VEGAS: Visual influEnce GrAph Summarization on Citation Networks

Lei Shi, *Member, IEEE,* Hanghang Tong, *Member, IEEE,* Jie Tang, *Senior Member, IEEE,* and Chuang Lin, *Senior Member, IEEE*

Abstract—Visually analyzing citation networks poses challenges to many fields of the data mining research. How can we summarize a large citation graph according to the user's interest? In particular, how can we illustrate the impact of a highly influential paper through the summarization? Can we maintain the sensory node-link graph structure in the representation while revealing the flow-based influence patterns and preserving a fine readability? The state-of-the-art influence maximization algorithms can detect the most influential node in a citation network, but fail to summarize a graph structure to account for this influence. On the other hand, existing graph summarization methods fold large graphs into clustered views, but can not reveal the hidden influence patterns underneath the citation network. In this paper, we first formally define the Influence Graph Summarization problem on citation networks. Second, we propose a matrix decomposition based algorithm pipeline to solve the IGS problem. Our method can not only highlight the flow-based influence patterns, but also easily extend to support the rich attribute information. A prototype system called VEGAS implementing this pipeline is also developed. Third, we present a theoretical analysis on our main algorithm, which is equivalent to the kernel k-mean clustering. It can be proved that the matrix decomposition based algorithm can approximate the objective of the proposed IGS problem. Last, we conduct comprehensive experiments with real-world citation networks to compare the proposed algorithm with classical graph summarization methods. Evaluation results demonstrate that our method significantly outperforms the previous ones in optimizing both the quantitative IGS objective and the quality of the visual summarizations.

Index Terms—influence summarization, visualization, citation network.

1 Introduction

Citation networks are indispensable in understanding modern research activities and have become a fundamental resource for the data mining field to analyze the interplay of researchers, venues and publications (e.g., their communities, research topics and trends, etc.). How to make sense of an individual's influence in the context of the citation network? Specially, how to summarize the underlying citation graph to represent this influence? This is referred to as the Influence Graph Summarization (IGS) problem we aim to address in this paper. Here an individual in the citation network can be a scientific paper, an author or a venue (conference, journal, etc.), also known as the source node in the citation graph. The term of citation graph is used interchangeably with the citation network, both represent a set of individuals (nodes) connected by directed citation links (edges).

While there have been quite a few measures to quantify an individual's influence in research communities, notably the total number of citations and the H-index [1], the graph-based summarization can deliver new insights to better understand the scientific advancements and evolutions. 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? All these questions point to the visual summarization in a similar form to Figure 1 which can characterize the influence flows from the source node over the entire citation graph.

Although closely related, the IGS problem on citation networks bears some subtle differences from the existing work in graph mining and visualization. We briefly review three most relevant topics. First (*graph summarization*), many interesting work has been done in the context of graph clustering and aggregation. These works typically look for coherent regions in the graph by optimizing a pre-defined loss function (e.g., minimizing the inter-cluster connections [2], maximizing the intra-cluster attribute homogeneity [3], minimizing the total description cost [4], etc). Despite their own success, most, if not all, of these graph summarization algorithms tend to group the graph nodes with a direct linkage together, but fails to reveal the influence flows important for the IGS problem. Second (*social graph simplification*), in the scenario of information diffusion over social networks such as Twitter and Facebook, researchers have studied the problem of extracting the most important social paths based on information propagation logs to optimize applications such as viral marketing [5]. This is significantly different from the IGS problem considered here. On citation

- *Lei Shi is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. Email: shil@ios.ac.cn.*
- *Hanghang Tong is with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University. Email: htong6@asu.edu.*
- *Jie Tang is with the Department of Computer Science and Technology, Tsinghua University. Email: jietang@tsinghua.edu.cn.*
- *Chuang Lin is with the Department of Computer Science and Technology, Tsinghua University. Email: chlin@tsinghua.edu.cn.*

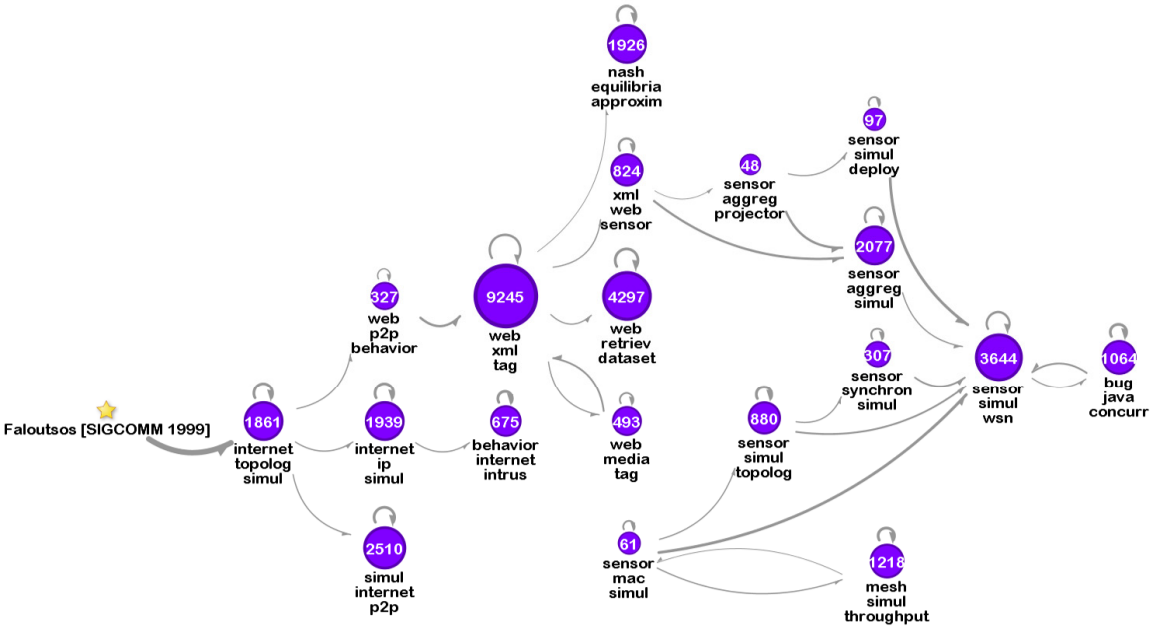


Fig. 1. The visual influence graph summarization on [Faloutsos SIGCOMM'1999] (#Cluster = 20). Paper citation relationship and venue information are integrated. Node label gives the cluster size and summary on paper title+abstract. Link thickness indicates the normalized flow rate.

networks, there is hardly an underlying social network (at least difficult to acquire or predict), over which the influence propagates. A new researcher can cite seminal papers in his field without connecting to the authors in person. In this sense, IGS is more an "unsupervised" summarization problem. Third (*influence maximization*), in the past decades, many elegant algorithms have been proposed for the so-called influence maximization problem [6]. While effective in identifying *who* are the most influential in the network, the question of *what makes them influential* largely remains open. We outline three design objectives that differentiate the IGS problem from existing works.

- *D1. Flow Rate Maximization*. The primary goal of IGS is to summarize the influence flows from a source node in citation graphs. For the effective visualization, it is defined as the objective of maximizing the overall flow rate given the number of flows to display. The consistency within the generated node cluster is not defined by the dense internal connection any more, but rather by the high topological similarity of all the nodes in the cluster. Under this objective, more edges will be cut across clusters than traditional methods, so as to highlight the inter-cluster flows that outline the influence patterns.
- *D2. Localized Summarization*. While a full citation graph can span millions of nodes and prohibit any readable visual summarization, in the IGS objective, we switch to summarize the influence of a single source node. This localized summarization problem is at least as important as the global summarization. Consider a user navigating the citation graph of computer science papers, after an overview of the entire field, most likely she will drill down to a few key papers and examine their influences separately.
- *D3. Rich Information*. The citation graphs have rich node attributes (e.g., the venue, research topic of a paper) and often evolve over time (e.g., the publication date). Incorporating these information to optimize the summarization result poses additional challenges to our work.

We should note that the definition of flows here is quite different from the flow network in graph theory [7], which physically mimics a source-to-destination transportation network. Our flow denotes the primitive form of connections from one group of nodes to another group. On citation graphs, the influence flows are reversed citation link groups. Figure 1 gives a visual summarization over the influence graph of the famous power-law paper presented at SIGCOMM'99. With this summarization, we can track the evolution of research topics, beyond simply detecting the hot topics through traditional graph clusterings.

To solve the IGS problem, we propose an algorithm pipeline, and over which, build a prototype system called VEGAS, to generate *flow-based*, *localized* Visual influEnce GrAph Summarization over large-scale citation networks. The algorithm pipeline is flexible and admits many existing graph mining algorithms for each of its building blocks. Meanwhile, the theoretical analysis shows that our main algorithm, which is equivalent to the kernel k-mean clustering, can approximate the IGS objective on the flow rate maximization with a carefully designed kernel matrix. Finally, we conduct comprehensive empirical evaluations to validate the effectiveness of the proposed algorithm. The main contributions of this paper can be summarized as:

- *Problem Definition*, to fulfill the design objectives for the IGS problem (Section 2);
- *An Algorithm Pipeline*, to solve the IGS problem (Section 3), and the prototype VEGAS system implementing this pipeline (Section 5, Section 7);
- *Theoretical Analysis*, to reveal the intrinsic relationship between the IGS problem and the matrix decomposition based main algorithm (Section 4);
- *Comprehensive Evaluation*, to demonstrate the effectiveness and efficiency of the proposed algorithm (Section 6).

TABLE 1
Notations.

SYMBOL	DESCRIPTION
I	citation 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}	topology adjacency matrix of G and its entries
A^D, a^D_{ij}	node attribute adjacency matrix of G and its entries
S	graph summarization of G
π_c, $\|\pi_c\|$, 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

2 Problem Definition

Table 1 lists the notations used throughout the problem definition. There are two input data in our problem: the influence graph I, which is defined by the underlying citation graph after reversing all the citation links; and the source node f, which can be the paper/author/venue selected by the user or detected by existing influence maximization algorithms. To summarize the influence of f on I, it is enough to consider a maximal influence graph G, which is the induced subgraph of I containing all the nodes reachable from f in I (including f). The maximality here states that G includes all the nodes directly or indirectly influenced by the source node f. Let G have n nodes, denoted by $\{v_i\}_{i=1}^n$. G can be represented by its topology adjacency matrix $A = \{a_{ij}\}_{i,j=1}^n$ in which a_{ij} denotes the link weight, $a_{ij} > 0$ indicates there is a nontrivial link from v_i to v_j.

2.1 Flow Rate Maximization

Before defining the IGS problem, we first introduce two important terminologies.

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

Note that S can be a partial summarization of G, with fewer flows ($l < k^2$) than a complete summarization ($l = k^2$). This is desirable in generating readable influence graph visualizations where a huge number of flows can cause unpleasant visual clutters due to the edge crossing.

Definition 2: The **normalized flow rate** of ξ_s is defined by

$$r(\xi_s) = \frac{\sum_{v_i \in \pi_{c(s)}, v_j \in \pi_{d(s)}} a_{ij}}{\sqrt{|\pi_{c(s)}||\pi_{d(s)}|}} \quad (1)$$

In the remaining paper, we will refer to $r(\cdot)$ as the flow rate for brevity.

Problem 1: The **primitive IGS problem** is defined as finding a graph summarization S of the maximal influence graph G, with k clusters and l flows, to maximize a objective function equaling the sum of flow rates:

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

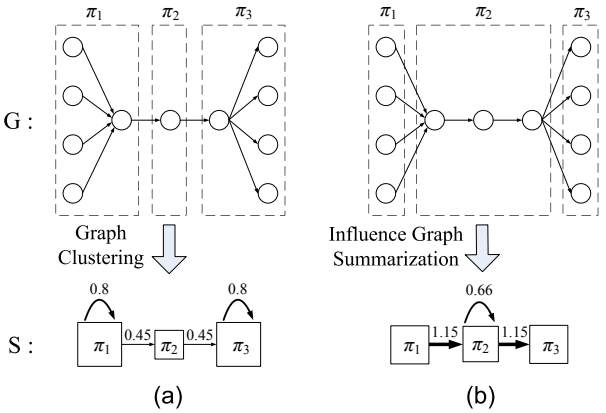


Fig. 2. The difference between the IGS objective and the traditional graph clustering objective. Each dash box in the original graph G becomes a square node (i.e., cluster) in the summarization S. (a) the graph clustering leading to large intra-cluster flows; (b) the influence graph summarization exposing both large intra- and inter-cluster 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 one in the original graph G.

Note that the flow rate defined in equation (1) can not be unnormalized, otherwise the IGS objective in equation (2) will be constant in a complete summarization. The rationale of the equation (2) can be explained in comparison to the objective function of the traditional ratio association graph clustering method, as shown below.

$$\max \quad \sum_{c=1}^{k} \sum_{i,j \in \pi_c} \frac{a_{ij}}{|\pi_c|} = \sum_{c=1}^{k} r(\xi_c) \quad (3)$$

Without loss of generality, ξ_c denotes the intra-cluster flow from π_c to itself, listed as the first k flows by index.

It is clear that the IGS objective in equation (2) is to maximize the sum of flow rates in l largest intra- or inter-cluster flows, corresponding to l densest blocks in the adjacency matrix. On the other hand, the ratio association objective maximizes the sum of flow rates in all the k intra-cluster flows, referring to k diagonal matrix blocks. In other words, the IGS objective is designed to detect a node clustering that maximizes the rate of all the l visible flows in the summarization which fits well the goal to reveal the flow-based influence patterns on the citation graph, while the traditional graph clustering is designed to detect a clustering with the densest internal connections. An illustrative example is given in Figure 2.

Through experiments on the primitive IGS objective, we find that in most cases, it can achieve the desired flow patterns through the summarization. However, in quite a few cases, there appears a redundant graph structure near the source node in the summarization, which is called the fragmented flows. As shown in Figure 3(a) on a small citation graph, two single-node clusters π_2 and π_3 have the same topological position in the graph, but they are not grouped together in the summarization. This effect can be exaggerated in larger graphs where most of the clusters by the primitive IGS are these single-node clusters in one-hop to the source node. We propose a simple yet effective improvement to the primitive IGS objective by applying a square function on each flow rate.

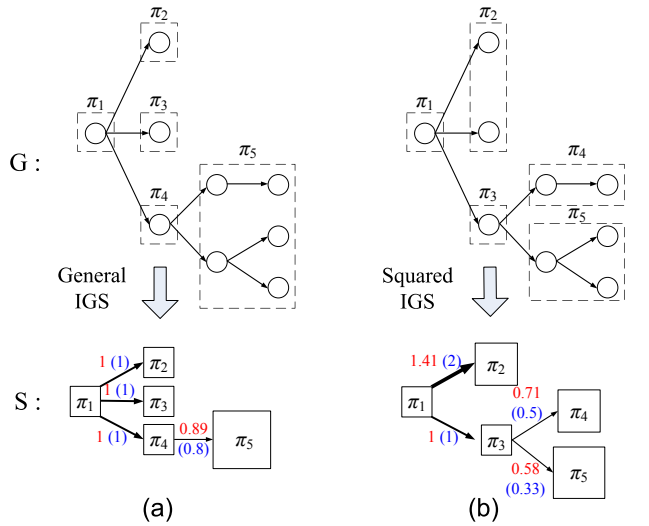


Fig. 3. The influence graph leading to fragmented flows near the source node in the summarization ($k = 5, l = 4$): (a) By the primitive IGS objective, the summarization has two identically-positioned clusters (π_2, π_3), the flow rate by equation (1) is labeled in red, favoring the primitive IGS summarization by a sum of $3.89 > 3.70$; (b) Applying the squared IGS objective, the two identical clusters will be merged and a finer-grained structure of the influence graph is revealed. The squared flow rate by equation (4) is labeled in blue parentheses, favoring the squared IGS summarization by a sum of $3.83 > 3.80$. (best viewed in color)

Problem 2: The **squared IGS problem** is defined as finding a graph summarization S of the maximal influence graph G, with k clusters and l flows, to maximize the objective function:

$$\max \quad \sum_{s=1}^{l} r(\xi_s)^2 \tag{4}$$

From the perspective of highlighting influence flows, the squared IGS objective is consistent with the primitive IGS. Importantly, no extra gain in the objective function is obtained by fragmenting these one-hop nodes apart, as illustrated in Figure 3. Moreover, the squared IGS objective favors large flows more than the primitive IGS objective. In this sense, it is better for the influence graph visualization scenario which has a tightly bounded flow number.

2.2 Incorporating Node Attributes

In the citation network, the network node often carries some additional information, such as the venue and publication date of a scientific paper, the research topic of an author. This information, beyond the network topology, can be critical in many scenarios. For example, the flow of information among papers in the same venue or in the adjacent years can be seen as more consistent when we summarize the internal evolution of certain research field. In contrast, to summarize the interdisciplinary advancement, we can prioritize the flows among papers/authors on different research topics. To serve this need, we propose a variant of the IGS problem by extending the flow rate definition with node attributes.

$$r(\xi_s) = \frac{\sum_{v_i \in \pi_{c(s)}, v_j \in \pi_{d(s)}} a_{ij} a^D_{ij}}{\sqrt{|\pi_{c(s)}||\pi_{d(s)}|}} \tag{5}$$

where $a^D_{ij} \in [0, 1]$ is the entry in the node attribute adjacency matrix of G, denoted by A^D. A^D defines the pairwise similarity between the nodes in G according to their attributes. The other part of the problem definition holds unchanged from Section 2.1.

3 Algorithm

3.1 End-to-End Pipeline

To solve the IGS problem, we propose an end-to-end algorithm pipeline, which decomposes the problem into several building blocks, as illustrated in Figure 4. 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, a few processing components work in parallel to generate several matrices from the graph: the topology similarity matrix M^G, the optional node attribute adjacency matrix A^D and the generalized similarity matrix M^D. The core of the algorithm pipeline is the decomposition of the similarity matrices 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. The optional node attribute adjacency matrices can be incorporated to ensure coherence on node attributes while still optimizing the proposed objective. The requirement of the l flows in the summarization is handled by the link pruning using either the ranking-based filtering algorithm 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.

3.2 Node Summarization by Topology Matrices

Node summarization is the core building block of the algorithm pipeline. First, we compute the topology similarity matrix M^G of the maximal influence graph G by:

$$M^G = \frac{AA^T + A^T A}{2} \tag{6}$$

where A is the adjacency matrix of G. In the context of the citation network, the entry in the M^G for the similarity of two nodes indicates their number of commonly cited and commonly citing nodes (i.e., neighboring nodes in the citation graph). Therefore, we name the main algorithm for the node summarization as the bidirectional CommonNeighbor. Meanwhile, two variants of the algorithm are supported, the forward CommonNeighbor algorithm by $M^G = AA^T$ which only considers the outgoing edges of each node, and the backward CommonNeighbor algorithm by $M^G = A^T A$ which only considers the incoming edges of each node. The bidirectional CommonNeighbor is also referred to as the forward+backword CommonNeighbor.

In the second stage, we propose a matrix decomposition based solution to generate k node clusters from the similarity matrix M^G. The decomposition employs a Symmetric version of the Nonnegative Matrix Factorization (SymNMF [8]) which optimizes:

$$\min_{H \geq 0} ||M^G - HH^T||^2_F \tag{7}$$

where $||\cdot||_F$ denotes the Frobenius norm of the matrix. $H = \{h_{ij}\}$ is a n by k matrix indicating the cluster membership

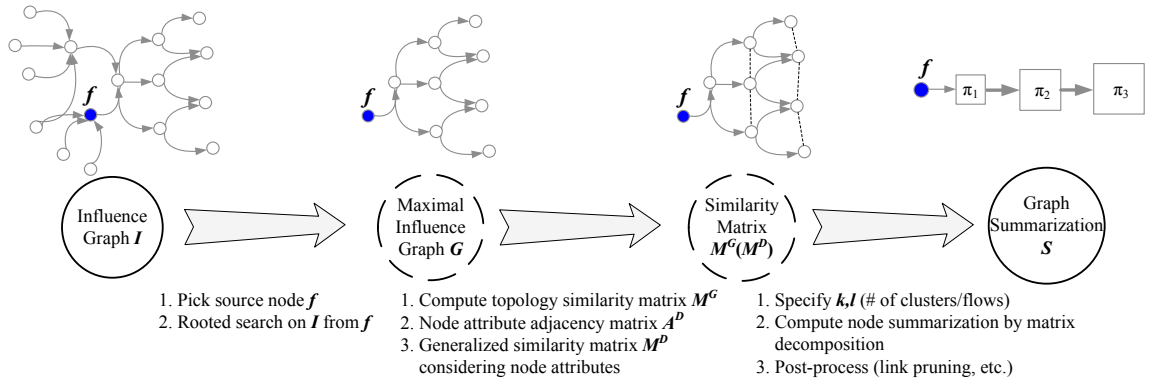


Fig. 4. The algorithm pipeline to solve the IGS problem.

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.

The rationality of our algorithm in optimizing the IGS objective and the details of the matrix decomposition will be discussed in Section 4 and 5, respectively.

3.3 Generalizations

Our algorithm can incorporate node attributes in generating the summarization. To optimize the IGS objective under the attributed definition of equation (5), we first compose the node attribute adjacency matrix A^D. Then the generalized similarity matrix M^D which considers the node attribute information is computed by

$$M^D = \frac{(A \odot A^D)(A \odot A^D)^T + (A \odot A^D)^T(A \odot A^D)}{2} \quad (8)$$

where $\odot$ indicates the Hadamard (by element) product of matrices. The corresponding SymNMF objective becomes

$$\min_{H \geq 0} ||M^D - HH^T||_F^2 \quad (9)$$

The construction of the node attribute adjacency matrix A^D can be customized by users. Here we give typical settings for two common scenarios. In the first scenario, denote the node attribute selected for consistency as D, each node v_i has a nominal value $D(v_i)$ on this attribute. For example, each scientific paper has a venue tag indicating the conference/journal in which the paper publishes. Then the entry of the node attribute adjacency matrix A^D is computed by

$$a_{ij}^D = \begin{cases} 1, & D(v_i) = D(v_j) \\ \lambda, & D(v_i) \neq D(v_j) \end{cases} \quad (10)$$

where $\lambda \in [0, 1)$ controls the degree of penalty for inconsistent node attribute values.

In the second scenario, we consider the node attribute D that has an interval value $D(v_i)$ on each node. For example, each paper has a publication year. The node attribute adjacency matrix is defined by the difference on the attribute values:

$$a_{ij}^D = \beta^{-|D(v_i)-D(v_j)|} \quad (11)$$

where $\beta > 1$ controls the rate of the similarity decay.

4 Equivalence Analysis

In this section, we present a theoretical analysis, to explain the rationality behind our matrix decomposition based algorithm. 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.

4.1 Approximation of IGS problem

Consider the objective function in equation (4), 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 large 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 6. Formally, the approximate objective function becomes:

$$\max \sum_{s=1}^{k^2} r(\xi_s)^2 = \sum_{c,d=1}^{k} \frac{(\sum_{i \in \pi_c, j \in \pi_d} a_{ij})^2}{|\pi_c||\pi_d|} \quad (12)$$

$$\max \sum_{s=1}^{l} r(\xi_s)^2 \qquad given\ \{\pi_c\}_{c=1}^k \quad (13)$$

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

4.2 Kernel K-Mean Clustering

According to [9], the kernel k-mean clustering (KM) is defined as follows. Given n data vectors $\{x_i\}_{i=1}^n$ with a kernel function $\phi(x_i)$, the KM method groups the data vectors into k non-overlapping clusters $\{\pi_c\}_{c=1}^k$ based on the objective function

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

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

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

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

$$\max \quad \sum_{c=1}^{k}\sum_{i,j\in\pi_c} \frac{\phi(x_i)\cdot\phi(x_j)}{|\pi_c|} \tag{14}$$

Introduce the heuristics of the number of bidirectional common neighbors as the similarity measure, we can compute a topology similarity matrix by

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

If we use K as the kernel matrix in the KM clustering and substitute k_{ij} for $\phi(x_i)\cdot\phi(x_j)$, equation (14) becomes

$$\begin{aligned}
\max \quad & \textstyle\sum_{c=1}^{k} \frac{1}{|\pi_c|} \sum_{i,j\in\pi_c} \sum_{t=1}^{n} \frac{a_{it}a_{jt}+a_{ti}a_{tj}}{2} \\
= \quad & \textstyle\sum_{c=1}^{k} \sum_{t=1}^{n} \sum_{i,j\in\pi_c} \frac{a_{it}a_{jt}+a_{ti}a_{tj}}{2|\pi_c|} \\
= \quad & \textstyle\sum_{c=1}^{k} \sum_{t=1}^{n} \frac{(\sum_{i\in\pi_c} a_{it})^2+(\sum_{i\in\pi_c} a_{ti})^2}{2|\pi_c|} \\
= \quad & \textstyle\sum_{c=1}^{k} \sum_{j=1}^{n} \frac{(\sum_{i\in\pi_c} a_{ij})^2+(\sum_{i\in\pi_c} a_{ji})^2}{2|\pi_c|} \\
= \quad & \textstyle\sum_{c,d=1}^{k} \sum_{j\in\pi_d} \frac{(\sum_{i\in\pi_c} a_{ij})^2+(\sum_{i\in\pi_c} a_{ji})^2}{2|\pi_c|}
\end{aligned} \tag{15}$$

4.3 Equivalence

Let us compare the objective functions in equation (12) and equation (15). They are in similar forms if we re-formulate equation (12) into

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

$$where \quad w_{ij}^{IGS} = \frac{\sum_{p\in\pi_c, q\in\pi_d} a_{pq}}{|\pi_c||\pi_d|} \;\; (i\in\pi_c, j\in\pi_d) \tag{16}$$

and re-formulate equation (15) into

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

$$where \quad w_{ij}^{KM} = \frac{\sum_{p\in\pi_c} a_{pj}}{2|\pi_c|} + \frac{\sum_{q\in\pi_d} a_{iq}}{2|\pi_d|} \;\; (i\in\pi_c, j\in\pi_d) \tag{17}$$

Both IGS and KM objectives aim to maximize the weighted sum of the 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 heuristics of the CommonNeighbor-based k-mean clustering is to put the graph nodes with similar in- and out-neighbors 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 the kernel k-mean clustering to the IGS problem.

Furthermore, the kernel k-mean clustering is equivalent to solving the trace maximization problem:

$$\max_{H^TH=I, H\geq 0} Tr(H^TKH)$$

where the kernel matrix K equals the topology similarity matrix M^G computed by the CommonNeighbor algorithm. The trace maximization problem can then be solved by SymNMF under spectral relaxations [8].

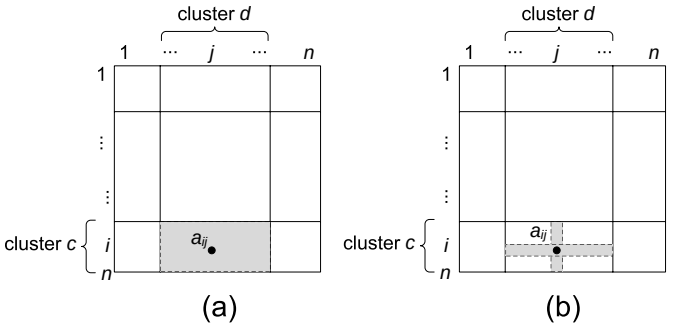


Fig. 5. The weighting schema comparison in two objective functions: (a) influence graph summarization using the entire block; (b) kernel k-mean using the block's column and row.

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
  $S \leftarrow S_0$;
  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

5 Implementation Details

In this section, we provide additional implementation details. As shown in Figure 4, our pipeline involves four kinds of algorithm-driven building blocks. The rooted graph

search follows the standard BFS/DFS implementation. Below we describe details on the similarity matrix computation, node summarization by SymNMF and the link pruning for post-processing of the summarization.

Similarity Matrix Computation. In Section 4, we have shown that using the heuristics of common neighbors to construct the similarity matrix can approximate the objective function of the IGS problem. 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 all the three variants of the algorithm and it is shown by the experiment result in Section 6 that the bidirectional CommonNeighbor algorithm is generally better than the one-directional forward or backward CommonNeighbor algorithm.

Node Summarization by SymNMF. The node summarization is done by applying SymNMF on the similarity matrix $M^G(M^D)$, and using the factorized matrix H for the cluster membership assignment. In our implementation, we apply the iterative SymNMF solver with the multiplicative updating rule in [8] which guarantees convergence.

$$h_{ij} \leftarrow h_{ij}\left(1-\beta+\beta\frac{(M^GH)_{ij}}{(HH^TH)_{ij}}\right) \tag{18}$$

where h_{ij} denotes the entry of H in (i,j)'s cell, β is set to 0.5. The iteration stops when $||M^G - HH^T||_F < \epsilon||M^G||_F$ where $\epsilon = 10^{-7}$. The maximal number of iteration is 500.

With this iterative solver, the initialization of H is critical to the final result. We introduce nonnegative eigenvalue decomposition similar to the method in [10] to compute a good initial factorization. Over this initialization, we compute the cluster assignment of the source node f by its largest entry in H, denoted by $\pi(f)$. The other entries of H that are related to the source node f and the cluster $\pi(f)$ are cleared to zero. Due to the nature of the multiplicative updating, the cluster assignment of the source node is guaranteed to be unchanged and isolated from the other nodes during the iteration.

Link Pruning. The graph summarization by SymNMF needs further post-processings to select l top flows for the final summarization S. According to equation (13), the top flows can be extracted after ranking by the 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 largest flow going to each node cluster in the summarization. An alternative choice is to use the maximum spanning tree (MST) algorithm [11].

We implement the initial VEGAS system backend in Java. The main computation routines are built on the ParallelColt package [12] 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 [13] through JNI invocations.

6 Evaluation

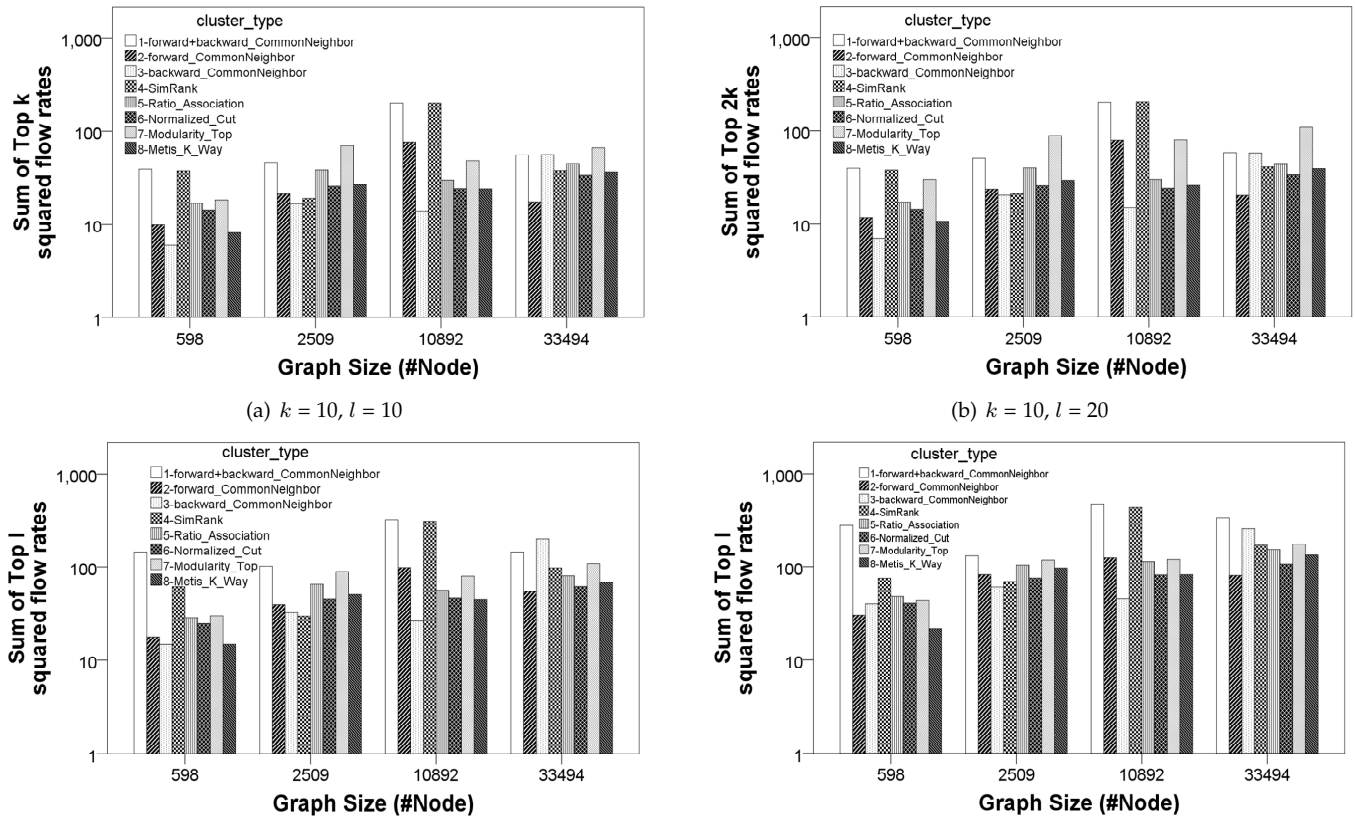
In this section, we evaluate the main algorithm on the node summarization by comparing with alternative graph summarization methods. Nine algorithms are considered: the first three are ours, using *CommonNeighbor* algorithms to compute the similarity matrix (i.e. forward+backward, forward, and backward settings) and then apply SymNMF for the summarization; the fourth uses the *SimRank* algorithm [14] to compute the similarity matrix for SymNMF; the next four are classical graph clustering algorithms with *Ratio Association*, *Normalized Cut* objectives [15], the agglomerative *Modularity*-based graph clustering [16], and the *Metis* K-way graph partition [17]; the last is Minimal Description Length (*MDL*) based graph summarization [4]. Note that Ratio Association and Normalized Cut are implemented using their equivalent similarity matrix computation for SymNMF [18]. Metis partition is implemented by the official open source software package [19]. Modularity clustering is executed agglomeratively until all clusters stop merging at the top level or the number of clusters reaches k, the desired number of clusters. For MDL, we implement the greedy algorithm in [4]. The MDL algorithm can not specify the number of clusters, in fact, it generates 4,937 clusters on one medium-sized influence graph as shown in Figure 9(g). It is known that increasing the number of clusters will raise the overall flow rate, to ensure a fair comparison, we exclude MDL from quantitative comparisons, but still present its visual summarization results. Note that all the approaches in comparison only differ in the graph summarization algorithm, the pre-processing (i.e. generation of the maximal influence graph) and the post-processing (e.g. link pruning) steps are the same.

In the experiment, the parameters for the summarization, namely the number of clusters (k) and the number of flows (l), are configured within a recommended range from the user's perspective. During the study with real users, we find that most of them consider a graph view with less than ten clusters to be less informative, while a graph with more than 20 clusters to be too complex to interpret. It also corresponds with the previous study result [20] that the node-link graph like our design with larger than 20 nodes will start to fall behind another representation by the adjacency matrix, in most graph analysis tasks. On the choice of the flow number, we set the lower bound to the number of clusters ($l = k$). Below this number, the summarization graph will be disconnected, even with the link recovery mechanism, the important flow patterns can be distorted. Meanwhile, the upper bound of the number of flows is set to $l = 2k$, because it is shown in our result that the largest $2k$ flows account for more than 99% flow rates in average defined by equation (4). From user's feedbacks, the extra number of flows also introduces additional overhead to understand the influence graph structure.

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. The raw experiment data are paper citation graphs collected from ArnetMiner [21]. The influence graphs are obtained by reversing the citation links.

6.1 Flow Rate Maximization

We evaluate the performance of these summarization algorithms in optimizing the numeric objectives defined in



(a) $k = 10, l = 10$ (b) $k = 10, l = 20$

(c) $k = 20, l = 20$ (d) $k = 40, l = 40$

Fig. 6. The IGS objective on four sample graphs. The flow rate is summed from all the flows in the (k, l) summarization.

TABLE 2
The initial four citation graphs used in the experiment.

Source paper title	Venue/Year	Node	Link
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

Section 2. Initially, four papers from the ArnetMiner data set are selected as the source node to generate maximal influence graphs, as listed in Table 2. These maximal influence graphs are summarized by each of the above algorithms. The sum of the squared flow rate on each summarization, which is the IGS objective defined in equation (4), is computed for comparison. Figure 6(a)∼(d) present the result of eight summarization algorithms (excluding MDL) under four carefully selected (k, l) settings.

The first group of results in Figure 6(a) compare on a minimal graph summarization ($k = 10, l = 10$). These results indicate that among three CommonNeighbor algorithms, the bidirectional setting almost always achieves the best performance in maximizing the IGS objective (at least $> 100\%$ gain [1]), 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 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.

After we double the number of flows ($k = 10, l = 20$), the sum of flow rates in Figure 6(b) does not increase much on all algorithms (in average $< 15\%$) and the overall comparative patterns stay unchanged. We also increase the number of clusters ($k = 20, l = 20$; $k = 40, l = 40$, beyond the recommended parameter range for performance test only), and the results shown in Figure 6(c)(d) reveal that the objective function increases much as the number of clusters increases (at least $> 30\%$, in average $> 90\%$, comparing Figure 6(c) with Figure 6(b)). The Modularity algorithm is an exception, whose objective function remains unchanged because the number of clusters is already larger than k, so that the flow rates are kept stable. For example, a sample influence graph with 33,494 nodes stops at 71 clusters by the Modularity algorithm. Meanwhile, with the larger number of clusters (Figure 6(c)(d)), the bidirectional CommonNeighbor regains a performance advantage over SimRank and Modularity.

During the experiment, we have executed each algorithm three times and report their best performance. However, the results in Figure 6 still show some randomness due to the nature of the iterative NMF solver. To obtain

1. Percentage of performance gain (drop) by $\frac{new\_number - original\_number}{original\_number} \times 100\%$, the same below.

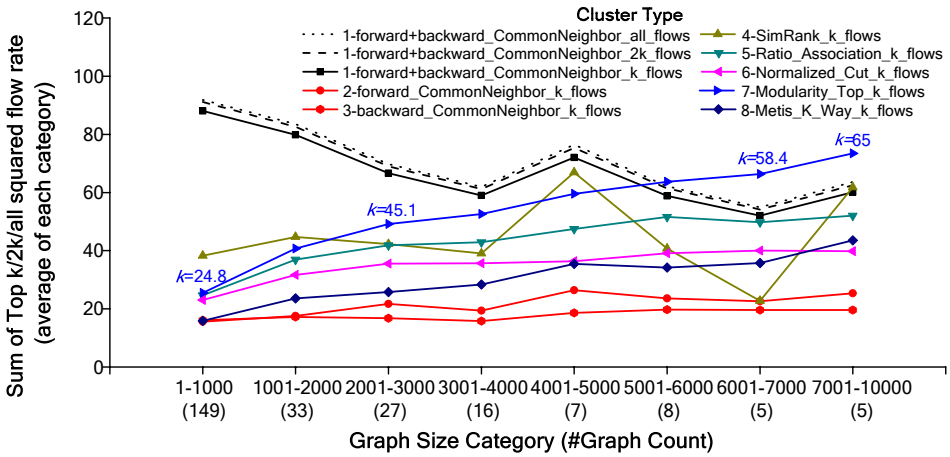


Fig. 7. The IGS objective on 250 citation graphs with the number of nodes ranging from 100 to 10000. The cluster number is set to k = 20.

more accurate results, we sample 250 most-cited papers published in KDD and ICDM from the ArnetMiner data set as the source nodes. The size of their maximal influence graphs are within the range of 100∼10,000 nodes (we have to remove a few largest graphs with more than 10,000 nodes due to performance consideration). On each graph, the same experiment is conducted under a fixed setting of $k = 20$. Finally in Figure 7, we categorize the summarization result on 250 graphs into 8 bins according to their original sizes. The average IGS objective function in each bin is reported for comparison. The results on this larger data set demonstrate the same patterns with those on four sample graphs. In the comparison under the setting of $l = k = 20$ (solid lines), the bidirectional CommonNeighbor in most cases are the best. The Modularity algorithm raises some exception, which performs better as the number of nodes increases beyond 5,000. As mentioned, this is because the Modularity algorithm generates much more clusters than the setting of $k = 20$ in all the other algorithms. As indicated by the labels above the Modularity performance (the blue line and text), the number of clusters by Modularity increases from 24.8 in the first category to 65 among the largest citation graphs. Meanwhile, we also plot in Figure 7 the performance of the CommonNeighbor algorithm in summarizing more flows ($l = 2k$, the dashed line; $l = k^2$, the dotted line). These results confirm that increasing the number of flows in CommonNeighbor does not optimize the objective function much. In average, the top $2k$ flows occupy more than 99% flow rates in the summarization.

We also evaluate the performance of CommonNeighbor algorithms under the generalized flow rate definition when the node attribute information is incorporated, as defined in equation (5). Here we assume a typical setting of penalizing the inconsistent node attribute according to equation (10). We test two CommonNeighor algorithms: one uses the bidirectional setting without the node attribute (the topology similarity matrix is computed by equation (6)), the other uses the generalized bidirectional CommonNeighbor that incorporates the extra node attribute information (the generalized similarity matrix is computed by equation (8)). All the other algorithms do not consider the node attribute by their default settings. The citation graph is that of the pattern mining paper in SIGMOD 2000 (Table 2). The cluster number is set to k = 20. The result on the IGS objective

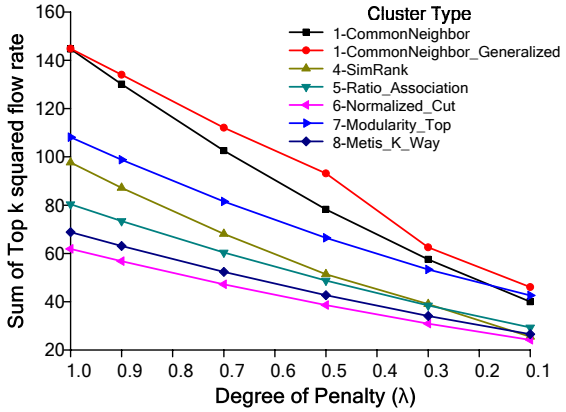
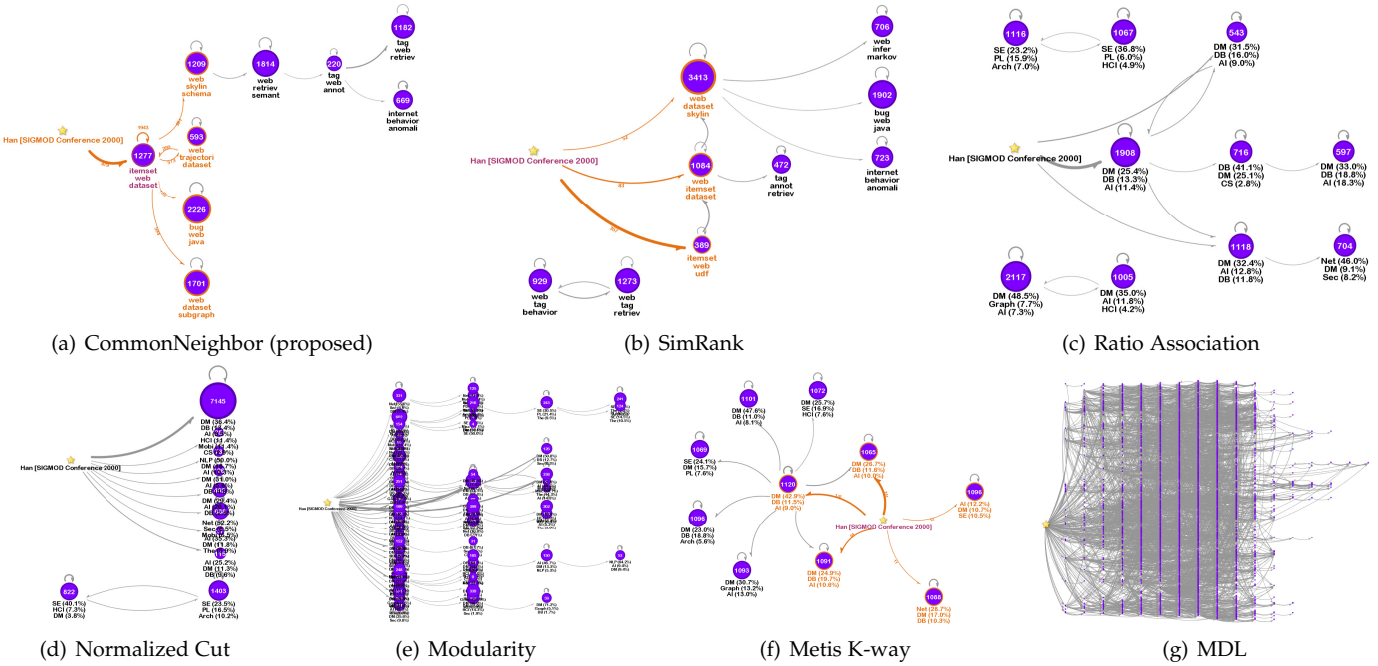


Fig. 8. The IGS objective after incorporating node attributes in the flow rate definition. The node attribute adjacency matrix is set according to equation (10). The cluster number is set to k = 20.

function is illustrated in Figure 8. While the CommonNeighbor algorithm still dominates the other algorithms under all degrees of penalty for the inconsistent node attribute, the generalized version of CommonNeighbor achieves even better performance. The gain is the largest under the medium degree of penalty (∼20% when $\lambda = 0.5$) and small when we apply a huge ($\lambda = 0.1$) or tiny penalty ($\lambda = 0.9$). This can be ascribed to the overestimate and underestimate of the attribute homogeneity in flow-based node clusters, though we admit that the honey spot of λ in our approach may change according to the choice of the citation graph .

6.2 Visualization

We evaluate the effectiveness of summarization algorithms also by comparing their visualization results: whether they produce a clean influence graph summarization with little visual clutter and whether the result is meaningful for domain users. Note that all the methods use the same link-pruning algorithm as in Algorithm 1 ($l = 2k$). We first pick the famous pattern mining paper in SIGMOD 2000 as the source node to generate the maximal influence graph. Then we execute seven typical summarization algorithms and depict their results in Figure 9(a)∼(g). At the first glance, the proposed bidirectional CommonNeighor method generates a connected tree-like influence graph summarization without edge crossing (Figure 9(a)). SimRank produces a similar visual form (Figure 9(b)), corresponding well to the



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

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

Fig. 9. Influence graph summarization results on [Han SIGMOD'2000] by different algorithms (k = 10, l = 20). The 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). The link thickness indicates the 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, almost a half of the visual complexity in the input graph.

IGS objective close to that of CommonNeighbor, but the generated graph is disconnected. The Metis result is also clean (Figure 9(f)), however all the clusters have a similar number of nodes, making the graph summarization almost impossible to deliver the true message. Ratio Association and Normalized Cut look inferior due to the poor graph connectivity (Figure 9(c)) and the flat influence hierarchy (Figure 9(d)). Modularity and MDL are the worst because of the visual clutter generated by the large number of clusters in the summarization (Figure 9(e)(g)).

Taking a closer look at these visual summarizations, we find that by CommonNeighbor, most flows in the summarization represent at least 300 citation links. While by SimRank, the critical flows directly from 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 of the highlighted flows only include 11 and 12 citations. We invite a senior researcher from the database and data mining community to evaluate these summarization results. She mainly compares the visual summarization by CommonNeighbor and SimRank. In this case, she prefers the result by CommonNeighbor in Figure 9(a) because the influence evolutions make more sense. The initial paper quickly raises much attention on the pattern mining research topics such as the itemset and association rule mining, then this thread splits into four streams: the general data management research (such as web and skyline analysis), trajectory analysis, subgraph analysis, and applications in the software engineering (e.g. bug analysis). The thread of web data analysis gradually moves to the topic of web retrieval and finally leads to the tag analysis and the research on the anomaly behavior detection. Compared with CommonNeighbor, SimRank creates some false links, e.g. the direct flow from the source-node paper on frequent pattern mining to skyline analysis (the cluster of 3413 nodes).

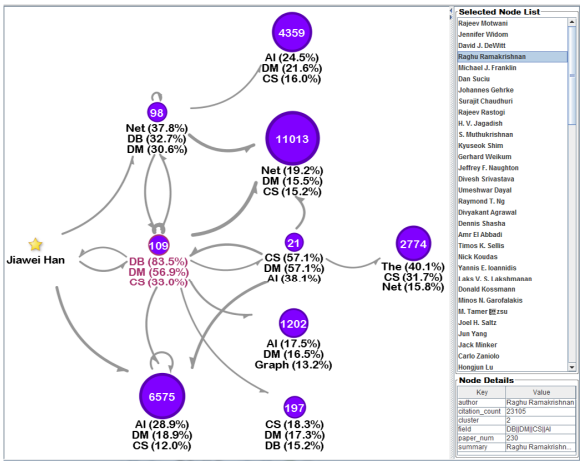


Fig. 10. The summarization of Prof. Jiawei Han's influence graph by the bidirectional CommonNeighbor algorithm ($k = 10$).

Furthermore, we invite another relevant researcher 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 topics together. From this visual summarization, the subject learns that the SIGCOMM paper directly influences the research on Internet topology and simulation. Next, over the Internet topology related 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 networks which corresponds well to his domain knowledge.

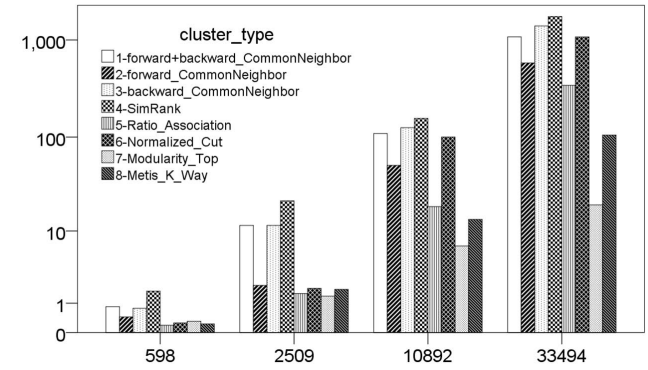
Our algorithm pipeline can also summarize the author's influence on the author citation graph. This is generated

by adding one citation link between two authors for each citation between their papers. The maximal author influence graph is then computed from a source author node by traversing the reversed author citation graph. As an example, we select Prof. Jiawei Han as the source author, and collect all the authors influenced by him within two hops. To limit the size of the influence graph, we only keep productive authors (i.e. ≥ 30 paper publications in the data set) which gives a graph of 26,349 author nodes. The summarization result applying the bidirectional CommonNeighbor algorithm ($k = 10$) is shown in Figure 10. Our invited researcher 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 shown in the right column of Figure 10, a list of 109 authors (e.g. Raghu Ramakrishnan). The most directly influenced field by the number of authors are DM and AI, 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.

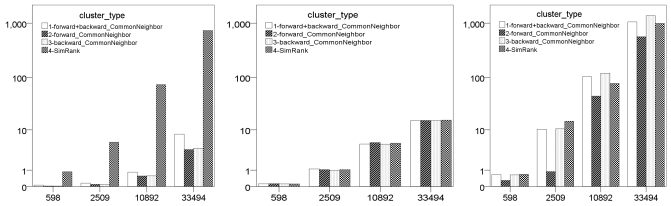
During the case studies with experts from the research community, most of them recognized this kind of visual summarization on citation networks to be quite helpful. The frequent terms they talked about were "clear views" and "new insights", from which aspects they found the visualization to greatly outperform previous methods. On the other hand, they did provide a lot of suggestions to improve the current design, primarily from the user experience of a potential system. First, the users are given static, preconfigured views for the analysis and comparison, only a few interactions are provided. For example, they can not switch between the different size of summarizations (changing k and l). This feature is among the top requests during the study. Second, on the quality of the summarization, quite a lot users would like to have some kind of visual annotation on the flows between paper clusters. What they want to know is why and how some papers influence the other papers. This corresponds to the detailed analysis on the context of each citation. It will be very helpful to also visualize when the influence of one paper goes beyond its original topic and leads to the multidisciplinary research. Part of the expert feedbacks is addressed in a working system - VEGAS, which is explained in Section 7; most other features are under discussion for the future work.

6.3 Scalability

The overall computation time for different summarization algorithms is illustrated in Figure 11(a). Our proposed CommonNeighbor algorithms are more costly than the efficient modularity-based clustering algorithm ($O(nlog(n))$ with small constant) and the 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 moderately above linear. Note that n denotes the size of the maximal influence



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



(b) Similarity matrix (c) Eigen decomposition (d) SymNMF

Fig. 11. The computation time of different summarization algorithms, k = 20: (a) Total time; (b)∼(d) Split time of four CommonNeighbor algorithms in our pipeline. The similarity matrix computation and SymNMF iteration dominate the cost.

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

In the experiment, the SimRank algorithm requires the longest computation time. To explain this, we have looked at the split time at three key steps of the algorithm pipeline, as shown in Figure 11. The eigenvalue decomposition (Figure 11(c), only top k eigenvectors are computed) are quite fast due to the sparsity of the influence graph matrix (Table 2). On the similarity matrix computation (Figure 11(b)), SimRank is the slowest 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 four-hop range. In contrast, CommonNeighbor is much faster on similarity computation, through the multi-threaded routine on sparse matrix multiplication. Finally, SymNMF computation (Figure 11(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 algorithm pipeline is less stringent. The similarity matrix computation and the iterative SymNMF need to store one 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 citation graph.

6.4 Summary

The experiment results demonstrate that most graph summarization algorithms specifying the number of clusters provide compact influence graph summarizations on citation networks. In contrast, typical graph compression and summarization methods such as MDL and Modularity can

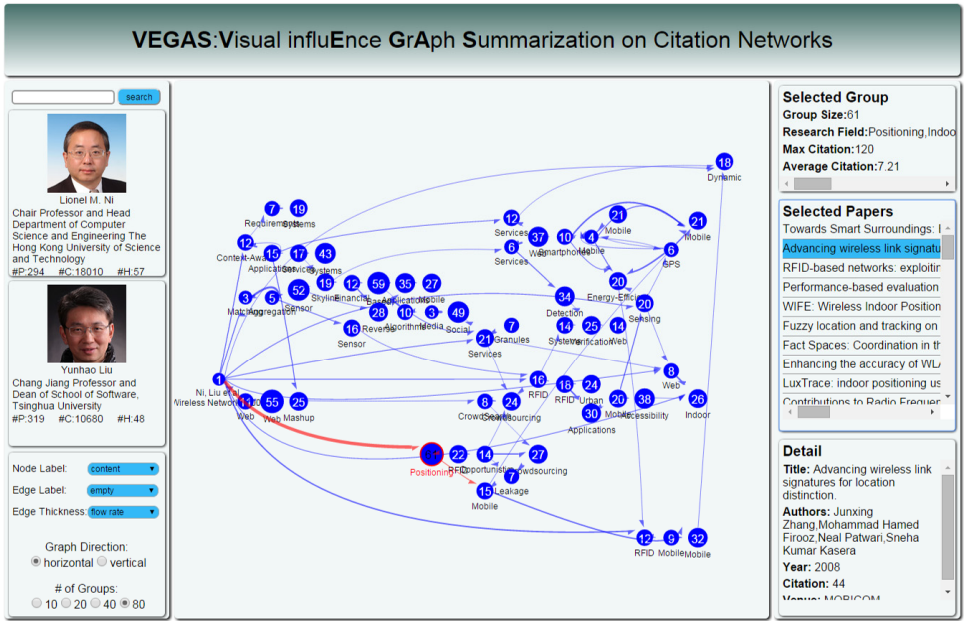


Fig. 12. The online VEGAS system interface on the "LANDMARC" paper.

lead to huge visual clutters that make it hard for people to interpret. Within the k-cluster methods, applying bidirectional CommonNeighbor algorithm 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 algorithm pipeline can achieve a comparable performance. In fact, SimRank has very close tie to our proposed algorithms. CommonNeighbor considers the topological similarity of two nodes within one hop, while SimRank computes their similarity in an infinite hop (pruned to four hops in this work for the performance consideration). Our results show that, though close to, SimRank is no better than CommonNeighbor in maximizing the IGS objective, also it suffers from a much higher computational complexity at $O(n^2d^2)$. Overall, we recommend the bidirectional CommonNeighbor algorithm in practice for its good balance among the visualization result, optimization quality and computational efficiency.

7 Online System Prototype and Discussion

The initial implementation of VEGAS is based on Java and operated as the desktop software in offline (Section 5). To meet the user's expectation of an open interactive system, we are building an online VEGAS prototype in JavaScript at the frontend. In this prototype, The user could start from a search interface on papers and authors, and then directs to the homepage of a particular paper/author in the search result. An example with the "LANDMARC" paper from the wireless networks journal is illustrated in Figure 12. The influence graph summarization is shown in the central panel with the default setting. These settings can be configured by users and updated online in the display. For example, they can switch the node label to the paper title+abstract summary, or the research field summary, to get more context on the graph. They can set the granularity of the summarization by changing the number of clusters. Due to the computational cost, only a few choices are provided now. On the visual summarization, most standard graph interactions are supported, including the zoom&pan, node drag&drop to fine-tune the graph layout, the node hovering for adjacency analysis, and the click-selection to access details on each cluster of papers. An instruction of the system usage is available as the video demo in the supplemental file and also on the first author's website.

For the future work on VEGAS, first we plan to analyze the content on each citation to discover its importance, sentiment and the topic drift. They will help to create more accurate and useful visual influence summarizations. Second, we plan to improve on the interactive exploration of the influence graph visualization, from one seminal paper to certain milestones on the same topic. Last, we plan to transfer the existing technique on the citation networks to study the more sophisticated information diffusion patterns on the social media.

8 Related Work

We review related work from three aspects: graph summarization and visualization, graph simplification, social influence maximization.

First, constructing smaller *summarizations* to represent a large graph has been a traditional research topic, notably using graph clustering and community detection [22] algorithms. These algorithms typically optimize certain association or cut measures, e.g., ratio association, ratio cut [2], normalized cut [15], and the modularity [23]. Most of these methods target at maximizing intra-cluster connections while minimizing inter-cluster connections. This is fairly different from the IGS problem, which is defined as maximizing the overall flow rate. Similarly, the graph summarization methods based on node attributes, such as SNAP [3] [24], ensure the content coherence on clusters, but again they are not tailored for the flow rate maximization objective in the IGS problem.

Meanwhile, there are many works on graph compression for the efficient graph storage and representation. In [4],

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 perspective. While MDL can successfully compress web graphs, on influence graphs that are much sparser, it suffers from a low compression rate and leads to huge visual clutters.

Graph visualization methods, especially those on large graphs, are often tightly combined with graph summarization algorithms. The most relevant works to ours are egocentric network visualizations which only consider the subgraph of nodes that have direct connection to a predefined ego [25]. While our IGS problem is also a localized summarization of the selected source node, the subgraph considered, which is called the maximal influence graph, is induced from all the nodes that have direct or indirect connection (path) to the source node. In this sense, the IGS problem has a much larger scope than the egocentric network visualization. In constructing user-friendly information maps, Shahaf et al. [26] [27] [28] studied the similar problem of summarizing large amount of information. They developed intriguing methods to detect hidden linkage and document clusters from the keyword frequency statistics. On a quite different focus, our work built on graphs with explicit linkage data while the textual content of each node can be absent or incomplete.

Second, *graph simplification* methods are another kind of techniques to generate the clearer view of large graphs. These methods identify important edges in the graph and prune the others to produce the summarization. In [7], the authors proposed two algorithms on directed and undirected flow networks. Through the construction of biconnected components and the dominance trees, the useless edges subject to either the source node or the sink node are located and removed. Though inspiring, these algorithms on flow networks can not be directly applied to the IGS problem that considers the overall flow rate, rather than the flows in the source-sink path. A more general work on weighted graphs defined the connectivity measure of a graph by the average connectivity quality between all pairs of nodes [29]. The authors proposed algorithms to find edges for removal in the objective of minimizing the loss of this connectivity measure. A brute-force approach was shown to achieve the best quality though it is costly in the computation complexity. Two other algorithms were designed for the better trade-off between the simplification quality and computation time.

The similar problem of sparsifying networks exists in the social influence scenario, which involves two types of data, the underlying social graph and the log of information propagation on this graph (e.g., memes and subscriptions). In [5], the authors characterized the information propagation by the independent cascade model [6] and inferred the model parameters by the propagation logs. Given the number of edges, a sparsified subnetwork can be extracted which best describes the observed propagation logs in theory. Though such a problem is shown to be NP-hard to approximate, the authors proposed a greedy algorithm that is both practical in complexity and performs closely to the optimal solution in quality. In another work [30], the scope of the network sparsification problem is extended by assuming no underlying information propagation model. The optimal simplified network are computed by maximizing the coverage of network links on the propagation logs. In contrast to the simplification approaches that try to preserve the overall network structure, another class of literature on the social influence analysis focuses on identifying frequent propagation patterns in either the coherent subnetwork structure [31] or the content-centric information flow [32].

The IGS problem differs from the graph simplification in that we target at building an overview of the large influence graph. Graph simplifications can reduce the number of edges, but can not construct a readable overview because they do not group the graph nodes together. Recently, Mehmood et al. proposed CSI [33], a model that generalizes the independent cascade model to the community level. CSI can produce similar group-level visual summaries to our result. However, the CSI model depends on both the underlying social graph and the information propagation log. In comparison, our method is unsupervised and we do not leverage the information propagation model or the associated log data in their scenario.

Last, considerable work has been conducted for studying the effects of *social influence*. For example, Bakshy et al. [34] conducted randomized controlled trials to identify the effect of social influence on consumer responses to advertising. Bond et al. [35] used a randomized controlled trial to verify the social influence on political voting behavior. Anagnostopoulos et al. [36] proposed a shuffle test to examine the existence of social influence. However, most of the methods focus on qualitatively study the existence of social influence in different networks. Tang et al. [37] presented a Topical Affinity Propagation (TAP) approach to quantify the topic-level social influence in large networks. Domingos and Richardson [38], [39] formally defined influence maximization as an algorithmic problem and prove its NP-hardness. Kempe et al. [6] 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 finding the most influential nodes in a network and do not target the summarization problem studied here.

9 Conclusions

In this paper, we consider the problem of summarizing influences in large citation networks under a flow-based and localized context. We formally define this as an optimization problem, study its linkage to the existing clustering methods, and present an algorithm pipeline as well as the prototype system VEGAS to solve it. VEGAS achieves all the three design objectives, including (1) flow rate maximization that highlights the flow of influence; (2) a localized visual summarization from the source node; and (3) easy to incorporate rich information on graphs such as the node attribute and the time. We describe both the matrix decomposition based main algorithm and the implementation details of VEGAS. Through comprehensive evaluations with real-world citation networks, we demonstrate that the proposed algorithm constantly outperforms classical methods, such as the graph clustering and compression algorithms, in both quantitative performance and qualitative visual effects.

ACKNOWLEDGMENTS

This work is supported by China National 973 project 2014CB340301, Natural Science Foundation of China (No. 61379088, 61222212), National Social Science Foundation of China (No. 13&ZD190).

REFERENCES

[1] J. E. Hirsch, "An index to quantify an individual's scientific research output," *Proceedings of the National academy of Sciences of the United States of America*, vol. 102, no. 46, pp. 16 569–16 572, 2005.
[2] 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.
[3] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization." in *SIGMOD*, 2008, pp. 567–580.
[4] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *SIGMOD*. ACM, 2008, pp. 419–432.
[5] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *KDD*, 2011, pp. 529–537.
[6] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.
[7] E. Misiolek and D. Z. Chen, "Two flow network simplification algorithms," *Information Processing Letters*, vol. 97, no. 5, pp. 197–202, 2006.
[8] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *SDM*, 2005.
[9] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Spectral relaxation for k-means clustering," in *NIPS*, 2001, pp. 1057–1064.
[10] C. Boutsidis and E. Gallopoulos, "SVD based initialization: A head start for nonnegative matrix factorization," *Pattern Recognition*, 2007.
[11] 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.
[12] "Parallelcolt," https://github.com/Danimoth/Parallel-Colt.
[13] "Intel math kernel library," http://software.intel.com/en-us/intel-mkl/.
[14] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *KDD*, 2002, pp. 538–543.
[15] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 1997.
[16] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, 2004.
[17] 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.
[18] D. Kuang, H. Park, and C. Ding, "Symmetric nonnegative matrix factorization for graph clustering," in *SDM*, 2012, pp. 106–117.
[19] "Metis - serial graph partitioning and fill-reducing matrix ordering," http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/.
[20] 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.
[21] 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.
[22] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.
[23] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004.
[24] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization." in *ICDE*, 2010, pp. 880–891.
[25] L. Shi, C. Wang, Z. Wen, H. Qu, C. Lin, and Q. Liao, "1.5D egocentric dynamic network visualization," *to appear in IEEE Transactions on Visualization and Computer Graphics*.
[26] 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.
[27] D. Shahaf, C. Guestrin, and E. Horvitz, "Trains of thought: Generating information maps," in *WWW*, 2012, pp. 899–908.
[28] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," in *KDD*, 2010, pp. 623–632.
[29] F. Zhou, S. Mahler, and H. Toivonen, "Network simplification with minimal loss of connectivity," in *IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 659–668.
[30] F. Bonchi, G. D. F. Morales, A. Gionis, and A. Ukkonen, "Activity preserving graph simplification," *Data Mining and Knowledge Discovery*, vol. 27, no. 3, pp. 321–343, 2013.
[31] L. Macchia, F. Bonchi, F. Gullo, and L. Chiarandini, "Mining summaries of propagations," in *IEEE International Conference on Data Mining (ICDM)*, 2013, pp. 498–507.
[32] K. Subbian, C. Aggarwal, and J. Srivastava, "Content-centric flow mining for influence analysis in social streams," in *CIKM*, 2013, pp. 841–846.
[33] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen, "Csi: Community-level social influence analysis," in *ECML/PKDD*, 2013, pp. 48–63.
[34] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn, "Social influence in social advertising: evidence from field experiments," in *EC*, 2012, pp. 146–161.
[35] 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.
[36] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks," in *KDD'08*, 2008, pp. 7–15.
[37] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD*, 2009, pp. 807–816.
[38] P. Domingos and M. Richardson, "Mining the network value of customers," in *KDD'01*, 2001, pp. 57–66.
[39] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *KDD'02*, 2002, pp. 61–70.

Lei Shi is an associate research professor in the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences. Previously, he was a research staff member and research manager at IBM Research - China, working on information visualization and visual analytics. He holds B.S. (2003), M.S. (2006) and Ph.D. (2008) degrees from the Department of Computer Science and Technology, Tsinghua University. His research interests span Information Visualization, Visual Analytics, Network Science and Networked Systems. He has published more than 50 papers in refereed conferences and journals. He is the recipient of the IBM Research Accomplishment Award on "Visual Analytics" and the VAST Challenge Award twice in 2010 and 2012.

Hanghang Tong is currently an assistant professor as School of Computing, Informatics and Decision Systems Engineering at Arizona State University. Before that, he was an assistant professor at Computer Science Department, City College, City University of New York, a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including best paper award in CIKM 2012, SDM 2008 and ICDM 2006. He has published over 80 referred articles and more than 20 patents. He has served as a program committee member in top data mining, databases and artificial intelligence venues.

Jie Tang is an associate professor at the Department of Computer Science and Technology, Tsinghua University. His main research interests include data mining algorithms and social network theories. He has been visiting scholar at Cornell University, Chinese University of Hong Kong, Hong Kong University of Science and Technology, and Leuven University. He has published over 100 research papers in major international journals and conferences including: KDD, IJCAI, AAAI, ICML, WWW, SIGIR, SIGMOD, ACL, Machine Learning Journal, TKDD, and TKDE.

Chuang Lin is a professor of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is a Honorary Visiting Professor, University of Bradford, UK. He received the Ph.D. degree in Computer Science from the Tsinghua University in 1994. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 400 papers in research journals and IEEE conference proceedings in these areas and has published four books.