDB* on the social graph through *Philip S. Yu*, *Jiawei Han*, *Charu C. Aggarwal*, *Kun-Lung Wu* and *Haixun Wang*, respectively. The seventh row specifies the influence of *Philip S. Yu* on the conference activity graph through *ICDM* ($45/88$), *KDD* ($37/100$), *SDM* ($32/63$), *SIGMOD* ($20/62$), *VLDB* ($26/70$) and *ICDE* ($51/141$). The minus entries in each row represent the amount of diffused influences.

Definition 4.4. [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 $\mathrm{f}(0)$ be an initial heat distribution on IG_i. The vertex's thermal capacity at time t is defined by an exponential function $\mathrm{f}(t)$ with variable t for constant $\mathrm{f}(0)$.

$$\left.\begin{array}{ll} \mathrm{f_i}(1) = & e^{\alpha \mathrm{H_i}} \mathrm{f_i}(0) \\ & \cdots \\ \mathrm{f_i}(t) = & e^{\alpha \mathrm{H_i}} \mathrm{f_i}(t-1) \end{array}\right\} \Rightarrow \mathrm{f_i}(t) = e^{\alpha t \mathrm{H_i}} \mathrm{f_i}(0) \tag{14}$$

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

$$e^{\alpha t \mathrm{H_i}} = \mathrm{I} + \alpha t \mathrm{H_i} + \frac{\alpha^2 t^2}{2!} \mathrm{H_i^2} + \frac{\alpha^3 t^3}{3!} \mathrm{H_i^3} + \cdots \tag{15}$$

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

Based on the non-propagating heat diffusion kernel $\mathrm{H}_{\mathrm{conf}}$ in Eq. 13, we calculate the propagating heat diffusion kernel $e^{\alpha t \mathrm{H}_{\mathrm{conf}}}$ for the conference influence graph shown in Figure 3 (b), where both α and t are set to 1. For presentation clarity, we only show the bidirectional influence flow between authors and conferences in $e^{\alpha t \mathrm{H}_{\mathrm{conf}}}$. Associated blue numbers in red edges quantify the influences of authors on the conference activity graph through conferences in $e^{\alpha t \mathrm{H}_{\mathrm{conf}}}$. Associated green numbers measure the influences of conferences on the social graph through authors in $e^{\alpha t \mathrm{H}_{\mathrm{conf}}}$.

In the next subsection, we will execute the iterative label propagation to infer the class membership of each member vertex with respect to the activity-based clusters. We perform the classification by using the propagating heat diffusion kernel $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ as the classifier and the clustered activities as the training data. It is known that a random walk on a graph with its transition matrix will converge to a stationary distribution. Many multi-label relational classification methods [Bhagat et al. 2011] are based on random walk model by iteratively propagating the class labels through performing random walks on the graph. Thus, we transform the propagating heat diffusion kernel $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ into a transition matrix by removing the negative entries and followed by row-wise normalization. Concretely, we first find the smallest element in the matrix $e^{\alpha \mathrm{H}_{\mathrm{i}}}$, denoted by $\min(e^{\alpha \mathrm{H}_{\mathrm{i}}})$. If $\min(e^{\alpha \mathrm{H}_{\mathrm{i}}}) \geq 0$, then all entries in $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ are non-negative and we directly perform the row-wise normalization (e.g., using the sum of all elements of the row as the denominator) such that each row of the normalized $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ sums to 1. Otherwise, we add $-\min(e^{\alpha \mathrm{H}_{\mathrm{i}}})$ to each entry of $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ such that all entries are non-negative. Then we perform the row-wise normalization to turn $e^{\alpha \mathrm{H}_{\mathrm{i}}}$ into a transition matrix and perform the multi-label relational classification on the normalized $e^{\alpha \mathrm{H}_{\mathrm{i}}}$.

4.3. Co-influence Model

Our co-influence model is to define pairwise activity-based influence similarity for members of the social graph. For example, in the DBLP dataset, author *A* has papers on *KDD* and *SIGIR*, and author *B* has publications on *KDD*. During the heat propagation process, the heat will be transferred from *SIGIR* to author *B* through the path *SIGIR* $\rightarrow$ *A* $\rightarrow$ *KDD* $\rightarrow$ *B*. We call this kind of activity-based influence phenomenon the co-influence. We have defined the propagating heat diffusion kernel $e^{\alpha t \mathrm{H}_{\mathrm{i}}}$ for the influence graph IG_i ($1 \leq i \leq N$). According to Eq.(14), in order to conduct heat diffusion on an influence graph and compute pairwise co-influence similarity, we need both $e^{\alpha t \mathrm{H}_{\mathrm{i}}}$ and $\mathrm{f}_{\mathrm{i}}(0)$ on IG_i. $\mathrm{f}_{\mathrm{i}}(0)$ 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. Intuitively, we can figure out a co-influence similarity score between two member vertices in terms of the social influence between any of two members and each of activities in the influence graph. Suppose that N_{SG} is the number of member vertices in the social graph SG and N_{AG_i} is the number of activity vertices in the activity graph AG_i, the computational cost of computing co-influence similarity scores will be approximately equal to $N_{SG}^2 \times N_{AG_i}$. When N_{AG_i} is very large, the computation of the co-influence similarity is similar to DNA microarray calculation. Thus it causes a non-trivial cost and the final co-influence similarity matrix is very dense. For example, a recent version of the DBLP bibliography data contains $964,166$ authors, $6,992$ conferences, $363,352$ keywords and $31,962,786$ heterogeneous links. When we compute keyword influence graph for this DBLP dataset, the final author-author co-influence similarity matrix based on the keyword influence graph will have $964,166$ non-zero entries and it is a full matrix.

To make good use of the topological information of AG_i, find good heat sources from AG_i and reduce the computational cost for large-scale activity graph, we propose to start by partitioning AG_i into M_i disjoint activity clusters, denoted by $c_{i1}, c_{i2}, \ldots, c_{iM_i}$. This partitioning operation also helps us have a better understanding of the social interactions of member vertices in each category of activities. SI-Cluster combines the self-influence and the multiple co-influence similarity scores, one per activity network, into a unified influence similarity scores, through weight assignment and iterative weight refinement, to performance iterative graph clustering until the convergence condition is met.

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.

$$\mathrm{f_{ij}}(0) = (p_{ij1}, p_{ij2}, \cdots, p_{ijN_{AG_i}}, 0, 0, \cdots, 0)^T \tag{16}$$

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 zeros. The total number of entries in $\mathrm{f_{ij}}(0)$ is $N_{AG_i}+N_{SG}$ and the last N_{SG} entries have the initial values of 0. Thus, the initial heat distribution matrix $\mathrm{f_i}(0)$ for the propagating heat diffusion kernel $e^{\alpha t \mathrm{H_i}}$ in Eq.(15) is defined as follow.

$$\mathrm{f_i}(0) = [\mathrm{f_{i1}}(0), \mathrm{f_{i2}}(0), \cdots, \mathrm{f_{iM_i}}(0)] = \begin{bmatrix} p_{i11} & p_{i21} & \cdots & p_{iM_i1} \\ p_{i12} & p_{i22} & \cdots & p_{iM_i2} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i1N_{AG_i}} & p_{i2N_{AG_i}} & \cdots & p_{iM_iN_{AG_i}} \\ p_{i1(N_{AG_i}+1)} & p_{i2(N_{AG_i}+1)} & \cdots & p_{iM_i(N_{AG_i}+1)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i1(N_{AG_i}+N_{SG})} & p_{i2(N_{AG_i}+N_{SG})} & \cdots & p_{iM_i(N_{AG_i}+N_{SG})} \end{bmatrix} \tag{17}$$

where each column in $\mathrm{f_i}(0)$ corresponds to one of the M_i disjoint activity clusters, such as conference clusters *DB* and *DM* in Figure 3 (c).

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

$$\mathrm{f_{conf}}(0) = [\mathrm{f_{DM}}(0), \mathrm{f_{DB}}(0)] = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T \tag{18}$$

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*).

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 class in the influence graph IG_i until approaching convergence, i.e., stable heat distribution. The heat distribution $\mathrm{f_i}(t)$ at time t is then defined as follow.

$$\mathrm{f_i}(t) = [\mathrm{f_{i1}}(t), \mathrm{f_{i2}}(t), \cdots, \mathrm{f_{iM_i}}(t)] = e^{\alpha t \mathrm{H_i}}[\mathrm{f_{i1}}(0), \mathrm{f_{i2}}(0), \cdots, \mathrm{f_{iM_i}}(0)] \tag{19}$$

where $\mathrm{f_i}(t)(j,k)$ denotes the probability of j^{th} activity vertex or member vertex in the class c_{ik} based on the i^{th} influence graph IG_i. Also the first N_{AG_i} rows in $\mathrm{f_i}(t)$ represent the probability distribution of activity vertices in each activity class and the last N_{SG} rows represent the probability distribution of member vertices based on activity class labels.

By setting α and time t equal to 1, we can generate the final heat distribution matrix $\mathrm{f_{conf}}(t)$ for them, which serve as their influence-based probabilities belonging to each of *DM* and *DB*.

$$\begin{aligned}\mathrm{f_{conf}}(t) &= e^{\alpha t \mathrm{H_{conf}}}\mathrm{f_{conf}}(0) = e^{\mathrm{H_{conf}}}\mathrm{f_{conf}}(0) \\ &= \begin{pmatrix} 0.62537 & 0.59262 & 0.68311 & 0.07281 & 0.09125 & 0.09832 & 0.39837 & 0.31718 & 0.26773 & 0.00361 & 0.11046 \\ 0.10230 & 0.09606 & 0.06515 & 0.68725 & 0.64891 & 0.22615 & 0.27052 & 0.33491 & 0.11775 & 0.03455 & 0.25032 \end{pmatrix}^T\end{aligned} \tag{20}$$

Given that the goal of our social influence based graph clustering is to partition the member vertices in the social graph (e.g., the coauthor graph or the consumer graph) into clusters, we further reduce the final heat distribution 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 redundant training rows and normalize the remaining test rows for $\mathrm{f'_i}(t)$ without loss of quality. This reduction also helps to reduce the computational complexity and improve memory consumption. The final influence-based classification matrix $\mathrm{f'_i}(t)$ is given as follow.

$$\mathrm{f'_i}(t) = \begin{bmatrix} p_{i1(N_{AG_i}+1)} & p_{i2(N_{AG_i}+1)} & \cdots & p_{iM_i(N_{AG_i}+1)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i1(N_{AG_i}+N_{SG})} & p_{i2(N_{AG_i}+N_{SG})} & \cdots & p_{iM_i(N_{AG_i}+N_{SG})} \end{bmatrix} \tag{21}$$

The reduced formula of $\mathrm{f'_{conf}}(t)$ is given as follow.

$$\mathrm{f'_{conf}}(t) = \begin{pmatrix} 0.59557 & 0.48641 & 0.69454 & 0.09460 & 0.30617 \\ 0.40443 & 0.51359 & 0.30546 & 0.90540 & 0.69383 \end{pmatrix}^T \tag{22}$$

Figure 3 (d) shows the heat distribution $\mathrm{f'_{conf}}(t)$ in the two different conference classes: *DM* and *DB*. The larger the blue number is, the more influence that the corresponding author has on the conference class. Charu Aggarwal has more influence on *DM* (0.69454) than *DB* (0.30546). Note that there is a red line between *Kun-Lung Wu* and *DM* with the value of 0.09460, even though *Kun-Lung Wu* does not have any publications on *DM* conferences. The influence between *Kun-Lung Wu* and *ICDM* is derived from the influence between *Kun-Lung Wu* and *ICDE* and the similarity between *ICDE* and *ICDM* in terms of the common set of authors who publish in both conferences, even if the similarity score between *ICDE* and *ICDM* is relatively small.

The pairwise vertex closeness is an important measure of clustering quality. Let $\mathrm{W_i}$ denote the co-influence vertex similarity matrix for influence graph IG_i, M_i be the number of activity classes in IG_i, and $\mathrm{f'_{im}}(t)(j)$ 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.

$$
\begin{aligned}
W_i(j,k) = W_i(k,j) &= 1 - \frac{\sqrt{\sum_{m_i=1}^{M}(\mathbf{f}'_{\mathbf{im}}(t)(j) - \mathbf{f}'_{\mathbf{im}}(t)(k))^2}}{\sum_{m_i=1}^{M}\mathbf{f}'_{\mathbf{im}}(t)(j) + \mathbf{f}'_{\mathbf{im}}(t)(k)} \\
&= 1 - \frac{\sqrt{\sum_{m_i=1}^{M}(p_{im(N_{AG_i}+j)} - p_{im(N_{AG_i}+k)})^2}}{\sum_{m_i=1}^{M} p_{im(N_{AG_i}+j)} + p_{im(N_{AG_i}+k)}}
\end{aligned}
\tag{23}
$$

The green numbers in Figure 3 (e) represents the co-influence similarity between author vertices based on the conference influence graph. *Philip S. Yu* and *Charu C. Aggarwal* have a higher co-influence similarity of $1 - \sqrt{(0.59557 - 0.69454)^2 + (0.40443 - 0.30546)^2}/((0.59557+0.69454)+(0.40443+0.30546)) = 0.93002$ since they both have more influence in *DM* than that in *DB*. On the other hand, *Philip S. Yu* and *Kun-Lung Wu* have a smaller co-influence similarity of $1 - \sqrt{(0.59557 - 0.09460)^2 + (0.40443 - 0.90540)^2}/((0.59557+0.09460)+(0.40443+0.90540)) = 0.64572$ since *Kun-Lung Wu* has much more influence in *DB* than that in *DM*.

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, $\mathrm{W_i}$ denote the co-influence similarity from the influence graph IG_i $(1 \leq i \leq N)$ with the weight ω_i. The unified similarity function W is defined as follow.

$$
\mathrm{W} = \mathrm{W}_0 + \omega_1 \mathrm{W}_1 + \cdots + \omega_N \mathrm{W_N} \tag{24}
$$

where $\mathrm{W}_0 = e^{\alpha t \mathrm{H}}$, $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$, $\alpha \geqslant 0$, $\omega_i \geqslant 0$, $i = 1, \cdots, N$.

By assigning an equal initial weight $\alpha = \omega_1 = \ldots = \omega_N = 1$ for each kind of social influence-based similarity matrices, we assume that each kind of social influence-based similarity matrices has the same degree of importance. However, different types of similarity scores may have different degree of contributions in calculating the unified influence-based similarity matrix, thus we need to adaptively update the weights after each clustering iteration in terms of their contributions towards the clustering convergence, i.e., under the fixed constraint of $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$. Such dynamic weight update may continuously increase the weights to important influence-based similarities and decrease the weights or assign zero weights to trivial influence-based similarities in terms of the result of each clustering iteration.

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

$$
\begin{aligned}
s(u_i, u_j) = W(i,j) &= e^{\alpha t H}(i,j) + \omega_1 W_1(i,j) + \cdots + \omega_N W_N(i,j) \\
&= (I(i,j) + \alpha t H(i,j) + \frac{\alpha^2 t^2}{2!} H^2(i,j) + \cdots) + \omega_1 W_1(i,j) + \cdots + \omega_N W_{\mathrm{N}}(i,j) \\
&= \sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j)
\end{aligned}
\tag{25}
$$

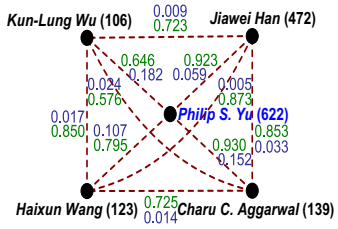


Fig. 4. A Unified Influence-based Similarity

The main challenge for the computation of the unified influence-based similarity is to set and tune the weight parameters α and ω_i ($1 \leq i \leq N$) as well as interpret the weighted similarity function. Figure 4 integrates the self-influence similarity in Figure 3 (a) and the co-influence similarity in Figure 3 (e) into a unified influence-based similarity. As previously mentioned, the blue numbers represent the self-influence similarity scores between authors based on the social graph SG. The green numbers, which are calculated in terms of Eqs.(22) and (23), specify the co-influence similarity scores between authors based on the conference influence graph. It is not clear to a user whether the weight of self-influence similarity matrix should be larger or smaller than the weight of conference-influence similarity matrix. It is even harder for the user to decide the weights quantitatively. We will discuss how to find an optimal solution for a nonlinear programming model with weight parameters in the next section.

5. SOCIAL INFLUENCE BASED CLUSTERING FRAMEWORK

We have presented our approach to compute the weighted unified social influence based similarity by combining the self-influence similarity matrix and the N activity-based co-influence similarity matrices for each pair of member vertices in the social graph. In this section we present our social influence based clustering framework, called SI-Cluster, which follows the *K-Medoids* clustering method [Kaufman and Rousseeuw 1987] by using the unified influence-based similarity with the initial weights as an input. It partitions a social graph SG by iteratively combining both self-influence and co-influence similarities with refined weight assignment through our unified similarity model on SG, its associated N activity graphs AG_i, and N influence graphs IG_i ($1 \leq i \leq N$). At each iteration, we perform three tasks: (1) assign the vertices to their closest centroids, (2) select the most centrally located point in a cluster as a centroid, and (3) update the $N + 1$ weights using the weight update method, which computes the weighted contribution by each influence-based similarity to the clustering objective for the social graph and the N associated activity-based influence graphs. The clustering process is repeated until the convergence condition is met.

We will first discuss the weight and centroid initialization of the SI-Cluster algorithm in Section 5.1, and the vertex assignment and centroid update in Section Section 5.3. Then we describe the clustering objective function in Section 5.2, the parameter-based optimization in Section 5.4 and iterative weight adjustment in Section 5.5, followed by the clustering algorithm in Section 5.6.

5.1. Initialization

In the initialization step of our SI-Cluster algorithm, we need to address two issues: (1) initial weight setup and (2) cluster centroid initialization.

Good weight assignment for our unified influence-based similarity measure is crucial to produce a good clustering result. Recall Equation (24) in Section 4.4, $\alpha \geqslant 0$ is the weight for self-influence similarity and $\omega_i \geqslant 0$ ($1 \leq i \leq N$) are the N weights for activity-based co-influence similarity scores and they satisfy the constraint $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$. In SI-Cluster, we assign the initial value of 1 to all $N + 1$ weights, i.e., $\alpha = \omega_1 = \cdots = \omega_N = 1$, based on the assumption that each kind of social influence-based similarity matrices has the same degree of importance. By assigning an equal initial weight, we start to combine the self-influence similarity matrix and the N co-influence similarity matrices into a unified influence-based similarity matrix with equal contributions. We will update the $N + 1$ weight values in the subsequent iteration of the clustering process using our dynamic weight update scheme, which continuously quantify and adjust the weights on self-influence similarity and on multiple co-influence similarity scores towards the clustering convergence, while maintaining the constraint $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$. As a result, at each iteration, weights to important influence-based similarities are increased while weights to trivial influence-based similarities are decreased or become zero, ensuring the clustering process progresses towards the convergence. Note that choosing a weight assignment randomly often results in incorrect clustering results. In Sections 5.4 and 5.5, we will show that there exists one and only one optimal weight assignment to maximize the clustering objective.

Good initial centroids are essential for the success of partitioning clustering algorithms. There are a number of studies by using various criteria, such as *DENCLUE* [Hinneburg and Keim 1998] and *K-Means++* [Arthur and Vassilvitskii 2007]. The idea of *K-Means++* [Arthur and Vassilvitskii 2007] is that the first center is chosen uniformly at random and each subsequent center is chosen from the remaining data points with probability proportional to its squared distance from the closest existing center. *DENCLUE* produces good cluster characteristics from noisy data by choosing the local maxima of the density function as centers. SI-Cluster follows the same motivation of *DENCLUE* to initialize cluster centroids. Concretely, in SI-Cluster, we first compute the density for each member vertex in the social graph and then find the local maxima of the density function as the centroids.

Definition 5.1. [Density Function] The density function of one vertex u_i is the sum of the unified similarity scores between u_i and all other vertices in U.

$$D(u_i) = \sum_{u_j \in U, u_j \neq u_i} s(u_i, u_j) \tag{26}$$

If one member vertex u_i has a large density value, it means that, either u_i connects to many member vertices through the links within the social graph SG, say coauthor links, or u_i has the similar activity participation behaviors with many member vertices through the links between SG and the activity graph AG_i ($i = 1, \ldots, N$). Based on the density value of each member vertices, we find the member vertices with a local maximum of the density value by following the same hill-climbing strategy in *DENCLUE*. A member vertex which has a local maximum of the density value often can diffuse its heat to many member vertices along multiple paths. A centroid-based cluster is thus formed when heat is diffused to the margin of the social graph. We sort all such member vertices in the descending order of their density values and select top-K member vertices as the initial K centroids $\{c_1^0, ..., c_K^0\}$.

5.2. Clustering Objective Function

The clustering objective function in SI-Cluster is defined by considering both the intra-cluster similarity and the inter-cluster similarity. Similar to most existing work, we

define the intra-cluster similarity with the sum of the similarity scores between vertices and their centroid within each cluster. There are two common ways to define inter-cluster similarity: (1) the sum of the similarity scores between pairwise centroids across clusters and (2) the sum of the similarity scores between pairwise vertices across clusters, both of which are used to quantitatively measure the extent of similarity between two clusters. Given that the pairwise vertex similarity in SI-Cluster is defined based on a complex weighted unifying similarity function, each cluster may have more representative points than just its centroid. Thus, SI-Cluster adopts the second definition as our inter-cluster similarity. Compared to traditional *K-Means* or *K-Medoids*, whose objective functions only consider the intra-cluster distance (similarity), SI-Cluster defines the objective function as the ratio of the intra-cluster similarity to the inter-cluster similarity to better capture the complexity of clustered graphs. We below formally define the inter-cluster similarity and our objective function.

Definition 5.2. [Inter-cluster Similarity] Let $SG = (U, E)$ be the social graph, $W(i, j)$ be 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.

$$s(U_p, U_q) = \sum_{u_i \in U_p, u_j \in U_q, u_i \neq u_j} s(u_i, u_j) = \sum_{u_i \in U_p, u_j \in U_q, u_i \neq u_j} W(i, j) \tag{27}$$

We below formally define the objective function as the ratio of the intra-cluster similarity to the inter-cluster similarity.

Definition 5.3. [Graph Clustering Objective Function] Let $SG = (U, E)$ be a social graph with the weight α and $IG_1, IG_2, \ldots, IG_N$ be N influence graphs with the weights $\omega_1, \ldots, \omega_N$ where ω_i is the weight for IG_i, and K is 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 $\forall 1 \leq i, j \leq K, i \neq j$, and the following objective function $O(\{U_l\}_{l=1}^K, \alpha, \omega_1, \ldots, \omega_N)$ is maximized.

$$\begin{aligned} O(\{U_l\}_{l=1}^K, \alpha, \omega_1, \ldots, \omega_N) &= \frac{\sum_{p=1}^K \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} s(u_i, u_j)}{\sum_{p=1}^K \sum_{q=1, q \neq p}^K s(U_p, U_q)} \\ &= \frac{\sum_{p=1}^K \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i, j) + \sum_{k=1}^N \omega_k W_k(i, j))}{\sum_{p=1}^K \sum_{q=1, q \neq p}^K \sum_{u_i \in U_p, u_j \in U_q} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i, j) + \sum_{k=1}^N \omega_k W_k(i, j))} \end{aligned} \tag{28}$$

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

The numerator of Eq.(28) captures the intra-cluster similarity by aggregating the intra-cluster similarities from all K clusters. The denominator of Eq.(28) represents the aggregated inter-cluster similarity of all K clusters. We define the clustering objective is to maximize the above objective function, i.e., maximize the ratio of the intra-cluster similarity to the inter-cluster similarity.

We model the graph clustering problem as the optimization of three subproblems: (1) vertex 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 in literature. The weight adjustment is unique because SI-Cluster defines pairwise vertex similarity by combining self-influence similarity score and co-influence similarity scores from multiple activity networks. In addition to improve the objective function for vertex assignment and centroid update in each iter-

ation, SI-Cluster also dynamically refines all weighting parameters towards the goal of maximizing the objective function.

5.3. Vertex Assignment and Centroid Update

Given that the goal of SI-Cluster is to maximize the ratio of the intra-cluster similarity to the inter-cluster similarity, which is different from that of traditional *K-Means* and *K-Medoids*, in order to guarantee that the objective function of SI-Cluster is improved by vertex assignment, we need to verify both the intra-cluster condition and the inter-cluster condition for vertex assignment. Given K centroids in the t^{th} iteration, we perform the intra-cluster similarity test on each vertex $u_i \in U$ by computing its closest centroid $c^* = argmax_{c_j^t} s^t(u_i, c_j^t)$, *i.e.*, a centroid $c^* \in \{c_1^t, ..., c_K^t\}$ with the largest unified similarity from u_i. We perform the inter-cluster similarity test by introducing an additional verification condition for vertex assignment.

Concretely, for each candidate vertex u_i to be assigned to a new centroid c_q^t, we first find $c_q^t = argmax_{c_j^t} s^t(u_i, c_j^t)$ such that $s^t(u_i, c_p^{t-1}) < s^t(u_i, c_q^t)$, and $c_p^{t-1} \neq c_q^t$, where c_p^{t-1} is the centroid to which u_i was assigned in the last iteration. Then we make the final decision of whether to assign u_i to this new c_q^t by checking whether the condition of $\sum_{u_j \in U_q, u_i \neq u_j} s^t(u_i, u_j) \geq \sum_{u_j \in U_p, u_i \neq u_j} s^t(u_i, u_j)$ is satisfied. This is because when this condition is satisfied, by assigning u_i to the cluster with the new centroid c_q^t, then the inter-cluster similarity between U_p and U_q in the denominator in the objective function will be reduced from $s^t(U_p, U_q)$ to $s^t(U_p, U_q) - \sum_{u_j \in U_q, u_i \neq u_j} s^t(u_i, u_j) + \sum_{u_j \in U_p, u_i \neq u_j} s^t(u_i, u_j)$ or remain unchanged. At the same time, the intra-cluster similarity about u_i in the numerator of the objective function will be increased from $s^t(u_i, c_p^{t-1})$ to $s^t(u_i, c_q^t)$ or remain unchanged. Thus, the total clustering objective function will be improved by vertex assignment. When u_i passes both tests during the vertex assignment phase, we will assign u_i to its closest centroid c_q^t.

Centroid update in SI-Cluster is similar to traditional *K-Means* and *K-Medoids* except that our pairwise vertex similarity is unique by combining both self-influence similarity score and co-influence similarity scores from multiple activity networks. When each vertex is 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" a_i of a cluster U_i in terms of the unified similarity matrix as

$$s(a_i, u_j) = \frac{1}{|U_i|} \sum_{u_k \in U_i} s(u_k, u_j), \forall u_j \in U_i \tag{29}$$

Thus, $s(a_i, :)$ 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

$$c_i^{t+1} = argmin_{u_j \in U_i} \|s(u_j, :) - s(a_i, :)\| \tag{30}$$

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.

Note that the numerator in the objective function is the sum of the similarity scores between vertices and their centroid. Thus, the centroid update operation will make the numerator increased or keep it unchanged. According to the definitions of the inter-cluster similarity and the objective function, the centroid update operation does not have any affect on (no change to) the denominator. Thus, the centroid update will improve the clustering objective function.

5.4. Parameter-based Optimization

The objective of our clustering algorithm is to maximize intra-cluster similarity and minimize inter-cluster similarity, i.e., maximize a quotient of two functions of multiple variables. We cannot directly solve this sophisticated nonlinear fractional programming problem (NFPP). Then we show that the NFPP problem is equivalent to a polynomial programming problem with polynomial constraints (PPPPC). The polynomial constraints of PPPPC make it very hard to solve. Thus, we further simplify the dynamic weight update problem as a nonlinear parametric programming problem (NPPP), which can obtain optimal weights through parameter-based optimization.

Definition 5.4. Suppose that $f(\alpha, \omega_1, \ldots, \omega_N) = \sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} (\sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j) + \sum_{k=1}^{N} \omega_k W_k(i,j))$ 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).

$$Max\ O(\{U_l\}_{l=1}^{K}, \alpha, \omega_1, \ldots, \omega_N) = \frac{f(\alpha, \omega_1, \ldots, \omega_N)}{g(\alpha, \omega_1, \ldots, \omega_N)} \tag{31}$$

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

The preliminary knowledge of convexity/concavity can be found in Chapters 2, 3, 4, 5 and 32 in [Rockafellar 1997].

THEOREM 5.5. *$f(\alpha, \omega_1, \ldots, \omega_N)$ is either or both of convex and concave 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 follow.

$$\Pi(f)_{ij}(\alpha, \omega_1, \ldots, \omega_N) = D_i D_j f(\alpha, \omega_1, \ldots, \omega_N) \tag{32}$$

where D_i is the differentiation operator with respect to the i^{th} argument and the Hessian becomes

$$\Pi(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial \alpha^2} & \frac{\partial^2 f}{\partial \alpha \partial \omega_1} & \cdots & \frac{\partial^2 f}{\partial \alpha \partial \omega_N} \\ \frac{\partial^2 f}{\partial \omega_1 \partial \alpha} & \frac{\partial^2 f}{\partial \omega_1^2} & \cdots & \frac{\partial^2 f}{\partial \omega_1 \partial \omega_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial \omega_N \partial \alpha} & \frac{\partial^2 f}{\partial \omega_N \partial \omega_1} & \cdots & \frac{\partial^2 f}{\partial \omega_N^2} \end{bmatrix} \tag{33}$$

Since there is only one non-linear term $\sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} \sum_{k=0}^{\infty} \frac{\alpha^k t^k}{k!} H^k(i,j)$, the final Hessian matrix is

$$\Pi(f) = \begin{bmatrix} \sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} \sum_{k=0}^{\infty} k(k-1) \frac{\alpha^{k-2} t^k}{k!} H^k(i,j) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (34)$$

Since this Hessian matrix is a diagonal matrix, all of its eigenvalues are non-negative (non-positive or zero) if the only diagonal entry $\sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} \sum_{k=0}^{\infty} k(k-1) \frac{\alpha^{k-2} t^k}{k!} H^k(i,j)$ *is positive (negative or zero). Thus, it is positive-semidefinite (negative-semidefinite, both positive-semidefinite and negative-semidefinite) for* $\forall \alpha$, ω_1, ..., $\omega_N \in S$ *if the only diagonal entry in the Hessian matrix is positive (negative or zero). Accordingly,* $f(\alpha, \omega_1, \dots, \omega_N)$ *is convex (concave, both convex and concave) on the convex set* S.

THEOREM 5.6. $g(\alpha, \omega_1, \dots, \omega_N)$ *is either or both of convex and concave on the set* $S = \{(\alpha, \omega_1, \dots, \omega_N) | \alpha + \sum_{i=1}^{N} \omega_i = N + 1, \alpha \geqslant 0, \omega_i \geqslant 0, i = 1, \cdots, N\}$.

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

According to Theorems 5.5-5.6, we know that it is very difficult to identify whether $f(\alpha, \omega_1, \dots, \omega_N)$ or $g(\alpha, \omega_1, \dots, \omega_N)$ is convex or concave on the set S. According to Definition 5.4, we know that our clustering objective is equivalent to maximize a quotient of $f(\alpha, \omega_1, \dots, \omega_N)$ and $g(\alpha, \omega_1, \dots, \omega_N)$. It is very hard to decide whether this quotient is convex or concave. We cannot easily perform function trend identification and estimation to determine the existence and uniqueness of solutions of the clustering objective such that we can not directly solve this sophisticated nonlinear fractional programming problem (NFPP). Therefore, we need to convert the original NFPP to an easily solvable optimization problems to improve the efficiency of SI-Cluster.

THEOREM 5.7. *The NFPP problem is equivalent to a polynomial programming problem with polynomial constraints (PPPPC).*

$$Max\ \gamma f(\alpha, \omega_1, \dots, \omega_N) \quad (35)$$

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

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

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

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 5.8. *A nonlinear parametric programming problem (NPPP) is defined as* $F(\beta) = Max\ \{f(\alpha, \omega_1, \dots, \omega_N) - \beta g(\alpha, \omega_1, \dots, \omega_N)\}$ *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.(31) 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* $(\overline{\alpha}, \overline{\omega}_1, \ldots, \overline{\omega}_N)$*.*

Recall Eq.(24), we define a compact constraint of $\alpha + \sum_{i=1}^{N} \omega_i = N + 1$. One necessary condition of Theorem 5.8 is that two equivalent problems in Theorem 5.8 should be defined on the same compact constraint space, i.e., the constraint space of the NFPP problem and the NPPP problem is a compact space, i.e., closed and bounded space. We have explained that it is very hard to solve the original NFPP problem. To ease the solving of the NFPP problem, we introduce a compact constraint space of $\alpha + \sum_{i=1}^{N} \omega_i = N+1$ and transform the NFPP problem into the NPPP problem. In addition, the weight learning process may not converge if there is no compact constraint.

Now we have successfully transformed the original NFPP in Eq.(31) 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 5.9. $F(\beta)$ *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 the definition of convexity in Chapter 4 in [Rockafellar 1997], we know that* $F(\beta)$ *is convex.*

THEOREM 5.10. $F(\beta)$ *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 5.11. $F(\beta) = 0$ *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 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

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 $U_1, ..., U_K$.

```
Calculate W_0, W_1, W_2, ..., W_N, and W;
Select K initial centroids with a local maximum of the density value;
Repeat until the objective function F(β) converges:
    Assign each vertex u_i to a cluster C* with a centroid c* where c* = argmax_{c_j} s(u_i, c_j);
    Update the cluster centroids with the most centrally located point in each cluster;
    Solve the NPPP of F(β);
    Update α, ω_1, ..., ω_N;
    Refine β = f(α, ω_1, ..., ω_N)/g(α, ω_1, ..., ω_N);
    Update W;
Return K clusters U_1, ..., U_K.
```

the next round. The algorithm repeats the above-mentioned iterative procedure until $F(\beta)$ converges to 0.

5.6. Clustering Algorithm

By assembling the vertex assignment, centroid update and adaptive weight adjustment together, we provide the pseudo code of our clustering algorithm - SI-CLUSTER in Algorithm 1. Our SI-Cluster algorithm is a variant of *K-Medoids* [Kaufman and Rousseeuw 1987] in the sense that we divide the original optimization problem into three optimization subproblems and iteratively solve each subproblem with the goal of optimizing the clustering objective.

THEOREM 5.12. *The objective function in Algorithm 1 converges to a local maximum in a finite number of iterations.*

Proof. Similar to the divide-and-conquer framework of K-Medoids *[Kaufman and Rousseeuw 1987], SI-Cluster solves the optimization problem by iteratively solving the following three subproblems: (1) given cluster centroids and weight assignment, the vertex assignment step will assign each vertex to its closest centroid; (2) given vertex assignment and weight assignment, the centroid update step will update cluster centroids with the most centrally located point in each cluster; and (3) given vertex assignment and cluster centroids, the weight adjustment step will refine the weight for each kind of influence-based similarity matrix with the optimal weight assignment.*

Let t be the current iteration, O_0^t be the clustering objective before vertex assignment in the t^{th} iteration, O_1^t be the clustering objective before centroid update, O_2^t be the clustering objective before weight adjustment, and O_3^t be the clustering objective after weight adjustment, then $O_3^t = O_0^{t+1}$. As discussed in Section 5.3, the first subproblem (vertex assignment) will make $\sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} s(u_i, u_j)$ increased or keep unchanged, and make $\sum_{p=1}^{K} \sum_{q=1, q \neq p}^{K} s(U_p, U_q)$ decreased or keep unchanged. Thus, $O_0^t \leq O_1^t$. In addition, the second subproblem (centroid update) will make $\sum_{p=1}^{K} \frac{1}{|U_p|-1} \sum_{u_i \in U_p, u_j = c_p, u_i \neq u_j} s(u_i, u_j)$ increased or keep unchanged, and keep the denominator of the objective function, $\sum_{p=1}^{K} \sum_{q=1, q \neq p}^{K} s(U_p, U_q)$, unchanged. Therefore, $O_1^t \leq O_2^t$.

Now we show that solving the subproblem (3) in SI-Cluster can also optimize the clustering objective of SI-Cluster. Note that the original clustering objective function in Eqs.(28) and (31) is $f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$, i.e., the ratio of the intra-cluster similarity to the inter-cluster similarity. On the other hand, β is updated

with $\beta = f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$ *at each iteration. Essentially,* β *is equivalent to the original clustering objective function. The algorithm starts with* $\alpha = \omega_1 = \ldots = \omega_N = 1$ *to calculate the unified influence-based similarity matrix* W *and do the graph clustering. Based on the initial* $\beta = 0$*, we solve the NPPP in Theorem 5.8 to generate the new weights* $\alpha, \omega_1, \ldots, \omega_N$ *and update* β *with the new ratio* $\beta = f(\alpha, \omega_1, \ldots, \omega_N)/g(\alpha, \omega_1, \ldots, \omega_N)$ *of the intra-cluster similarity to the inter-cluster similarity. Notice that* $F(\beta)$ *is monotonic decreasing in terms of Theorem 5.10. Thus,* $F(\beta)$ *keeps decreasing but* β *keeps increasing, i.e., the original clustering objective function in Eq.(31) keeps increasing and* $O_2^t \leq O_3^t$*. In each iteration, SI-Cluster repeats the above process until* $F(\beta)$ *converges to zero: utilize* β *obtained at the last iteration to figure out the new weights* $\alpha, \omega_1, \ldots, \omega_N$ *with the goal of maximizing the clustering objective, and update* β *with the new weights* $\alpha, \omega_1, \ldots, \omega_N$ *helps the algorithm enter the next iteration. Thus,* β *constantly keeps increasing, i.e., the original clustering objective function in Eq.(31) continuously keeps increasing during the clustering process. To sum up,* $O_0^t \leq O_1^t \leq O_2^t \leq O_0^{t+1} \leq \cdots$.

In summary, the entire algorithm iteratively executes the above three tasks until convergence: (1) utilize improved cluster centroids and weight assignment to assign vertices through both intra-cluster similarity test and inter-cluster similarity test; (2) use refined vertex assignment and weight assignment to update centroids, improving intra-cluster similarity without changing inter-cluster similarity; and (3) make use of elevated vertex assignment and cluster centroids to refine the weights for self-influence similarity score and co-influence similarity scores such that the pairwise vertex similarity is improved towards the clustering objective. As a result, the overall objective function keeps increasing and can converge to a local maximum in a finite number of iterations.

6. SPEED UP PROPAGATING HEAT KERNEL COMPUTATION

In real applications, the non-propagating heat kernel H or $\mathrm{H_i}$ may be a high dimension matrix and very dense (recall Eqs.(4) and (13). To compute the propagating heat diffusion kernel $e^{\alpha t \mathrm{H}}$ in Eq.(7) or $e^{\alpha t \mathrm{H_i}}$ in Eq.(15), we have to calculate H^2, H^3, H^4, $\cdots$, or $\mathrm{H_i}^2$, $\mathrm{H_i}^3$, $\mathrm{H_i}^4$, $\cdots$, i.e., a mass of matrix multiplication operations in total. To improve the computational cost of self-influence similarity matrix and co-influence similarity matrices, we develop diagonalizable-matrix approximation to speed up the computation of propagating heat kernels. For a symmetric non-propagating kernel, by directly utilizing similarity transformation in Chapter 3 in [Arfken et al. 2005] to convert it to a diagonal matrix, the computational cost of exact propagating kernel is reduced from $O((l-1)N_{SG}^3)$ to $O(N_{SG}^3)$ for $e^{\alpha t \mathrm{H}}$ or from $O((l-1)(N_{SG}+N_{AG_i})^3)$ to $O((N_{SG}+N_{AG_i})^3)$ for $e^{\alpha t \mathrm{H_i}}$ where l is the highest order of the power of H in $e^{\alpha t \mathrm{H}}$ or $\mathrm{H_i}$ in $e^{\alpha t \mathrm{H_i}}$. For an asymmetric non-propagating kernel, by proposing diagonalizable-matrix approximation to generate its diagonalizable approximation, the computational cost of approximate propagating kernel is reduced from $O((l-1)N_{SG}^3)$ to $O(N_{SG}^3)$ for $e^{\alpha t \mathrm{H}}$ or from $O((l-1)(N_{SG}+N_{AG_i})^3)$ to $O(N_{AG_i}^3 + 2(N_{SG}+N_{AG_i})^3)$ for $e^{\alpha t \mathrm{H_i}}$. Extensive experimental evaluation demonstrates that our SI-Cluster-Opt approach can achieve a good approximation of SI-Cluster while meeting the guarantee of high density, low entropy and low DBI.

6.1. Undirected Heat Kernel Computation

Notice that the non-propagating kernel H or $\mathrm{H_i}$ for an undirected graph is often a symmetric matrix. In this subsection, based on the symmetric non-propagating kernel, we aim to calculate the propagating kernel by using the similarity transformation to reduce the number of matrix multiplication operations.

We first introduce some preliminary concepts before discussing the fast computation of propagating heat kernel. Given a square matrix A, $\sum_{i=0}^{\infty} c_i \mathrm{A}^i$ is called the *Power Series* of A (assume $\mathrm{A}^0 = \mathrm{I}$). In particular, when all coefficients c_i ($i = 0, 1, \ldots$) are equal to the reciprocal of factorial of i, the Power series of A is converted to $\sum_{i=0}^{\infty} \frac{1}{i!}\mathrm{A}^i$. We call it the *Taylor series* for the exponential function e^{A}. In related literature, we can find the following theorems on diagonal transformation. Theorems 6.2-6.15, Corollaries 6.4 and 6.16, and the detailed proofs can be found in Chapters 1, 2 and 4 in [Horn and Johnson 1990], Chapters 5 and 7 in [Golub and Loan 1996], Chapters 2, 4-7 in [Strang 2005] and Chapter 18 in [Manning et al. 2008].

Definition 6.1. Two n-by-n square matrices A and B are called similar if $\mathrm{B} = \mathrm{P}^{-1}\mathrm{AP}$ for some invertible n-by-n square matrix P.

Similar matrices represent the same linear transformation under two different bases, with P being the change of basis matrix. Similarity is an equivalence relation on the space of square matrices. Thus,

THEOREM 6.2. *A real symmetric matrix* A *can always be diagonalized in the form of* $\mathrm{A} = \mathrm{PDP}^{-1}$*, where* P *is a orthogonal matrix and* D *is diagonal with* n *eigenvalues of* A *as diagonal entries.*

The non-propagating kernel H or $\mathrm{H_i}$ for an undirected graph is often a real symmetric matrix. Theorem 6.2 provides a feasible method to transform a dense matrix A with at most n^2 non-zero elements into a sparse diagonal matrix B with up to n non-zero entries. Thus, the computation of the power of A with the complexity of $O(n^3)$ can also be reduced to the calculation of the power of B with the complexity of $O(n)$.

The Gram-Schmidt algorithm in Chapter 3 in [Arfken et al. 2005] implements diagonalization by computing its eigenvalues, orthogonal matrix and diagonal matrix of a diagonalizable matrix. Due to the space constraint, we only present the detailed speed-up formulae for the propagating heat diffusion kernel $e^{\alpha t \mathrm{H_i}}$. We can adopt the similar steps to compute the speed-up formulae for $e^{\alpha t \mathrm{H}}$.

$$\mathrm{D_i} = \mathrm{P}^{-1}\mathrm{H_i}\mathrm{P} \tag{36}$$

$$\mathrm{D_i} = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_{(N_{SG}+N_{AG_i})} \end{pmatrix} \tag{37}$$

where $\lambda_1, \lambda_2, \cdots, \lambda_{(N_{SG}+N_{AG_i})}$ are $N_{SG} + N_{AG_i}$ eigenvalues of $\mathrm{H_i}$.

The calculation of power of non-propagating heat kernel is then simplified as follow.

$$\mathrm{H_i^n} = \mathrm{PD_i^n P^{-1}} \tag{38}$$

Thus, we revise the propagating heat kernel $e^{\alpha t \mathrm{H_i}}$ in Eq.(15) with diagonal transformation as $e^{\alpha t \mathrm{H_i}}$ in Eq.(39), where $\lambda_1, \lambda_2, \cdots, \lambda_{(N_{SG}+N_{AG_i})}$ are $N_{SG} + N_{AG_i}$ eigenvalues of $\mathrm{H_i}$. The invertible matrices P is used to do similarity transformation for $\mathrm{H_i}$.

Our optimization techniques can dramatically reduce the computational complexity and the memory consumption through diagonalization of dense matrix since the

matrix exponential can be computed simply by taking each diagonal entry to the exponential in question. Suppose that l is the possible highest order for the power of $\mathrm{H_i}$. To compute $e^{\alpha t\mathrm{H}}$ or $e^{\alpha t\mathrm{H_i}}$, we need to calculate H^2, H^3, $\cdots$, H^l, or $\mathrm{H_i}^2$, $\mathrm{H_i}^3$, $\cdots$, $\mathrm{H_i}^l$. Thus the number of matrix multiplication operations can be reduced from $(l-1)$ to 2 which is the cost of diagonal-to-symmetric transformation. However, these two matrix multiplications include one multiplication between one regular matrix and one diagonal matrix, and one regular multiplication. The first regular-diagonal matrix multiplication has the complexity of $O((N_{SG}+N_{AG_i})^2)$. In addition, the cost of the exponential of eigenvalues is equal to $O(N_{SG}+N_{AG_i})$. We can safely ignore the computational cost of these two operations since both are $\ll O((N_{SG}+N_{AG_i})^3)$. The additional cost refers to the Gram-Schmidt computation for generating its eigenvectors, eigenvalues, orthogonal matrix and diagonal transformation. When there are m principal eigenvectors to be determined, it has a complexity of $O((N_{SG}+N_{AG_i})m^2)$ [Trefethen and Bau 1997], which is also $\ll O((N_{SG}+N_{AG_i})^3)$. To sum up, the total cost of propagating heat kernel computation is reduced from $O((l-1)(N_{SG}+N_{AG_i})^3)$ to $O((N_{SG}+N_{AG_i})^3)$.

$$
\begin{aligned}
e^{\alpha t\mathrm{H_i}} &= \mathrm{I}+\alpha t\mathrm{H_i}+\frac{\alpha^2t^2}{2!}\mathrm{H_i}^2+\frac{\alpha^3t^3}{3!}\mathrm{H_i}^3+\cdots \\
&= \mathrm{P}(\mathrm{I}+\alpha t\mathrm{D_i}+\frac{\alpha^2t^2}{2!}\mathrm{D_i}^2+\frac{\alpha^3t^3}{3!}\mathrm{D_i}^3+\cdots)\mathrm{P}^{-1} \\
&= \mathrm{P}(\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} + \begin{pmatrix} \alpha t\lambda_1 & 0 & \dots & 0 \\ 0 & \alpha t\lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha t\lambda_{(N_{SG}+N_{AG_i})} \end{pmatrix} \\
&+ \begin{pmatrix} \frac{\alpha^2t^2}{2!}\lambda_1^2 & 0 & \dots & 0 \\ 0 & \frac{\alpha^2t^2}{2!}\lambda_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\alpha^2t^2}{2!}\lambda_{(N_{SG}+N_{AG_i})}^2 \end{pmatrix} \\
&+ \begin{pmatrix} \frac{\alpha^3t^3}{3!}\lambda_1^3 & 0 & \dots & 0 \\ 0 & \frac{\alpha^3t^3}{3!}\lambda_2^3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\alpha^3t^3}{3!}\lambda_{(N_{SG}+N_{AG_i})}^3 \end{pmatrix} + \cdots)\mathrm{P}^{-1} \\
&= \mathrm{P}(\begin{pmatrix} e^{\alpha t\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\alpha t\lambda_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\alpha t\lambda_{(N_{SG}+N_{AG_i})}} \end{pmatrix})\mathrm{P}^{-1}
\end{aligned}
\tag{39}
$$

6.2. Directed Heat Kernel Computation

In the above computation defined in Eq.(39), we assume that the non-propagating kernel $\mathrm{H_i}$ is a symmetric matrix. Note that the non-propagating kernel $\mathrm{H_i}$ for a directed graph maybe an asymmetric matrix. If it is not symmetric or diagonalizable, the diagonal transformation formula can not be directly applied to solve the problem. In this subsection, we will propose diagonalizable-matrix approximation to address this problem. The core idea is to attempt to find multiple smaller symmetric matrices to implement the diagonal transformation formula respectively.

Recall Subsection 4.3, we partition the activity graph AG_i into M_i disjoint activity clusters of c_{i1}, c_{i2}, ..., c_{iM_i}. To maintain a high-quality approximation, based on the above-mentioned activity partitions, we split the non-propagating heat kernel $\mathrm{H_i}$ for a directed influence graph IG_i into two matrices: $\mathrm{H_{i1}}$ with all intra-cluster links and $\mathrm{H_{i2}}$ with all inter-cluster links. The intra-cluster matrix $\mathrm{H_{i1}}$ can be obviously organized as the following block diagonal matrix.

$$\mathrm{H_{i1}} = \begin{bmatrix} \mathrm{H_1} & \mathrm{O} & \dots & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{H_2} & \dots & \mathrm{O} & \mathrm{O} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{H_{M_i}} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{O} & \mathrm{D} \end{bmatrix} \tag{40}$$

where submatrices $\mathrm{H_1}$, $\mathrm{H_2}$, $\cdots$, $\mathrm{H_{M_i}}$ contain intra-cluster links in clusters c_{i1}, c_{i2}, ..., c_{iM_i}, respectively. The submatrix D is the diagonal matrix D in the non-propagating heat kernel $\mathrm{H_i}$ in Eq.(9).

For the example of the non-propagating heat diffusion kernel $\mathrm{H_{conf}}$ on the conference influence graph in Eq.(13), we split it into the following two matrices.

$$\mathrm{H_{conf1}} = \left(\begin{array}{ccc|ccc|ccccc} -0.64606 & 0.00967 & 0.00897 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00967 & -0.77344 & 0.00905 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00897 & 0.00905 & -0.53857 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -0.51132 & 0.00977 & 0.00947 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.00977 & -0.57204 & 0.00937 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.00947 & 0.00937 & -2.14662 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -2.44501 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.09861 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.70480 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.07801 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.67357 \end{array}\right) \tag{41}$$

$$\mathrm{H_{conf2}} = \begin{pmatrix} 0 & 0 & 0 & 0.00408 & 0.00397 & 0.00464 & 45/211 & 28/180 & 0 & 0 & 15/61 \\ 0 & 0 & 0 & 0.00407 & 0.00393 & 0.00460 & 37/211 & 43/180 & 20/61 & 0 & 0 \\ 0 & 0 & 0 & 0.00408 & 0.00392 & 0.00443 & 32/211 & 14/180 & 17/61 & 0 & 0 \\ 0.00408 & 0.00407 & 0.00408 & 0 & 0 & 0 & 20/211 & 28/180 & 0 & 0 & 14/61 \\ 0.00397 & 0.00393 & 0.00392 & 0 & 0 & 0 & 26/211 & 28/180 & 9/61 & 0 & 7/61 \\ 0.00464 & 0.00460 & 0.00443 & 0 & 0 & 0 & 51/211 & 39/180 & 15/61 & 11/11 & 25/61 \\ 45/88 & 37/100 & 32/63 & 20/62 & 26/70 & 51/141 & 0 & 0 & 0 & 0 & 0 \\ 28/88 & 43/100 & 14/63 & 28/62 & 28/70 & 39/141 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20/100 & 17/63 & 0 & 9/70 & 15/141 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11/141 & 0 & 0 & 0 & 0 & 0 \\ 15/88 & 0 & 0 & 14/62 & 7/70 & 25/141 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{42}$$

Although $\mathrm{H_{i1}}$ is a block diagonal matrix, $\mathrm{H_1}$, $\mathrm{H_2}$, $\cdots$, $\mathrm{H_{M_i}}$ and the inter-cluster matrix $\mathrm{H_{i2}}$ may be asymmetric matrices such that they may be non-diagonalizable matrices. We attempt to find diagonalizable matrices $\mathrm{H'_1}$, $\mathrm{H'_2}$, $\cdots$, $\mathrm{H'_{M_i}}$ and $\mathrm{H'_{i2}}$ to approximate these non-diagonalizable matrices respectively while minimizing the difference between the two matrices.

THEOREM 6.3. *An n-by-n matrix* A *over the field F is diagonalizable if and only if the sum of the dimensions of its eigenspaces is equal to n, which is the case if and only if there exists a basis of F^n consisting of eigenvectors of* A.

Theorem 6.3 presents the necessary and sufficient condition whether a square matrix is diagonalizable.

COROLLARY 6.4. *An n-by-n matrix* A *is diagonalizable over the field F if it has n distinct eigenvalues in F, i.e., if its characteristic polynomial has n distinct roots in F. Opposite may be false.*

Corollary 6.4 tells us that we can eliminate duplicate eigenvalues of a real asymmetric matrix A to generate its diagonalizable approximation A'.

The following definitions and theorems give us the theoretical fundamental of finding the eigenvalues of a real asymmetric matrix A by doing Schur Decomposition.

Definition 6.5. Two n-by-n square matrices A and B are called congruent if $\mathrm{B} = \mathrm{P}^{\mathrm{T}}\mathrm{AP}$ for some invertible n-by-n square matrix P.

Definition 6.6. Two n-by-n real square matrices A and B are called orthogonally similar if $\mathrm{B} = \mathrm{P}^{\mathrm{T}}\mathrm{AP}$, i.e., if $\mathrm{B} = \mathrm{P}^{-1}\mathrm{AP}$, where P is orthogonal and B is upper quasi-triangular.

Orthogonal similarity implies both similarity and congruence, i.e., Definition 6.6 implies Definition 6.5. An upper quasi-triangular matrix is a block upper triangular matrix with either 1-by-1 or 2-by-2 blocks on the diagonal. Thus, B contains up to $(n^2 + 3n - 2)/2$ non-zero elements. Definition 6.6 renders a possible solution to transform a dense real matrix A with at most n^2 non-zero elements into a sparse matrix B with at most $(n^2+3n-2)/2$ non-zero entries. The orthogonal similarity transformation is famous as Real Schur Decomposition, i.e., B is a real Schur form of A.

Definition 6.7. An n-by-n square matrix P is a unitary matrix if $\mathrm{P}^{\mathrm{H}}\mathrm{P} = \mathrm{PP}^{\mathrm{H}} = \mathrm{I}$ where I is an n-by-n identity matrix.

P^{H} is the conjugate transpose (or Hermitian transpose) of P. It is often used to transform a regular matrix into an upper triangular matrix.

Definition 6.8. Two n-by-n complex square matrices A and B are called unitarily similar if $\mathrm{B} = \mathrm{P}^{\mathrm{H}}\mathrm{AP}$, i.e., if $\mathrm{B} = \mathrm{P}^{-1}\mathrm{AP}$, where P is unitary and B is upper triangular.

Definition 6.8 can be used to transform a dense complex matrix A with at most n^2 non-zero elements into a sparse matrix B with at most $n(n+1)/2$ non-zero entries. The unitary similarity transformation is also known as Complex Schur Decomposition, i.e., B is a complex Schur form of A.

THEOREM 6.9. *If* B *is a real Schur form of* A*, then the upper quasi-triangular matrix* B *has a block upper triangular structure where each diagonal block is of size either 1-by-1, corresponding to a real eigenvalue, or 2-by-2, corresponding to a pair of complex eigenvalues that are conjugates of one another.*

Theorem 6.9 tells us that we can obtain all eigenvalues of A from the diagonal of B by doing Schur Decomposition of $\mathrm{B} = \mathrm{P}^{-1}\mathrm{AP}$ and eliminate duplicate eigenvalues to generate a diagonalizable approximation of A. Formally, the Real Schur Decomposition is defined as follow.

Definition 6.10. **Real Schur Decomposition**. If A is an n-by-n real matrix, then there exists an n-by-n real orthogonal matrix P such that

$$\mathrm{P^T AP = B} = \begin{bmatrix} \mathrm{B_{11}} & \mathrm{B_{12}} & \dots & \mathrm{B_{1m}} \\ \mathrm{O} & \mathrm{B_{22}} & \dots & \mathrm{B_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{B_{mm}} \end{bmatrix} \tag{43}$$

where B is a real upper quasi-triangular matrix, which is called a real Schur form of A. Since B is similar to A, it has the same multiset of eigenvalues. Each diagonal block matrix $\mathrm{B_{ii}}$ is either 1-by-1 (in which case they represent real eigenvalues) or 2-by-2 (in which case they are derived from complex conjugate eigenvalue pairs).

THEOREM 6.11. *If* A *is an* n*-by-*n *real asymmetric matrix, then its eigenvalues are either real or complex.*

According to Theorem 6.11, there are two possible cases to be discussed separately as follows.

Case 1: A with all real eigenvalues.

Thus, all $\mathrm{B_{ii}}$s in B are of size 1-by-1, corresponding to all eigenvalues.

$$\mathrm{P^T AP = B} = \begin{bmatrix} \mathrm{B_{11}} & \mathrm{B_{12}} & \dots & \mathrm{B_{1n}} \\ \mathrm{O} & \mathrm{B_{22}} & \dots & \mathrm{B_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{B_{nn}} \end{bmatrix} \tag{44}$$

According to Corollary 6.4, A is non-diagonalizable since it has duplicate eigenvalues. Suppose that A has s duplicate eigenvalues and $n-s$ distinct eigenvalues, for ease of presentation, we use a set S of positive integers represents the indices of duplicate eigenvalues in B such that $|S| = s$ and $i \in \{1, 2, \dots, n\}$ for $\forall i \in S$.

Next, we will use Real Schur Decomposition to create a diagonalizable matrix $\mathrm{A'}$ to approximate the non-diagonalizable matrix A. Since $\mathrm{A'}$ is diagonalizable, we call this problem the *diagonalizable-matrix approximation* problem. Recall Eq.(39), we know that the exponential of a diagonalizable matrix can be reduce to the exponential of eigenvalues. A good diagonalizable approximation is crucial to speed up the computation of propagating heat diffusion kernel while meeting the guarantee of high-quality approximation. Given D is a diagonal matrix consisting of eigenvalues of the non-diagonalizable matrix A, and $\mathrm{D'}$ is a diagonal form of the diagonalizable approximation $\mathrm{A'}$, the matrix difference between D and $\mathrm{D'}$ is measured by the Frobenius norm of $\mathrm{X = D' - D}$. The Frobenius norm can be defined as:

$$\|\mathrm{X}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}^2} \tag{45}$$

Our goal is to find a new diagonal matrix $\mathrm{D'}$ to approximate D, while constraining $\mathrm{A'}$ is diagonalizable. The following five steps are proposed to solve this diagonalizable-matrix approximation problem.

— **Step 1**: Reduce a non-diagonalizable matrix A to an upper Hessenberg matrix H which is zero below the first subdiagonal. The reduction can be implemented with $\mathrm{H = Q^{-1}AQ}$ where Q is orthogonal.

— **Step 2**: Reduce the upper Hessenberg matrix H to a real Schur form B. The reduction can be implemented with $\mathrm{B = R^T HR}$ where R is orthogonal.

— **Step 3**: A new matrix B' is derived by substituting the s duplicate diagonal entries $\mathrm{B_{ii}}$ with $\mathrm{B_{ii}} \pm \epsilon_i$ for $\forall i \in S$, where ϵ represents a very small positive number, i.e., $\epsilon_i > 0$ and $\epsilon_i \approx 0$, such that $\mathrm{B_{ii}} \neq \mathrm{B_{jj}}$ for $\forall i, j \in \{1, 2, \ldots, n\}, i \neq j$.
— **Step 4**: Calculate the diagonalizable-matrix approximation $\mathrm{A}' = (\mathrm{QR})\mathrm{B}'(\mathrm{QR})^{-1}$.
— **Step 5**: Reduce the approximate matrix A' to a diagonal matrix $\mathrm{D}' = \mathrm{P}^{-1}\mathrm{A}'\mathrm{P}$ where P is orthogonal.

Theoretical analysis of this diagonalizable-matrix approximation is presented as follow.

THEOREM 6.12. *A' is the diagonalizable approximation of* A *with a negligible error represented by the Frobenius norm of* $\mathrm{X} = \mathrm{D}' - \mathrm{D}$ *where* D' *is a diagonal form of* A'*,* D *is a diagonal matrix consisting of eigenvalues of* A*, and diagonal entries in* D' *and diagonal entries in* D *are in one-to-one correspondence.*

Proof. Suppose that a set S of positive integers represents the indices of duplicate eigenvalues in D *such that $|S| = s$ and $i \in \{1, 2, \ldots, n\}$ for $\forall i \in S$. Let $\mathrm{D}'_{\mathrm{ii}} = \mathrm{D_{ii}} \pm \epsilon_i$ for $\forall i \in S$, where $\epsilon_i > 0$ and $\epsilon_i \approx 0$, the error of diagonalizable approximation can be generated as follow.*

$$\mathrm{D}' - \mathrm{D}\|_F = \sqrt{\sum_{i=1}^{n} (\mathrm{D}'_{\mathrm{ii}} - \mathrm{D_{ii}})^2} \tag{46}$$

Given the assumption that $\mathrm{D}'_{\mathrm{ii}} = \mathrm{D_{ii}} \pm \epsilon_i$ for $\forall i \in S$, the above equation is equivalent to:

$$\sqrt{\sum_{i=1}^{n} (\mathrm{D}'_{\mathrm{ii}} - \mathrm{D_{ii}})^2} = \sqrt{\sum_{i=1, i\in S}^{n} (\mathrm{D}'_{\mathrm{ii}} - \mathrm{D_{ii}})^2 + \sum_{i=1, i\notin S}^{n} (\mathrm{D}'_{\mathrm{ii}} - \mathrm{D_{ii}})^2} = \sqrt{\sum_{i=1, i\in S}^{n} \epsilon_i^2} \approx 0 \tag{47}$$

Therefore, A' is the diagonalizable approximation of A *with a near-zero error represented by the Frobenius norm of* $\mathrm{X} = \mathrm{D}' - \mathrm{D}$ *when $\epsilon_i > 0$ and $\epsilon_i \approx 0$ for $\forall i \in S$.*

Case 2: A with both complex and real eigenvalues.
Therefore, each $\mathrm{B_{ii}}$ in B is either 1-by-1 (real eigenvalues) or 2-by-2 (complex eigenvalues).

THEOREM 6.13. *Let* A *be an n-by-n real asymmetric square matrix. If* A *has complex eigenvalues, then they must occur in complex conjugate pairs, meaning that if $a+bi$ is an eigenvalue, where a and b are real, then so is $a - bi$. Also the corresponding eigenvectors occur in conjugate pairs.*

Theorem 6.13 describes that the eigenvalues of each 2-by-2 $\mathrm{B_{ii}}$ are a complex conjugate eigenvalue pair.
Different from Case 1, A in this case has both duplicate complex eigenvalues and duplicate real eigenvalues. In addition, its real Schur form is defined as Eq.(43) since each diagonal block matrix $\mathrm{B_{ii}}$ is either 1-by-1 or 2-by-2. Suppose that A has s duplicate real eigenvalues, t duplicate complex eigenvalues, and $n-s-t$ distinct eigenvalues, we use a set S of positive integers represents the indices of 1-by-1 diagonal block containing duplicate real eigenvalues in B such that $|S| = s$ and $i \in \{1, 2, \ldots, m\}$ for $\forall i \in S$. Similarly, a set T of positive integers denotes the indices of 2-by-2 diagonal block holding duplicate complex eigenvalues in B such that $|T| = t$ and $i \in \{1, 2, \ldots, m\}$ for $\forall i \in T$. Notice that t should be an even number in terms of Theorem 6.9 and Definition 6.10.

The following nine steps are adopted to solve this diagonalizable-matrix approximation problem.

— **Step 1**: The same as Step 1 in Case 1.
— **Step 2**: The same as Step 2 in Case 1.
— **Step 3**: The same as Step 3 in Case 1.
— **for each** 2-by-2 diagonal block matrix B_{jj} in B
— **Step 4**: Reduce B_{jj} to a diagonal matrix $E_{jj} = O_{jj}^{-1} B_{jj} O_{jj}$ where O_{jj} is orthogonal.
— **Step 5**: A new matrix E'_{jj} is derived by substituting two complex conjugate eigenvalues of $a_j + b_j i$ and $a_j - b_j i$ with $\lambda_{jj1} = a_j - b_j + \delta_j$ and $\lambda_{jj2} = a_j - b_j - \delta_j$ respectively, where a_j and b_j are real, δ_j represents a very small positive number, i.e., $\delta_j > 0$ and $\delta_j \approx 0$, such that two approximate eigenvalues λ_{jj1} and λ_{jj2} are different from other actual or approximate eigenvalues of A.
— **Step 6**: Calculate the approximation $B'_{jj} = O_{jj} E'_{jj} O_{jj}^{-1}$.
— **Step 7**: Substitute B_{jj} in B' with B'_{jj}.
— **Step 8**: The same as Step 4 in Case 1.
— **Step 9**: The same as Step 5 in Case 1.

THEOREM 6.14. *A' is the diagonalizable-matrix approximation of* A *with a negligible error represented by the Frobenius norm of* $X = D' - D$ *where* D' *is a diagonal form of* A', D *is a diagonal matrix consisting of eigenvalues of* A*, and diagonal entries in* D' *and diagonal entries in* D *are in one-to-one correspondence.*

Proof. Suppose that a set S of positive integers represents the indices of duplicate real eigenvalues in D *such that $|S| = s$ and $j \in \{1, 2, \ldots, n\}$ for $\forall j \in S$. Also a set T of positive integers denotes the indices of duplicate complex eigenvalues in* D *such that $|T| = t$ and $j \in \{1, 2, \ldots, n\}$ for $\forall j \in T$. Assuming that $D'_{jj} = D_{jj} + \epsilon_j$ for $\forall j \in S$, where $\epsilon_j > 0$ and $\epsilon_j \approx 0$. Let $a_{jk} + b_{jk} i$ and $a_{jk} - b_{jk} i$ be the complex conjugate eigenvalues represented by D_{jj} and D_{kk} respectively, and $a_{jk} - b_{jk} + \delta_{jk}$ and $a_{jk} - b_{jk} - \delta_{jk}$ be the corresponding approximate real eigenvalues represented by D'_{jj} and D'_{kk} for $\forall j, k \in T, j \neq k, \overline{D_{jj}} = D_{kk}$, where $\delta_{jk} > 0$ and $\delta_{jk} \approx 0$, the error of diagonalizable-matrix approximation can be generated as follow.*

$$
\begin{aligned}
\|D' - D\|_F &= \sqrt{\sum_{j=1}^{n} (D'_{jj} - D_{jj})^2} \\
&= \sqrt{\sum_{j=1, j\in S, j\notin T}^{n} (D'_{jj} - D_{jj})^2 + \sum_{j=1, j\in T, j\notin S}^{n} (D'_{jj} - D_{jj})^2 + \sum_{j=1, j\notin S, j\notin T}^{n} (D'_{jj} - D_{jj})^2} \\
&= \sqrt{\sum_{j=1, j\in S, j\notin T}^{n} (D'_{jj} - D_{jj})^2 + \sum_{\overline{D_{ij}} = D_{kk}, j,k\in T, j,k\notin S} (D'_{jj} - D_{jj})^2 + (D'_{kk} - D_{kk})^2} \\
&= \sqrt{\sum_{j=1, j\in S, j\notin T}^{n} \epsilon_j^2 + \sum_{\overline{D_{ij}} = D_{kk}, j,k\in T, j,k\notin S} (-b_{jk} - b_{jk} i + \delta_{jk})^2 + (-b_{jk} + b_{jk} i - \delta_{jk})^2} \\
&= \sqrt{\sum_{j=1, j\in S, j\notin T}^{n} \epsilon_j^2 + \sum_{\overline{D_{ij}} = D_{kk}, j,k\in T, j,k\notin S} 2\delta_{jk}^2 - 4 b_{jk} \delta_{jk} i} \approx 0
\end{aligned}
\tag{48}
$$

Therefore, A' *is the diagonalizable-matrix approximation of* A *with a near-zero error represented by the Frobenius norm of* $\mathrm{X} = \mathrm{D}' - \mathrm{D}$ *when* $\epsilon_j > 0$ *and* $\epsilon_j \approx 0$ *for* $\forall j \in S$*, and* $\delta_{jk} > 0$ *and* $\delta_{jk} \approx 0$ *for* $\forall j, k \in T, j \neq k, \overline{\mathrm{D_{jj}}} = \mathrm{D_{kk}}$.

THEOREM 6.15. *If* A *is a block diagonal matrix over the field* F, $\mathrm{A} = diag[\mathrm{A}_1, \mathrm{A}_2, \cdots, \mathrm{A_k}]$*, and the function* $f(t)$ *is defined on the spectrum of* A*, then* $f(\mathrm{A}) = diag[f(\mathrm{A}_1), f(\mathrm{A}_2), \cdots, f(\mathrm{A_k})]$.

COROLLARY 6.16. *If* A *is a block diagonal matrix over the field* F, $\mathrm{A} = diag[\mathrm{A}_1, \mathrm{A}_2, \cdots, \mathrm{A_k}]$*, then* $e^{\mathrm{A}} = diag[e^{\mathrm{A}_1}, e^{\mathrm{A}_2}, \cdots, e^{\mathrm{A_k}}]$.

Theorem 6.15 and Corollary 6.16 describes the theoretical fundamental of fast computation of matrix exponential. The exponential of $\mathrm{H_{i1}}$ in Eq.(40) is thus rewritten as follow.

$$e^{\alpha t \mathrm{H_{i1}}} = \begin{bmatrix} e^{\alpha t \mathrm{H}_1} & \mathrm{O} & \dots & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & e^{\alpha t \mathrm{H}_2} & \dots & \mathrm{O} & \mathrm{O} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & e^{\alpha t \mathrm{H_{M_i}}} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{O} & e^{\alpha t \mathrm{D}} \end{bmatrix} \tag{49}$$

To sum up, we rewrite the propagating heat kernel $e^{\alpha t \mathrm{H_i}}$ in Eq.(15) as diagonalizable-matrix approximation as $e^{\alpha t \mathrm{H_i}}$ in Eq.(50), where λ'_{jk} represents the k^{th} actual or approximate eigenvalue in the approximate matrix $\mathrm{H'_j}$, D_{jj} specifies the j^{th} diagonal entries in the diagonal matrix D, and λ'_1, λ'_2, $\cdots$, $\lambda'_{(N_{SG}+N_{AG_i})}$ are $N_{SG} + N_{AG_i}$ actual or approximate eigenvalues of $\mathrm{H'_{i2}}$. N_1, N_2, $\cdots$, N_{M_i} are the dimensions of H_1, H_2, $\cdots$, $\mathrm{H_{M_i}}$ respectively, i.e., the number of activity vertices in clusters c_{i1}, c_{i2}, ..., c_{iM_i}, respectively. The orthogonal matrices P', P'_1, P'_2, $\cdots$, $\mathrm{P}'_{\mathrm{M_i}}$ are used to do similarity transformation for $\mathrm{H'_{i2}}$, H'_1, H'_2, $\cdots$, $\mathrm{H}'_{\mathrm{M_i}}$ respectively. For the computation of $e^{\alpha t \mathrm{H_i}}$ in Eq.(50), it is the worst case when $M_i = 1$. In this situation, the number of matrix multiplication operations can be reduced from $(l-1)$ to 5 (2 regular-diagonal matrix multiplications and 3 regular matrix multiplications), which has an approximate cost of $O(N^3_{AG_i} + 2(N_{SG}+N_{AG_i})^3)$. Therefore, we can make use of the diagonalizable-matrix approximation to speed up the propagating heat kernel computation of a directed graph.

7. EVALUATION

In this section we provide both experimental evaluation and complexity analysis of our algorithms. We perform extensive experiments to evaluate the performance of SI-CLUSTER on real graph datasets. All experiments were performed on a PC with 3.00GHz Core 2 Quad CPU and 8GB main memory running Red Hat Linux 5.7. To differentiate the optimized SI-Cluster algorithm from the baseline SI-Cluster algorithm, we refer to the optimized algorithm as SI-Cluster-Opt and our experimental results show that SI-Cluster-Opt significantly improves the time complexity of SI-Cluster baseline algorithm while maintaining high quality of clustering results. We report the effectiveness and efficiency of our social influence based clustering algorithms: SI-Cluster and SI-Cluster-Opt by comparing them with several state-of-the-art graph clustering algorithms. We present three important results. First, our social influence based clustering approach converges very quickly on several real datasets, while offering high quality of clustering results in terms of density, entropy and Davies-Bouldin Index (DBI). Second, both SI-Cluster and SI-Cluster-Opt can scale to large graphs

with good clustering performance and high quality clustering results, whereas existing algorithms suffer from "out of memory" problem and fail to complete due to large intermediate result size. Third, SI-Cluster-Opt can effectively reduce the runtime of SI-Cluster baseline algorithm by two-thirds for large datasets.

$$
\begin{aligned}
& e^{\alpha t \mathrm{H_i}} = e^{\alpha t(\mathrm{H_{i1}}+\mathrm{H_{i2}})} = e^{\alpha t \mathrm{H_{i1}}} \times e^{\alpha t \mathrm{H_{i2}}} \\
&= \begin{bmatrix} e^{\alpha t \mathrm{H}_1} & \mathrm{O} & \dots & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & e^{\alpha t \mathrm{H}_2} & \dots & \mathrm{O} & \mathrm{O} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & e^{\alpha t \mathrm{H}_{\mathrm{M_i}}} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{O} & e^{\alpha t \mathrm{D}} \end{bmatrix} \times e^{\alpha t \mathrm{H_{i2}}} \approx \begin{bmatrix} e^{\alpha t \mathrm{H}'_1} & \mathrm{O} & \dots & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & e^{\alpha t \mathrm{H}'_2} & \dots & \mathrm{O} & \mathrm{O} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{O} & \mathrm{O} & \dots & e^{\alpha t \mathrm{H}'_{\mathrm{M_i}}} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \dots & \mathrm{O} & e^{\alpha t \mathrm{D}} \end{bmatrix} \times e^{\alpha t \mathrm{H}'_{\mathrm{i2}}} \\
&= \begin{bmatrix} \mathrm{P}'_1 \left(\begin{pmatrix} e^{\alpha t \lambda'_{11}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e^{\alpha t \lambda'_{1 N_1}} \end{pmatrix} \right) \mathrm{P}'^{-1}_1 & \mathrm{O} & \dots & \mathrm{O} \\ \vdots & \ddots & \vdots & \vdots \\ \mathrm{O} & \dots & \mathrm{P}'_{\mathrm{M_i}} \left(\begin{pmatrix} e^{\alpha t \lambda'_{M_i 1}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e^{\alpha t \lambda'_{M_i N_{M_i}}} \end{pmatrix} \right) \mathrm{P}'^{-1}_{\mathrm{M_i}} & \mathrm{O} \\ \mathrm{O} & \dots & \mathrm{O} & \begin{pmatrix} e^{\alpha t D_{11}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e^{\alpha t D_{N_{SG} N_{SG}}} \end{pmatrix} \end{bmatrix} \\
&\times \mathrm{P}' \left(\begin{pmatrix} e^{\alpha t \lambda'_1} & 0 & \dots & 0 \\ 0 & e^{\alpha t \lambda'_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\alpha t \lambda'_{(N_{AG_i}+N_{SG})}} \end{pmatrix} \right) \mathrm{P}'^{-1}
\end{aligned}
\tag{50}
$$

7.1. Experimental Datasets

We use a full version of the DBLP bibliography data with $964,166$ authors from all research areas (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 multi-typed clustering framework, RankClus [Sun et al. 2009], 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.

7.2. Comparison Methods and Evaluation

We compare **SI-Cluster** and **SI-Cluster-Opt** with three recently developed representative graph clustering algorithms, **BAGC** [Xu et al. 2012], **SA-Cluster** [Zhou et al. 2009] and **Inc-Cluster** [Zhou et al. 2010], 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 interconnected activities into a unified influence-based model. The optimized version SI-Cluster-Opt based on similarity transformation and diagonalizable-matrix approximation is also tested for the evaluation of both effectiveness and efficiency. BAGC constructs a Bayesian probabilistic model to capture both structural and attribute aspects and transform the original clustering problem into a probabilistic inference problem solved by an efficient variational algorithm. Both SA-Cluster and Inc-Cluster combine both structural and attribute similarities in the clustering decisions by estimating the importance of attributes. Inc-Cluster, a optimized version of SA-Cluster, generates the same clustering results and differs only in time complexity. 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.

$$density(\{U_l\}_{l=1}^K) = \sum_{i=1}^{K} \frac{|\{(u_p, u_q)|u_p, u_q \in U_i, (u_p, u_q) \in E\}|}{|E|} \tag{51}$$

density measures the cohesiveness within clusters based on the self-influence similarity (or topological structure) within the social collaboration graph. A large density value means the clusters have cohesive intra-cluster self-influence (or structure).

An information-theoretic measure *entropy* reflects the homogeneity of individual clusters in different semantic environments, i.e., measures how the various co-influences (or static attributes) with individual semantics are distributed within each cluster.

$$entropy(\{U_l\}_{l=1}^K) = \sum_{i=1}^{N} \frac{\omega_i}{\sum_{p=1}^{N} \omega_p} \sum_{j=1}^{K} \frac{|U_j|}{|U|} entropy(a_i, U_j) \tag{52}$$

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. A small entropy value indicates the generated clusters have higher co-influence (or attribute) homogeneity.

Davies-Bouldin Index (DBI) measures the uniqueness of clusters with respect to the unified similarity measure integrating the self-influence similarity and multiple co-influence similarities. It tends to identify set of clusters that are compact and well separated in terms of the unified similarity.

$$DBI(\{U_l\}_{l=1}^K) = \frac{1}{K} \sum_{i=1}^{K} max_{j \neq i} \frac{s(c_i, c_j)}{\sigma_i + \sigma_j} \tag{53}$$

where c_x is the centroid of U_x, $s(c_i, c_j)$ is the unified similarity between c_i and c_j, σ_x is the average similarity of vertices in U_x to c_x. A clustering with high intra-cluster similarity and low inter-cluster similarity will have a low *DBI* value.

7.3. Complexity Analysis

In this subsection, we will perform some complexity analysis to demonstrate how SI-Cluster-Opt is significantly faster than SI-Cluster and the state-of-the-art graph clustering approaches. Let N_{SG} be the number of vertices in the social graph, N be the number of influence graphs, N_{AG_i} be the number of vertices in the i^{th} activity graph, l be the highest order of Taylor series in the heat diffusion kernel calculation process in SI-Cluster-Opt or SI-Cluster, or be the length limit of a random walk in SA-Cluster or Inc-Cluster, K be the number of member clusters, M_i be the number of activity clusters of the i^{th} activity graph in SI-Cluster-Opt or SI-Cluster, t_i be the number of iterations in the process of influence propagation based on the non-propagating heat diffusion kernel $\mathrm{H_i}$, and t be the number of iterations in the clustering process.

(1) SI-Cluster-Opt and SI-Cluster. The total cost of SI-Cluster can be expressed as $T_{diffusion\_kernel} + T_{influence\_propagation} + T_{influence\_similarity} + t(T_{unified\_similarity} + T_{vertex\_assignment} + T_{centroid\_update} + T_{weight\_update})$, where $T_{diffusion\_kernel}$ is the cost of computing $N+1$ propagating heat diffusion kernels $e^{\alpha t\mathrm{H}}, e^{\alpha t_1\mathrm{H_1}}, \cdots, e^{\alpha t_N\mathrm{H_N}}$, $T_{influence\_propagation}$ is the cost of executing the influence propagating on each of N heat diffusion kernels $\mathrm{H_1}, \cdots, \mathrm{H_N}$, $T_{influence\_similarity}$ is the cost of computing N co-influence similarity matrices $\mathrm{W_1}, \mathrm{W_2}, \cdots, \mathrm{W_N}$, $T_{unified\_similarity}$ is the cost of combining $N+1$ influence-based similarity matrices, $T_{vertex\_assignment}$ is the cost of assigning all vertices to cluster centroids, $T_{centroid\_update}$ is the cost of updating cluster centroids, and T_{weight_update} is the cost of updating weights.

$T_{unified\_similarity}$ is equal to $O(2NN_{SG}^2)$ since there are N scalar multiplications of matrix and N matrix additions. $T_{vertex\_assignment}$ is bounded by $O(KN_{SG})$ since the operation performs a linear scan of the graph vertices for each centroid. $T_{centroid\_update}$ is equal to $O(N_{SG}^2)$ since it needs to calculate an average similarity score between each vertex and other vertices within each cluster. T_{weight_update} is bounded by $O(l-1+N)$ since the KKT condition calculation of a polynomial programming problem is dominated by the solving of the variable with the highest order and the solving of a linear equation is bounded by $O(1)$. As the co-influence similarity score between any pair of vertices based on the influence graph IG_i is generated by comparing the member distribution in each of M_i class, $T_{influence\_similarity}$ is equal to $\sum_{i=1}^{N} N_{SG}^2 M_i$. $T_{influence\_propagation}$ is bounded by $\sum_{i=1}^{N} t_i(N_{SG}+N_{AG_i})^2 M_i$ since the influence propagation is executed on M_i class synchronously in each iteration. According to Eqs.(7) and (15), $T_{diffusion\_kernel}$ is bounded by $O((l-1)N_{SG}^3 + \sum_{i=1}^{N}(l-1)(N_{SG}+N_{AG_i})^3 + 2lN_{SG}^2 + \sum_{i=1}^{N} 2l(N_{SG}+N_{AG_i})^2)$ because the propagating heat diffusion kernel calculation consists of a series of matrix multiplications and additions. It is clear that $T_{diffusion\_kernel}$ is the dominant factor in the clustering process. Since $T_{influence\_propagation}, T_{influence\_similarity}, T_{unified\_similarity}, T_{vertex\_assignment}$, $T_{centroid\_update}, T_{weight\_update} \ll T_{diffusion\_kernel}$, the total cost of SI-Cluster is approximately equal to $O((l-1)N_{SG}^3 + \sum_{i=1}^{N}(l-1)(N_{SG}+N_{AG_i})^3)$ by only focusing on the cube terms in $T_{diffusion\_kernel}$.

We adopt two optimization techniques, aiming at optimizing the computation of Taylor series for the computation of propagating heat diffusion kernel: similarity transformation and diagonalizable-matrix approximation in SI-Cluster-Opt. Recall that the similarity transformation technique is defined in Section 6.1, for any positive order l even if $l \to +\infty$, SI-Cluster-Opt can make the total cost of propagating heat kernel computation reduced from $O((l-1)N_{SG}^3)$ to $O(N_{SG}^3)$, or from $O((l-1)(N_{SG}+N_{AG_i})^3)$ to $O((N_{SG}+N_{AG_i})^3)$ for undirected non-propagating heat kernel. On the other hand, in terms of the diagonalizable-matrix approximation in Section 6.2, the time complexity of propagating heat kernel computation is reduced

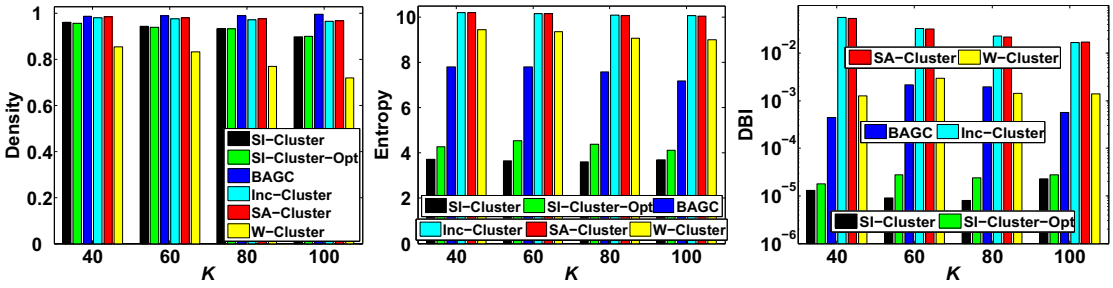
to $O(N_{SG}^3)$ or $O(N_{AG_i}^3 + 2(N_{SG} + N_{AG_i})^3)$ for directed non-propagating heat kernel at the worst case. To sum up, the total cost of SI-Cluster-Opt is approximately equal to $O(N_{SG}^3 + \sum_{i=1}^{N}(N_{AG_i}^3 + 2(N_{SG} + N_{AG_i})^3))$ at the worst case.

(2) SA-Cluster and Inc-Cluster. The total cost of SA-Cluster is equal to $t(T_{randomwalk\_similarity} + T_{vertex\_assignment} + T_{centroid\_update} + T_{weight\_update})$. The cost of Inc-Cluster is equal to $T_{randomwalk\_similarity} + (t-1)T_{incremental\_similarity} + t(T_{vertex\_assignment} + T_{centroid\_update} + T_{weight\_update})$. Similarly, due to $T_{vertex\_assignment}, T_{centroid\_update}, T_{weight\_update} \ll T_{randomwalk\_similarity}$ or $T_{incremental\_similarity}$, we mainly focus on $T_{randomwalk\_similarity}$ and $T_{incremental\_similarity}$. $T_{randomwalk\_similarity}$ is bounded by $O((l-1)(N_{SG} + \sum_{i=1}^{N} N_{AG_i})^3 + (2l-1)(N_{SG} + \sum_{i=1}^{N} N_{AG_i})^2)$ and $T_{incremental\_similarity}$ is equal to $T_{randomwalk\_similarity}$ at the worst case. We have observed that $T_{incremental\_similarity}$ is about 1/3 - 2/3 of $T_{randomwalk\_similarity}$ in our experiments. Thus, by only keeping the terms with the highest order, the total cost of SA-Cluster is approximately bounded by $O(t(l-1)(N_{SG} + \sum_{i=1}^{N} N_{AG_i})^3)$ and the total cost of Inc-Cluster is approximately equal to $O(t(l-1)(N_{SG} + \sum_{i=1}^{N} N_{AG_i})^3)$ at the worst case. However, the cost of SI-Cluster is totally independent of the value of t. In addition, even if we overlook t, we have noticed that $(l-1)(N_{SG} + \sum_{i=1}^{N} N_{AG_i})^3 \gg (l-1)N_{SG}^3 + \sum_{i=1}^{N}(l-1)(N_{SG} + N_{AG_i})^3$ when each activity graph contains a lot of activities, or there are a mass of values for each vertex attribute.

(3) BAGC. Assuming that $T_{\widetilde{\xi}}$, $T_{\widetilde{\gamma}}$, $T_{\widetilde{\mu}}$, $T_{\widetilde{\nu}}$ and $T_{\widetilde{\beta}}$ represents the time complexity of computing the mentioned parameters $\widetilde{\xi}$, $\widetilde{\gamma}$, $\widetilde{\mu}$, $\widetilde{\nu}$ and $\widetilde{\beta}$ respectively in Algorithm 2 in the BAGC paper [Xu et al. 2012], the cost of BAGC can be expressed as $t(T_{\widetilde{\xi}} + T_{\widetilde{\gamma}} + T_{\widetilde{\mu}} + T_{\widetilde{\nu}} + T_{\widetilde{\beta}})$. $T_{\widetilde{\xi}}$, $T_{\widetilde{\gamma}}$, $T_{\widetilde{\mu}}$, $T_{\widetilde{\nu}}$ and $T_{\widetilde{\beta}}$ are equal to $O(KN_{SG})$, $O(KN_{SG}\sum_{i=1}^{N} N_{AG_i})$, $O(K^2N_{SG}^2)$, $O(K^2N_{SG}^2)$, and $O(KN_{SG}\sum_{i=1}^{N} N_{AG_i} + K^2N_{SG}^2)$ respectively. Thus, the total cost of BAGC is equal to $O(t(2KN_{SG}\sum_{i=1}^{N} N_{AG_i} + 3K^2N_{SG}^2 + KN_{SG})) \approx O(t(2KN_{SG}\sum_{i=1}^{N} N_{AG_i} + 3K^2N_{SG}^2))$. Notice that l (the highest order of Taylor series in SI-Cluster) is comparable to t (the number of clustering iterations in BAGC). If we ignore l and t, then the cost of BAGC is approximately bounded by $O(2KN_{SG}\sum_{i=1}^{N} N_{AG_i} + 3K^2N_{SG}^2)$ and the cost of SI-Cluster is approximately equal to $O(N_{SG}^3 + \sum_{i=1}^{N}(N_{SG} + N_{AG_i})^3)$. When K, N and N_{AG_i} are very large, $O(2KN_{SG}\sum_{i=1}^{N} N_{AG_i} + 3K^2N_{SG}^2) \gg O(N_{SG}^3 + \sum_{i=1}^{N}(N_{SG} + N_{AG_i})^3)$. For the DBLP 964,166 dataset containing 6,992 conferences and 363,352 keywords, $K^2N_{SG}^2 = 10,000^2 \times 964,166^2 \gg N_{SG}^3 = 964,166^3$ or $K^2N_{SG}^2 = 10,000^2 \times 964,166^2 \gg \sum_{i=1}^{N}(N_{AG_i} + N_{SG})^3 = (6,992 + 964,166)^3 + (363,352 + 964,166)^3$ when $K = 10,000$. When K is relatively small, say $K < \sqrt{N_{SG}}$, the cost of BAGC is comparable to the cost of SI-Cluster. BAGC works very well when K, N and N_{AG_i} are very small.

7.4. Cluster Quality Evaluation

Figure 5 (a) shows the density comparison on Amazon $20,000$ Products by varying the number of clusters $K = 40, 60, 80, 100$. The density values achieved by SI-Cluster and SI-Cluster-Opt are very similar at different K values. This is because the Amazon product co-purchasing network is an undirected graph and the non-propagating heat kernel H on social graph is thus a symmetric matrix. We can directly make use of diagonal transformation to compute the accurate propagating heat kernel $e^{\alpha t\mathbf{H}}$ through Eq.(39). Therefore, both algorithms can achieve the same self-influence similarity matrix such that the similar density values are obtained by them at each of different

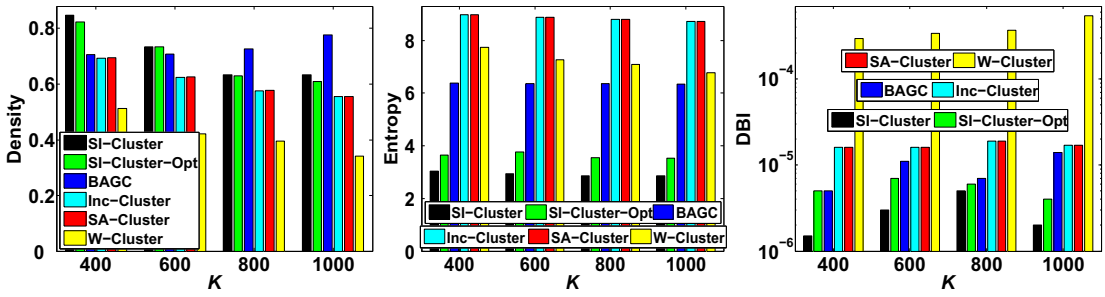


(a) density (b) entropy (c) DBI

Fig. 5. Cluster Quality on Amazon 20,000 Products

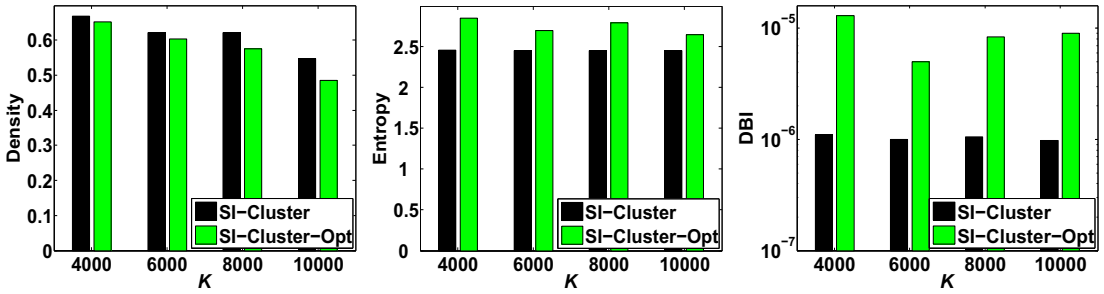
K values. The density values by SI-Cluster, SI-Cluster-Opt, BAGC, Inc-Cluster and SA-Cluster remains 0.89 or higher even when K is increasing. This demonstrates that these methods can find densely connected components. The density values of W-Cluster is relatively lower, in the range of 0.72-0.85 with increasing K, showing that the generated clusters have a very loose intra-cluster structure. Figure 5 (b) shows the entropy comparison on Amazon $20,000$ Products with $K = 40, 60, 80, 100$. SI-Cluster has the lowest entropy, while SI-Cluster-Opt achieves slightly higher entropy values. Although the Amazon product co-purchasing network is an undirected graph, recall Eq.(13), the non-propagating heat kernel $\mathrm{H_i}$ on influence graph is often an asymmetric matrix. We need to utilize diagonalizable-matrix approximation to generate an approximate propagating heat kernel $e^{\alpha t \mathrm{H_i}}$ through Eq.(50). Thus, the generated co-influence similarity matrix by SI-Cluster-Opt is also approximate. Other four algorithms have a much higher entropy than SI-Cluster and SI-Cluster-Opt, since our social influence based graph clustering framework 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. Figure 5 (c) shows the DBI comparison on Amazon $20,000$ Products with different K values. SI-Cluster and SI-Cluster-Opt have the lowest DBI of around $0.000008 - 0.000028$, while other methods have a much higher DBI than them. This demonstrates that SI-Cluster and SI-Cluster-Opt can achieve both high intra-cluster similarity and low inter-cluster similarity. This is because they integrate self-influence similarity as well as co-influence similarity with the optimal weight assignment by parameter-based optimization. They both fully utilize the connections between activities and the interactions between members and activities such that the generated clusters not only own similar collaborative patterns but also have similar interaction patterns with activities.

Figures 6 (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 5 (a), (b) and (c) respectively. SI-Cluster and SI-Cluster-Opt obtain almost the same density values, and the similar entropy values with different K values. As shown in the figures, SI-Cluster and SI-Cluster-Opt achieve high density values (> 0.61), which are slightly lower than that of BAGC since the probabilistic clustering method partitions vertices into each possible cluster such that the density value by it often increases with K. SI-Cluster and SI-Cluster-Opt achieve a very low entropy around 2.86-3.77, which is obviously better than the other methods (> 6.35). As K increases, the entropy of SI-Cluster and SI-Cluster-Opt remains stable, while the density by them decreases. In addition, SI-Cluster and SI-Cluster-Opt achieve the lowest D-



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

Fig. 6. Cluster Quality on DBLP 100,000 Authors



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

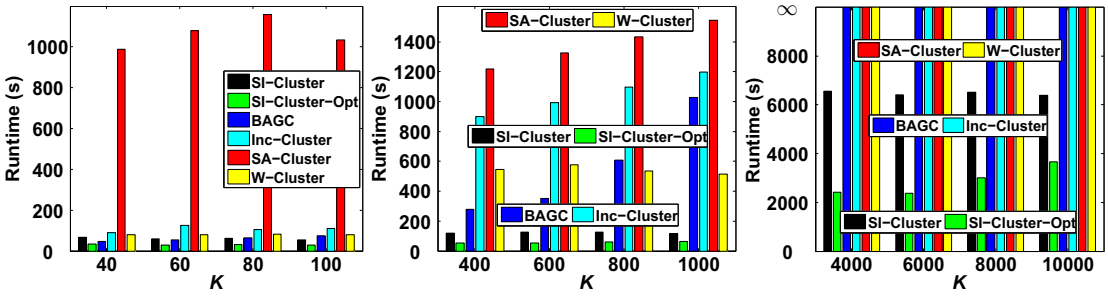
Fig. 7. Cluster Quality on DBLP 964,166 Authors

BI (< 0.000007) among different methods, while the DBI values by other methods are obviously larger than > 0.000005.

Figures 7 (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 and SI-Cluster-Opt do not work on this large dataset due to the "out of memory" problem with our $8G$ main memory machine. Thus, we do not plot these four methods in Figures 7 (a), (b) and (c). On the other hand, SI-Cluster and SI-Cluster-Opt still show good performance with varying K. They achieve similar high density values (> 0.49), much lower entropy of about 2.45-2.85, and very low DBI (≈ 0) for different K.

7.5. Clustering Efficiency Evaluation

Figures 8 (a), (b) and (c) show the clustering time on Amazon 20,000 Products, DBLP $100,000$ and $964,166$ Authors respectively. SI-Cluster-Opt outperforms all other algorithms in all experiments. When facing with an extremely large dataset, such as DBLP $964,166$ Authors, other algorithms can not work due to the "out of memory" problem, while SI-Cluster-Opt and SI-Cluster scale well with large graphs and show good performance with varying K. Thus, for BAGC, Inc-Cluster, SA-Cluster and W-Cluster we plot their running time with ∞ in Figure 8 (c). We make the following observations on the runtime costs of different methods. First, SI-Cluster-Opt significantly reduces the runtime cost of SI-Cluster to around 1/3-1/2, while achieving almost the same clustering quality. This result shows that, with the speed-up techniques of similarity transformation and diagonalizable-matrix approximation, SI-Cluster-Opt only leads to a relatively small overhead compared with SI-Cluster. Second, SA-Cluster is obviously worst than other methods since it needs to perform the repeating random walk dis-



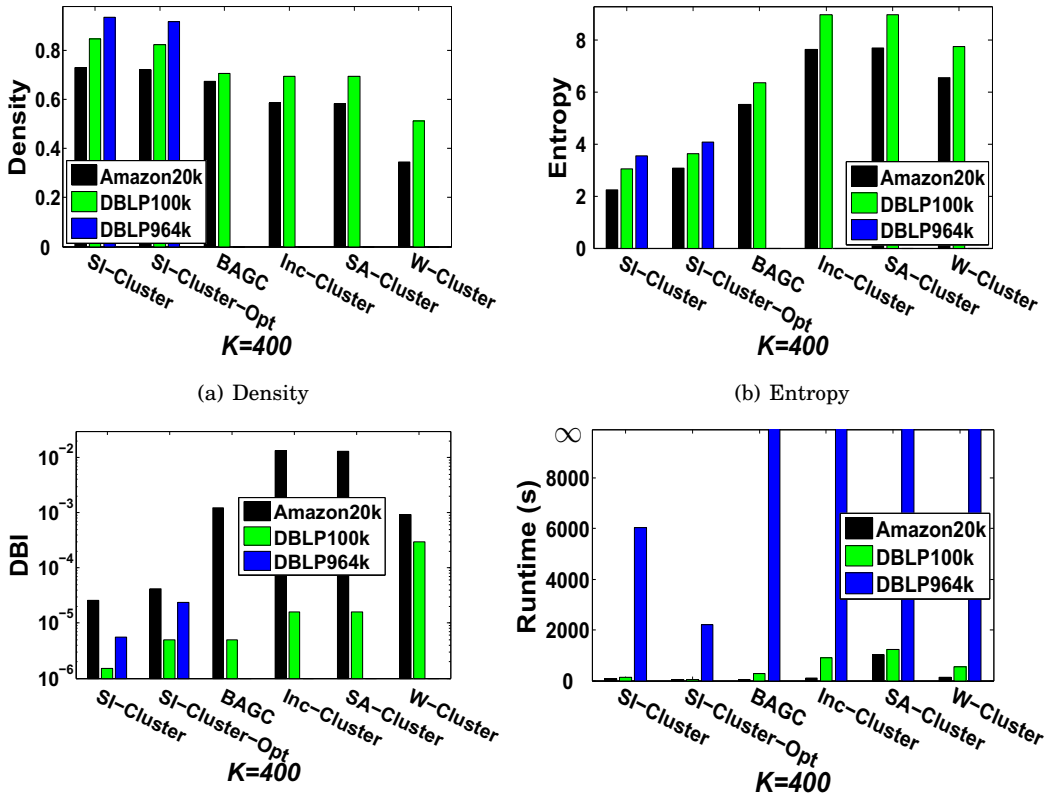
(a) Amazon 20,000 (b) DBLP 100,000 (c) DBLP 964,166

Fig. 8. Clustering Efficiency

tance calculation during each iteration of the clustering process and the distance computation takes more than 80% of the total clustering time. Third, Inc-Cluster, an optimized version of SA-Cluster, is much slower than SI-Cluster-Opt, SI-Cluster, BAGC and W-Cluster since it still needs to incrementally calculate the random walk distance. Fourth, although W-Cluster compute the random walk distance only once, it still runs on a large scale matrix. Fifth, the performance by BAGC is better than other approaches except SI-Cluster-Opt and 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-Opt and SI-Cluster reorganize a large scale heterogeneous network into multiple small scale subgraphs without loss of information. They reduce the computational cost by partitioning activities into clusters with the topological information of the activity graph. Furthermore, SI-Cluster-Opt and SI-Cluster calculate influence-based similarity matrices only once. According to Theorems 5.8-5.11, solving $F(\beta)$ for a given β is a polynomial programming problem which can be sped up by existing fast polynomial programming model, such as gradient algorithms or sequential approximation algorithms in Chapter 13 in [Hillier and Lieberman 1995].

We address the memory consumption of SI-Cluster or SI-Cluster-Opt in two steps. First, we use an in-memory compression technique with fast compression and decompression speed for handling large-scale graphs. For the DBLP 964,166 dataset containing 6,992 conferences, 363,352 keywords, and 31, 962,786 heterogeneous links, the original adjacency list storage (a txt file which is from the original dblp.xml after removing irrelevant information) is 471.7MB. With our in-memory compression, it is only 107.7MB and reduces to less than a quarter of the original size. In addition, it takes only 1.736 seconds for read and 4.237 seconds for write on a regular hard disk when we sequentially access this entire DBLP dataset with in-memory compression turned on. This in-memory compression technique is very useful to address large-scale graphs. Notice that we calculate the influence-based similarity matrices $\mathrm{W}_0, \mathrm{W}_1, \cdots, \mathrm{W}_\mathrm{N}$ in Eq.(24) only once. If there is not enough working memory, we will write the compressed influence-based similarity matrices to disk. Similarly, we use the same strategy to address the P matrix in SI-Cluster-Opt. We found that with compression, the I/O cost is trivial compared to the computational cost. For example, the running time by SI-Cluster and SI-Cluster-Opt on the DBLP 964,166 dataset ranges between around 2400-6500 seconds while it takes only 1.736 seconds to sequentially read this compressed DBLP dataset from disk to memory.

Figures 9 (a), (b) and (c) compare the clustering performance among different methods on different datasets for a fixed $K = 400$. The density value or the entropy value



(a) Density (b) Entropy

(c) DBI (d) Runtime

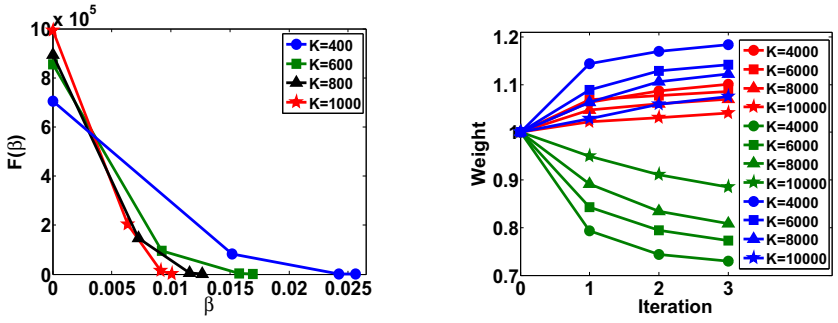
Fig. 9. Clustering Comparison on Different Datasets

by each of different methods increases with the dataset scalability since the bigger the dataset with the same K is, the more vertices are included in each cluster. Thus, the clusters from the bigger dataset are obviously denser. On the other hand, vertices within each cluster on the bigger dataset may have more different attribute values or more co-influences on different activities such that the generated clusters have lower homogeneity. To avoid to achieve the extreme clustering results, the choice of the cluster number K should be proportional to the dataset scalability and should be in a suitable range. According to DBI values, $K = 400$ fits well with DBLP $100,000$ Authors. In addition, the runtime by each of different methods increases quickly with the dataset scalability since the matrix multiplication is the dominant factor in the clustering process and it has the time complexity of $o((N_{SG})^3)$.

7.6. Clustering Convergence

Figure 10 (a) exhibits 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 5.8-5.11.

Figure 10 (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



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

Fig. 10. Clustering Convergence on DBLP 964,166 Authors

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.

7.7. Case Study

In this experiment, we examine some details of our social influence clustering results on DBLP $964,166$ Authors when we set $k = 100$ for both conferences and keywords. Given the predefined keyword similarity and the precalculated keyword partitions, Table II exhibits influence scores of authors based on the social influence propagation between a part of DBLP authors and partitions of all DBLP keywords. We only present most prolific DBLP experts ($\#publications > 190$) who have publications in the area of database or data mining for ease of presentation. The influence scores in Table II are normalized by different authors for each keyword partition. When social influence propagation converges, each column specifies the influence distribution of different authors in the same keyword category. This influence distribution is considered as a local ranking result. For example, *Christos Faloutsos* achieves a higher ranking score than *Hector Garcia-Molina* in the keyword category *DM* since *Christos Faloutsos* has more publications with respect to DM topics and more influence interactions with DM keywords.

Table III presents influence scores of authors normalized by different partitions of all DBLP conferences for each author. Each row represents the social influence distribution in each conference category when the social influence propagation achieves a equilibrium after enough time. We can look upon this social influence distribution as a multi-label classification result with conference clusters as the training dataset. For example, *Jeffrey D. Ullman* is assigned into the cluster *DB* with the probability of 0.8369 and partitioned into *DM* with the probability of 0.1631.

Table IV shows influence scores of authors normalized by different partitions of selected top conferences for each author. We choose three top conferences from four research areas of database (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI), respectively. The detailed conference list is, DB: SIGMOD, VLDB, ICDE; DM: KDD, ICDM, SDM; IR: SIGIR, CIKM, ECIR; AI: IJCAI, AAAI, ECAI. Table III actually presents an unbalanced social influence propagation result since the propagation process is based on the complete DBLP conference list. We know academic research in the area of database has a longer history than data mining research and

Table II. Influence Scores of Authors Normalized by Different Authors Based on Partitions of All Keywords from DBLP

Author	Cluster 1 (DB)	Cluster 2 (DM)
Elisa Bertino	0.0568	0.0249
Michael J. Carey	0.0515	0.0023
Christos Faloutsos	0.0465	0.0746
Jiawei Han	0.0585	0.0960
Theo Härder	0.0386	0.0136
Won Kim	0.0405	0.0176
Vipin Kumar	0.0146	0.0545
Tao Li	0.0116	0.0604
Bing Liu	0.0153	0.0511
Ling Liu	0.0386	0.0249
David Maier	0.0474	0.0079
Hector Garcia-Molina	0.0603	0.0047
S. Muthukrishnan	0.0323	0.0136
M. Tamer Özsu	0.0408	0.0111
Jian Pei	0.0386	0.0653
Raghu Ramakrishnan	0.0509	0.0343
Elke A. Rundensteiner	0.0561	0.0236
Dan Suciu	0.0496	0.0111
Jeffrey D. Ullman	0.0391	0.0136
Ke Wang	0.0305	0.0572
Wei Wang	0.0192	0.0314
Xindong Wu	0.0192	0.0561
Qiang Yang	0.0201	0.0527
Philip S. Yu	0.0606	0.0991
Jeffrey Xu Yu	0.0478	0.0567
Chengqi Zhang	0.0159	0.0431

Table III. Influence Scores of Authors Normalized by Different Conference Partitions Based on Partitions of All Conferences from DBLP

Author	Cluster 1 (DB)	Cluster 2 (DM)
Elisa Bertino	0.8034	0.1966
Michael J. Carey	0.9877	0.0123
Christos Faloutsos	0.5273	0.4727
Jiawei Han	0.5215	0.4785
Theo Härder	0.8352	0.1648
Won Kim	0.8048	0.1952
Vipin Kumar	0.3237	0.6763
Tao Li	0.2562	0.7438
Bing Liu	0.3237	0.6763
Ling Liu	0.7351	0.2649
David Maier	0.9150	0.0850
Hector Garcia-Molina	0.9234	0.0766
S. Muthukrishnan	0.8093	0.1907
M. Tamer Özsu	0.8677	0.1323
Jian Pei	0.5137	0.4863
Raghu Ramakrishnan	0.7265	0.2735
Elke A. Rundensteiner	0.8097	0.1903
Dan Suciu	0.8885	0.1115
Jeffrey D. Ullman	0.8369	0.1631
Ke Wang	0.4876	0.5124
Wei Wang	0.5215	0.4785
Xindong Wu	0.3790	0.6210
Qiang Yang	0.3956	0.6044
Philip S. Yu	0.5223	0.4777
Jeffrey Xu Yu	0.6010	0.3990
Chengqi Zhang	0.3970	0.6030

Table IV. Influence Scores of Authors Normalized by Different Conference Partitions Based on Partitions of Selected Top Conferences from DBLP

Author	AI Cluster	DB Cluster	DM Cluster	IR Cluster
Elisa Bertino	0.0047	0.7135	0.0055	0.2763
Michael J. Carey	0.0015	0.9820	0.0133	0.0032
Christos Faloutsos	0.0012	0.4267	0.3950	0.1771
Jiawei Han	0.0883	0.3724	0.3766	0.1628
Theo Härder	0.0024	0.6102	0.0044	0.3830
Won Kim	0.0133	0.8155	0.1667	0.0045
Vipin Kumar	0.2511	0.1342	0.5198	0.0949
Tao Li	0.1781	0.0673	0.3983	0.3563
Bing Liu	0.2648	0.1001	0.4004	0.2347
Ling Liu	0.0842	0.4125	0.1883	0.3150
David Maier	0.1570	0.8290	0.0117	0.0023
Hector Garcia-Molina	0.0031	0.8217	0.0075	0.1677
S. Muthukrishnan	0.0011	0.8456	0.1446	0.0087
M. Tamer Özsu	0.0017	0.5506	0.1080	0.3397
Jian Pei	0.0876	0.3768	0.3717	0.1639
Raghu Ramakrishnan	0.0756	0.6635	0.1852	0.0756
Elke A. Rundensteiner	0.0747	0.5790	0.0056	0.3407
Dan Suciu	0.1205	0.7009	0.0034	0.1752
Jeffrey D. Ullman	0.0122	0.7183	0.1409	0.1387
Ke Wang	0.0749	0.2901	0.4368	0.1982
Wei Wang	0.0024	0.5036	0.2908	0.2032
Xindong Wu	0.3001	0.1415	0.4584	0.1000
Qiang Yang	0.3741	0.0807	0.2965	0.2487
Philip S. Yu	0.0972	0.3504	0.3763	0.1761
Jeffrey Xu Yu	0.0012	0.4860	0.2881	0.2248
Chengqi Zhang	0.3208	0.2198	0.4539	0.0067

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. For instance, *Jiawei Han* is assigned into the cluster *DM* with the probability of 0.3766 and partitioned into *DB* with the probability of 0.3724. Notice both *Jiawei Han* and *Philip S. Yu* have higher influence scores on the conference category *DB* shown in Table III. However, they both achieve higher influence scores on the conference cluster *DM* in Table IV when we adopt a more balanced strategy to execute the social influence propagation.

Table V exhibits the top-4 cluster list for each author on DBLP $100,000$ Authors by four graph clustering algorithms. According to [Chakraborty et al. 2013], we set $K = 24$, i.e, partition authors into 24 computer science fields: AI, AIGO, ARC, BIO, CV, DB, DIST, DM, EDU, GRP, HCI, IR, ML, MUL, NLP, NW, OS, PL, RT, SC, SE, SEC, SIM, WWW. Thereinto, the computer science fields of AI, ALGO, DB, DIST, DM, IR, ML, NW, OS, PL, SEC and WWW in Table V represent Artificial Intelligence, Algorithms and Theory, Databases, Distributed and Parallel Computing, Data Mining, Information Retrieval, Machine Learning and Pattern Recognition, Networking, Operating Systems, Programming Languages, Security and Privacy, and World Wide Web, respectively. Since Inc-Cluster, a optimized version of SA-Cluster, generates the same clustering results and differs only in time complexity, we put SA-Cluster and Inc-Cluster into the same rows. In addition, SI-Cluster, BAGC, SA-Cluster and Inc-Cluster are all partitioning-based clustering methods, i.e., each author is partitioned into one and only one cluster. However, we can modify the above clustering algorithms to generate a ranking list of the possible clusters for each vertex. For SI-Cluster, SA-Cluster and Inc-Cluster, we sort the closest centroids for each vertex in the descending order of their similarity scores and generate the ranking list of clusters. For BAGC, we sort the variational parameter $\widetilde{\beta}_{ik}$ (cluster membership) for each vertex in

Table V. Top-4 Clusters for Each Author by Different Graph Clustering Algorithms

Cluster Rank		1	2	3	4
Elisa Bertino	SI-Cluster	DB	SEC	DIST	WWW
	BAGC	DB	SEC	WWW	IR
	SA-Cluster/Inc-Cluster	DB	SEC	WWW	IR
Jiawei Han	SI-Cluster	DM	DB	IR	AI
	BAGC	DB	DM	IR	AI
	SA-Cluster/Inc-Cluster	DB	DM	IR	AI
Tao Li	SI-Cluster	IR	DM	NW	AI
	BAGC	DM	IR	NW	AI
	SA-Cluster/Inc-Cluster	IR	DM	AI	NW
Bing Liu	SI-Cluster	DM	IR	AI	ML
	BAGC	DM	AI	IR	ML
	SA-Cluster/Inc-Cluster	DM	IR	AI	DB
Hector Garcia-Molina	SI-Cluster	DB	DIST	WWW	IR
	BAGC	DB	IR	DIST	WWW
	SA-Cluster/Inc-Cluster	DB	IR	WWW	DIST
M. Tamer Özsu	SI-Cluster	DB	DIST	DM	IR
	BAGC	DB	OS	DIST	IR
	SA-Cluster/Inc-Cluster	DB	DIST	OS	DM
Dan Suciu	SI-Cluster	DB	ALGO	PL	SEC
	BAGC	DB	ALGO	SEC	IR
	SA-Cluster/Inc-Cluster	DB	ALGO	SEC	DM
Jeffrey D. Ullman	SI-Cluster	DB	ALGO	PL	DM
	BAGC	ALGO	DB	PL	DM
	SA-Cluster/Inc-Cluster	ALGO	DB	PL	OS
Qiang Yang	SI-Cluster	AI	DM	IR	ML
	BAGC	AI	DM	IR	ML
	SA-Cluster/Inc-Cluster	AI	DM	ML	IR
Philip S. Yu	SI-Cluster	DM	DB	DIST	SEC
	BAGC	DM	DB	IR	NW
	SA-Cluster/Inc-Cluster	DB	DM	IR	NW

the descending order of their values and generate the ranking list of clusters. The clustering results by SI-Cluster are consistent with the main research topics of each author and the corresponding orders of research topics. However, other three clustering approaches can not always produce the correct results. This is because SI-Cluster treats conference and keyword as author's dynamic and inter-connected activities and pre-partition conferences and keywords into clusters. On the other hand, other three clustering approaches treat conference and keyword as author's static attribute such that they often can not generate a good similarity measure and produce a good clustering result. For example, suppose that there are only two authors: A with 5 papers on "KDD" and B with 2 publications on "ICDM", other three clustering approaches can not discover the relationship between two authors based on the static attributes, since there does not exist a path between A and B through conference, whether "KDD" or "ICDM". However, in our SI-Cluster framework, we first partition partition "KDD" and "ICDM" into cluster *DM*, shrink multiple data mining conference vertices into a supernode (conference cluster *DM*), and summarize the links between authors and conference vertices as the links between authors and conference supernode. Thus, there exists a path between A and B through this conference supernode and we will generate a positive co-influence similarity score between A and B. In Table V, *Jiawei Han* has more publications on database conferences than data mining conferences. However, by examining the keyword similarity network, most of title keywords in his papers are DM specific. Most importantly, SI-Cluster derives a lot of co-influence similarity scores between *Jiawei Han* and other DM researchers through keyword cluster supernode. However, in other three clustering approaches, *Jiawei Han* are reachable to other DM researchers through individual keyword vertices. As discussed above, we may not gen-

erate the correct relationship between *Jiawei Han* and other experts. Similar example is *Jeffrey D. Ullman*, who has more publications on theory conferences and journals than database conferences and journals. However, most of title keywords in his papers are DB specific.

8. CONCLUSIONS

We have presented an efficient approach for social influence based clustering over heterogeneous information networks with four unique features. First, we define the concept of influence-based similarity in terms of propagating heat diffusion kernel on both social collaboration graph and its associated multiple influence graphs to capture multiple types of social influences through both direct and indirect social relationships. Second, we introduce a weighted unified influence-based similarity matrix that utilizes a weight function with an iterative weight refinement method to integrate self-influence and co-influence similarities. Third, we design a dynamic learning algorithm for social influence based graph clustering, which dynamically refines the K clusters by continuously quantifying and adjusting the weighted contributions from different information networks until reaching convergence. Finally, we develop the SI-Cluster-Opt algorithm to further speed up the performance of the SI-CLUSTER baseline algorithm. We transform a sophisticated nonlinear fractional programming problem with respect to multiple weights into a straightforward nonlinear parametric programming problem of single variable. We develop two optimization techniques on the computation of both self-influence similarity and co-influence similarities to make the influence-based similarity computation more efficient. Extensive experimental evaluation on three real graphs demonstrates that our social influence based graph clustering approach not only achieves a very good balance between self-influence and co-influence similarities but also scales extremely well for clustering large graphs in terms of time complexity.

REFERENCES

http://www.emarketer.com/.

Lada Adamic and Eytan Adar. 2003. Friends and Neighbors on the Web. *Social Networks* 25, 3 (2003), 211–230.

Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. 2008. Influence and Correlation in Social Networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'08)*. Las Vegas, NV, 7–15.

George B. Arfken, Hans J. Weber, and Frank E. Harris. 2005. *Mathematical Methods for Physicists* (6th ed.). Academic Press.

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The Advantages of Careful Seeding. In *Proc. 2007 Annual ACM-SIAM Symp. Discrete Algorithms (SODA'07)*. New Orleans, LA, 1027–1035.

Mikhail Belkin and Partha Niyogi. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation* 15, 6 (2003), 1373–1396.

Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. 2011. Node Classification in Social Networks. *Social Network Data Analytics* (2011), 115–148.

Rajendra Bhatia. 1997. *Matrix Analysis*. Springer.

Vincent D. Blondel, Anahi Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. 2004. A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. *SIAM Rev.* 46, 4 (2004), 647–666.

Marco Bodrato. 2010. A Strassen-like Matrix Multiplication Suited for Squaring and Higher Power Computation. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC'10)*. Munich, Germany, 273–280.

Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. 2005. Community Mining from Multi-relational Networks. In *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*. Porto, Portugal, 445–452.

Tanmoy Chakraborty, Sandipan Sikdar, Vihar Tammana, Niloy Ganguly, and Animesh Mukherjee. 2013. Computer Science Fields as Ground-truth Communities: Their Impact, Rise and Fall. In *Proceedings of*

the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'13). Niagara Falls, Canada, 426–433.

Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'10)*. Washington, DC, 1029–1038.

Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Christopher Umans. 2005. Group-theoretic algorithms for matrix multiplication. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. Pittsburgh, Pennsylvania, USA.

D. Coppersmith and S. Winograd. 1990. Matrix Multiplication via Arithmetic Progressions. *Journal of Symbolic Computation* 9 (1990), 251–280.

Lee Raymond Dice. 1945. Measures of the Amount of Ecologic Association Between Species. *Ecology* 26, 3 (1945), 297–302.

P. Domingos and M. Richardson. 2001. Mining the network value of customers. In *Proc. 2001 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'01)*. San Francisco, CA, 57–66.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (3rd ed.). Johns Hopkins University Press.

Frederick S. Hillier and Gerald J. Lieberman. 1995. *Introduction to Operations Research (IBM)*. Mcgraw-Hill College.

A. Hinneburg and D. A. Keim. 1998. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*. New York, NY, 58–65.

Roger A. Horn and Charles R. Johnson. 1990. *Matrix Analysis*. Cambridge University Press.

Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *KDD*. ACM, New York, NY, USA, 538–543. DOI:http://dx.doi.org/10.1145/775047.775126

Ming Ji, Jiawei Han, and Marina Danilevsky. 2011. Ranking-based classification of heterogeneous information networks. In *Proc. 2011 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'11)*. San Diego, CA, 1298–1306.

L. Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18 (March 1953), 39–43.

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

D. Kempe, J. Kleinberg, and E. Tardos. 2003. Maximizing the spread of influence through a social network. In *Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*. Washington, DC, 137–146.

Edward Casey Kenley and Young-Rae Cho. 2011. Entropy-Based Graph Clustering: Application to Biological and Social Networks. In *Proceedings of the 11th International Conference on Data Mining (ICDM'11)*. Vancouver, Canada, 1116–1121.

J. M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46 (1999), 604–632.

Risi Imre Kondor and John D. Lafferty. 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Proc. 2002 Int. Conf. Machine Learning (ICML'02)*. Sydney, Australia, 315–322.

John Lafferty and Guy Lebanon. 2005. Diffusion Kernels on Statistical Manifolds. *Journal of Machine Learning Research* 6 (2005), 129–163.

E. Scott Larsen and David McAllister. 2001. Fast Matrix Multiplies using Graphics Hardware. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (SC'01)*. Denver.

E. A. Leicht, Petter Holme, and M. E. J. Newman. 2006. Vertex Similarity in Networks. *Physical Review E* 73, 2 (2006).

Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. Mining Social Networks Using Heat Diffusion Processes for Marketing Candidates Selection. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. Napa Valley, CA, 233–242.

Kathy Macropol and Ambuj Singh. 2010. Scalable Discovery of Best Clusters on Large Graphs. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB'10) / PVLDB 3(1)*. Singapore, 693–702.

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

Seth A. Myers, Chenguang Zhu, and Jure Leskovec. 2012. Information Diffusion and External Influence in Networks. In *Proc. 2012 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'12)*. Beijing, China, 33–41.

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

R. T. Rockafellar. 1997. *Convex Analysis*. Princeton University Press.

Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. 2010. Suggesting Friends Using the Implicit Social Graph. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*. Washington, DC, 233–242.

G. Salton. 1989. *Automatic Text Processing*. Addison-Wesley.

V. Satuluri and S. Parthasarathy. 2009. Scalable Graph Clustering Using Stochastic Flows: Applications to Community Discovery. In *Proc. 2009 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'09)*. Paris, France.

J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, no. 8. 888–905.

Motoki Shiga, Ichigaku Takigawa, and Hiroshi Mamitsuka. 2007. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *Proc. 2007 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'07)*. San Jose, CA, 647–656.

Gilbert Strang. 2005. *Linear Algebra and Its Applications, 4th edition*. Brooks Cole.

Volker Strassen. 1969. Gaussian Elimination is not Optimal. *Numer. Math.* 13 (1969), 354–356.

Yizhou Sun, Charu C. Aggarwal, and Jiawei Han. 2012. Relation Strength-Aware Clustering of Heterogeneous Information Networks with Incomplete Attributes. *Proceedings of the VLDB Endowment (PVLDB)* 5, 5 (2012), 394–405.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proceedings of the VLDB Endowment (PVLDB)* 4, 11 (2011), 992–1003.

Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. 2009. RankClus: Integrating Clustering with Ranking for Heterogenous Information Network Analysis. In *Proc. 2009 Int. Conf. Extending Database Technology (EDBT'09)*. Saint Petersburg, Russia.

Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S. Yu, and Xiao Yu. 2012. Integrating Meta-Path Selection with User-Guided Object Clustering in Heterogeneous Information Networks. In *Proc. 2012 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'12)*. Beijing, China, 1348–1356.

P. Tan, M. Steinbach, and V. Kumar. 2005. *Introduction to Data Mining*. Addison Wesley.

B. Taskar, E. Segal, and D. Koller. 2001. Probabilistic Classification and Clustering in Relational Data. In *Proc. 2001 Int. Joint Conf. Artificial Intelligence (IJCAI'01)*. Seattle, WA, 870–878.

Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient aggregation for graph summarization. In *Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'08)*. Vancouver, Canada, 567–580.

Lloyd N. Trefethen and David Bau. 1997. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics.

Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. 2007. SCAN: a structural clustering algorithm for networks. In *Proc. 2007 Int. Conf. Knowledge Discovery and Data Mining (KDD'07)*. San Jose, CA, 824–833.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A Model-based Approach to Attributed Graph Clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. Scottsdale, Arizona, 505–516.

Haixuan Yang, Irwin King, and Michael R. Lyu. 2005. NHDC and PHDC: Non-propagating and Propagating Heat Diffusion Classifiers. In *Proc. 12th International Conference on Neural Information Processing, ICONIP2005*. Taipei, Taiwan, 394–399.

Haixuan Yang, Irwin King, and Michael R. Lyu. 2007. DiffusionRank: a Possible Penicillin for Web Spamming. In *Proc. 2007 Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'07)*. Amsterdam, Netherlands, 431–438.

Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009. Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'09)*. Paris, France, 927–936.

Xiao Yu, Yizhou Sun, Peixiang Zhao, and Jiawei Han. 2012. Query-driven Discovery of Semantically Similar Substructures in Heterogeneous Networks. In *Proc. 2012 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'12)*. Beijing, China, 1500–1503.

Ning Zhang, Yuanyuan Tian, and Jignesh M. Patel. 2010. Discovery-Driven Graph Summarization. In *Proceedings of the 26th International Conference on Data Engineering (ICDE'10)*. Long Beach, CA, 880–891.

Peixiang Zhao, Jiawei Han, and Yizhou Sun. 2009. P-Rank: A Comprehensive Structural Similarity Measure over Information Networks. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*. Hong Kong, China, 553–562.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. In *Proc. 2009 Int. Conf. on Very Large Data Base (VLDB'09)*. Lyon, France, 718–729.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2010. Clustering Large Attributed Graphs: An Efficient Incremental Approach. In *Proc. 2010 Int. Conf. on Data Mining (ICDM'10)*. Sydney, Australia, 689–698.

Yang Zhou and Ling Liu. 2013. Social Influence Based Clustering of Heterogeneous Information Networks. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'13)*. Chicago, IL, 338–346.

Yang Zhou and Ling Liu. 2014. Activity-edge Centric Multi-label Classification for Mining Heterogeneous Information Networks. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'14)*. New York, NY, 1276–1285.

Yang Zhou, Ling Liu, Chang-Shing Perng, Anca Sailer, Ignacio Silva-Lepe, and Zhiyuan Su. 2013. Ranking Services by Service Network Structure and Service Attributes. In *Proceedings of the 20th International Conference on Web Service (ICWS'13)*. Santa Clara, CA, 26–33.

Received November 2013; accepted January 2015