Localized 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 for social good. 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 local details and fine readability?

To answer these questions, we first formally define the localized influence graph summarization problem. Second, we propose an end-to-end framework to solve the new problem. Our algorithms can not only highlight localized 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 approximate the proposed influence flow maximization objective while outperforming previous methods in a typical scenario of visually mining academic citation networks.

1. 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 an influential research paper impact several research areas; and consequentially, how do these different areas interact with each other and lead to a new multi-disciplinary research direction? How does a rumor on twitter mislead people from different regions and cause public panic?

Although closely related, IGS problem bears some subtle difference from the existing work. We briefly review three mostly related ones. First (*influence maximization*), in the past decades, many elegant algorithms have been proposed for the so-called influence maximization problem [13, 28, 22]. 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 clutters 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 three design objectives that differentiate our IGS problem from existing works.

- *D1. Maximizing Flow Rate.* Quite different from extracting dense clusters on graph, the goal of IGS is to highlight the flow of influence moving across and within clusters for visual perception.
- *D2. View localization.* The influence graph to be summarized is asymmetrically weighted, subjecting to the focus node which defines the source of the influence. The remote nodes spanning multiple hops from the focus can receive less influence than the neighboring nodes of the focus. Having a finer granularity on the clusters closer to the focus will greatly enhance the effectiveness of visual perception.
- *D3. Rich Information.* Most influence graphs have rich attributes (e.g., the topic, venue and publication date of a scientific paper) and often evolve over time. Incorporating these attributes to enhance the summarization 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 our IGS problem by decompos-

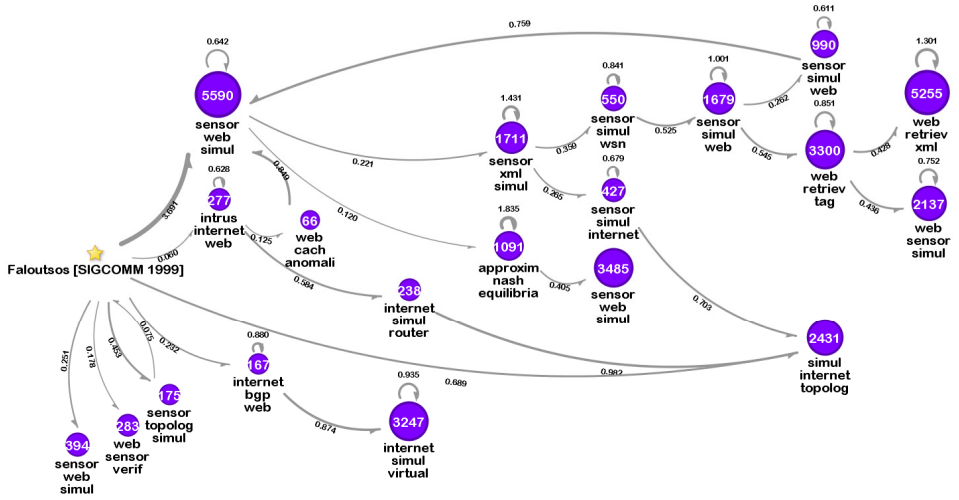


Figure 1: Localized influence graph summarization on [Faloutsos SIGCOMM'1999] by our framework. Topology similarity and venue information are integrated. Node label gives the size and summary on paper title+abstract normalized by keyword frequency. Edge label and link thickness indicate normalized flow rate.

ing it into several key building blocks. It is flexible and admits many existing graph mining algorithms for each of its building blocks. We conducted extensive theoretic analysis as well as empirical evaluations to validate the effectiveness and efficiency of the framework. The main contributions of the paper can be summarized as

- *Problem Definition*, to fulfill all the three design objectives listed above for localized visual summarization of large influence graphs;
- *A Unified Framework and Implementation Details*, to solve the IGS problem;
- *Theoretic Analysis*, to reveal the intrinsic relationship between IGS problem and the existing work;
- *Empirical Evaluation*, to demonstrate the effectiveness and efficiency of the proposed framework.

The rest of the paper is organized as follow. We formally define the IGS problem in Section 2, and present our framework in Section 3. We analyze the equivalence between IGS problem and the existing work in Section 4. We present the implementation details and empirical evaluations in Section 5 and Section 6, respectively. Finally, we review related work in Section 7 and conclude the paper in Section 8.

2. PROBLEM DEFINITION

Table 1 lists the notations used throughout the paper. The raw inputs are the influence graph I and the focus 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.

Table 1: Notations.

SYMBOL	DESCRIPTION
I	influence graph as input
f	focus 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
W, ω_{ij}	localized weight 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, $\lvert\pi_c\rvert$, k	clusters, cluster size and # of clusters in S
ξ_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
$\omega(\xi_s)$	localized flow rate of ξ_s

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 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 for users.

2.1 Flow Rate Maximization

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 below

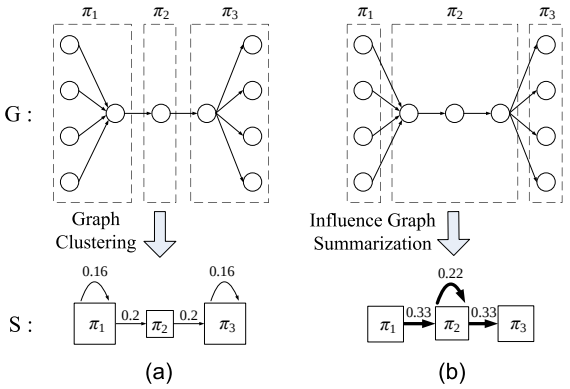


Figure 2: Difference between IGS problem and traditional graph clustering problem. Each dash box in the original graph G becomes a square node in the summarization graph S. (a) traditional graph clustering leading to more intra-cluster flows; (b) influence graph summarization exposing denser flows. In 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.

objective function:

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

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.

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

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 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 and 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 simple 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:

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

From the perspective of highlighting influence flows, the squared IGS objective is consistent with the general IGS.

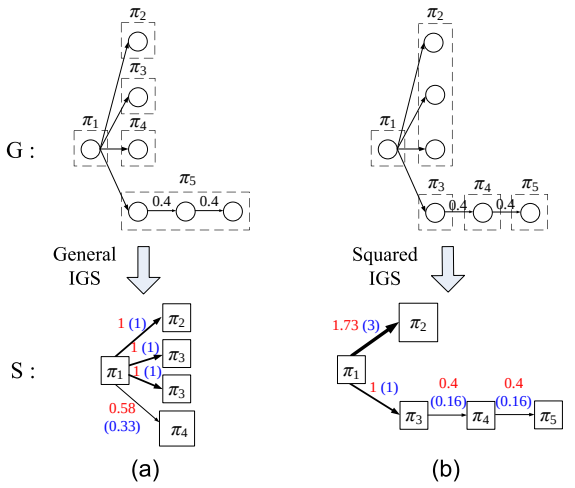


Figure 3: An illustrative graph leading to fragmented flows by general IGS objective ($k = 5$, $l = 4$), the links far away from the focus node (π_1) have a lighter weight of 0.4. (a) Graphs are summarized into four clusters at 1-hop, the normalized flow rate by (1) is labeled in red, favoring this summarization by a sum of $3.58 > 3.53$; (b) Applying the squared IGS objective, more structures of the influence graph can be revealed. The squared flow rate by (2) is labeled in blue parentheses, having a sum of $4.32 > 3.33$. (best viewed in color)

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.

2.2 Localization

Consider that the maximal influence graph G is derived from the focus node f, the summarization of G should also be subject to the focus node f. Here we define a weight matrix W on G localized to the focus node f. The entries of W is denoted as $\{\omega_{ij}\}_{i,j=1}^{n}$, indicating the localized weight of each link subject to f. As we shall discuss later, W will be constructed to favor the contribution of closer nodes and links to f.

Problem 3: The **localized IGS problem** incorporates the weight matrix W to generate a localized graph summarization subject to the focus node f. The objective function is written as:

$$\max \quad \sum_{s=1}^{l} \omega(\xi_s)^2|\pi_{c(s)}||\pi_{d(s)}| \tag{3}$$

where $\omega(\xi_s)$ denotes the localized flow rate of ξ_s weighted by the square root of the weight matrix entries to accommodate the squared IGS objective.

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

3. FRAMEWORK

In this section, we propose a unified framework to solve IGS problems, including an end-to-end pipeline, algorithms to summarize the influence from the graph topology, and the natural extensions to incorporate graph attributes and

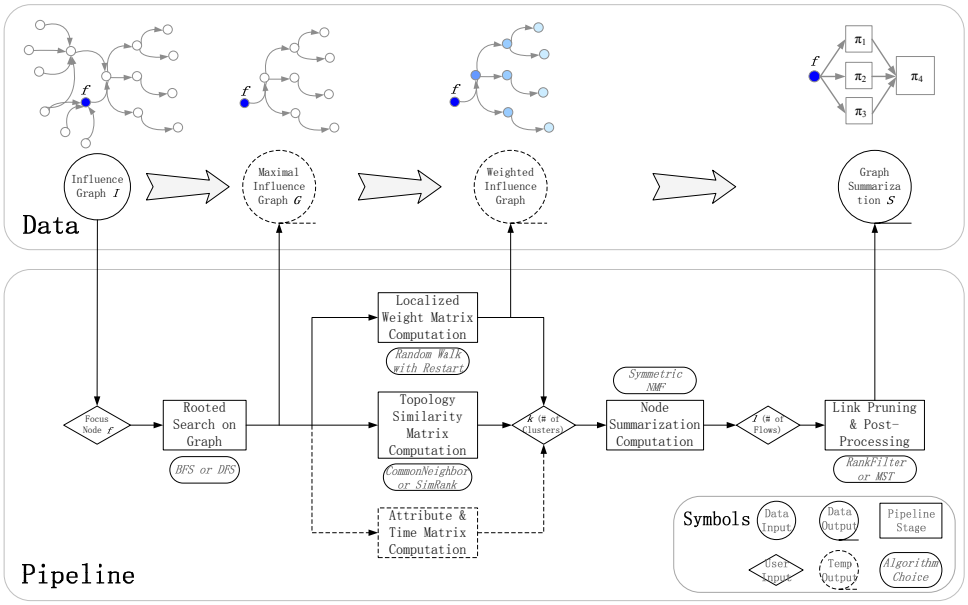


Figure 4: The framework and algorithm pipeline to solve the localized influence graph summarization problem. Dashed parts indicate the extensions and the temporary output of the framework.

time information. We will leave the theoretic analysis and the implementation details to Section 4 and Section 5, respectively.

3.1 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/components. Initially, the maximal influence graph G is computed from the input graph I by a breadth-first or depth-first search starting from the focus node f. Over the rooted influence graph G, three processing components work in parallel to generate three matrices on the graph: the localized weight matrix, the topology similarity matrix and the attribute/time matrix. 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 maximization objective (See Section 4 for detailed analysis). By incorporating the weight matrix, the localized objective function is also satisfied. Optionally, the attribute/time matrix is used to augment the matrix for decomposition so that the resulting summarization is optimized over graph attributes. The requirement of the l flows in the summarization is further fulfilled by link pruning using either ranking-based filtering or the maximum spanning tree algorithm. The proposed pipeline is flexible and admits many existing scalable 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.

3.2 Node Summarization

In the proposed pipeline, the key building block is the node summarization. It takes the topology similarity matrix M^G as the basic 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 provided in Section 4 and 5, respectively. Note that the algorithm is executed bidirectionally that both forward and backward neighborhoods contribute equally to the final similarity matrix M^G.

Over the similarity matrix M^G, the decomposition employs a Symmetric version of the Nonnegative Matrix Factorization (SymNMF [12]) which optimizes:

$$\min_{H \geq 0} ||M^G - HH^T||_F^2 \tag{4}$$

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

To solve the localized version of IGS problems, the weight matrix of G (denoted as W) is incorporated. The SymNMF decomposition becomes the weighted SymNMF:

$$\min_{H \geq 0} ||W \odot (M^G - HH^T)||_F^2 \tag{5}$$

where $\odot$ indicates the Hadamard (by element) product of matrices. In our framework, W is computed by random walk with restart (RWR [34]) on G from the focus node f. The RWR weight is known to well represent the ranking of each node subject to the focus node f. Formally, the weight matrix W is constructed by

$$\omega_{ij} = \begin{cases} \omega_{ii}, & i = j \\ \sqrt{\omega_{ii}\omega_{jj}}, & i \neq j \end{cases}$$

where the diagonal elements of W are set by the RWR weight of each node from f. Under this construction, the objective function in (5) is converted to that of a standard SymNMF.

$$\min_{H \geq 0} ||W \odot M^G - (W_2 \odot H)(W_2 \odot H)^T||_F^2 \tag{6}$$

where $W_2 = \{\sqrt{\omega_{ii}}\}_{i,j=1}^n$. This provides us extra advantage because the algorithm for weighted SymNMF does not converge as well as that of the standard SymNMF.

3.3 Variants and Generalizations

As shown in the dashed part of Figure 4, our framework can be extended to incorporate node attributes and their time information on graph (e.g., the "research field" attribute and 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

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

where Λ_{aug} controls the degree of augmentation, by default set to 2.

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 nearby years will have a larger 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. By using the median of cited half-life measure of sci-indexed computer science journals, we compute a default value of $\lambda_{Decay} = 1.1144$.

The attribute and time matrices are then used to extend the topology similarity matrix. The unified similarity matrix M^F is computed by

$$M^F = M^G \odot M^D \odot M^T \tag{7}$$

The final SymNMF objective becomes

$$\min_{H \geq 0} ||W \odot (M^F - HH^T)||_F^2 \tag{8}$$

4. EQUIVALENCE ANALYSIS

In this section, we present theoretic analysis, to explain the rationality behind our matrix decomposition based solution for graph summarization. We start with deriving an approximate objective function of the squared 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.

4.1 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 assignment; and $\{\xi_s\}_{s=1}^l$, the selected top flows. However, the simultaneous optimization of these two classes of variables is hard due to the non-linear and combinatorial nature of the problem. Therefore, we consider an 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 given an appropriate l (e.g. $l = k$), because the top l flows will contribute the most part in the overall sum after applying the square function to flow rate. Formally, the approximate objective function becomes:

$$\max \quad \sum_{s=1}^{k^2} r(\xi_s)^2 |\pi_{c(s)}||\pi_{d(s)}| = \sum_{c,d=1}^{k} \frac{(\sum_{i \in \pi_c, j \in \pi_d} a_{ij})^2}{|\pi_c||\pi_d|} \tag{9}$$

$$\max \quad \sum_{s=1}^{l} r(\xi_s)^2 |\pi_{c(s)}||\pi_{d(s)}| \qquad given \ \{\pi_c\}_{c=1}^k \tag{10}$$

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

4.2 Kernel K-Mean Clustering

According to [35], 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 will cluster the data vectors into k non-overlapping clusters $\{\pi_c\}_{c=1}^k$ based on the objective function

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

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

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

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

$$\max \quad \sum_{c=1}^{k} \sum_{i,j \in \pi_c} \frac{\phi(x_i) \cdot \phi(x_j)}{|\pi_c|} \tag{11}$$

Consider the one-hop bidirectional algorithm using the number of common neighbors as the similarity (CommonNeighbor), the corresponding graph topology similarity matrix is computed by

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

Substitute K_{ij} for $\phi(x_i) \cdot \phi(x_j)$ when K is used as the kernel matrix, (11) becomes

$$\max \quad \sum_{c,d=1}^{k} [\sum_{j \in \pi_d} \frac{(\sum_{i \in \pi_c} a_{ij})^2}{|\pi_c|} + \sum_{i \in \pi_c} \frac{(\sum_{j \in \pi_d} a_{ij})^2}{|\pi_d|}]/2 \tag{12}$$

4.3 Equivalence

Let us compare the objective functions in (9) and (12). They are actually in similar mathematical forms if we reformulat them as

$$\max \quad \sum_{i,j=1}^{n} a_{ij} w_{ij}^{IGS}$$

$$where \quad w_{ij}^{IGS} = \frac{\sum_{p \in \pi_c, q \in \pi_d} a_{pq}}{|\pi_c||\pi_d|} \ (i \in \pi_c, j \in \pi_d) \tag{13}$$

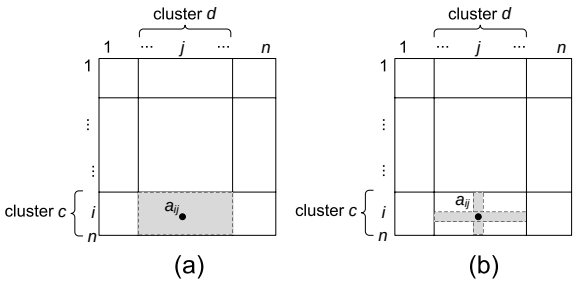


Figure 5: The area of density to weight the graph entries in two objective functions: (a) influence graph summarization using the entire block; (b) kernel k-mean using the block's column and row.

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

$$where\ w_{ij}^{KM} = [\frac{\sum_{p\in\pi_c} a_{pj}}{|\pi_c|} + \frac{\sum_{q\in\pi_d} a_{iq}}{|\pi_d|}]/2\ (i \in \pi_c, j \in \pi_d) \quad (14)$$

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 more 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.

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

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

where the kernel matrix K equals the topology similarity matrix M^G we compute by CommonNeighbor. The trace maximization problem can then be solved by SymNMF under spectral relaxations [12].

The localized IGS problem can be studied similarly. Compare the objective function in (3) with (2), we can see that the only difference is to replace a_{ij} with $\sqrt{\omega_{ij}} a_{ij}$. After the same analysis as above, the SymNMF to solve becomes:

$$\min_{H\geq 0} ||W \odot M^G - HH^T||_F^2$$

which leads to the solution to (6), because only the relative ranking of entries in each row of H is used for the cluster assignment.

5. IMPLEMENTATION DETAILS

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

Similarity Matrix Computation. We consider two alternatives to compute the similarity matrix M^G from the adjacency matrix A of G. In Section 4, we have shown that using the number of common neighbors as the similarity (CommonNeighbor) can approximate the objective function of the squared IGS problem. This algorithm can run fast even for very large graphs due to its complexity of $O(md^2)$ where m is the number of links in G and d is the average node degree. Bidirectional CommonNeighbor will be generally better than one-directional forward or backward CommonNeighbor. On broadcast-style influence graphs (e.g. reversed citation network), the adjacency matrix usually has dense flows on flat blocks with larger width than height (Figure 5(b)), i.e. a few papers influencing a lot of succeeding papers. Then the diversity among vertical strips of the block is smaller than the horizontal strips. In such cases, using the forward CommonNeighbor can be better than the backward one.

We also implement SimRank [19], a classical algorithm improving over the 1-hop similarity computation. SimRank is shown to be slightly better in maximizing the IGS objective function, though generally it is slower on large graphs, with a complexity of $O(n^2d^2)$. The performance of these algorithms are studied and compared in Section 6.

Node Summarization. The node summarization is done by applying SymNMF on similarity matrix $M^G(M^F)$, and then using the factorized matrix H for cluster assignment. Here we apply the multiplicative updating rule in [12] for SymNMF which guarantees convergence. However, due to the nature of multiplicative updating, the initialization of H is critical to the final result. We introduce nonnegative double eigen-decomposition (NNDED) similar to NNDSVD in [9] 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 (10), 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 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;
end

Table 2: Citation graphs used in the experiment.

Seed 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

algorithm [23] which computes a tree out of the summarization with maximal overall flow rates.

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 [3] 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 [1] through JNI invocations.

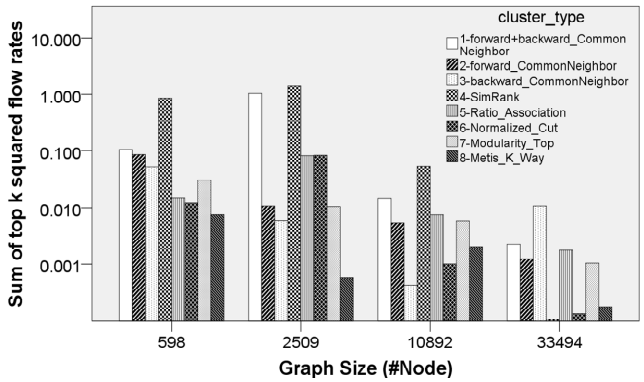
6. EXPERIMENT

We compare nine graph summarization methods: three using *CommonNeighbor* algorithms to compute the similarity matrix for SymNMF (i.e. forward+backward, forward and backward settings), one using *SimRank* [19] to compute the similarity matrix for SymNMF, three classical graph clustering algorithms with *Ratio Association* objective, *Normalized Cut* objective [29], agglomerative *Modularity*-based graph clustering [26], *Metis* K-way graph partition [21] and the Minimal Description Length (*MDL*) based graph summarization [25]. Note that Ratio Association and Normalized Cut are implemented using their equivalent similarity matrix computations for SymNMF [24], Modularity clustering and Metis partition are generated by official open source software package [2] and feed their results to our framework. We implement the greedy algorithm in [25] for MDL. The MDL algorithm can not specify the number of clusters, in fact, it generates 1248 clusters on one medium-sized influence graph. For this reason, we only present a visualization example of MDL summarization, and do not include it in performance comparisons.

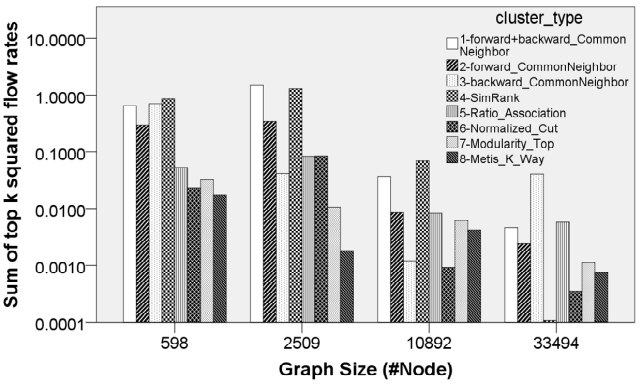
All the experiments are conducted on the same Linux server with two 8-core 2.9GHz Intel Xeon E5-2690 CPU and 384GB of memory. All the LAPACK and ARPACK libs are compiled locally to provide machine-optimized performance. Note that the modularity and Metis implementations are using native-version software package, not guaranteed with multi-threading. The sample data are citation graphs collected from ArnetMiner [32]. Their citation links are reversed to be the influence direction. Five of such maximal influence graphs are extracted from the corresponding seed papers, as summarized in Table 2.

6.1 Performance Evaluation

Figure 6 shows the performance comparison in maximizing the localized IGS objective function in (3). Our approaches, with either CommonNeighbor or SimRank similarity computation, are consistently better than the other four classical graph summarization algorithms. The best of



(a) $k = 10, l = 10$



(b) $k = 20, l = 20$

Figure 6: The performance in maximizing localized IGS objective. The squared flow rate is summed from the top $l = k$ flows. Higher is better.

our method in each case is at least ten times better than the best competitor. Within our own methods, SimRank is generally better than the bidirectional CommonNeighbor. The latter combines the advantages of forward and backward CommonNeighbor method. The only exception happens at the largest influence graph, where the backward CommonNeighbor receives the best performance. Note that as the graph becomes large, the total squared flow rate drops significantly. This is because RWR will split weights among more nodes on large graphs, leading to reduced weight on individual links.

Varying the number of flows and clusters in the summarization does not change the relative performance much. We sum up the top 20 flows under 10 clusters, and receive almost the same result with Figure 6(a). With the finer summarizations with 20 clusters in Figure 6(b), CommonNeighbor methods receive a little performance gain compared with SimRank and the others, though the improvement is not significant.

The overall computation time of these summarization methods is illustrated in Figure 7. Our proposed algorithms are more costly than efficient modularity clustering algorithm ($O(nlog(n))$ with small constant) and Metis k-way graph partition ($O(n + m)$). However, our methods can generate the summarization of a 10000-node graph in 100 seconds, and the overall complexity is only slightly above linear. This is feasible for most citation graphs that can be found in the community.

Within our methods, the SimRank algorithm requires the

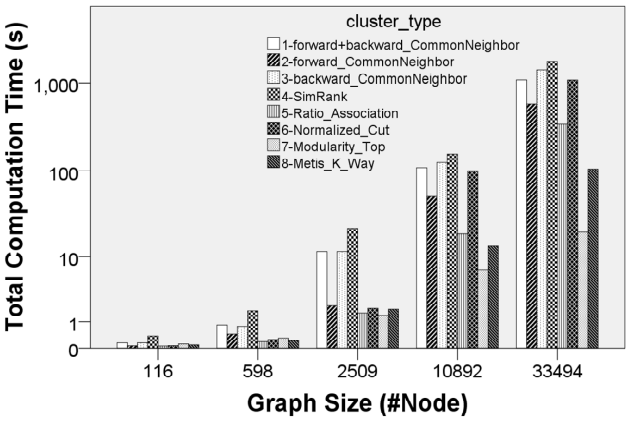
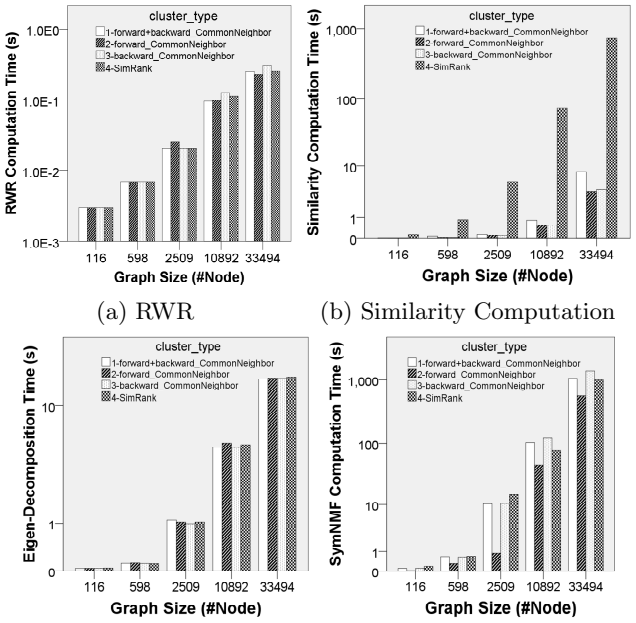


Figure 7: The computation time of different summarization methods, cluster number $k = 20$.

longest computation time. To explain this, we have looked at the split time at four key steps, as shown in Figure 8. RWR and eigen-decomposition (only top k eigen-vectors are computed) are quite fast due to the sparsity of the influence graph matrix (Table 2). On similarity matrix computation, SimRank is very 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. CommonNeighbor is much faster on similarity computation, with the multi-threaded routine on sparse matrix multiplication. Finally, SymNMF is the most costly step in our framework. In each iteration, there are a few matrix-matrix multiplications. Overall, we recommend CommonNeighbor in practice for its good balance between the optimization quality and the computational efficiency.

6.2 Visualization

We also evaluate the effectiveness of our methods by comparing the summarization results in visual forms. We first choose the TC'1988 paper by Lin on stochastic high-level Petri Nets. This paper does not have much direct citations in the data set, but due to its focus on fundamental theory, the total influence expanded by the citation tree reaches 2,508 papers through 1988∼2013. Figure 9 shows seven summarizations by representative methods. Our CommonNeighbor algorithm is able to generate a much cleaner view with little visual clutter (Figure 9(a)). The summarized graph is also comprehensive, with the desired style of detailed patterns near the seed paper along with coarse-grained context farther away. Having a close look at the graph, we find two development lines: one from 30 Mobile/Networking papers, another from 5 miscellaneous papers, both influenced quite a lot software engineering papers. The major contributions of the TC'1988 paper are two-fold, first to advance the theory of Petri-Net and second to build a foundation for performance analysis on wired and wireless networking. Our summarization results are consistent with the main contribution of TC'1998 paper. The 5 miscellaneous papers are all on the theory of performance evaluation and modeling (interdisciplinary topic that can be published in many research fields), and 30 Mobile/Networking papers are mostly on performance analysis of networked systems, potentially applied the theory in TC'1988 paper. After these publications, at around 2000∼2005, the community grows interest on the modeling of software systems as they become more complex. This fact is illustrated by the second stage of the influence graph in Figure 9(a).



(a) RWR (b) Similarity Computation

(c) Eigen-initialization (d) SymNMF

Figure 8: The split computation time of our approaches in four key steps, cluster number $k = 20$. The time cost mainly lies in calculating the similarity matrix and iteratively computing SymNMF.

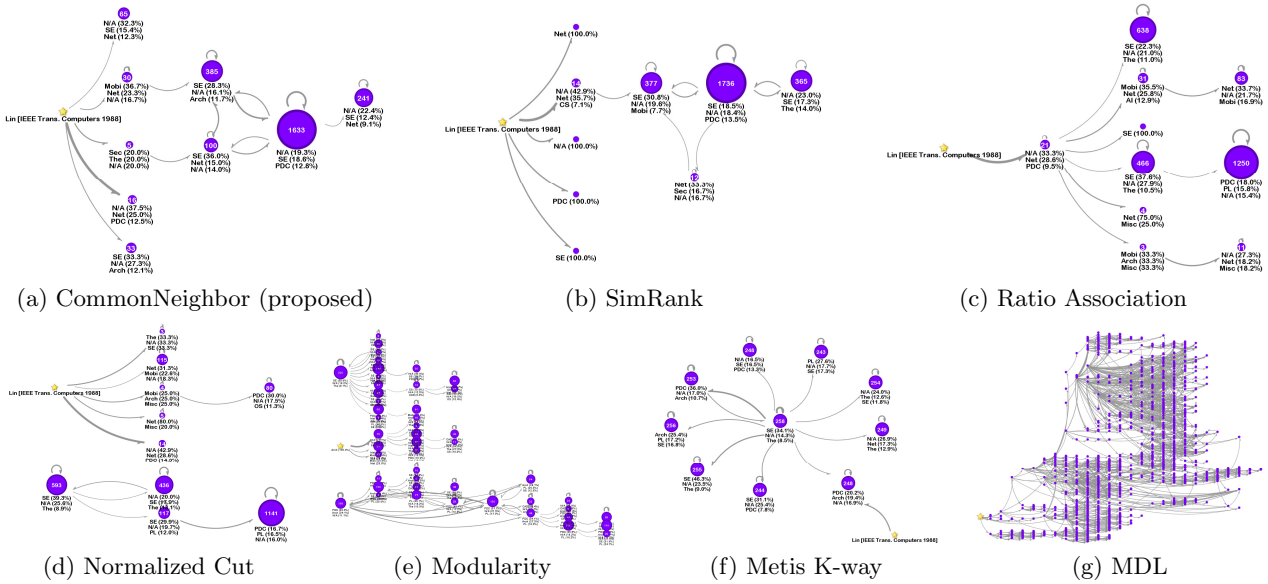
In contrast, the summarizations produced by other methods are much less competitive. Though SimRank has a similar gain on IGS objective function, the result in Figure 9(b) does not capture the two development lines, possibly due to the fragmented flows near the focus; Ratio Association (Figure 9(c)) has similar but worse visual effect, grouping all one-hop nodes together; Normalized Cut and Modularity (Figure 9(d)(e)) fail to reveal the overall influence patterns; Metis k-way partition generates 10 dense clusters, leaving the seed node at the corner (Figure 9(f)); MDL obtains a nice summarization in detail (Figure 9(g)), but can not make it compact for human perception.

In Figure 10, we apply our method on a larger graph influenced by the famous Internet power-law paper in SIGCOMM'1999. We can learn from the summarization that the major direct influence of the paper is on the data mining venues, as indicated by the chain of 6339 and 12841 paper groups, though it is originally published on a networking conference. In the meanwhile, the long-lasting and multi-hop influences, as indicated by the rightmost paper groups in the graph, are more on networking/mobile venues. It is better perceived in Figure 1 (go back to the second page) when we increase the number of clusters to 20, and integrate node attribute (paper venue) similarity matrices for summarization. The text summary is also switched to that by title+abstract. The middle-staged paper groups are mostly on the topic of "sensor (network)" on networking/mobile venues, indicating the hottest topic along this research line.

7. RELATED WORK

We review the related work from three aspects.

First, *graph visualization*, over the past few decades, the



(a) CommonNeighbor (proposed) (b) SimRank (c) Ratio Association

(d) Normalized Cut (e) Modularity (f) Metis K-way (g) MDL

Figure 9: Influence graph summarization results on [Lin TC'1988] by different methods. Node label gives the size and summary by the papers' venue field. The number of clusters is set to 10. Some methods generate 11 clusters to keep the seed paper as a separate cluster. Note that the modularity method stops at 46 clusters and can not merge any further. MDL produces 1,248 clusters, very close to the view of input graph.

methodology to draw node-link graphs has reached its maturity [17]. On graphs with less than a few hundred nodes, the planar graph drawing approach [7] and the force-directed algorithms with spring [20] or spring-electrical model [14] 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 [18] 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 in this paper, the resulting graph layout still has numerous edge crossings. This leads to overwhelming visual clutters detrimental to visual data mining tasks. Some recent literatures try to create graph visualizations over existing summarization results (e.g. graph clustering [4][30]). They improve the viewing experience on large graphs, but neither do they develop any new graph summarization method, nor are they optimized for influence graph visualization.

Second, *graph summarization*, constructing a smaller abstraction to represent the large graph has been a traditional research topic [16]. In the data mining field, it is best known as the graph clustering problem. This problem is usually formalized as optimizing certain association or cut measure during the k-way graph partition. Several measures have been proposed, e.g. ratio association, ratio cut [10] and normalized cut [29]. The general data clustering methods, such as spectral clustering, are shown to be equivalent to graph cut methods [11]. A related topic is community detection topic [15]. Modularity is the most popular quality function to access a community [27]. Based on this function, greedy algorithms have been proposed to merge sub-groups in a agglomerative manner, generating the best communities that can be found [26]. Overall, most of the clustering 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, in data management and web mining field, there is a need to compress large graphs for efficient storage and representation. In [25], the MDL-based compression presents the graph with an aggregated graph 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 less than 3), it performs similarly to a structural equivalence based grouping, leaving large visual clutters unsettled. Another algorithm, SNAP [33], considers the node attributes on graph, but again is not tailored for the influence graph scenario.

Third, considerable work has been conducted for studying the effects of *social influence*. For example, Bakshy et al. [6] conducted randomized controlled trials to identify the effect of social influence on consumer responses to advertising, and Bond et al. [8] used a randomized controlled trial to verify the social influence on political voting behavior. Anagnostopoulos et al. [5] proposed a shuffle test to examine the existence of social influence. Most of the methods focus on qualitatively study the existence of social influence in different networks. Tang et al. [31] presented a Topical Affinity Propagation (TAP) approach to quantify the topic-level social influence in large networks. Domingos and Richardson [13, 28] formally defined influence maximization as an algorithmic problem and prove its NP-hardness. Kempe et al. [22] 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. However, all these aforementioned works focus on the information diffusion process and do not consider the summarization problem. To be specific, while these existing work aim to detect *which* nodes are most influential, the proposed work aims to explain/interprete *what makes them influential* and *how they influence others*.

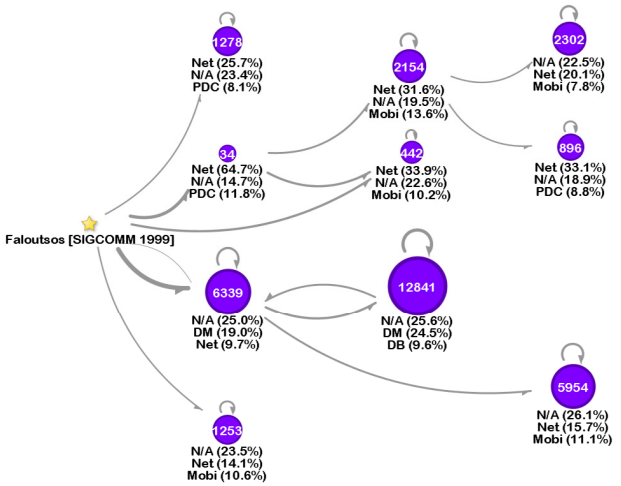


Figure 10: Influence graph summarization on [Faloutsos SIGCOMM'1999] by CommonNeighbor algorithm (backward). Cluster number is set to 10.

8. CONCLUSIONS

In this paper, we study the localized influence graph summarization problem and present a unified framework to solve it. The framework achieves all the three design objectives, including (1) flow rate maximization that highlights the evolvement of influence; (2) a localized view for focused visualization; 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 implementation details and theoretic equivalence analysis. Through empirical evaluations with real-world influence graphs, we demonstrate that our framework outperforms classical methods, such as graph clustering and compression, in both quantitative performance and visual effectiveness.

9. REFERENCES

[1] Intel Math Kernel Library. http://software.intel.com/en-us/intel-mkl/.

[2] Metis - serial graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/.

[3] ParallelColt. https://github.com/danimoth/parallel-colt.

[4] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[5] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *KDD'08*, pages 7–15, 2008.

[6] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn. Social influence in social advertising: evidence from field experiments. In *EC'12*, pages 146–161, 2012.

[7] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[8] 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*, 489:295–298, 2012.

[9] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 2007.

[10] 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*, 13(9):1088–1096, 1994.

[11] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical report, 2004.

[12] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*, 2005.

[13] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD'01*, pages 57–66, 2001.

[14] P. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[15] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[16] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2001.

[17] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6:24–43, 2000.

[18] Y. Hu. Efficient and high quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[19] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *KDD'02*, pages 538–543, 2002.

[20] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[21] G. Karypis, V. Kumar, and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[22] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*, pages 137–146, 2003.

[23] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

[24] D. Kuang, H. Park, and C. H. Q. Ding. Symmetric nonnegative matrix factorization for graph clustering. In *SDM*, pages 106–117, 2012.

[25] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, pages 419–432. ACM, 2008.

[26] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[27] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[28] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD'02*, pages 61–70, 2002.

[29] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[30] L. Shi, N. Cao, S. Liu, W. Qian, L. Tan, G. Wang, J. Sun, and C.-Y. Lin. HiMap: Adaptive visualization of large-scale online social networks. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 41–48, 2009.

[31] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD'09*, pages 807–816, 2009.

[32] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

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

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

[35] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001.