TOPIC: TOward Perfect InfluenCe Graph Summarization

Lei Shi*, Sibai Sun†, Yuan Xuan‡, Yue Su*, Hanghang Tong§, Shuai Ma¶ and Yang Chen‡
*SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
†Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China
‡School of Computer Science, Fudan University, Shanghai 200433, China
§School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA
¶SKLSDE, Beihang University, Beijing 100191, China
Email: {shil,suyue}@ios.ac.cn, niasw@pku.edu.cn, {yxuan12,chenyang}@fudan.edu.cn, htong6@asu.edu, mashuai@buaa.edu.cn

***Abstract*—Summarizing large influence graphs is crucial for many graph visualization and mining tasks. Classical graph clustering and compression algorithms focus on summarizing the nodes by their structural-level or attribute-level similarities, but usually are not designed to characterize the flow-level pattern which is the centerpiece of influence graphs. On the other hand, the social influence analysis has been intensively studied, but little is done on the summarization problem without an explicit focus on social networks. Building on the recent study of the Influence Graph Summarization (IGS), this paper presents a new perspective of the underlying flow-based heuristic. It establishes a direct linkage between the optimal summarization and the classic eigenvector centrality of the graph nodes. Such a theoretic linkage has important implications on numerous aspects in the pursuit of a perfect influence graph summarization. In particular, it enables us to develop a suite of algorithms that can: 1) achieve a *near-optimal* IGS objective, 2) support *dynamic* summarizations balancing the IGS objective and the stability of transition in navigating the summarization, and 3) scale to million-node graphs with a *near-linear computational complexity*. Both quantitative experiments on real-world citation networks and the user studies on the task analysis experience demonstrate the effectiveness of the proposed summarization algorithms.**

I. Introduction

Influence, originally indicating the action or fact of flowing in [1], has extended its scope to the cyberspace to describe the online dissemination of ideas and opinions. For example, on the websites such as `Twitter` and `Facebook`, one can post messages, get read and circulated by their friends, and even accessed by strangers. On the other hand, many kinds of influence previously happened offline have been brought online through the digitalization. For example, the research influence (i.e., scientific impact) by paper citations is now recorded in electronic databases and becomes viable for analysis. In many cases, a single dose of influence can be represented by one directed link from a source to a target, e.g., from one paper to the paper citing it, and from the user of a Twitter post to another user forwarding the post. Assembling individual links on the same subject together generates the so-called *influence graph* which provides a holistic picture of the evolution of the subject. In citation analysis, the influence of one scientific paper can be roughly measured by its citation count, yet an open question lies in *why* and *how* the paper is influential. In social network analysis, the popularity of one Twitter post can be estimated by the number of retweets, yet an equally important question is *how* the post becomes popular and *whom* it has influenced. Understanding the underlying influence graph can be the key stepping stone to answer these questions.

Making sense of a graph with tens of nodes can be affordable for most human users. However, the scale of an online influence graph can grow millions of times larger. Facebook has billions of registered users, the scientific knowledge base in the computer science domain [2][3] has recorded tens of millions of papers. Interpreting these ultra-large influence graphs in the full detail is extremely hard for ordinary users, if not impossible at all. In our earlier work [4], we made a first attempt on this problem, i.e., the static Influence Graph Summarization (IGS) in large scale, by formally defining the problem through an effective flow-based heuristic (refer to Section III for a review). In short, the objective of IGS aims to maximize the flow rate among different node groups/clusters, whereas a typical graph clustering objective aims to maximize the intra-cluster connectivity and/or minimize the inter-cluster connectivity. Take Figure 1 as an example, by IGS-based algorithms, large influence flows and paths can be exposed in the summarization (top-right of Figure 1), while by the classic graph clustering, more clusters with dense internal connections are detected (bottom-right of Figure 1).

In this paper, our contribution can be summarized in three dimensions. First (*optimality*), although demonstrating superior empirical performance, the theoretic optimality of the original algorithm in [4] remains unknown. By re-formulating the IGS objective, we derive theoretically near-optimal algorithms to solve the static IGS problem (Section IV). Second (*flexibility*), thanks to the new formulation, we are able to extend the proposed algorithms on the dynamic IGS problem to balance the trade-off between the static IGS objective and the stability of transition upon the user navigation (Section V). Third (*scalability*), the worst-case space and time complexity of the algorithm in [4] is *quadratic*. In contrast, our newly proposed algorithm has a complexity of $O(n \cdot logn)$ (Section VI-B). On all the three technical dimensions, we conducted comparative evaluations on the IGS objective, computational overhead, and the usability for visualization (Section VI). Results demonstrate the effectiveness of the proposed algorithm and the IGS summarization in real-life analysis tasks.

This work was performed when Sibai Sun/Yuan Xuan were SKLCS interns.

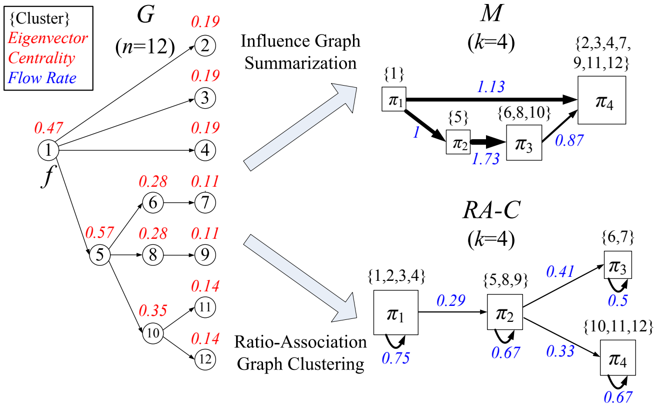


Fig. 1. An illustrative example of the summarization M (top-right) from the maximal influence graph G (left). Large flows are detected in M among clusters, where the flow rate is visually coded as the link thickness and displayed in blue labels. The summarization M is equivalent to the node grouping by their eigenvector centralities in G (labeled in red). A comparative graph clustering C by the ratio association objective is shown in the bottom-right part where only intra-cluster large flows are detected.

II. Related Work

Constructing smaller abstractions to represent the large graph has been a traditional research topic. Most existing methods focus on summarizing the graph nodes through algorithms such as graph clustering [5] and community detection [6]. These algorithms work by looking for coherent/homogeneous node regions in the graph through the optimization of certain loss functions (e.g., minimizing the inter-cluster connection [7], maximizing the intra-cluster attribute homogeneity [8], minimizing the total description cost [9], maximizing the community modularity [10], etc). Among all these methods, the thread on the graph compression is the most relevant. In general, graph compression algorithms try to detect and reduce redundancies in the graph structure. The pioneering work dates back to the notion of structural equivalence [11], which explicitly groups nodes having the same neighbors together. In a probabilistic version, a node similarity score can be computed by the SimRank algorithm [12], which is further used in the structure-based node grouping [13]. In another MDL-based graph compression [9], the graph is represented by an aggregated structure plus an error correction list. This has been proved to be the best summary under the information-theoretic objective. In the SNAP algorithm [8], user-selected node attributes are incorporated in the graph summarization. Navigation interactions are supported to control the resolution of the summarization and provide drill-down and roll-up operations. This is very similar to the dynamic summarization scenario in our work. An extension of the SNAP algorithm proposed the discovery-driven graph summarization by automatically categorizing numerical attributes and recommending the most interesting summarization to the user [14]. On the other hand, Batson et al. proposed the spectral sparsification on weighted graphs [15], which can be leveraged in our approach to select useful edges after the summarization. However, on large influence graphs, the sparsification method alone can not reduce the visual complexity into a user-interpretable level because it does not reduce the number of nodes.

While existing summarization methods can effectively abstract graph structures for storage, computation and representation, they mostly base on the notion to summarize the graph nodes. On the influence graph analysis scenario, the flow pattern among the node clusters is at least as important as the node coherence inside the cluster. In the literature, the influence analysis on graphs and networks has been focused on the social influence, e.g., the information diffusion over social networks. For example, Bond et al. [16] used a randomized trial to verify the social influence on political voting behavior. Tang et al. [17] presented a Topical Affinity Propagation (TAP) approach to quantify the topic-level social influence in large networks. Kempe et al. [18] proposed to use a submodular function to formalize the influence maximization problem and develop a greedy algorithm to solve the problem with provable approximation guarantee. Most of these works focus on the existence of social influence or the nature of the information diffusion process and do not consider the summarization problem. Recently, Mathioudakis et al. proposed the method of influence network sparsification [19], which extracts the most important social paths based on information propagation logs. Mehmood et al. proposed CSI [20], a model that generalizes the classical Independent Cascade model to the community level. Both the influence network sparsification and the community-level cascading model can produce similar visual forms to our work. However, their results are computed from the information propagation log customized to the social influence scenario. In comparison, we focus more on the unsupervised summarization of large influence graphs and we do not leverage the domain-specific information propagation model and the associated log data.

TABLE I. IGS Notations.

SYMBOLS	DEFINITION
I	raw influence graph
f	source node initiating the influence
$G(f)$	maximal influence graph of f in I
n, v_i	# of nodes and the ith node in G
A, a_{ij}	G's topology adjacency matrix and the entry in the ith row and jth column
$M(f)$	influence graph summarization of $G(f)$
k, π_i, $\lvert\pi_i\rvert$	# of clusters in M, the ith cluster and its cluster size
l, ξ_i, $r(\xi_i)$	# of flows in M, the ith flow and its flow rate
$C(v_i)$, $S(\xi_i)$, $T(\xi_i)$	v_i's cluster index, ξ_i's source and target cluster index

III. Preliminaries

In this section, we provide 1) a concise review of the influence graph summarization (IGS) problem for the sake of the completeness of the paper, 2) highlight the difference between IGS and graph clustering problems, and 3) summarize the open challenges this paper aims to address.

A. Key Notations and Concepts

Table I lists the notations in the IGS definition. There are two external inputs: 1) the raw influence graph I, e.g., the graph by reversing the citation relationship among scientific papers, and 2) the source node f on which we want to study its influence, e.g., the seminal paper or key researcher who has a good impact on the research community. Over the input graph I, we consider the maximal influence graph $G(f)$, which is I's induced subgraph composed of all the nodes reachable from f (including f) and their internal links. $G(f)$ will be denoted as G in short. The maximality means that G covers all the nodes that f can potentially influence. An example of the maximal influence graph is shown in Figure 1 (Left). Let G have n nodes, $v_1, \cdots, v_n$. G's topology can be represented by the adjacency matrix $A = \{a_{ij}\}_{i,j=1}^{n}$ where a_{ij} is the link weight from v_i to v_j, $a_{ij} > 0$ indicates a nontrivial link.

The IGS problem is to compute a *summarization* $M(f)$ over the influence graph $G(f)$, denoted by M in short. As shown in the top-right part of Figure 1, M contains k disjoint and exhaustive node clusters of G, denoted by $\pi_1, \cdots, \pi_k$. The cluster size is denoted by $|\pi_1|, \cdots, |\pi_k|$, and the clustering assignment is defined by $C(v_i)$, the cluster index of v_i in M. Let M contain l flows, which are the links among all the k nodes of M (i.e., clusters in G). These flows are denoted by $\xi_1, \cdots, \xi_l$, and the source and target cluster indices of a flow ξ are defined by $S(\xi)$ and $T(\xi)$. The flow ξ represents the collection of all links in G from the nodes in cluster $\pi_{S(\xi)}$ to the nodes in cluster $\pi_{T(\xi)}$. The *flow rate* of ξ is defined by the size-normalized aggregation of relevant link weights:

$$r(\xi) = \frac{\sum_{v_i \in \pi_{S(\xi)}, v_j \in \pi_{T(\xi)}} a_{ij}}{\sqrt{|\pi_{S(\xi)}||\pi_{T(\xi)}|}} \qquad (1)$$

For example, the flow rate is given as blue labels in Figure 1.

B. IGS Problem

PROBLEM 1: INFLUENCE GRAPH SUMMARIZATION

Given: 1) *A, the topology adjacency matrix (representing the maximal influence graph G from the source node f)*; 2) *k* $(k \leq n)$*, the number of clusters in the summarization*; 3) *l* $(l \leq k^2)$*, the number of flows in the summarization*;

Compute: 1) $C(v_i)$ $(i = 1, \cdots, n)$*, the cluster index of all nodes in G*; 2) ξ_i $(i = 1, \cdots, l)$*, the l flows kept in the summarization*;

By optimizing *the flow rate maximization objective*:

$$\max \quad \sum_{i=1}^{l} r(\xi_i) \qquad (2)$$

In our following study, we consider the full IGS summarization problem $(l = k^2)$. The variation on the partial summarization $(l < k^2)$ can be solved by filtering flows additionally.

C. IGS vs. Graph Clustering

We explain briefly the rationale of the flow rate maximization objective. In the influence summarization scenario, the user's central interest is on detecting the flows among the clusters (e.g, the influence of one paper to several communities), instead of focusing on grouping nodes into communities. Mathematically, the proposed IGS objective, though closely related, bears some fundamental difference from the classic graph clustering objective. The objective in Equation (2) for the full IGS summarization can be expanded to reveal the linkage between IGS and classic graph clustering problems.

$$\sum_{i=1}^{k^2} r(\xi_j) = \sum_{i=1}^{k} r(\hat{\xi}_i) + \sum_{i=1}^{k^2-k} r(\tilde{\xi}_i)$$
$$s.t. \;\; S(\hat{\xi}_i) = T(\hat{\xi}_i), \;\; S(\tilde{\xi}_i) \neq T(\tilde{\xi}_i) \qquad (3)$$

The first term by $\hat{\xi}_i$ is the sum of all intra-cluster flow rates, which is exactly the objective of ratio-association graph clustering. The second term by $\tilde{\xi}_i$ considers the flow rate of all inter-cluster flows, which corresponds to the influence flow among clusters. In this sense, the IGS problem balances between the classic graph clustering objective and the unique requirement to highlight the pattern of large flows.

D. Open Challenges

In a pre-work of this paper [4], we have proposed a matrix decomposition based method to compute the influence graph summarization. This method also supports incorporating rich graph attributes. By aligning with the kernel k-mean clustering, it is proved that the decomposition based method can approximate the IGS objective under certain conditions.

Despite the success of the existing IGS algorithm, there remains several open challenges that prohibit its large-scale deployment for the everyday analytic task of end users. First, it is still unknown when the optimality of the IGS objective can be achieved and whether the heuristic IGS algorithm can provide performance guarantees in approaching the objective. Second, from the end user's perspective, the visual summarization can not stay with one static view. Smooth navigations should be supported to analyze the summarization in multiple granularities, which is related to a fresh-new dynamic IGS problem. Third, the complexity of the decomposition based algorithm constantly exceeds $O(n^2)$ time, where n is the number of nodes in the input graph. It takes a few hours to process the largest influence graph on the citation network data set, while the real-world deployment will need to summarize tens of thousands of such graphs.

In this paper, we reinforce the theoretical foundation of the flow-based heuristics by presenting a new analysis of the IGS objective. Our analysis demonstrates that the IGS optimality has a direct linkage to the eigenvector node centrality on influence graphs. As shown in Figure 1 (Left), the red label above each node gives its eigenvector centrality in G. If we group all the nodes by their similarity on this centrality, the clustering output is exactly the same with the one by the IGS algorithm (Figure 1, top-right). We will show in Section IV this effect is not coincidental, certain equivalence exists between the flow rate maximization and the eigenvector centrality based node clustering. By definition, the eigenvector centrality is a measure of the influence of a node in the graph, and one node will receive a high centrality if it connects to other high influential nodes in the graph [21]. Our findings indicate that to maximize the total flow rate, high-influential nodes should be placed in smaller clusters and therefore are more prominent in the summarization, while low-influential nodes should be aggregated into large clusters showing little details. This again conforms well to the goal to make large influence flows visible in the summarization.

IV. OPTIMAL IGS

In this section, we present a new theoretical analysis to derive the optimal condition for the IGS objective. Though this condition is shown to be infeasible due to the discreteness of the node clustering, a near-optimal solution has been found to offer lower-bound guarantees. Based on this solution, both an exact dynamic programming algorithm and two fast heuristic algorithms are proposed.

A. Analysis of IGS Objective and Optimality

Consider the IGS objective in Equation (2), it can be optimized by a divide-and-conquer approach in two steps: 1) maximize the objective for $l = k^2$, i.e., the full summarization; 2) select $l < k^2$ largest flows out of all the k^2 flows. It is shown in Ref. [4] that, the top $2k$ flows in the summarization can occupy more than 99% of the overall flow rate, so that the divide-and-conquer approach is effective if only l is large.

As below, we start from the expanded form of Equation (2) under $l = k^2$ by replacing the flow rate with Equation (1).

$$\max \quad \sum_{s=1}^{k^2} r(\xi_s) = \sum_{s=1}^{k^2} \frac{\sum_{v_i \in \pi_{S(\xi_s)}, v_j \in \pi_{T(\xi_s)}} a_{ij}}{\sqrt{|\pi_{S(\xi_s)}||\pi_{T(\xi_s)}|}} \tag{4}$$

Because the node clustering is disjoint and exhaustive, Equation (4) is actually the weighted sum of all the entries in A.

$$\max \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{|\pi_{C(v_i)}||\pi_{C(v_j)}|}} \tag{5}$$

It is important to notice that if we define a normalized cluster size vector x of length n by $x = (|\pi_{C(v_1)}|^{-\frac{1}{2}}, \cdots, |\pi_{C(v_n)}|^{-\frac{1}{2}})^T$, the IGS objective in Equation (5) can be written as the quadratic form:

$$\max \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j = x^T A x = x^T A^* x \tag{6}$$

where $A^* = \frac{A+A^T}{2}$ is the symmetric version of the adjacency matrix. By the min-max theorem [22], we have

$$\max \quad \frac{x^T A^* x}{x^T x} = \lambda_1$$

where λ_1 is the largest eigenvalue of A^* (a real symmetric matrix). Because $x^T x$ is constant by

$$x^T x = \sum_{i=1}^{n} |\pi_{C(v_i)}|^{-1} = \sum_{j=1}^{k} |\pi_j|^{-1} \cdot |\pi_j| = k$$

We finally obtain

$$\max \quad x^T A^* x = k \cdot \lambda_1 \tag{7}$$

The optimality is achieved when $x = \sqrt{k} q_1$, where q_1 denotes the largest eigenvector of A^* after normalized to a length of one. The optimal x is then parallel to this eigenvector. Note that λ_1 and all the components in q_1 are guaranteed to be nonnegative by the Perron-Frobenius theorem [22].

Though the optimal IGS condition has been derived, we further show that this condition is infeasible in practice due to the discreteness of the node clustering. By the optimal summarization, we should have $x_i = |\pi_{C(v_i)}|^{-\frac{1}{2}} = \sqrt{k} q_{1i}$ where q_{1i} is the ith component of q_1. This leads to $|\pi_{C(v_i)}| = k^{-1} q_{1i}^{-2}$, which means that the optimal cluster size of each node v_i can be computed in close form. In the final clustering, because $C(v_i) = C(v_j) \Rightarrow |\pi_{C(v_i)}| = |\pi_{C(v_j)}|$ where C denotes the optimal clustering. We should have $C(v_i) = C(v_j) \Rightarrow q_{1i} = q_{1j}$. This is almost impossible for the vector q_1 with continuous value on all components.

Below we present an analysis to approach the optimality by maximizing the largest component of the objective. We start from the eigen-decomposition of the real symmetric matrix A^* by $A^* = Q \Lambda Q^T$, where Λ is the diagonal matrix composed by $diag(\lambda_1, \cdots, \lambda_n)$. λ_i is the ith largest eigenvalue of A^*, and $Q = (q_1, \cdots, q_n)$ is an orthogonal matrix having $Q^T Q = Q Q^T = I$ where q_i is the ith eigenvector of A^*. Substitute the decomposed form for A^* in Equation (6) and define a new vector $z = Q^T x$ which is a linear transformation of the vector x by the matrix Q, the IGS objective is reduced to:

$$\max \quad x^T A^* x = x^T Q \Lambda Q^T x = z^T \Lambda z = \sum_{i=1}^{n} \lambda_i z_i^2 \tag{8}$$

Recall $x^T x = k$, we obtain the constraint for (8) as:

$$z^T z = x^T Q Q^T x = x^T x = k = \sum_{i=1}^{n} z_i^2 \tag{9}$$

Consider this constraint optimization problem, because $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ where λ_1 is ensured to be nonnegative, we can approach the optimality of the IGS objective by only maximizing z_1. The resulting objective is guaranteed to have a lower bound compared to the maximal IGS objective of $k \cdot \lambda_1$.

Theorem 1 (Performance Guarantee): Define the cluster size vector to maximize z_1 by x^*. For any other feasible cluster size vector x, the following performance lower bound holds for x^*.

$$\frac{{x^*}^T A^* x^*}{x^T A^* x} \geq \frac{(\max z_1)^2}{k} + [1 - \frac{(\max z_1)^2}{k}] \cdot \frac{\min_{i=2}^{n} \lambda_i}{\lambda_1} \tag{10}$$

Proof: See Appendix A[1].

B. k-Segmentation Problem

Now the IGS problem is converted into maximizing z_1, which can be decomposed into

$$\max z_1 = q_1^T x = \sum_{i=1}^{n} q_{1i} |\pi_{C(v_i)}|^{-\frac{1}{2}} = \sum_{j=1}^{k} |\pi_j|^{-\frac{1}{2}} \sum_{C(v_i)=j} q_{1i} \tag{11}$$

We will show in this part that Equation (11) is equivalent to a k-segmentation problem on the components of q_1. Without loss of generality, we assume vector q_1 as sorted, i.e., $q_{11} \geq \cdots \geq q_{1n} \geq 0$. Then the cluster order lemma holds.

Lemma 1 (Cluster Order): There exists a clustering that maximizes z_1 and satisfies the below inequality.

$$i \leq i' \Rightarrow q_{1i} \geq q_{1i'} \Rightarrow |\pi_{C(v_i)}| \leq |\pi_{C(v_{i'})}| \tag{12}$$

Proof: See Appendix B.

The cluster order lemma implies that the node with a smaller index after the descent sorting (i.e., a larger eigenvalue centrality by the component on q_1) should be placed in the node cluster with a smaller size.

Theorem 2 (k-Segmentation): There exists a clustering that maximizes z_1 and forms a k-segmentation over the full node index list (i.e., [1,...,n]). The k-segmentation means that the

[1] The proof of all the lemmas and theorems are available in the Appendix of http://lcs.ios.ac.cn/%7eshil/paper/ICDE16Long.pdf

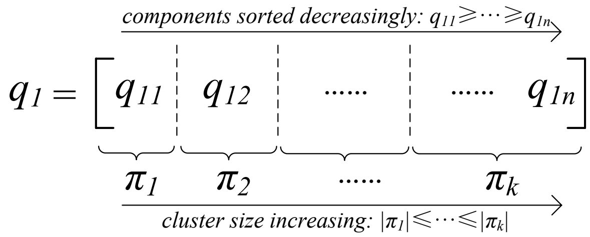

components sorted decreasingly: $q_{11} \geqslant \cdots \geqslant q_{1n}$
$q_1 =$
q_{11}
q_{12}
......
......
q_{1n}
π_1
π_2
......
π_k
cluster size increasing: $|\pi_1| \leqslant \cdots \leqslant |\pi_k|$

Fig. 2. An illustration of the optimal node clustering strategy. q_1, the largest eigenvector of A^* is segmented into k continuous intervals.

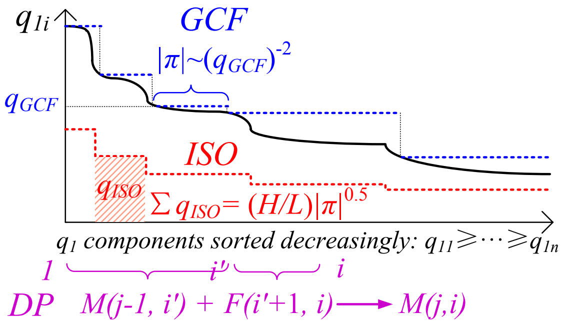


Fig. 3. Example of the proposed algorithms for the k-segmentation.

node indices of each cluster must fill a continuous index interval. Mathematically, the below inequality holds:

$$I_{min}(1) \leq I_{max}(1) < I_{min}(2) \leq I_{max}(2) < \cdots \leq I_{max}(k) \tag{13}$$

where $I_{min}(j)$ ($I_{max}(j)$) denote the minimal (maximal) node index of cluster j. The clusters are indexed incrementally by their minimal node index $I_{min}(j)$.

Proof: See Appendix C.

By Theorem 2, the optimal node clustering is equivalent to a k-segmentation of the ordered component list on q_1, as depicted in Figure 2. An extended analysis is conducted in Appendix D which shows that the best k-segmentation is achieved when both between-cluster and within-cluster variances on the components of q_1 are minimized.

C. Algorithms

To solve the k-segmentation problem, three algorithms are proposed by taking different level of results in our analysis.

Dynamic Programming (DP). Examine the objective in Equation (11). If we define the gain from a cluster $\pi_j = [s_j, s_{j+1})$ by $F(\pi_j) = |\pi_j|^{-\frac{1}{2}} \sum_{C(v_i)=j} q_{1i} = F(s_j, s_{j+1})$, the objective is written as the sum of gain from all clusters. The segmentation problem can be solved by the standard dynamic programming algorithm [23] under the following equation.

$$M(j,i) = \max_{1 \leq i' \leq i-1} [M(j-1, i') + F(i'+1, i)] \tag{14}$$

where $M(j,i)$ denotes the maximal objective achieved in segmenting the first i eigenvector components into j segments. DP algorithm can reach the optimality of z_1, but the complexity is as high as $O(n^2 k)$, which inhibits its usage in large scale.

Greedy Curve Fitting (GCF). Recall Equation (7), the optimal cluster size vector x should be in parallel with the largest eigenvector of A^*. The GCF algorithm combines this condition with the k-segmentation requirement and manage to draw the closest line of x to the target eigenvector. An example is illustrated as the blue dashed lines in Figure 3. The algorithm starts by adding a horizontal line from the highest eigenvector component to represent the first cluster. The cluster size is computed in a greedy manner by $|\pi| = (Len \cdot q)^{-2}$ to meet the parallelism, where Len is the parameter to tune the curve fitting. The following lines are drawn in a similar way until all nodes are clustered. After each iteration, the total number of clusters is computed as k'. The variation of k' from the desired cluster number k is used to update the parameter Len in the next iteration, until k clusters are obtained in the GCF output. Algorithm 1 gives the algorithm description.

Algorithm 1: Greedy Curve Fitting $\sim$ **GCF**(A, n, k).

```
Input  : $A$ (adjacency matrix), $n, k$ (# of nodes/clusters)
Output: $\Pi = \{|\pi_i|\}_{i=1}^{k}$ (size of each cluster)
$q_1 \leftarrow largest\_eigenv(\frac{A+A^T}{2})$, sort $\{q_{1i}\}_{i=1}^{n}$ dec.
$Len \leftarrow \sqrt{k}$, $k' \leftarrow 0$, $best\_r \leftarrow k$                // init
while $k' \neq k$ do
    /* outer iteration to adjust clusters */
    $s_1 \leftarrow 1$, $s_{k+1} \leftarrow n+1$       // separators, $\pi_j = [s_j, s_{j+1})$
    for $j \leftarrow 1$ to $k$ do
        /* inner iteration to fit the size of cluster $j$ */
        $\varphi_j \leftarrow \min(\lfloor (Len \cdot q_{1s_j})^{-2} \rfloor, n+1-s_j)$
        $s_{j+1} \leftarrow s_j + \varphi_j$
        if $s_{j+1} \geq n+1$ or $j \geq k$ then
            $s_{j+1} \leftarrow n+1$, $k' \leftarrow j$, break
    $Len \leftarrow Len \cdot \sqrt{\frac{k}{k'}}$, $cur\_r \leftarrow |k'-k|$       // update
    if $cur\_r < best\_r$ then
        $best\_r \leftarrow cur\_r$, $|\pi_1| \cdots |\pi_{k'}| \leftarrow \varphi_1 \cdots \varphi_{k'}$
return $|\pi_1|, \cdots, |\pi_{k'}|$
```

Iterative Stepwise Optimization (ISO). By the analysis in Appendix D, the IGS optimality is achieved when

$$q_{1i} = \frac{H}{L \cdot \sqrt{|\pi_j|}} \quad s.t. \;\; C(v_i) = j \tag{15}$$

where H is constant and L depends on the node clustering.

Under this condition, the desired cluster size for each node can be explicitly computed from the corresponding component on q_1, though it also relates to the overall clustering assignment through L. The ISO algorithm starts from the initial clustering by a uniform k-segmentation and iteratively meets the requirement of Equation (15) for each cluster, from π_1 to π_k. After each round, L is updated and the iteration is repeated until no cluster change can increase the IGS objective. The red lines in Figure 3 illustrates an example of the ISO algorithm. Appendix D gives a detailed algorithm derivation.

V. Dynamic IGS

The previous section has solved the static IGS problem that given the size of clustering (k), we can compute an optimal summarization that maximizes the influence flows in the graph. However, in real-world usages (e.g., on the citation graph), most users are not satisfied with one static summarization for each graph. Instead, to complete complex analytical tasks, they need to access multiple summarizations on the same

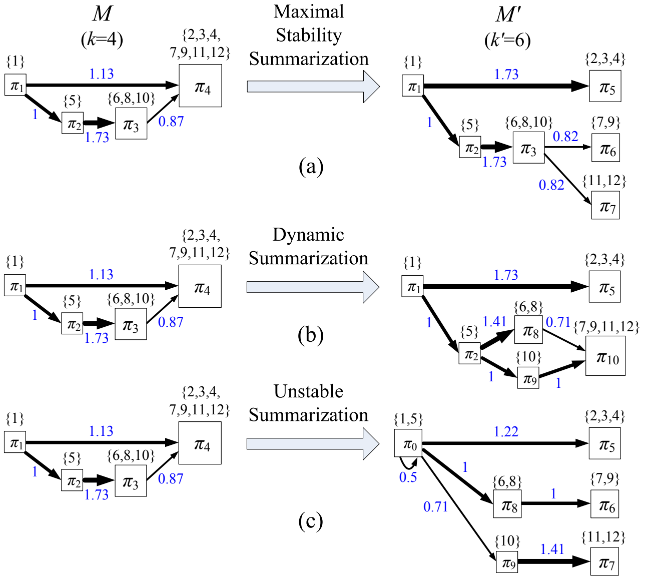


Fig. 4. Illustrative examples on the dynamic visual summarization: (a) by the algorithm to maximize stability ($\Gamma = 0.75, \sum r = 6.1$); (b) by the extended IGS algorithm to satisfy the dynamic update rule while gaining a larger total flow rate ($\Gamma = 0.5, \sum r = 6.85$); (c) by the algorithm that violates the dynamic update rule ($\Gamma = 0, \sum r = 5.84$).

graph, to examine, compare and reason the flow patterns in different granularities. Computing static summarizations in varying k can be a trivial solution, but the user will suffer from the context-losing problem when the displayed summarization changes much across different settings. In this section, we seek computational methods for the *dynamic* summarization which ensures the smooth visual transition between summarizations apart from optimizing the IGS objective.

A. Problem

Before formulating the dynamic summarization problem, we first define the key measure of visual stability.

Definition 1 (Visual Stability of Transition): Consider two summarizations M and M' on the same graph, whose clustering results are denoted by $\prod = \{\pi_1, \cdots, \pi_k\}$ and $\prod' = \{\pi'_1, \cdots, \pi'_{k'}\}$ respectively. Both π_j and π'_j denote a single cluster containing several nodes. The visual stability of transition from M to M' is defined by the ratio of unchanged clusters in the former summarization M:

$$\Gamma_{unweighted}(M \to M') = \frac{|\prod \bigcap \prod'|}{|\prod|}$$
$$\Gamma_{weighted}(M \to M') = \frac{||\prod \bigcap \prod'||}{||\prod||} \qquad (16)$$

where $|\cdot|$ denotes the number of clusters and $||\cdot||$ denotes the number of nodes in all clusters. This definition is valid mainly because we target at the visualization-level stability. In the typical unweighted definition, each cluster is drawn as a single element and the stability is defined as the ratio of unchanged visual elements. Meanwhile, the stability of the reversed transition $M' \to M$ is proportional to the stability of $M \to M'$, by a factor of $\frac{k}{k'}$. Therefore, the optimization of the dynamic summarization in both directions will lead to the same result, and in the following we mainly consider the case with a increasing number of clusters (i.e., $k' > k$).

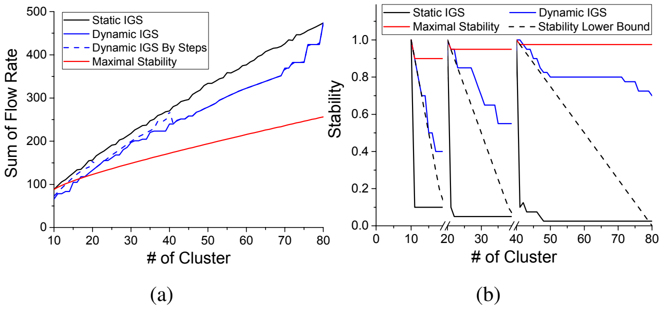


Fig. 5. The trade-off between (a) the sum of flow rate and (b) the stability of transition in dynamic summarizations when the number of clusters increases from 10 to 80. The result is obtained on a 10892-node graph from ArnetMiner.

By Definition 1, the maximal stability of transition can reach $\frac{k-1}{k}$ where only one cluster is changed in the previous summarization. An example is given in Figure 4(a), from M with $k = 4$ clusters, the next summarization M' with $k' = 6$ clusters carries $k - 1 = 3$ clusters from M without any change (π_1, π_2, π_3). The other $k' - (k - 1) = 3$ clusters in M' (π_5, π_6, π_7) are split from the remaining cluster in M (π_4). When the static IGS objective is considered, the best guess is that the new summarization after the transition with the maximal stability can also maximize the sum of flow rate among all the summarizations with k' clusters. Unfortunately, this guess is far from the truth: the experiment result in Figure 5(a) shows that, the summarization algorithm to maximize the stability (the red line) quickly falls behind the static IGS algorithm (the black line) in the sum of flow rate, when the number of clusters increases from $k = 10$. On the last trial of $k' = 80$, the flow rate of the maximal stability algorithm is only 54.1% of that of the static IGS algorithm. On another extreme, the static IGS algorithm maximizing the flow rate (using Algorithm 1, the greedy curve fitting) does not maintain the stability of transition. In Figure 5(b), the stability (the black line) drops to almost zero immediately when the number of clusters increases. Notice that, the stability is computed in multiple steps: $k' = 10 \sim 19$ is compared with $k = 10$; $k' = 20 \sim 39$ is compared with $k = 20$; $k' = 40 \sim 80$ is compared with $k = 40$.

The preliminary result indicates that the stability of transition and the sum of flow rate are almost two conflicting objectives and there is no single algorithm that can optimize them simultaneously. To be able to reduce these objectives into a solvable problem, we define the dynamic update rule that turns the stability requirement into an inequality constraint.

Definition 2 (Dynamic Update Rule): Consider two summarizations M and M' on the same graph, whose clustering results are denoted by $\prod = \{\pi_1, \cdots, \pi_k\}$ and $\prod' = \{\pi'_1, \cdots, \pi'_{k'}\}$ respectively ($k' > k$). The transition $M \to M'$ is said to satisfy the dynamic update rule if for any cluster in M', denoted by $\pi'_{j'} \in \prod'$, there exists only one cluster in M, denoted by $\pi_j \in \prod$, that meets $\pi_j \supseteq \pi'_{j'}$.

The key of this rule is to suggest that for the dynamic summarization with an increasing size, the clusters can only be split, but not merged. Conversely, for the case with a decreasing size, the original clusters can only be merged, but not split. For example, the summarization by the dynamic IGS algorithm in Figure 4(b) obeys this rule, while the counterexample in Figure 4(c) violates the rule with both split and merge of clusters in the transition. Compared with the method to maximize stability, the dynamic update rule loosens the requirement into a stability lower bound.

Lemma 2 (Stability Lower Bound): For any transition from the summarization M with k clusters to the summarization M' with k' clusters ($k' > k$), there exists a lower bound of $2 - \frac{k'}{k}$ for the stability of transition, if only the transition satisfies the dynamic update rule.

Proof: See Appendix E.

Figure 5(b) depicts this stability lower bound. Notice that, the bound will drop below zero when $k' > 2k$. Therefore, we do not apply the dynamic summarization when the number of new clusters is too far from the initial setting. This is the reason why we examine the stability of transition by steps in Figure 5(b). Now we formally define the dynamic summarization problem as a constrained optimization problem.

PROBLEM 2: DYNAMIC INFLUENCE GRAPH SUMMARIZATION

Given: 1) A, *the topology adjacency matrix of the maximal influence graph* G; 2) *an initial summarization* M *specified by the cluster index of all nodes in G, denoted by* $C(v_i)$ $(i = 1, \cdots, n,\ |C(v_i)| = k \le n)$; 3) k' $(n \ge k' > k)$, *the expected number of clusters in the dynamic summarization* M';

Compute: $C'(v_i)$ $(i = 1, \cdots, n)$ *where* $|C'(v_i)| = k'$, *the cluster index of all nodes in the dynamic summarization* M';

By optimizing *constrained flow rate maximization objective*:

$$\max \quad \sum_{i=1}^{k'^2} r(\xi'_i) \quad s.t. \quad M \to M' \ obeys \ Definition \ 2 \qquad (17)$$

where ξ'_i $(i = 1, \cdots, k'^2)$ denotes all the flows in M'. Notice that we do not consider the selection of flows here because this has been solved in the second stage of the static IGS problem.

Solving the dynamic IGS problem leads to a better trade-off between the flow rate objective and the stability requirement, as illustrated in the example of Figure 4(b). A preview of the performance measure is shown in Figure 5. The dynamic IGS algorithm (the blue line) obtains a larger flow rate compared with the algorithm to maximize stability (Figure 5(a)), and achieves a flow rate of at least 70.3% of the static IGS algorithm. When the computation is performed in 3 steps (10∼19, 20∼39, 40∼80), the total flow rate increases to 82.3% of the static IGS algorithm at a minimum (the dashed line in Figure 5(a)). In Figure 5(b), the stability of the dynamic IGS is between the static IGS and the maximal stability, and higher than the theoretical lower bound in Lemma 2.

B. Solution Framework and Analysis

To solve the dynamic IGS problem, we propose a bottom-up solution framework. Compared with the straightforward method that starts from a high-level summarization and grows the number of clusters in a top-down manner, our framework takes an opposite strategy. We start from a fine-grained low-level summarization and agglomeratively merge the node clusters to obtain coarser-grained summarizations in a bottom-up manner. The reason for this bottom-up summarization framework is two-fold. First, by this design, both the theoretic analysis and the summarization algorithm can be naturally extended from those in solving the static IGS problem, ensuring a coherent methodology in our work. Second, through the experiments in Section VI, the bottom-up framework enjoys a much lower computational complexity and better trade-offs in the quantitative performance, compared with the top-down dynamic summarizations.

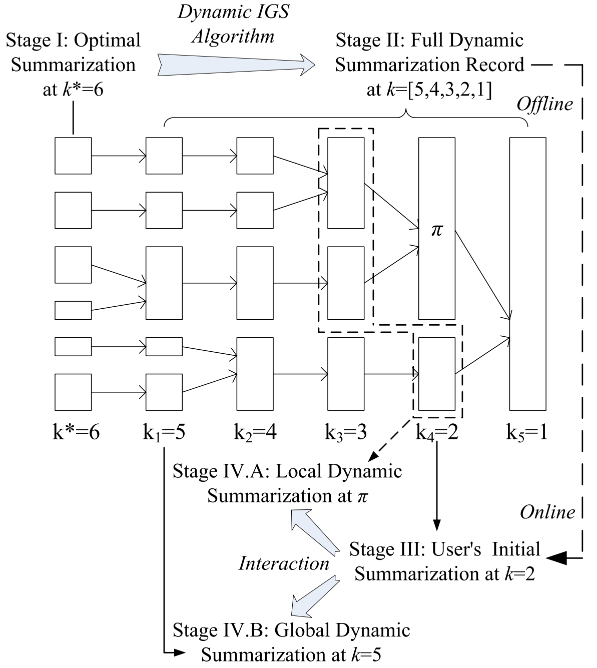

Dynamic IGS Algorithm
Stage I: Optimal Summarization at k*=6
Stage II: Full Dynamic Summarization Record at k=[5,4,3,2,1]
Offline
π
k*=6
k1=5
k2=4
k3=3
k4=2
k5=1
Stage IV.A: Local Dynamic Summarization at π
Online
Interaction
Stage III: User's Initial Summarization at k=2
Stage IV.B: Global Dynamic Summarization at k=5

Fig. 6. The agglomerative dynamic summarization framework.

Figure 6 illustrates the detailed steps of the proposed framework. In Stage I, we pick a number of cluster (k^*) that is large enough for the user's analysis task, then the static IGS algorithm in Section IV is applied to compute an optimal summarization. In the next stage, we decrease the number of clusters to create multiple dynamic summarizations in coarser granularities (e.g., $k^* \to k_1 \to \cdots \to k_5$). The result is the clustering tree shown in the center of Figure 6. Users are plugged in from Stage III, in which they start from an initial granularity of summarization (e.g., $k = 2$). The framework supports two user interactions for smooth navigation of the summarization. In the first interaction scenario (Stage IV.A), the user can select a set of nodes in the current summarization and expand them locally into the next hierarchy according to the clustering tree (e.g., the cluster π). In the second interaction scenario (Stage IV.B), the user can specify a larger number of clusters to enrich the graph globally (e.g., $k = 6$).

By this agglomerative summarization framework, the dynamic IGS problem can be analyzed as the same quadratic optimization problem with Section IV-A. Take the first-level dynamic summarization in stage II as an example, the input is the optimal summarization M^* with k^* clusters, and define the desired output as the new summarization M with k clusters ($k^* > k$). According to the dynamic update rule, the clusters in M^* can only be merged but not split. Therefore, the dynamic summarization from M^* to M can be seen as the k-clustering over the aggregated graph of M^*, with the objective to maximize the sum of flow rate in M.

According to (5), the sum of flow rate in M is written as:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sqrt{|\pi_{C(v_i)}||\pi_{C(v_j)}|}} \tag{18}$$

where π_i $(i = 1, \cdots, k)$ denote the clusters in M, $C(v_i)$ $(i = 1, \cdots, n)$ defines the clustering of all the nodes in M.

Define a $k^* \times k^*$ flow rate matrix B as the output of the optimal summarization M^* where each entry b_{ij} has

$$b_{ij} = \frac{\sum_{s\in\pi_i^*}\sum_{t\in\pi_j^*} a_{st}}{\sqrt{|\pi_i^*||\pi_j^*|}} \tag{19}$$

where π_i^* $(i = 1, \cdots, k^*)$ denote the clusters in M^*. If we define a dynamic cluster size vector by $y = (\sqrt{\frac{|\pi_1^*|}{|\pi_{C(\pi_1^*)}|}}, \cdots, \sqrt{\frac{|\pi_{k^*}^*|}{|\pi_{C(\pi_{k^*}^*)}|}})$, then the total flow rate in (18) can be constructed as the quadratic form again:

$$\sum_{i=1}^{k^*}\sum_{j=1}^{k^*} b_{ij}y_iy_j = y^TBy = y^TB^*y \tag{20}$$

where $B^* = \frac{B+B^T}{2}$ is the symmetric version of the flow rate matrix. The objective becomes a similar form to the static IGS.

$$\max\ y^TB^*y \quad s.t. \quad y^Ty = k \tag{21}$$

Applying the same analysis with Section IV-A, the maximality is achieved when $y = \sqrt{k}q_1$, parallel to the unit vector q_1, which is the largest eigenvector of B^*.

C. Algorithm

Due to the similarity with the static IGS problem, the principle of the algorithms proposed in Section IV-C can be extended to solve the dynamic IGS problem. In Algorithm 2, we adapt the GCF algorithm to the dynamic scenario. Three changes are made for such an adaptation: 1) the input adjacency matrix A of G is replaced by the flow rate matrix B of M^* (line 2); 2) the sorting applies on $\{\frac{q_{1i}}{\sqrt{|\pi_i^*|}}\}_{i=1}^{k^*}$ now (line 3); 3) the initial cluster size of $|\pi_i^*|$ is taken into account when fitting the requirement of the target clusters (line 10).

There is another notable difference from the static IGS problem: Theorem 2 is violated because the initial clusters have unequal size now. Therefore, it is not necessary to stay with continuous segmentations over the sorted eigenvector. In other words, GCF is not guaranteed to approach the optimality in the dynamic scenario. With this observation, we propose another packing-based algorithm for more elaborate clustering. Though the performance is slightly improved in some cases on the flow rate objective, this tiny gain does not justify its increased complexity in computing more elaborate dynamic summarizations (Section VI).

VI. Evaluation

We evaluate the proposed methods in four dimensions: 1) the performance measures (e.g., flow rate and stability) achieved by both the optimal and the dynamic IGS algorithms; 2) the algorithm scalability by complexity analysis and the empirical computation time; 3) the usefulness of static/dynamic influence summarizations by case studies; 4) the effectiveness in completing real-life tasks through user studies.

Algorithm 2: Dynamic GCF $\sim$ **GCF_GCF**(A, n, k^*, k).

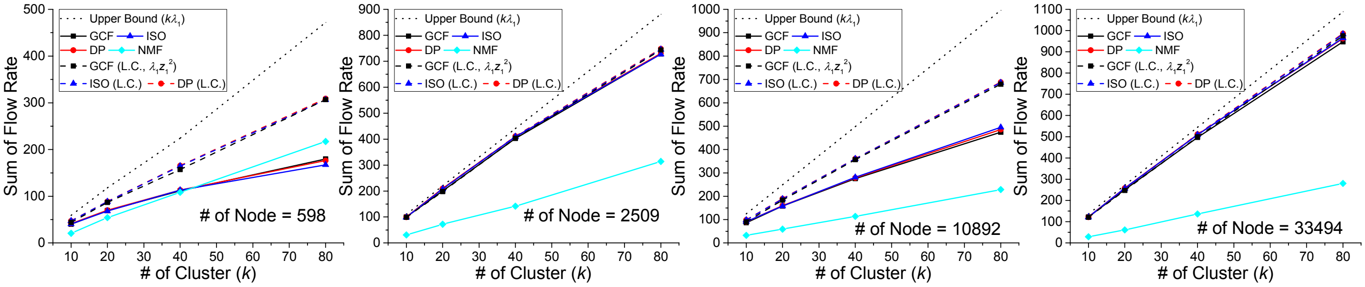
```
   Input  : $A$ (adjacency matrix), $n$ (# of nodes), $k^*, k$ (# of initial/target clusters)
   Output: $\Pi^* = \{|\pi_i^*|\}_{i=1}^{k^*}$ (size of initial clusters),
           $\Pi = \{|\pi_i|\}_{i=1}^{k}$ (size of target clusters)
 1 $\Pi^* \leftarrow$ GCF($A$, $n$, $k^*$)                     // initial clustering
 2 $B \leftarrow new\_flowrate\_matrix(A, \Pi^*)$
 3 $q_1 \leftarrow largest\_eigenv(\frac{B+B^T}{2})$, sort $\{q_{1i}/\sqrt{|\pi_i^*|}\}_{i=1}^{k^*}$
 4 $Len \leftarrow \sqrt{k}$, $k' \leftarrow 0$, $best\_r \leftarrow k$                  // init
 5 while $k' \neq k$ do
       /* outer iteration to adjust clusters                          */
 6     $s_1 \leftarrow 1$, $s_{k+1} \leftarrow k^* + 1$        // separator, $\pi_j = [s_j, s_{j+1})$
 7     for $j \leftarrow 1$ to $k$ do
           /* inner iteration to fit the size of cluster $j$           */
 8         $\Phi_j \leftarrow (Len \cdot q_{1s_j})^{-2}$, $\varphi_j \leftarrow 0$
 9         while $\varphi_j \leq k^* - s_j$ and $\Phi_j > 0$ do
10             $\Phi_j \leftarrow \Phi_j - |\pi_{s_j+\varphi_j}^*|$, $\varphi_j \leftarrow \varphi_j + 1$
11         $s_{j+1} \leftarrow s_j + \varphi_j$
12         if $s_{j+1} \geq k^* + 1$ or $j \geq k$ then
13             $s_{j+1} \leftarrow k^* + 1$, $k' \leftarrow j$, break
14     $Len \leftarrow Len \cdot \sqrt{\frac{k}{k'}}$, $cur\_r \leftarrow |k' - k|$                 // update
15     if $cur\_r < best\_r$ then
16         $best\_r \leftarrow cur\_r$, $|\pi_1| \cdots |\pi_{k'}| \leftarrow \varphi_1 \cdots \varphi_{k'}$
17 return $\Pi^* = \{|\pi_i^*|\}_{i=1}^{k^*}$, $\Pi = \{|\pi_i|\}_{i=1}^{k}$
```

TABLE II. Sample influence graphs by citation relationship.

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

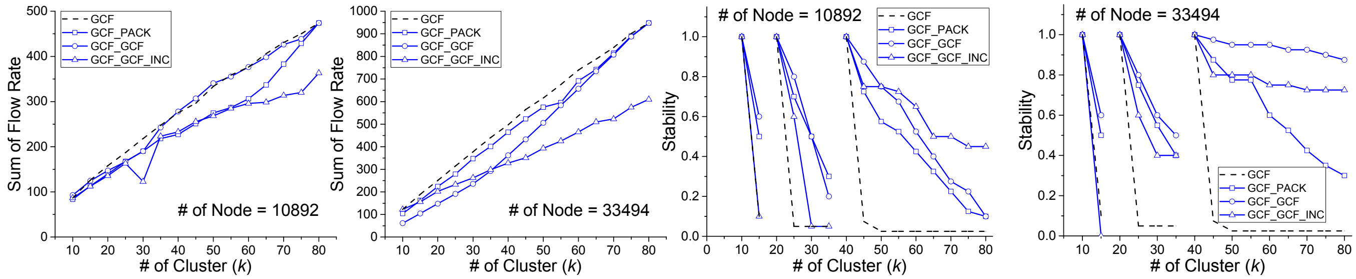
A. Performance Comparison

We collect influence graphs over the paper citation relationship from two mainstream academic search databases, ArnetMiner [3] and CiteSeerX [2]. Each database contains multi-million papers/citations. To compare with the result in Ref. [4], we start from the same citation graphs from ArnetMiner, as listed in Table II. Four static summarization algorithms are executed initially: Greedy Curve Fitting (GCF) and Iterative Stepwise Optimization (ISO) which we propose to approximate the optimal IGS objective, Dynamic Programming (DP) which is proved to achieve the optimality, and the algortihm using the symmetric version of Nonnegative Matrix Factorization (NMF) in Ref. [4], which has been shown to greatly outperform classic graph clustering algorithms on the flow rate objective. GCF/ISO/DP are tested in single thread on a modern desktop with 4-core Intel i7-4790 CPU and 16GB of memory. NMF is run in 32 threads with the same setup as Ref. [4], on a server with two 8-core 2.9GHz Intel Xeon E5-2690 CPU and 384GB of memory, because NMF requires large memory and parallel computing environment to complete. On each graph, we test four number of clusters: 10, 20, 40 and 80. The flow rate is summed from all flows, i.e., $l = k^2$.



(a) (b) (c) (d)

Fig. 7. Sum of flow rate in the static summarization of sample citation influence graphs (ArnetMiner).



(a) (b) (c) (d)

Fig. 8. Sum of flow rate and stability by the dynamic summarization algorithms (ArnetMiner).

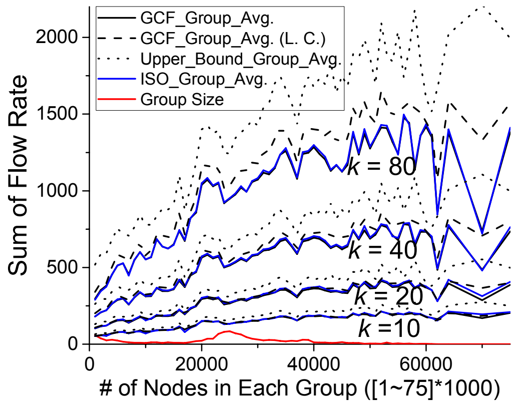
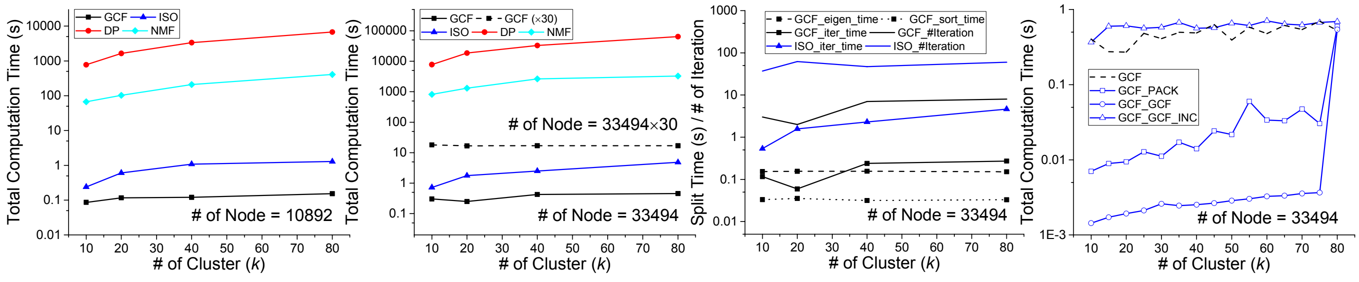


Fig. 9. Sum of flow rate in 1296 citation influence graphs (CiteSeerX).

Figure 7(a)$\sim$(d) depicts the achieved IGS objective on four sample graphs. The solid lines in different symbols/colors show the result from each of the four algorithms. GCF, ISO and DP are quite close in every sample, with the largest difference of 7.5% happened at $k = 80$ on the 598-node graph (the case with the smallest per-cluster number of nodes). The result by NMF is significantly different from our proposed algorithms. In most cases, NMF leads to a much smaller flow rate, less than 50% of the proposed algorithms. The only exception happens again on the case with the smallest per-cluster number of nodes. The three dashed lines on top of the solid lines depict the largest component (L.C.) of the flow rate by the proposed algorithms, i.e., $\lambda_1 z_1^2$ in (8). In two cases, the largest component is almost the same with the flow rate (less than 2% difference); in the other two cases, the largest component is moderately higher. This shows that our algorithms to maximize the largest component of flow rate is valid, because either the largest component can represent the flow rate, or the other smaller components (eigenvalue) are possibly negative. The topmost dotted line shows the theoretical upper bound ($k\lambda_1$) of the total flow rate. Though our analysis proves the upper bound can not be achieved, the performance result indicates that the actual flow rate obtained is comparable and in most cases larger than 50% of this bound.

Figure 9 provides more results on the data set of CiteSeerX. We collect all ICDE/TKDE papers in CiteSeerX that can influence more than 1000 papers both directly and indirectly. This generates 1296 influence graphs with size varying from 1001 to 75847. We categorize these graphs into groups by an interval of 1000-node in graph size, i.e., [1001, 2000],$\cdots$,[75001, 76000]. The graph size has a near-Gaussian distribution with a heavier head, as shown in the red line at the bottom of Figure 9. The other two solid lines plot the average flow rate achieved in each group by GCF and ISO. The result shows that on more cases in the CiteSeerX dataset, the two proposed algorithms still achieve almost the same performance on the flow rate objective. Again, the largest component is very close to the sum of flow rate, from the group of 1000 to 50000 nodes, the average difference is below 10%. For the group beyond 50000 nodes, the group size is too small to provide stable results, as shown by the large jitters at the end of all lines. We do not compare with DP and NMF in more samples because of their high computational complexity.

In another scenario, we test four IGS algorithms to generate dynamic influence graph summarizations as the number of clusters increases from $k = 10$ to $k = 80$ with an increment of 5. All the algorithms use GCF to compute the initial static summarization at $k = 80$ (or $k = 10$). Two dynamic algorithms by cluster packing (GCF_PACK) and iterative GCF (GCF_GCF) start from $k = 80$ and incrementally merge clusters until $k = 10$. Another is the reversed version of GCF_GCF (GCF_GCF_INC), by starting from the smallest setting of $k = 10$ and invoking a constrained version of GCF to compute summarizations in larger k. The last algorithm implements the static GCF, i.e., running an independent result at each k. Figure 8(a)(b) shows the sum of flow rate achieved by each algorithm as the number of clusters changes. The two merge-based dynamic algorithms obtain better results than the split-based



(a) (b) (c) (d)

Fig. 10. Total computation time in summarizing sample citation influence graphs (ArnetMiner).

TABLE III. Computational complexity of IGS algorithms.

Algorithm	Time Complexity			Space Complexity
	Matrix Oper.	IGS Opt.	Dynamic Update	
DP	$O(n \cdot logn)$	$O(n^2k)$	$O(n^2k)$	$O(nk)$
ISO/Pack.	$O(n \cdot logn)$	$O(n \cdot \#I)$	$O(k^{*2} \cdot \#I)$	$O(n)$
GCF	$O(n \cdot logn)$	$O(k \cdot \#I)$	$O(k^{*2} \cdot logk^*)$	$O(n)$
NMF	$O(n^2)$	$O(nk^2 \cdot \#I)$	$O(n^2)$	$O(n^2)$

algorithm. These two algorithms also achieve moderately close performance to the independent optimal IGS algorithm. On the stability measure between the dynamic summarizations, as depicted in Figure 8(c)(d), all three dynamic IGS algorithms perform much better than the static GCF without considering the stability. GCF_GCF is generally better in most cases.

B. Algorithm Complexity

In Table III, we list the theoretical complexity of the three static algorithms proposed here, and the NMF algorithm in Ref. [4] for comparison. The input maximal influence graph is assumed to be sparse, i.e., the number of edges is in a similar scale with the number of nodes (n). The time complexity in each static summarization is mainly composed of two parts: 1) the operations on matrix, including the decomposition and the sorting of eigenvector; 2) the iteration for IGS optimization. The dynamic update refers to the dynamic computation time of each k' after the initial static summarization (Section V).

On the three static algorithms, the time for the matrix operation is the same. Both finding the largest eigenvector [24] and sorting it takes $O(n \cdot logn)$ time. NMF requires $O(n^2)$ as it computes AA^T. In the second part, DP takes $O(n^2k)$ time to complete all iterations. ISO and GCF require $O(n)$ and $O(k)$ time in each iteration respectively. The number of iterations (#I) is unpredictable, but as shown in Figure 10(c), GCF takes much less iterations than ISO. NMF spends $O(nk^2)$ time in each multiplicative iteration and the number of iterations can be hundreds of times larger than ISO/GCF algorithms.

On the dynamic summarization, DP and NMF do not explicitly support, so they take the same time with the static algorithm, using the highest complexity between the matrix operation and the IGS optimization. GCF takes a similar complexity form to its static version, but the number of nodes is replaced by k^* (the largest number of clusters in the dynamic update), leading to $O(k^{*2} \cdot logk^*)$ (the eigenvector computation time, because the initial summarization graph can be dense). ISO takes the same eigenvector time, but it requires $O(k^{*2})$ time in each iteration of the IGS optimization, which in total can be larger than the eigenvector time.

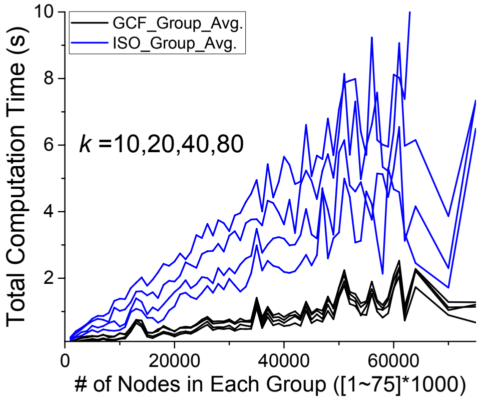
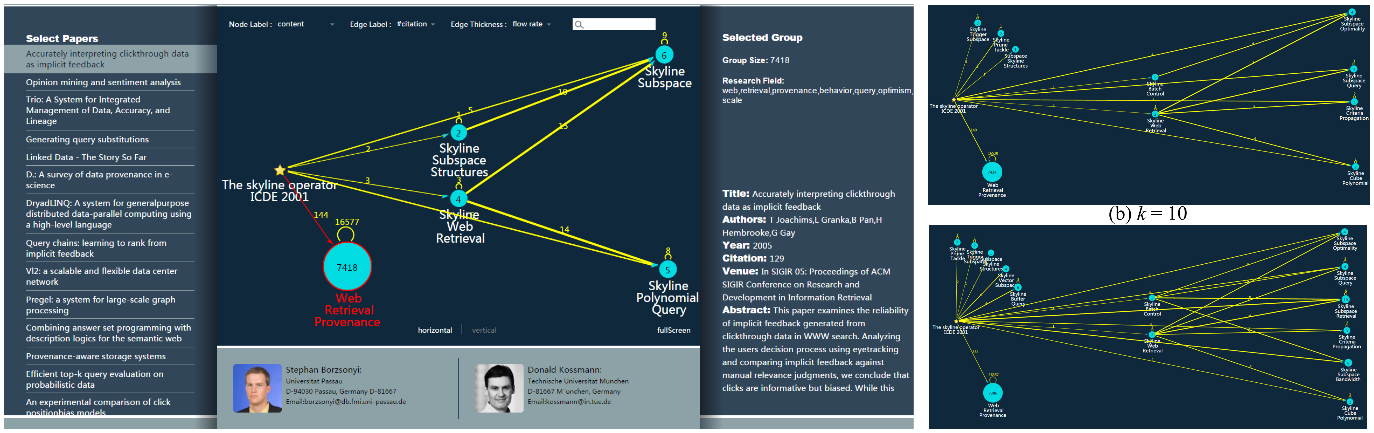


Fig. 11. Computation time to summarize 1296 citation graphs (CiteSeerX).

On the space requirement, ISO/GCF only stores one sparse matrix in $O(n)$ size. DP explicitly requires $O(n^2)$ to store the table F, but it can be reduced to $O(nk)$ by trading-off the computation time. NMF needs to store a near-dense matrix of AA^T, so the space complexity is $O(n^2)$.

We also evaluate the algorithm complexity through experiments. On the total computation time, Figure 10(a)(b) depicts the comparison among four IGS algorithms. Though DP achieves the maximal flow rate, it suffers from the high computation overhead of $O(n^2k)$. NMF is one magnitude better, because it requires $O(nk^2)$ time for each iteration, and the sparsity of the matrix reduces the decomposition time. However, we caution that the performance of NMF is obtained on a server machine with 32 threads, and even then, it requires an hour to compute on a 30000-node graph. This is not feasible to deploy in the large-scale influence summarization website hosting more than 100,000 graphs of such size. Alternatively, the proposed algorithm of ISO is two-magnitude faster than NMF, and GCF is three-magnitude faster, aligning well with the theoretical analysis. GCF only spends 0.46 seconds to compute the summarization of a 30000-node graph. Though the influence graph in our data set seldom exceeds a size of 30000 nodes, we proceed by replicating the 33494-node graph 30 times. As shown in the dashed line of Figure 10(b), the GCF algorithm only takes 16∼18 seconds to summarize the million-node graph under all cluster size settings. This makes it feasible to be deployed in large scale. The split time in Figure 10(c) shows that the major time cost of GCF is spent on the eigenvector computation and the iterative IGS optimization. We have repeated the performance test on 1296 graphs from the CiteSeerX dataset. As shown in Figure 11, the computation time by GCF and ISO is only slightly larger than the linear time to the number of nodes, which corresponds well to the above sampled test and the complexity analysis.



Select Papers

- Accurately interpreting clickthrough data as implicit feedback
- Opinion mining and sentiment analysis
- Trio: A System for Integrated Management of Data, Accuracy, and Lineage
- Generating query substitutions
- Linked Data - The Story So Far
- D.: A survey of data provenance in e-science
- DryadLINQ: A system for generalpurpose distributed data-parallel computing using a high-level language
- Query chains: learning to rank from implicit feedback
- Vl2: a scalable and flexible data center network
- Pregel: a system for large-scale graph processing
- Combining answer set programming with description logics for the semantic web
- Provenance-aware storage systems
- Efficient top-k query evaluation on probabilistic data
- An experimental comparison of click positionbias models

Node Label : content

Edge Label : #citation

Edge Thickness : flow rate

Stephan Borzsonyi:
Universitat Passau
D-94030 Passau, Germany D-81667
Email:borzsonyi@db.fmi.uni-passau.de

Donald Kossmann:
Technische Universitat Munchen
D-81667 M¨unchen, Germany
Email:kossmann@in.tue.de

Selected Group

Group Size: 7418

Research Field: web,retrieval,provenance,behavior,query,optimism,scale

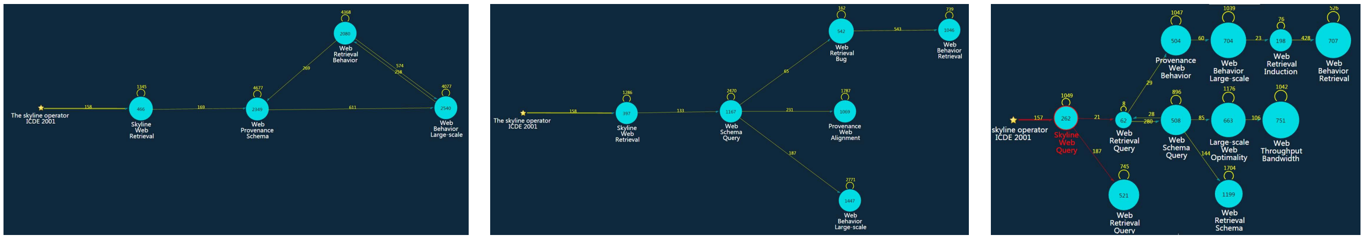
Title: Accurately interpreting clickthrough data as implicit feedback
Authors: T Joachims,L Granka,B Pan,H Hembrooke,G Gay
Year: 2005
Citation: 129
Venue: In SIGIR 05: Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval
Abstract: This paper examines the reliability of implicit feedback generated from clickthrough data in WWW search. Analyzing the users decision process using eyetracking and comparing implicit feedback against manual relevance judgments, we conclude that clicks are informative but biased. While this

(b) k = 10

(a) k=5

(c) k =15

Fig. 12. Dynamic summarizations by the GCF algorithm in varying settings of k.



(a) k=5

(b) k=7

(c) k=12

Fig. 13. Static summarizations by the NMF-based method in varying settings of k.

On dynamic IGS algorithms (Figure 10(d)), both the static GCF and the split-based GCF (GCF_GCF_INC) have a nearly constant computation time at each number of clusters (k). Their dynamic computation will suffer from a huge increase of time cost when every k is to be executed. On the other hand, the dynamic IGS by merge-operations spends much shorter time after the initial computation on the largest k. The best algorithm by the iterative GCF (GCF_GCF) only requires less than 0.1% of the initial computation time at each smaller k. Computing the dynamic IGS from $k = 79$ to $k = 10$ takes almost the same time with only computing $k = 80$.

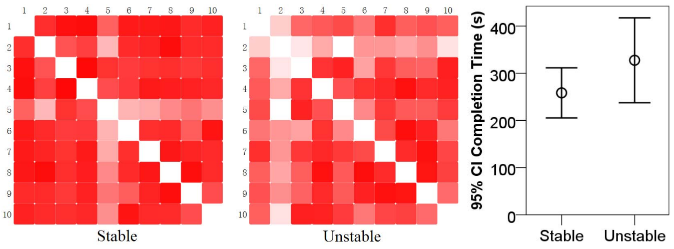
In summary, due to the better trade-off on the flow rate objective, the visual stability and the computational complexity, we recommend GCF in the static computation of influence graph summarization, and the iterative version of GCF for the dynamic update of influence graph summarizations.

C. Case Study

We apply our GCF method on the winner of ICDE'11 Influential Paper Award entitled "The Skyline Operator", which was published in ICDE 2001. This paper opened a new research topic by framing the skyline concept in the database setting. Figure 12(a) gives an initial high-level summarization of this paper's influence in the CiteSeerX database. All the 7435 papers directly or indirectly citing this work are grouped into 5 clusters. The largest cluster highlighted in red contains 7418 papers which are mostly loosely related to the skyline topic, e.g., web information retrieval (the main keyword displayed), sentiment analysis, etc. The other four clusters are on the core topic of skyline query, which are further divided into two hierarchies. When we drill down to finer granularities by setting $k = 10, 15$ in Figure 12(b)(c), more closely related skyline works are split from the large cluster and they construct a more detailed two-hierarchy influence pattern. As shown in Figure 12(c), the first hierarchy contains three other influential papers on this topic: "Shooting stars in the sky" in VLDB'02, "An optimal and progressive algorithm for skyline queries" in SIGMOD'03 and "Efficient distributed skylining for web information systems" in EDBT'04. In the second hierarchy, most of the four clusters published in 2006∼2011 are related to the skyline subspace analysis, varying from retrieval to optimality and bandwidth.

In comparison, we apply the previous NMF method on the same influence graph. Figure 13 gives the summarization result in three similar settings. Quite differently, these graphs group all the skyline works in the same cluster, one-hop from the original ICDE'01 paper. Multiple hierarchies are detected, which stress more on the research indirectly influenced by the skyline technique. For example, in Figure 13(c), after the skyline research, two main threads are detected, both related to web retrieval and large-scale processing. Though we can not judge which method (GCF or NMF) is better for various types of end users, in the objective of analyzing the original topic evolution on the source paper, the GCF method proposed here is more relevant. As discussed in Ref. [4], NMF is equivalent to the kernel k-mean clustering, which tries to preserve a balanced cluster size, so that more influence on multiple hops can be summarized. In contrast, the GCF method optimizing the total flow rate isolates the most influential nodes in the graph to create small clusters and provide a finer granularity on the most related nodes to the source.

Another difference comparing Figure 12 and Figure 13 lies in the stability of the dynamic summarization. The GCF method maintains a much better mental map in switching from the high-level picture of Figure 12(a) to more details in Figure 12(b)(c), while the summarization results by NMF are less



(a) Influence Path Similarity (b) Completion Time

Fig. 14. User study results on comparing stable and unstable summarization.

stable in both graph structure and the composition of clusters.

D. User Experiment

We conduct a controlled experiment to quantify the impact of graph stability on the user's ability to understand the influence patterns and evolvements. On the same influence graph, we apply the static and dynamic version of GCF algorithms separately. Two groups of summarization are generated, each with five graphs in increasing granularities. The first group corresponds to the stable summarization and the second group is unstable. We recruited 20 users with prior knowledge on the node-link graph and asked them to read the graphs and write down at most four influence paths from the source paper (by the topic keyword). The study follows a between-subject design with 10 users on each group.

Figure 14(a) depicts the pairwise cosine similarity between the user answered influence path vector. The stable group in the left has a relatively larger average similarity and smaller variance (0.89±0.084) than the unstable group in the right (0.82±0.11). Figure 14(b) indicates that both the average and the standard error of the task completion time in the stable group (258.4±53.0) is smaller than the unstable group (327.5±89.2), though the difference is not statistically significant, $F(1, 18) = 2.243, p = 0.152$. These results suggest that by the stable summarization, the user is more likely to retrieve consistent information than working with the unstable summarization. Users also require more cognitive efforts reading the unstable summarization.

VII. Conclusion

This paper proposes a new suite of algorithms for the influence graph summarization. Through an in-depth analysis on the IGS objective, it is shown that the flow rate maximization objective implicitly advocates separating and augmenting the flow pattern of key influencers in the summarization, defined by their eigenvector centrality in the graph. Compared with the sparse existing work in this area, our method bears superiority in several aspects, including the consistently better optimality, the near-linear computational complexity that allows the processing of million-node influence graphs, and the dynamic summarization finding a good trade-off between the summarization quality (i.e., the sum of flow rate) and the viewing stability upon user's interactive navigation. Experiment results on real-life citation networks validate the theoretical result and suggest the advantages of our method for end users. First, it helps focus on the topic-relevant part of the influence graph, and second, the consistency of the information gain is improved when the dynamic summarization method is applied.

Acknowledgment

This work is supported by China National 973 project 2014CB340301, Natural Science Foundation of China No. 61379088, 61322207, 61421003 and Special Funds of Beijing Municipal Science & Technology Commission. We gratefully acknowledge the data from CiteSeerX and ArnetMiner.

References

[1] "Influence, Oxford English Dictionary, http://www.oed.com/view/entry/95519."

[2] "CiteSeerX, http://citeseerx.ist.psu.edu/."

[3] 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.

[4] L. Shi, H. Tong, J. Tang, and C. Lin, "VEGAS: Visual influEnce GrAph Summarization on Citation Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3417–3431, 2015.

[5] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[6] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[7] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1096, 1994.

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

[9] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *SIGMOD*, 2008, pp. 419–432.

[10] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004.

[11] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of mathematical sociology*, vol. 1, no. 1, pp. 49–80, 1971.

[12] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *KDD*, 2002, pp. 538–543.

[13] L. Shi, Q. Liao, X. Sun, Y. Chen, and C. Lin, "Scalable network traffic visualization using compressed graphs," in *IEEE BigData*, 2013, pp. 606–612.

[14] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization." in *ICDE*, 2010, pp. 880–891.

[15] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[16] R. M. Bond, C. J. Fariss, J. J. Jones, and et al., "A 61-million-person experiment in social influence and political mobilization," *Nature*, vol. 489, pp. 295–298, 2012.

[17] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *KDD*, 2009, pp. 807–816.

[18] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.

[19] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *KDD*, 2011, pp. 529–537.

[20] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen, "CSI: Community-level social influence analysis," in *ECML/PKDD*, 2013, pp. 48–63.

[21] "Eigenvector centrality, https://en.wikipedia.org/wiki/centrality."

[22] R. A. Brualdi and H. J. Ryser, *Combinatorial matrix theory*. Cambridge University Press, 1991, vol. 39.

[23] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Communications of the ACM*, vol. 4, no. 6, p. 284, 1961.

[24] J. Kuczynski and H. Wozniakowski, "Estimating the largest eigenvalue by the power and lanczos algorithms with a random start," *SIAM journal on matrix analysis and applications*, vol. 13, no. 4, pp. 1094–1122, 1992.

Appendix A
Proof of Theorem 1

Theorem 1 (Performance Guarantee): Define the cluster size vector to maximize z_1 by x^*. For any feasible cluster size vector x, x^* must reach a lower bound of the achievable IGS objective.

$$\frac{{x^*}^T A^* x^*}{x^T A^* x} \geq \frac{(\max z_1)^2}{k} + [1 - \frac{(\max z_1)^2}{k}] \cdot \frac{\min_{i=2}^n \lambda_i}{\lambda_1}$$

Proof: First by Equation (7), $x^T A^* x \leq k \cdot \lambda_1$ for any x. Substitute x^* into Equation (8), we have

$$\begin{aligned}{x^*}^T A^* x^* = & \quad \textstyle\sum_{i=1}^n \lambda_i {z_i^*}^2 \\ = & \quad \lambda_1 \cdot (z_1^*)^2 + \textstyle\sum_{i=2}^n \lambda_i {z_i^*}^2 \\ \geq & \quad \lambda_1 \cdot (\max z_1)^2 + \min_{i=2}^n \lambda_i \cdot \textstyle\sum_{i=2}^n {z_i^*}^2 \\ = & \quad \lambda_1 \cdot (\max z_1)^2 + \min_{i=2}^n \lambda_i \cdot [k - (\max z_1)^2]\end{aligned}$$

where $z_i* = q_i \cdot x^*$. The theorem holds by dividing the minimum of ${x^*}^T A^* x^*$ with the maximum of $x^T A^* x$. ■

Appendix B
Proof of Lemma 1

Lemma 1 (Cluster Order): There exists a clustering that maximizes z_1 and satisfies the below inequality.

$$i \leq i' \Rightarrow q_{1i} \geq q_{1i'} \Rightarrow |\pi_{C(v_i)}| \leq |\pi_{C(v_{i'})}|$$

Proof: The first half of the lemma holds because the components in q_1 are ordered decreasingly. In the second half, we use the reduction to absurdity to complete the proof. In case that the lemma does not hold, for any clustering that maximizes z_1, there exists at least one pair of node indices $< i, i' >$ $(i \leq i')$ that satisfy $|\pi_{C(v_i)}| > |\pi_{C(v_{i'})}|$. If the cluster assignments of the node v_i and $v_{i'}$ are switched, the increase of z_1 can be computed from Equation (11). Note that the cluster size of all the other nodes except v_i and $v_{i'}$ does not change by such a switch.

$$\begin{aligned}\Delta z_1 = & \quad |\pi_{C(v_{i'})}|^{-\frac{1}{2}} \cdot q_{1i} + |\pi_{C(v_i)}|^{-\frac{1}{2}} \cdot q_{1i'} \\ & -|\pi_{C(v_i)}|^{-\frac{1}{2}} \cdot q_{1i} - |\pi_{C(v_{i'})}|^{-\frac{1}{2}} \cdot q_{1i'} \\ = & \quad (q_{1i} - q_{1i'})(|\pi_{C(v_{i'})}|^{-\frac{1}{2}} - |\pi_{C(v_i)}|^{-\frac{1}{2}})\end{aligned}$$

where $C(\cdot)$ denotes the cluster assignment before the switch. There are two cases, on one hand, $q_{1i} > q_{1i'}$, then $\Delta z_1 > 0$ and switching the cluster assignment will increase z_1, which contradicts the assumption that the clustering is optimal. On the other hand, $q_{1i} = q_{1i'}$, switching the cluster assignment does not affect z_1. Then we can switch v_i and $v_{i'}$ to produce a new optimal clustering $C'(\cdot)$ that has $|\pi_{C'(v_i)}| < |\pi_{C'(v_{i'})}|$. This new clustering satisfies all the conditions of this lemma. ■

Appendix C
Proof of Theorem 2

Theorem 2 (Segmentation): There exists a clustering that maximizes z_1 and forms a k-segmentation over the full node index list (i.e., [1,…,n]). In other words, the node index set of each cluster must be a continuous interval on the node index list. Mathematically, the below inequality holds:

$$I_{min}(1) \leq I_{max}(1) < I_{min}(2) \leq I_{max}(2) < \cdots \leq I_{max}(k)$$

where $I_{min}(j), I_{max}(j)$ denote the minimal and maximal node index in the cluster j and all clusters are numbered incrementally by their minimal node index $I_{min}(j)$.

Proof: Again we use the reduction to absurdity for the proof. In case that the theorem does not hold, for any clustering, we can find a minimal cluster index j that satisfies $i' = I_{max}(j) > I_{min}(j+1) = i$. Notice that $I_{max}(j) \neq I_{min}(j+1)$ because the clusters do not overlap. By Lemma 1, we have

$$i \leq i' \Rightarrow |\pi_{C(v_i)}| \leq |\pi_{C(v_{i'})}| \Rightarrow |\pi_{j+1}| \leq |\pi_j|$$

Also, let $i' = I_{min}(j) < I_{min}(j+1) = i$ and substitute into Lemma 1, we have

$$i' \leq i \Rightarrow |\pi_{C(v_{i'})}| \leq |\pi_{C(v_i)}| \Rightarrow |\pi_j| \leq |\pi_{j+1}|$$

The only chance is to have $|\pi_j| = |\pi_{j+1}|$. Therefore, we can switch the cluster assignment of the two nodes $v_{I_{max}(j)}$ and $v_{I_{min}(j+1)}$ to obtain a new clustering, while still preserving the maximality of the objective z_1. We can repeat this operation until there is no cluster index j having $I_{max}(j) > I_{min}(j+1)$. The final clustering will satisfy Equation (22) and lead to the contradiction. ■

Appendix D
Derivation of Theorem 2 for the ISO algorithm

We look at the desired property of the optimal k-segmentation by Theorem 2. Define a list of number $\{BG\}_{j=1}^k$ by $BG_j = |\pi_j|^{-\frac{1}{2}} \sum_{C(v_i)=j} q_{1i}$. Using the equality of root mean square, Equation (11) becomes:

$$\begin{aligned}\max \quad & \textstyle\sum_{j=1}^k BG_j = k \cdot \overline{BG} = k\sqrt{\overline{BG^2} - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k |\pi_j|^{-1} (\sum_{C(v_i)=j} q_{1i})^2 - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k |\pi_j| (\frac{\sum_{C(v_i)=j} q_{1i}}{|\pi_j|})^2 - \sigma_{BG}^2} \qquad (22)\end{aligned}$$

Define k list of number $(IG_1, \cdots, IG_k)$ where $IG_j = \{q_{1i}\}_{C(v_i)=j}$, i.e., the jth list includes all the components in q_1 corresponding to the jth cluster. Equation (22) becomes:

$$\begin{aligned}\max \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k |\pi_j| \cdot \overline{IG_j}^2 - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k |\pi_j| \cdot (\overline{IG_j^2} - \sigma_{IG_j}^2) - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k |\pi_j| \cdot \overline{IG_j^2} - \tfrac{1}{k} \sum_{j=1}^k |\pi_j| \cdot \sigma_{IG_j}^2 - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{j=1}^k \sum_{C(v_i)=j} q_{1i}^2 - \tfrac{1}{k} \sum_{j=1}^k |\pi_j| \cdot \sigma_{IG_j}^2 - \sigma_{BG}^2} \\ = \quad & k\sqrt{\tfrac{1}{k} \textstyle\sum_{i=1}^n q_{1i}^2 - \tfrac{1}{k} \sum_{j=1}^k |\pi_j| \cdot \sigma_{IG_j}^2 - \sigma_{BG}^2} \qquad (23)\end{aligned}$$

where $\sum_{i=1}^n q_{1i}^2$ is constant, σ_{BG}^2 is the between-cluster variance of q_1's components, $\sigma_{IG_j}^2$ represents the within-cluster variance of q_1's components in the jth cluster. The optimality is achieved when the between-cluster and all the k within-cluster variances are minimized.

In the following, we further derive a heuristic to compute this optimal k-segmentation, which is used in the ISO algorithm. First consider the between-cluster variance, the minimization is reached when $BG_1 = \cdots = BG_k$. Under this condition, we have $(\sum_{j=1}^k \sqrt{|\pi_j|}) \cdot BG_1 = \sum_{j=1}^k \sqrt{|\pi_j|} \cdot BG_j =$

$\sum_{i=1}^{n} q_{1i}$, which is constant. Let $H = \sum_{i=1}^{n} q_{1i}$, $L = \sum_{j=1}^{k} \sqrt{|\pi_j|}$, we have

$$BG_1 = \cdots = BG_k = \frac{H}{L}$$
$$\frac{\sum_{C(v_i)=j} q_{1i}}{|\pi_j|} = \frac{BG_j}{\sqrt{|\pi_j|}} = \frac{H}{L \cdot \sqrt{|\pi_j|}} \quad s.t. \quad j = 1, \cdots, k \tag{24}$$

On the within-cluster variances, any variance is minimized if all the components in q_1 belonging to the same cluster are equal:

$$q_{1i} = q_{1j} \quad s.t. \quad C(v_i) = C(v_j) \tag{25}$$

Combining (24) and (25), we obtain

$$q_{1i} = \frac{\sum_{C(v_i)=j} q_{1i}}{|\pi_j|} = \frac{H}{L \cdot \sqrt{|\pi_j|}} \quad s.t. \quad C(v_i) = j$$

Appendix E
Proof of Lemma 2

Lemma 2 (Stability Lower Bound): For any transition from the summarization M with k clusters to the summarization M' with k' clusters ($k' > k$), there exists a lower bound of $2 - \frac{k'}{k}$ for the stability of transition, if only the transition satisfies the incremental update rule.

Proof: By the dynamic update rule, the clusters in the original summarization M can only split. Notice that splitting each of such cluster will generate at least two new clusters in the updated summarization M' and increase the number of clusters by one. Denote the number of splits by C_s, then $C_s \leq k' - k$. The number of unchanged clusters, denoted by C_u, is lower bounded by $C_u \geq 2k - k'$, which proves the bound of the stability by $\frac{C_u}{k} \geq 2 - \frac{k'}{k}$. ■