

Flow-based Influence Graph Visual Summarization

Lei Shi
SKLCS, Institute of Software
Chinese Academy of Sciences
Beijing 100190, China
shil@ios.ac.cn

Hanghang Tong
Computer Science
City College, CUNY
New York, USA
tong@cs.ccny.cuny.edu

Jie Tang and Chuang Lin
Computer Science
Tsinghua University
Beijing 100084, China
{jietang, chlin}@tsinghua.edu.cn

Abstract—Visually mining a large influence graph is appealing yet challenging. People are amazed by pictures of newscasting graph on Twitter, engaged by hidden citation networks in academics, nevertheless often troubled by the unpleasant readability of the underlying visualization. Existing summarization methods enhance the graph visualization with blocked views, but have adverse effect on the latent influence structure. How can we visually summarize a large graph to maximize influence flows? In particular, how can we illustrate the impact of an individual node through the summarization? Can we maintain the appealing graph metaphor while preserving both the overall influence pattern and fine readability?

To answer these questions, we first formally define the influence graph summarization problem. Second, we propose an end-to-end framework to solve the new problem. Our method can not only highlight the flow-based influence patterns in the visual summarization, but also inherently support rich graph attributes. Last, we present a theoretic analysis and report our experiment results. Both evidences demonstrate that our framework can effectively approximate the proposed influence graph summarization objective while outperforming previous methods in a typical scenario of visually mining academic citation networks.

I. INTRODUCTION

Graphs are prevalent and have become a prevalent platform for the masses to interact and disseminate a variety of information (e.g., influence, memes, opinions, rumors, etc.). *How to make sense of an individual's influence in the context of such graphs?* This, which is referred as Influence Graph Summarization (IGS) problem, is the central problem we aim to address in this paper. For example, how does a highly-cited paper impact the research community to raise several topic threads; and consequentially, how do these topics interact with each other and lead to a new multi-disciplinary research direction? How does a senior researcher contribute to multiple research areas by influencing others?

Although closely related, IGS problem bears some subtle difference from the existing work. We briefly review three most relevant topics. First (*influence maximization*), in the past decades, many elegant algorithms have been proposed for the so-called influence maximization problem [1]. While effective in identifying *who* are most influential in the graph, the question of *what makes them influential* largely remains open. Second (*graph visualization*), many elaborate layout algorithms have been designed and widely applied in recent years. They can draw medium-sized graphs aesthetically and faithfully, but can not avoid the huge visual clutter on large influence graphs. Third (*graph summarization*), many interesting work has been

done in the context of graph clustering and compression. These works typically look for coherent/homogeneous regions in graphs by optimizing a pre-defined loss function (e.g., minimizing the inter-cluster connection, maximizing the intra-cluster density, minimizing the total description cost, etc.). Despite their own success, most, if not all, of the existing work on graph summarization tends to ignore the specific characteristics of influence graphs and how the end user would visually perceive/read/consume the summarization results.

To be specific, we outline the following design objectives that differentiate our IGS problem from existing works.

- *D1. Flow Rate Maximization.* Quite different from extracting dense clusters on graph, the goal of IGS is to highlight the flow of influence not only within but also across clusters. By maximizing the overall flow rate, IGS-based summarization outlines the strongest interaction among groups of nodes on a graph. For example, Figure 1 depicts the influence of the famous power-law paper presented at SIGCOMM'99. The evolution of research topics is revealed, rather than the hot topics themselves.
- *D2. Localized Visualization.* While a large graph can span millions of nodes and prohibit any readable visual summarization, in IGS objective, we switch to summarize the influence of a single node on the graph (called the source node). This localized visualization problem is at least as important as the overall summarization problem. Consider a user navigating the citation graph of computer science papers, after an overview of the entire field, likely she will drill down to a few interested papers and examine their influence separately.
- *D3. Rich Information.* Most influence graphs have rich attributes (e.g., the topic, venue of a scientific paper) and often evolve over time (e.g., the publication date). Incorporating these attributes to enhance the IGS performance poses additional challenges to our work.

In this paper, we propose a unified framework to generate *flow-based, localized visual* summarization over large-scale influence graphs. The framework provides a seamless, end-to-end pipeline to solve the IGS problem by decomposing it into several key building blocks. It is flexible and admits many existing graph mining algorithms for each of its building blocks. Meanwhile, theoretic analysis shows that our method is equivalent to the kernel k-mean clustering with a carefully designed kernel matrix so that the intra-cluster consistency is also preserved. Finally, we conduct extensive empirical evaluations to validate the effectiveness of the framework. The main contributions of the paper can be summarized as:

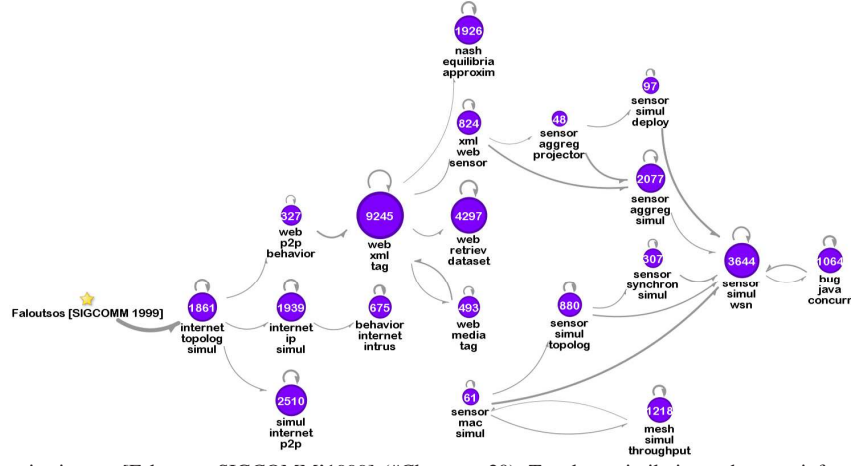


Fig. 1. Influence graph summarization on [Faloutsos SIGCOMM'1999] (#Cluster = 20). Topology similarity and venue information are integrated. Node label gives the cluster size and summary on paper title+abstract normalized by keyword frequency. Link thickness indicates the normalized flow rate.

TABLE I. NOTATIONS.

SYMBOL	DESCRIPTION
I	influence graph as input
f	source node selected by user or algorithm
G	maximal influence graph of f in I
$v_i, N(i), n$	nodes, neighbor set and # of nodes in G
A, a_{ij}	adjacency matrix of G and its entries
M^G, M^D, M^T	similarity, attribute and time matrix of G
S	graph summarization of G
π_c, π_c , k	clusters, cluster size and # of clusters in S
$\xi_s, r(\xi_s), l$	flows, flow rate and # of flows in S
$\pi_{c(s)}, \pi_{d(s)}$	the source and target cluster of flow ξ_s

- *Problem Definition*, to fulfill the design objectives listed above for flow-based visual summarization of large influence graphs (Section II);
- *A Unified Framework and Implementation Details*, to solve the IGS problem (Section III and Section V);
- *Theoretic Analysis*, to reveal the intrinsic relationship between IGS problem and the existing work (Section IV);
- *Comprehensive Evaluation*, to demonstrate the effectiveness and efficiency of the proposed framework (Section VI).

II. PROBLEM DEFINITION

Table I lists the notations used throughout the paper. The raw inputs are the influence graph I and the source node f either selected by the user or detected by any existing influence maximization algorithm. Without loss of generality, it is enough to consider a maximal influence graph G of f which is an induced subgraph of I containing all the nodes reachable from f in I (including f). Though it is easy to extend the definition to a maximal origin graph by reversing all the links in I or using an union of the two definitions, for relevancy to the IGS problem we stick to the maximal influence graph definition in this paper. Let G have n nodes, denoted as $\{v_i\}_{i=1}^n$. G is represented by the adjacency matrix $A = \{a_{ij}\}_{i,j=1}^n$ in which a_{ij} denotes the link weight. $a_{ij} > 0$ if there is a link from v_i to v_j .

Definition 1: The **graph summarization** of G , denoted as S , is a super node-link graph of G . The node set of S contains k disjoint and exhaustive node clusters of G , denoted as $\{\pi_c\}_{c=1}^k$ where $|\pi_c|$ indicates the number of nodes in cluster π_c . The link set of S contains l flows between the nodes in S (i.e., clusters in G), denoted as $\{\xi_s\}_{s=1}^l$. Each flow ξ_s

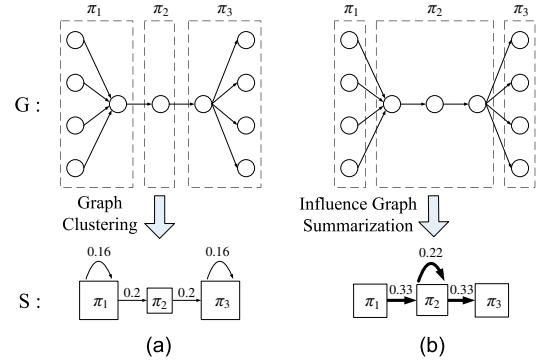


Fig. 2. Difference between IGS problem and traditional graph clustering problem. Each dash box in the original graph G becomes a square node in the summarization graph S . (a) traditional graph clustering leading to more intra-cluster flows; (b) influence graph summarization exposing denser flows. In S , the flow rate is labeled above each link and is mapped to the link thickness visually. We assume a uniform link weight of 1 in the original graph G .

represents the collection of all the links in G from nodes in cluster $\pi_{c(s)}$ to nodes in cluster $\pi_{d(s)}$. The flow rate of ξ_s is defined by

$$r(\xi_s) = \frac{\sum_{v_i \in \pi_{c(s)}, v_j \in \pi_{d(s)}} a_{ij}}{|\pi_{c(s)}| |\pi_{d(s)}|}$$

Note that S can be a partial summarization of G , with fewer flows ($l < k^2$) than a full summarization ($l = k^2$). This is desirable for influence graph visualization where huge number of flows and edge crossings can cause unpleasant visual clutter.

Problem 1: The **general IGS problem** is defined as finding a graph summarization S with k clusters and l top flows of the maximal influence graph G to optimize the objective function:

$$\max \sum_{s=1}^l r(\xi_s) \sqrt{|\pi_{c(s)}| |\pi_{d(s)}|} \quad (1)$$

The general IGS problem defined in (1), although seemingly similar to, is different from the traditional graph clustering problems. Let us explain their difference using the classic ratio association graph clustering problem, whose objective function is shown below.

$$\max \sum_{c=1}^k \sum_{i,j \in \pi_c} \frac{a_{ij}}{|\pi_c|} = \sum_{c=1}^k r(\xi_c) |\pi_c|$$

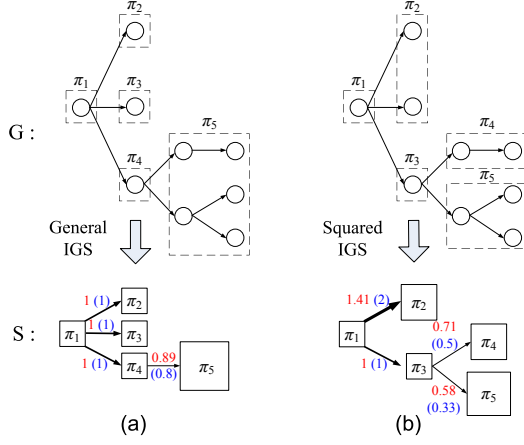


Fig. 3. The sample influence graph leading to fragmented flows in the summarization ($k = 5, l = 4$): (a) By the general IGS objective, the resulting graph has two identically-positioned clusters at 1-hop from the source node (π_2, π_3), the normalized flow rate by (1) is labeled in red, favoring this summarization by a sum of $3.89 > 3.70$; (b) Applying the squared IGS objective, the two identical clusters can be merged and more structure of the influence graph is revealed. The squared flow rate by (2) is labeled in blue parentheses, having a sum of $3.83 > 3.80$. (best viewed in color)

where ξ_c denotes the intra-cluster flow from π_c to itself.

The IGS objective function is designed to maximize the sum of l selected flows between or within clusters, corresponding to l arbitrary blocks in the adjacency matrix. On the other hand, the ratio association objective maximizes the sum of intra-cluster flows at all the k diagonal matrix blocks. In other words, IGS finds dense flows through summarization which fits well the goal to highlight flows of influence across the graph. This is quite different from the traditional graph clustering objective that finds dense node clusters. An example is given in Figure 2 for visual comparison.

Note that both objective functions are normalized by the square root of the size of clusters/blocks in the adjacency matrix. While this is good for classical graph clustering heuristics, applying the same normalization method on IGS can lead to fragmented flows on the summarization. Figure 3 illustrates a case with a small influence graph.

Problem 2: The squared IGS problem improves the definition of flow contributions by their squared and normalized flow rate. The new objective function is written as:

$$\max \sum_{s=1}^l r(\xi_s)^2 |\pi_{c(s)}| |\pi_{d(s)}| \quad (2)$$

From the perspective of highlighting influence flows, the squared IGS objective is consistent with the general IGS. Moreover, by applying the square function to the flow rate, it favors large flows more than the general objective. In this sense, heuristically it is better for our influence graph summarization problem with bounded flow number.

III. FRAMEWORK

In this section, we propose a unified framework to solve the IGS problem, including an end-to-end pipeline, the algorithm to summarize influence structure from graph topology, and the extension to incorporate graph attribute and time information.

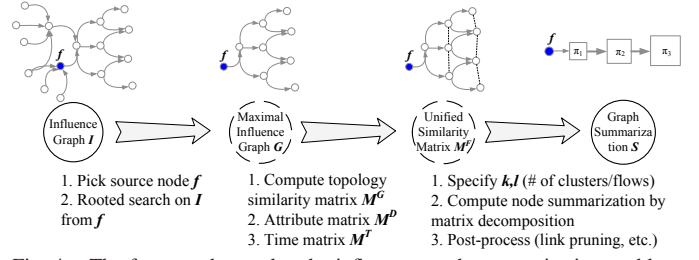


Fig. 4. The framework to solve the influence graph summarization problem.

A. End-to-End Pipeline

We propose an end-to-end pipeline, shown in Figure 4. The framework decomposes the IGS problem into several building blocks. Initially, the maximal influence graph G is computed from the input graph I by a breadth-first or depth-first search starting from the source node f . Over the maximal influence graph G , three processing components work in parallel to generate three matrices on the graph: the topology similarity matrix, and the optional attribute and time matrices. The core of our framework is the decomposition of the topology similarity matrix to generate k node clusters for the summarization. We carefully design the topology similarity matrix to ensure that the graph summarization approximates the flow rate maximization objective (See Section IV for detailed analysis). The attribute and time matrices can be incorporated to augment the similarity matrix before decomposition so as to optimize the summarization towards graph attributes. The requirement of the l flows in the summarization is handled by link pruning using either ranking-based filtering or the maximum spanning tree algorithm. The proposed pipeline is flexible and admits many existing graph mining algorithms for each of its building blocks. On the other hand, by itself, none of these existing algorithms is sufficient to solve the IGS problem.

B. Node Summarization

Node summarization is the key building block of our proposed pipeline. It takes the topology similarity matrix M^G as the input and generates k node clusters. We propose a matrix decomposition based solution. Its rationality as well as the details of the similarity matrix M^G will be discussed in Section IV and V, respectively.

Over the similarity matrix M^G , the decomposition employs a Symmetric version of the Nonnegative Matrix Factorization (SymNMF [2]) which optimizes:

$$\min_{H \geq 0} \|M^G - HH^T\|_F^2 \quad (3)$$

where $\|\cdot\|_F$ denotes the Frobenius norm of the matrix. $H = \{h_{ij}\}$ is a n by k matrix indicating the cluster membership assignment of nodes in G : v_i will be clustered into π_c if h_{ic} is the largest entry in the i th row of H .

C. Generalizations

Our framework can be extended to incorporate node attributes and their time information on graph (e.g., the research field attribute and the publication date of a paper). The extension takes in two separate matrices computed from the influence graph G . The attribute augmentation matrix is constructed to reflect the pairwise similarities among graph nodes in terms of the specific attribute. Consider an attribute

affinity adjacency matrix $A^D = \{a_{ij}^D\}_{i,j=1}^n$, where $a_{ij}^D > 0$ if v_i and v_j have the same value on a selected node attribute. Denote the attribute augmentation matrix as $M^D = \{m_{ij}^D\}_{i,j=1}^n$, M^D is computed from A^D by

$$m_{ij}^D = \begin{cases} \Lambda_{aug}, & i = j \text{ or } a_{ij}^D > 0 \\ 1, & a_{ij}^D = 0 \end{cases}$$

where Λ_{aug} controls the degree of augmentation. We find $\Lambda_{aug} = 2$ is an effective setting in general.

Similarly, we construct the time decaying matrix $M^T = \{m_{ij}^T\}_{i,j=1}^n$ to reflect the similarity among graph nodes on the associated time. For example, the papers in the same year will have a high similarity entry on M^T . Denote the time attribute of graph nodes on G as $\{t_i\}_{i=1}^n$ (unit year), the time decaying matrix M^T is computed by

$$m_{ij}^T = \lambda_{Decay}^{-|t_i - t_j|}$$

where λ_{Decay} controls the rate of similarity decaying over time. Using the median of cited half-life time of sci-indexed CS journals, we compute a default value of $\lambda_{Decay} = 1.11$.

The attribute and time matrices are then used to extend the topology similarity matrix to a unified form.

$$M^F = M^G \odot M^D \odot M^T \quad (4)$$

The final SymNMF objective becomes

$$\min_{H \geq 0} \|M^F - HH^T\|_F^2 \quad (5)$$

IV. EQUIVALENCE ANALYSIS

In this section, we present theoretic analysis, to explain the rationality behind our matrix decomposition based solution. We start with deriving an approximate objective function of the IGS problem. Then we show that such an objective is equivalent to the kernel k-mean clustering by choosing an appropriate kernel matrix. Finally, the kernel k-mean clustering can be solved by SymNMF.

A. Approximation of IGS problem

Consider the objective function in (2), the optimization requires maximizing over two types of variables: $\{\pi_c\}_{c=1}^k$, the node cluster membership assignment; and $\{\xi_s\}_{s=1}^l$, the selected top flows. The simultaneous optimization of these two classes of variables is hard due to the non-linear and combinatorial nature of the problem. Here we consider a two-step approximation that first maximizes the sum of all the flows over the node cluster assignment, then maximizes the sum of the top l flows given the cluster assignment. This is feasible with an appropriate l (e.g. $l = 2k$), because the top l flows contribute the most part of the overall flow rate after applying the square function, as shown in Section VI. Formally, the approximate objective function becomes:

$$\max \sum_{s=1}^{k^2} r(\xi_s)^2 |\pi_{c(s)}| |\pi_{d(s)}| = \sum_{c,d=1}^k \frac{(\sum_{i \in \pi_c, j \in \pi_d} a_{ij})^2}{|\pi_c| |\pi_d|} \quad (6)$$

$$\max \sum_{s=1}^l r(\xi_s)^2 |\pi_{c(s)}| |\pi_{d(s)}| \quad \text{given } \{\pi_c\}_{c=1}^k \quad (7)$$

The second part of the optimization can be solved by selecting l top flows with the largest size-normalized flow rate.

B. Kernel K-Mean Clustering

According to [3], the kernel k-mean clustering (KM) is defined as follows. Given n data vectors $\{x_i\}_{i=1}^n$ with kernel function $\phi(x_i)$, KM method groups the data vectors into k non-overlapping clusters $\{\pi_c\}_{c=1}^k$ based on the objective function

$$\min \sum_{c=1}^k \sum_{i \in \pi_c} \|\phi(x_i) - m_c\|^2 \quad \text{where } m_c = \frac{\sum_{i \in \pi_c} \phi(x_i)}{|\pi_c|}$$

Expand $\|\phi(x_i) - m_c\|^2$ into

$$\phi(x_i) \cdot \phi(x_i) - \frac{2 \sum_{j \in \pi_c} \phi(x_i) \cdot \phi(x_j)}{|\pi_c|} + \frac{\sum_{j, l \in \pi_c} \phi(x_j) \cdot \phi(x_l)}{|\pi_c|^2}$$

Because

$$\sum_{c=1}^k \sum_{i \in \pi_c} \frac{\sum_{j \in \pi_c} \phi(x_i) \cdot \phi(x_j)}{|\pi_c|} = \sum_{c=1}^k \sum_{i \in \pi_c} \frac{\sum_{j, l \in \pi_c} \phi(x_j) \cdot \phi(x_l)}{|\pi_c|^2}$$

The objective function of KM clustering can be written as

$$\min \sum_{c=1}^k \sum_{i \in \pi_c} [\phi(x_i) \cdot \phi(x_i) - \frac{\sum_{j \in \pi_c} \phi(x_i) \cdot \phi(x_j)}{|\pi_c|}]$$

As $\sum_{c=1}^k \sum_{i \in \pi_c} \phi(x_i) \cdot \phi(x_i)$ is constant, it is equivalent to

$$\max \sum_{c=1}^k \sum_{i, j \in \pi_c} \frac{\phi(x_i) \cdot \phi(x_j)}{|\pi_c|} \quad (8)$$

Introduce the heuristic of 1-hop bidirectional common neighbor as the similarity measure (CommonNeighbor), we can compute a topology similarity matrix by

$$K = \frac{AA^T + A^T A}{2} \quad \text{where } k_{ij} = \sum_{t=1}^n \frac{a_{it}a_{jt} + a_{ti}a_{tj}}{2}$$

If we use K as the kernel matrix in KM clustering and substitute k_{ij} for $\phi(x_i) \cdot \phi(x_j)$, (8) becomes

$$\begin{aligned} \max & \sum_{c=1}^k \frac{1}{|\pi_c|} \sum_{i, j \in \pi_c} \sum_{t=1}^n \frac{a_{it}a_{jt} + a_{ti}a_{tj}}{2} \\ = & \sum_{c=1}^k \sum_{t=1}^n \sum_{i, j \in \pi_c} \frac{a_{it}a_{jt} + a_{ti}a_{tj}}{2|\pi_c|} \\ = & \sum_{c=1}^k \sum_{t=1}^n \frac{(\sum_{i \in \pi_c} a_{it})^2 + (\sum_{i \in \pi_c} a_{ti})^2}{2|\pi_c|} \\ = & \sum_{c=1}^k \sum_{j=1}^n \frac{(\sum_{i \in \pi_c} a_{ij})^2 + (\sum_{i \in \pi_c} a_{ji})^2}{2|\pi_c|} \\ = & \sum_{c,d=1}^k \sum_{j \in \pi_d} \frac{(\sum_{i \in \pi_c} a_{ij})^2 + (\sum_{i \in \pi_c} a_{ji})^2}{2|\pi_c|} \end{aligned} \quad (9)$$

C. Equivalence

Let us compare the objective functions in (6) and (9). They are in similar forms if we re-formulate (6) into

$$\sum_{c,d=1}^k \sum_{i \in \pi_c, j \in \pi_d} a_{ij} \left(\frac{\sum_{p \in \pi_c, q \in \pi_d} a_{pq}}{|\pi_c| |\pi_d|} \right) = \sum_{i,j=1}^n a_{ij} w_{ij}^{IGS}$$

$$\text{where } w_{ij}^{IGS} = \frac{\sum_{p \in \pi_c, q \in \pi_d} a_{pq}}{|\pi_c| |\pi_d|} \quad (i \in \pi_c, j \in \pi_d) \quad (10)$$

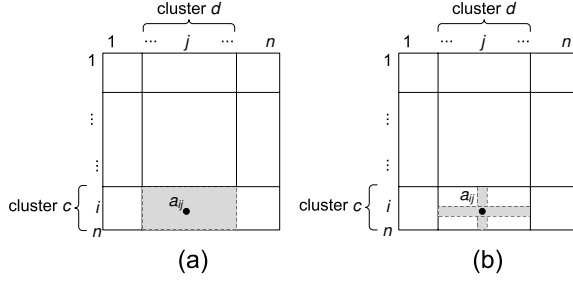


Fig. 5. The weighting schema comparison in two objective functions: (a) influence graph summarization using the entire block; (b) kernel k-mean using the block's column and row.

and re-formulate (9) into

$$\sum_{c,d=1}^k \frac{1}{2|\pi_c|} \left[\sum_{j \in \pi_d} \sum_{i \in \pi_c} a_{ij} \left(\sum_{p \in \pi_c} a_{pj} \right) + \sum_{j \in \pi_d} \sum_{i \in \pi_c} a_{ji} \left(\sum_{q \in \pi_c} a_{jq} \right) \right]$$

$$= \sum_{i,j=1}^n a_{ij} w_{ij}^{KM}$$

$$\text{where } w_{ij}^{KM} = \frac{\sum_{p \in \pi_c} a_{pj}}{2|\pi_c|} + \frac{\sum_{q \in \pi_d} a_{iq}}{2|\pi_d|} \quad (i \in \pi_c, j \in \pi_d) \quad (11)$$

Thus, both IGS and KM aim to maximize the weighted sum of graph adjacency matrix entries. In IGS, the weight of each entry is defined by the density of the belonging matrix block (or flow). In KM, the weight is defined by the average density of the column and row of the belonging matrix block. This is illustrated in Figure 5. Note that the heuristic of the CommonNeighbor based k-mean clustering is to put the graph nodes with similar in- and out-neighborhoods together. The resulting matrix blocks after the clustering tend to have uniform density distributions inside each block. Therefore, the density of the cross shape area in Figure 5(b) is a good approximation of the density of the shaded block area in Figure 5(a), which explains the rationality of using kernel k-mean clustering to the general IGS problem.

Furthermore, it is known that the kernel k-mean clustering problem is equivalent to the trace maximization problem:

$$\max_{H^T H = I, H \geq 0} \text{Tr}(H^T K H)$$

where the kernel matrix K equals the topology similarity matrix M^G computed by CommonNeighbor. The trace maximization problem can then be solved by SymNMF under spectral relaxations [2].

V. IMPLEMENTATION DETAILS

In this section, we provide some additional implementation details. As shown in Figure 4, our framework involves four kinds of algorithm-driven building blocks. The rooted graph search follows the standard BFS/DFS implementation. Below we describe details for similarity matrix computation, node summarization and the link pruning for post-processing of the summarization.

Similarity Matrix Computation. In Section IV, we have shown that using the heuristic of common neighbors to construct the similarity matrix (CommonNeighbor) can approximate the objective function of the squared IGS problem.

Algorithm 1: Link Pruning Algorithm.

Input : Initial summarization $S_0 \sim \{V, E\}$, # of flows l , $V = \{\pi_i\}_{i=1}^k$, $E = \{\xi_s\}_{s=1}^{k^2}$, flow rate $r(\xi_s)$

Output: Final summarization S

RankFilter(S_0) :

begin

$S \leftarrow S_0$;

for $s \leftarrow 1$ **to** k^2 **do** // rate normalization

$r(\xi_s) \leftarrow r(\xi_s) \sqrt{|\pi_c| |\pi_d|}$, $\xi_s \sim (\pi_c, \pi_d)$;

sort E by $r(E)$ in decreasing order;

for $s \leftarrow l+1$ **to** k^2 **do** // pruning

remove $E(s)$ from S ;

for $i \leftarrow 1$ **to** k **do** // link recovery

$E_i \leftarrow$ subset of E having π_i as destination;

sort E_i by $r(E_i)$ in decreasing order;

if $E_i(0) \notin S$ **then**

add $E_i(0)$ to S ;

This algorithm runs fast even for very large graphs due to a complexity of $O(md^2)$ where m is the number of links in G and d is the average node degree. We have implemented three versions of the algorithm and it is shown that bidirectional CommonNeighbor is generally better than one-directional forward or backward CommonNeighbor.

Node Summarization with SymNMF. The node summarization is done by applying SymNMF on similarity matrix $M^G(M^F)$, and using the factorized matrix H for cluster membership assignment. In our implementation, we apply the iterative SymNMF solver with the multiplicative updating rule in [2] which guarantees convergence. In this iterative algorithm, the initialization of H is critical to the final result. We introduce nonnegative eigenvalue decomposition similar to the method in [4] to compute a good initial factorization.

Link Pruning. The graph summarization by SymNMF needs further post-processing to select l top flows for the final summarization S . According to (7), the top flows can be extracted after ranking by the normalized flow rate. The other flows are then filtered out. This is illustrated in Algorithm 1. Notice that in the link recovery section of the algorithm, we introduce a constraint to keep a connected graph in the summarization. It is achieved by adding back the most dense flow going to each node cluster. An alternative choice is to use the maximum spanning tree (MST) algorithm [5].

We implement the proposed framework and algorithms in Java, which provides excellent UI library for visualization. The main computation routines are built on ParallelColt package [6] to optimize for multi-threading and sparse matrix operations. The speed of some core matrix decompositions (e.g., Eigenvalue) are further improved by invoking ARPACK (for sparse matrix) and LAPACK (for dense matrix) implementation [7] through JNI invocations.

VI. EVALUATION

In this section, we evaluate the proposed IGS framework and the CommonNeighbor algorithms by comparing

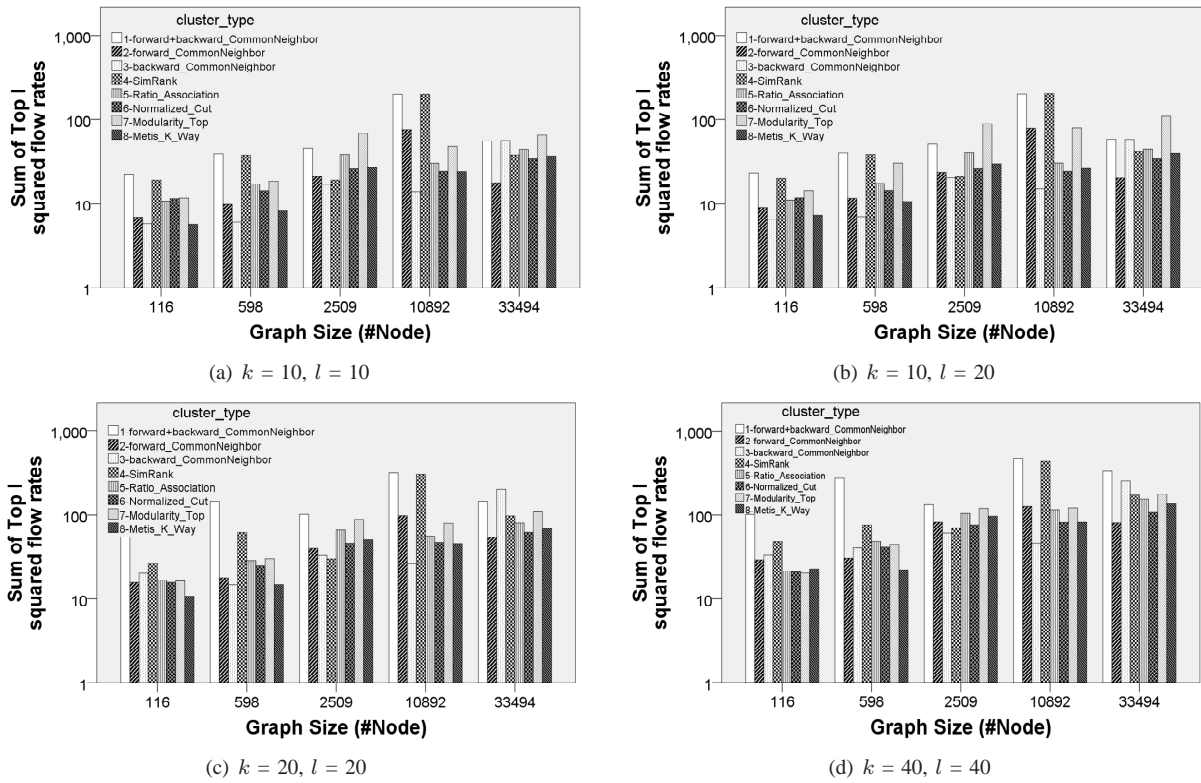


Fig. 6. The performance in maximizing the IGS objective on five sample graphs. The flow rate is summed from the top l flows between all the k clusters.

with alternative graph summarization methods. Nine approaches are considered: three using *CommonNeighbor* algorithms to compute the similarity matrix for SymNMF (i.e. forward+backward, forward, and backward settings), one using *SimRank* algorithm [8] to compute the similarity matrix for SymNMF, the classical graph clustering algorithm with *Ratio Association* and *Normalized Cut* objectives [9], agglomerative *Modularity*-based graph clustering [10], *Metis* K-way graph partition [11] and the Minimal Description Length (*MDL*) based graph summarization [12]. Note that Ratio Association and Normalized Cut are implemented using their equivalent similarity matrix computation for SymNMF [13]. Metis partition is implemented by official open source software package [14]. Modularity clustering is executed agglomeratively until all clusters stop merging at the top level or the number of clusters reaches k . For MDL, we implement the greedy algorithm in [12]. The MDL algorithm can not specify the number of clusters, in fact, it generates 4,937 clusters on one medium-sized influence graph. To ensure fair comparison (a larger number of clusters will lead to a much higher overall flow rate), we exclude MDL from numeric comparisons, but present its visual summarization results.

All the experiments are conducted on the same Linux server with two 8-core 2.9GHz Intel Xeon E5-2690 CPU and 384GB of memory. All the LAPACK and ARPACK libs are compiled locally to provide machine-optimized performance. Note that the modularity and Metis implementations are using native-version software package, not guaranteed to be optimized for multi-threading. The raw experiment data are paper citation graphs collected from ArnetMiner [15]. The influence graphs are obtained by reversing the citation links.

TABLE II. CITATION GRAPHS USED IN THE EXPERIMENT.

Source paper title	Venue/Year	Node	Link
Analysis of a hybrid cutoff priority scheme ...	Wireless Networks 1998	116	148
Manifold-ranking based image retrieval	ACM Multimedia 2004	598	895
Stochastic High-Level Petri Nets and Applications	IEEE TC 1988	2509	5256
Mining Frequent Patterns without Candidate Generation	SIGMOD 2000	10892	22301
On Power-law Relationships of the Internet Topology	SIGCOMM 1999	33494	86398

A. Flow Rate Maximization

We first pick five source papers from the data set to generate maximal influence graphs, as listed in Table II. These influence graphs are summarized into k clusters, between which the top l flow rates are summed according to the squared IGS objective in (2). Figure 6(a)~(d) present the comparisons among eight summarization methods on the numeric objective function.

The initial result in Figure 6(a) with a minimal graph summarization ($k = 10, l = 10$) suggests that among three *CommonNeighbor* algorithms, the bidirectional setting almost always achieves the best performance in maximizing the IGS objective (at least $> 100\%$ gain¹), except on the largest graph (#Node=33,494), the backward *CommonNeighbor* obtains a tiny advantage (1%). Further, comparing the bidirectional *CommonNeighbor* to traditional graph summarization methods, *CommonNeighbor* achieves much better performance than

¹Percentage of performance gain (drop) by $\frac{\text{new_number} - \text{original_number}}{\text{original_number}} \times 100\%$, the same below.

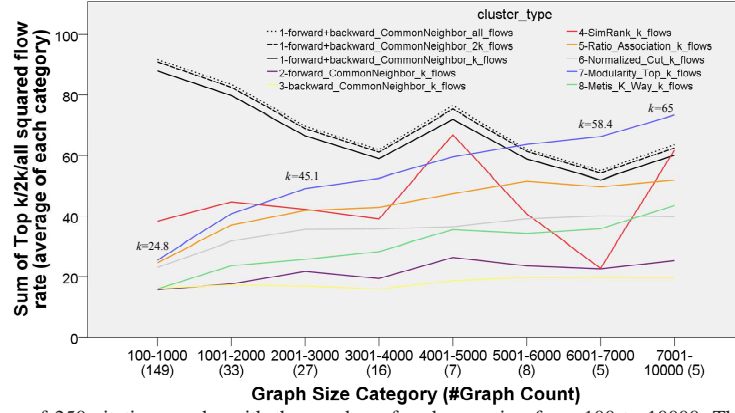


Fig. 7. The squared IGS performance of 250 citation graphs with the number of nodes ranging from 100 to 10000. The cluster number is set to $k = 20$.

Ratio Association, Normalized Cut and Metis (at least $> 20\%$, in average $> 100\%$). In some cases, the performance of CommonNeighbor is matched by SimRank ($< 10\%$ gain) or outperformed by Modularity.

When we double the number of flows ($k = 10, l = 20$) in Figure 6(b), the sum of flow rates does not increase much on all algorithms (in average $< 15\%$) and the overall comparative patterns stay unchanged. This shows that the top k flows already capture most of the flow rates on the graph summarization. We then increase the number of clusters ($k = 20, l = 20$; $k = 40, l = 40$). The results in Figure 6(c)(d) reveal that the objection function increases much as the number of clusters increases (at least $> 30\%$, in average $> 90\%$, comparing Figure 6(c) with Figure 6(b)), except for Modularity, which remains unchanged because their number of clusters are already larger than k and kept stable. For example, the sample graph with 33,494 nodes stops at 71 clusters in the top modularity level. On the comparative pattern, bidirectional CommonNeighbor regains performance advantage over SimRank and Modularity under a large number of clusters.

During the experiment, we have executed each algorithm case three times and report their average performance. However, the results in Figure 6 still show some randomness due to the nature of iterative NMF solver. To obtain more accurate result, we carefully sample 250 well-cited source papers published in KDD and ICDM from the ArnetMiner data set. The size of their maximal influence graphs are within the range of 100~10,000 nodes. On each graph, similar experiments are conducted as above given a setting of $k = 20$. Finally in Figure 7, 250 graphs are categorized into 8 bins according to their size. The average performance in each bin are reported for comparison. Results on the larger data set demonstrate the same pattern with the five sample graphs. Bidirectional CommonNeighbor in most cases are the best, except for Modularity, which becomes better as the number of nodes increases beyond 5,000. As mentioned, this is because the Modularity algorithm generates more clusters than the initial setting of $k = 20$. As indicated by the labels above the Modularity performance (blue line), the number of clusters increases from 24.8 in the first category to 65 among the largest graphs. Increasing the number of flows for CommonNeighbor does not optimize the objective function much.

B. Visualization

We evaluate the effectiveness of summarization methods also by comparing their visualization results: whether they produce a clean influence graph summarization with little visual clutter and whether the results are meaningful for users with domain knowledge. We first pick the famous frequent pattern mining paper by Prof. Jiawei Han et al. as the source to generate the maximal influence graph. Then we execute seven typical summarization methods and depict their results in Figure 8(a)~(g). At the first glance, the proposed bidirectional CommonNeighbor method generates a connected tree-like influence graph summarization without edge crossing (Figure 8(a)). Compared to that, SimRank gets a similar visual form (Figure 8(b)) due to the comparable objective function result, but the generated graph is not connected. The Metis result is also clean (Figure 8(f)), but all the clusters have a similar number of nodes, making the summarized graph impractical for usage. Ratio Association and Normalized Cut look inferior due to the poor graph connectivity (Figure 8(c)) and the flat influence hierarchy (Figure 8(d)). Modularity and MDL are the worst because of the visual clutter generated from the large number of clusters remained in the summarization (Figure 8(e)(g)).

Taking a closer look at the visual summarizations, we find that by CommonNeighbor, most flows represent at least 300 citation links. While by SimRank, the critical flows linking the source node are fragmented, two of which only include 52 and 83 citations. The same deficiency is found in the result by Metis, where two highlighted flows only have 11 and 12 citations. We also invite a senior researcher from the database and data mining community to evaluate the summarization result. With our interactive tool, she can switch between the title+abstract summary and the research field summary. She can also access paper details in each node cluster with a sorted list by citation count. She mainly compares the visual summarization by CommonNeighbor and SimRank. In this case, she prefers the result by CommonNeighbor in Figure 8(a) because the influence evolutions make more sense: the initial paper quickly raises much attention on pattern mining research such as itemset and association rule mining, then the thread splits into four streams on general data management research (such as web and uncertainty skyline analysis), trajectory analysis, subgraph analysis and application in software engineering (e.g. bug analysis). The thread of web data analysis gradually moves to web retrieval and finally leads to tag analysis and anomaly behavior detection. Compared with CommonNeighbor, Sim-

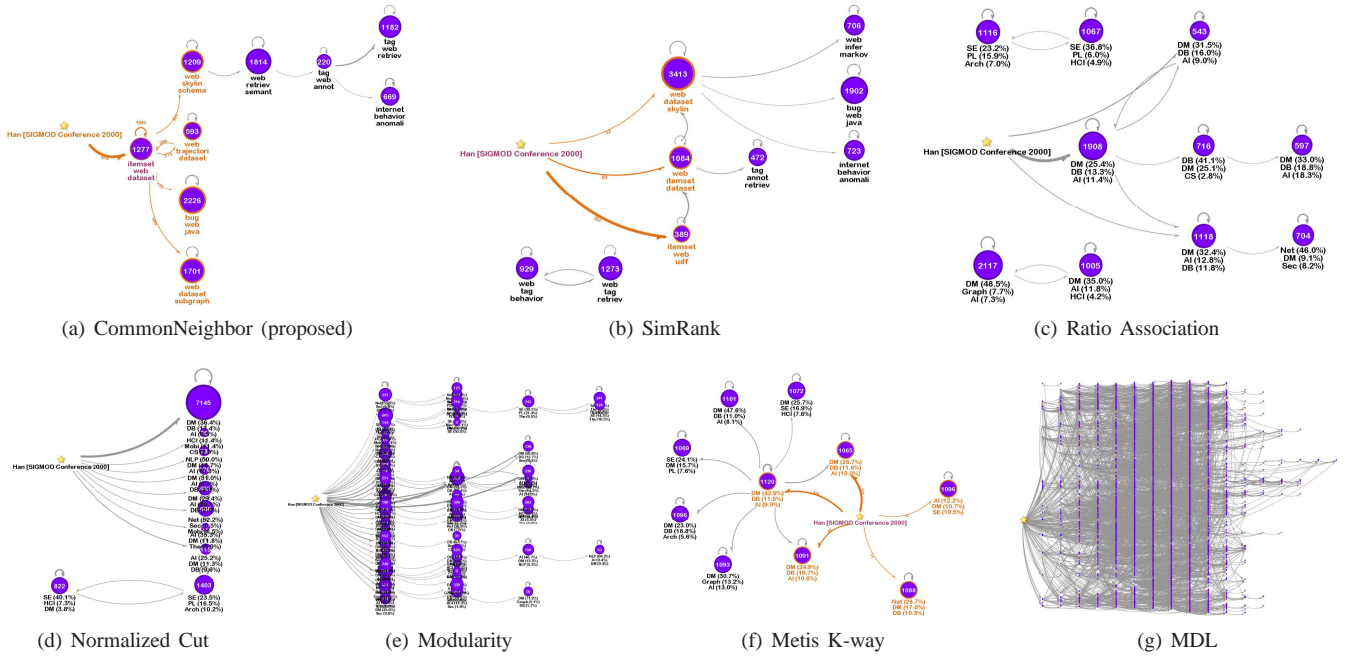


Fig. 8. Influence graph summarization results on [Han SIGMOD'2000] by different methods ($k = 10$, $l = 20$). Node label gives the number of papers in each cluster and their content summary by either title+abstract keywords in (a),(b) or the top 3 research fields in (c)~(f). Link thickness indicates the normalized flow rate. Some part of the graph is highlighted to show the number of citations as edge labels. Note that the modularity algorithm stops at 62 clusters and can not merge any further. MDL produces 4,937 clusters, leaving a half of the visual complexity from the input graph.

Rank creates some false links, e.g. the direct flow from the frequent pattern mining paper to uncertainty data analysis.

Furthermore, we ask our invited users to study the influence of the well-known Internet power-law paper in SIGCOMM'1999. The maximal influence graph is summarized by the bidirectional CommonNeighbor algorithm into Figure 1 (in the second page). Note that in this case the influence graph topology is augmented by the “venue” field of each paper to group the papers with similar research topic together. From the visual summarization, she learns that the SIGCOMM paper directly influences the research on Internet topology and simulation. Next, over the Internet topology topics, the P2P research becomes popular and after that the web-related research and XML. The most recent hot topic in this thread appears to be sensor network which corresponds well to his domain knowledge.

Our framework can also visualize one author's influence by summarizing the author influence graph. This graph is generated by adding one influence link between two authors for each citation between their papers. The maximal author influence graph is then computed from a source author by traversing the influence graph. As an example, we select Prof. Jiawei Han as the source author, and collect the influenced authors within 2 hops. To limit the size of the influence graph, we only keep productive authors (i.e. ≥ 30 paper publications) which gives a graph of 26,349 author nodes. The summarization result applying bidirectional CommonNeighbor algorithm ($k = 10$) is shown in Figure 9. Our invited user acknowledges the validity of the result: Prof. Han has influenced multiple fields with his research, mainly data mining (DM), database (DB), AI and networking (Net). On his contribution to DB and DM fields, the influence is bidirectional, i.e. he is also heavily influenced by the researchers there, as indicated by the group of 109 authors in the picture (e.g. Raghu Ramakrishnan). The

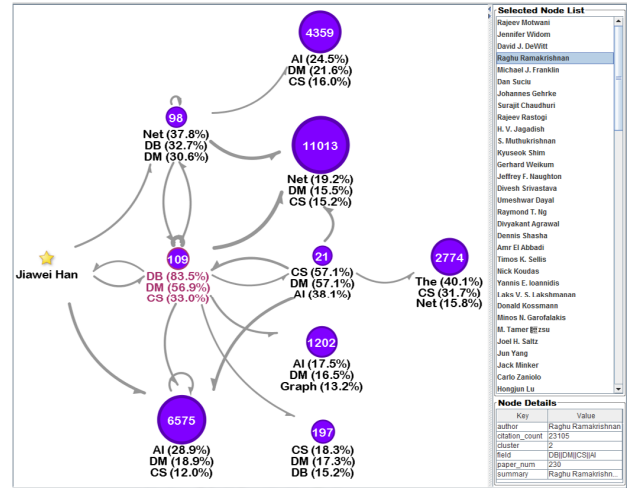
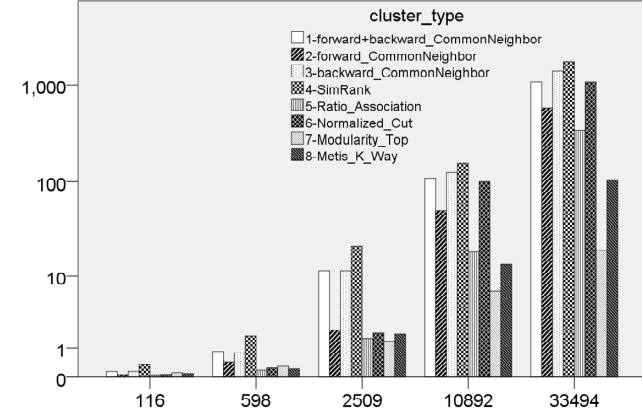


Fig. 9. The summarization of Prof. Jiawei Han's influence graph by bidirectional CommonNeighbor ($k = 10$).

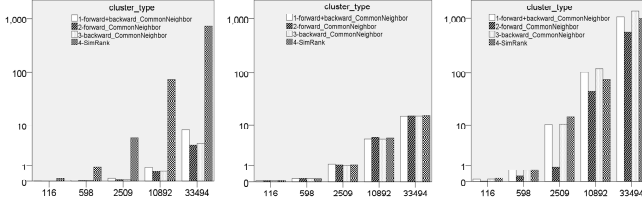
most directly influenced field by the number of authors are AI and DM, as indicated by the group of 6,575 authors. The most indirectly influenced field are Net and DM, as indicated by the group of 11,013 authors. Through the bridging of a group of 21 authors (e.g. Rakesh Agrawal), he also impacts the Theory (The) research, represented by the group of 2,774 authors.

C. Scalability

The overall computation time for different summarization methods is illustrated in Figure 10(a). Our proposed algorithms are more costly than efficient modularity clustering algorithm ($O(n \log(n))$ with small constant) and Metis k-way graph partition ($O(n + m)$). However, the best of our methods can summarize a 10,000-node maximal influence graph in 100 seconds, and the overall time complexity is only slightly above linear. Note that n here denotes the size of the maximal



(a) Total computation time (second) by graph size (#node)



(b) Similarity Matrix

(c) Eigen Initialization

(d) SymNMF

Fig. 10. The time cost of different summarization methods, $k = 20$: (a) Total time; (b)~(d) Split time of four algorithms in our framework. The similarity matrix computation and SymNMF iteration dominate the cost.

influence graph, which is much smaller than the size of the original graph. Most citation graphs from a single paper are no larger than the magnitude of 10,000 nodes, while the entire data set has millions of papers.

Within our framework, the SimRank algorithm requires the longest computation time. To explain this, we have looked at the split time at three key steps, as shown in Figure 10. The eigenvalue decomposition (Figure 10(c), only top k eigenvectors are computed) are quite fast due to the sparsity of the influence graph matrix (Table II). On similarity matrix computation (Figure 10(b)), SimRank is slow because in worst case it needs to compute an all-to-all similarity matrix ($O(n^2d^2)$), though we have optimized it to only compute within a 4-hop range. In contrast, CommonNeighbor is much faster on similarity computation, through the multi-threaded routine on sparse matrix multiplication. Finally, SymNMF (Figure 10(d)) is the most costly step. In each iteration, there are a few sparse matrix-matrix multiplication computations.

Compared with the time complexity, the space requirement of our framework is less stringent. The similarity matrix computation and iterative SymNMF each needs to store a dense matrix at most, giving a space complexity of $O(n^2)$ with small constant. The eigenvalue decomposition by dsyevx routine in LAPACK only needs $O(n)$ space with a relatively large constant. Recall that n is the number of nodes in the maximal influence graph and can be hundreds of times smaller than the original input graph.

D. Summary and Discussions

First, our experiment results demonstrate that the summarization methods specifying the number of clusters provide compact influence graph summarizations. In contrast, typical graph compression and summarization methods such as MDL

and Modularity can lead to huge visual clutters that make it hard for user to interpret. Within the k -cluster methods, applying bidirectional CommonNeighbor algorithm in our framework is shown to be the best in maximizing the IGS objective, constantly superior than traditional graph partition and clustering algorithms, such as Ratio Association, Normalized Cut and Metis. In a few cases, plugging SimRank into our framework can achieve comparable performance. In fact, SimRank has very close tie to our method. CommonNeighbor considers the similarity of two nodes in one hop beyond, while SimRank computes their similarity in an infinite hop (pruned to four hops in this work). Our results show that, though close to, SimRank is not better than CommonNeighbor in maximizing the IGS objective, but also it suffers from a higher computational complexity of $O(n^2d^2)$.

Second, we note that the parameter k and l in the summarization can be critical for both the objective and user performance. As k becomes large, for example from $(k = 20, l = 40)$ to $(k = 40, l = 40)$ in the cases of Table II, the IGS objective increases 56% in average while the visual complexity doubles. We recommend to set $k \leq 20$ on this trade-off, because at a size larger than 20, the node-link graph visualization may not be a good choice for many graph visual analysis tasks [16]. On the choice of l , we find that the optimization of IGS objective is not significant after $l \geq 2k$, therefore $l = 2k$ can be an appropriate setting.

Last, we target on the academic data sets in this work. It seems straightforward to apply our framework also to social influence graphs on Twitter and Facebook. However, we caution that the basic retweeting influence graph can be summarized clearly by MDL or structural equivalence [17] based node grouping, because such graphs are all standard trees with large structural redundancy.

VII. RELATED WORK

First, *graph summarization*, constructing a smaller abstraction to represent the large graph has been a traditional research topic, e.g. using graph clustering algorithms. These algorithms usually optimize certain association or cut measure during the k -way graph partition. Several measures have been proposed, e.g. ratio association, ratio cut [18] and normalized cut [9]. The similar problem is also studied in the context of community detection by interdisciplinary researchers [19], in which modularity is one of the most popular quality function to access a community [20]. However, most of the clustering and community detection methods on graph target at maximizing intra-cluster connections while minimizing inter-cluster connections. This is fairly different from the IGS problem studied here. On the other hand, there are also plenty of works in compressing large graphs for efficient storage and representation. In [12], MDL-based compression was proposed to present the graph with an aggregated structure and an error correction list. It is proved to be the best summary from the information-theoretic objective. While MDL approach can successfully compress web graphs, on influence graphs which are much sparser (the citation graphs have an average degree of less than 3), it performs similarly to a structural equivalence based grouping [17], leaving huge visual clutters unsettled. Another algorithm, SNAP [21], considers the node attribute on graph, but again is not tailored for the influence graph scenario.

Second, *visualization*, over the past few decades, the methodology to draw node-link graphs has reached its maturity. On graphs with less than a few hundred nodes, the planar graph drawing approach [22] and the force-directed algorithm [23] can produce visually pleasant graph layouts in real time, mainly by minimizing edge crossings. On large graphs with a thousand or more nodes, the force-based algorithms can be extended by multilevel coarsening and fast force approximation [24] and still generate a layout in reasonable time (e.g. less than a minute for million-node graphs). However, on real-world large graphs with small-world nature, including the influence graph discussed here, the resulting graph layout still has numerous edge crossings. This leads to overwhelming visual clutters detrimental to visual data mining tasks. Meanwhile, Shahaf et al. [25][26][27] studied the similar problem of summarizing large amount of information into user-friendly visual maps. They developed intriguing methods to detect hidden linkage and document clusters from the keyword frequency statistics. On a quite different focus, our method is built on the graph with explicit linkage data while the textual content of each node can be absent or incomplete.

Third, considerable work has been conducted for studying the effects of *social influence*. For example, Bakshy et al. [28] conducted randomized controlled trials to identify the effect of social influence on consumer responses to advertising. Bond et al. [29] used a randomized controlled trial to verify the social influence on political voting behavior. Tang et al. [30] presented a Topical Affinity Propagation (TAP) approach to quantify the topic-level social influence in large networks. Kempe et al. [1] proposed to use a submodular function to formalize the influence maximization problem and develop a greedy algorithm to solve the problem with provable approximation guarantee. Most of these works focus on the existence of social influence or the nature of the information diffusion process and do not consider the summarization problem. Recently, Mehmood et al. proposed CSI [31], a model that generalizes the classical Independent Cascade model to the community level, built from the cascade-based community detection method [32]. CSI can produce similar visual forms to our result. However, the CSI model is computed from the probabilistic social influence graph and the information propagation log more engaged to the social influence scenario. In comparison, our method is more focused on the visual summarization of large influence graphs in the objective of maximizing flows. We do not leverage the information propagation model and the associated log data in such scenarios.

VIII. CONCLUSIONS

In this paper, we propose the influence graph summarization problem, study its linkage to the existing clustering methods, and present a unified framework to solve it. The framework achieves all the three design objectives, including (1) flow rate maximization that highlights the evolution of influence; (2) a localized visualization from the source node; and (3) easy to incorporate rich information on graph such as node attribute and time. The framework is comprehensive and flexible. We provide both the SymNMF based solution and implementation details. Through comprehensive evaluations with real-world academic citation graphs, we demonstrate that our framework constantly outperforms classical methods,

such as graph clustering and compression algorithms, in both quantitative performance and qualitative visual effects.

REFERENCES

- [1] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.
- [2] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *SDM*, 2005.
- [3] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Spectral relaxation for k-means clustering," in *NIPS*, 2001, pp. 1057–1064.
- [4] C. Boutsidis and E. Gallopoulos, "SVD based initialization: A head start for nonnegative matrix factorization," *Pattern Recognition*, 2007.
- [5] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50.
- [6] "Parallelcolt," <https://github.com/Danimoth/Parallel-Colt>.
- [7] "Intel math kernel library," <http://software.intel.com/en-us/intel-mkl/>.
- [8] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *KDD*, 2002, pp. 538–543.
- [9] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 1997.
- [10] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, 2004.
- [11] G. Karypis, V. Kumar, and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed Computing*, vol. 48, pp. 96–129, 1998.
- [12] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *SIGMOD*. ACM, 2008, pp. 419–432.
- [13] D. Kuang, H. Park, and C. Ding, "Symmetric nonnegative matrix factorization for graph clustering," in *SDM*, 2012, pp. 106–117.
- [14] "Metis - serial graph partitioning and fill-reducing matrix ordering," <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/>.
- [15] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *KDD*, 2008, pp. 990–998.
- [16] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "A comparison of the readability of graphs using node-link and matrix-based representations," in *InfoVis*, 2004, pp. 17–24.
- [17] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.
- [18] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1096, 1994.
- [19] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.
- [20] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004.
- [21] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *SIGMOD*, 2008, pp. 567–580.
- [22] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.
- [23] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989.
- [24] Y. Hu, "Efficient and high quality force-directed graph drawing," *Mathematica Journal*, vol. 10, no. 1, pp. 37–71, 2005.
- [25] D. Shahaf, J. Yang, C. Suen, J. Jacobs, H. Wang, and J. Leskovec, "Information cartography: creating zoomable, large-scale maps of information," in *KDD*, 2013, pp. 1097–1105.
- [26] D. Shahaf, C. Guestrin, and E. Horvitz, "Trains of thought: Generating information maps," in *WWW*, 2012, pp. 899–908.
- [27] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," in *KDD*, 2010, pp. 623–632.
- [28] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn, "Social influence in social advertising: evidence from field experiments," in *EC*, 2012, pp. 146–161.

- [29] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. I. Kramer, C. Marlow, J. E. Settle, and J. H. Fowler, "A 61-million-person experiment in social influence and political mobilization," *Nature*, vol. 489, pp. 295–298, 2012.
- [30] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD*, 2009, pp. 807–816.
- [31] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen, "Csi: Community-level social influence analysis," in *ECML/PKDD*, 2013, pp. 48–63.
- [32] N. Barbieri, F. Bonchi, and G. Manco, "Cascade-based community detection," in *WSDM*, 2013, pp. 33–42.