



## Incremental spectral clustering by efficiently updating the eigen-system

Huazhong Ning<sup>a,\*</sup>, Wei Xu<sup>b</sup>, Yun Chi<sup>b</sup>, Yihong Gong<sup>b</sup>, Thomas S. Huang<sup>a</sup>

<sup>a</sup>ECE Department, University of Illinois at Urbana-Champaign, USA

<sup>b</sup>NEC Laboratories America, Inc., USA

### ARTICLE INFO

#### Article history:

Received 20 November 2008

Received in revised form 31 May 2009

Accepted 5 June 2009

#### Keywords:

Incremental clustering

Spectral clustering

Incidence vector/matrix

Graph

Web-blogs

### ABSTRACT

In recent years, the spectral clustering method has gained attentions because of its superior performance. To the best of our knowledge, the existing spectral clustering algorithms cannot incrementally update the clustering results given a small change of the data set. However, the capability of incrementally updating is essential to some applications such as websphere or blogsphere. Unlike the traditional stream data, these applications require incremental algorithms to handle not only insertion/deletion of data points but also similarity changes between existing points. In this paper, we extend the standard spectral clustering to such evolving data, by introducing the *incidence vector/matrix* to represent two kinds of dynamics in the same framework and by incrementally updating the eigen-system. Our incremental algorithm, initialized by a standard spectral clustering, continuously and efficiently updates the eigenvalue system and generates instant cluster labels, as the data set is evolving. The algorithm is applied to a blog data set. Compared with recomputation of the solution by the standard spectral clustering, it achieves similar accuracy but with much lower computational cost. It can discover not only the stable blog communities but also the evolution of the individual multi-topic blogs. The core technique of incrementally updating the eigenvalue system is a general algorithm and has a wide range of applications—as well as incremental spectral clustering—where dynamic graphs are involved. This demonstrates the wide applicability of our incremental algorithm.

© 2009 Elsevier Ltd. All rights reserved.

### 1. Introduction

Spectral clustering is notable both for its theoretical basis of graph theory and for its practical success. It recently has many applications in data clustering, image segmentation, web ranking analysis, and dimension reduction. Spectral clustering can handle very complex and unknown cluster shapes in which cases the commonly used methods such as  $K$ -means and learning a mixture model using EM may fail. It relies on analyzing the eigen-structure of an affinity matrix, rather than on estimating an explicit model of the data distribution [1,2]. In other words, the top eigenvectors of the graph Laplacian can unfold the data manifold to form meaningful clusters [3].

However, nearly all existing spectral approaches are off-line algorithms, and hence they cannot be directly applied to dynamic data set. Therefore, to handle evolving data set, e.g., web data, there is a need to develop efficient algorithms for inductive spectral clustering to avoid expensive recomputation of the solution from the scratch.

An intuitive approach is fixing the graph on the training data and assigning new test points to their corresponding clusters by the nearest neighbor in the training data [3]. However, the error will accumulate quickly when more test points that are close to the cluster boundaries are added. In this paper, we extend the spectral clustering to handle evolving data by incrementally updating the eigenvalue system, which achieves more accurate results while requires low computational cost.

There exist incremental clustering algorithms [4–6] that are designed to handle only insertion of new data points. However, data sets, such as web pages and blogs, require the incremental algorithms to handle not only insertion/deletion of nodes but also similarity changes between existing nodes. Fig. 1 gives a toy example where a graph evolves from (a) to (b), with a similarity change of 0.5 added to the edge  $CD$  and a new node  $G$  connected to node  $F$ . In Fig. 1(a), the graph should be cut at the edge  $CD$ ; while in Fig. 1(b) the cut edge is  $DE$  due to the similarity change on edge  $CD$ .

We handle the two kinds of dynamics in the same framework by representing them with the *incidence vector/matrix* [7]. The Laplacian matrix can be decomposed into the production of two incidence matrices. A similarity change can be regarded as an incidence vector appended to the original incidence matrix. And an insertion/deletion of a data point is decomposed into a sequence of similarity changes.

\* Corresponding author. Tel.: +1 217 417 6820.

E-mail addresses: [hning2@ifp.uiuc.edu](mailto:hning2@ifp.uiuc.edu), [hning2@uiuc.edu](mailto:hning2@uiuc.edu) (H. Ning)

URL: <http://www.ifp.uiuc.edu/~hning2>.

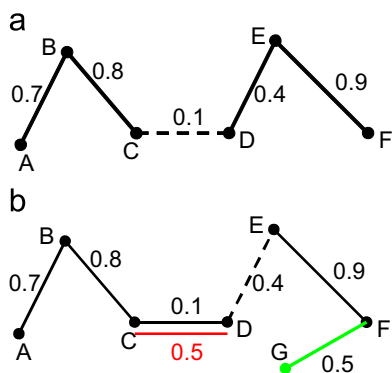


Fig. 1. A toy example of evolving data. (a) Before evolution. (b) After evolution. The dash lines are the cut edges.

Each newly added incidence vector (similarity change) may induce increment to the Laplacian and degree matrixes, and we approximate the corresponding increments of the eigenvalues and eigenvectors by omitting the influence of the data points outside the spatial neighborhood of the updating data points. In this way, the eigen-system and the cluster labels are incrementally updated as data points are inserted/deleted or similarity changes occur.

This approach is useful to the applications where the similarity matrix is sparse and where both the data points and their similarities are dynamically updated. An example is the community discovery of the web-blogs. The key observation is that a link reference from an entry of a source blog to an entry of a destination blog serves as an endorsement of the similarity between the two blogs. A graph can be constructed based on the similarities between the web-blogs, and communities (clusters) can be discovered by spectral clustering. However, web-blogs are evolving, and new blogs and new links are added or removed every day. Therefore, the standard spectral clustering cannot be used to online monitor the web-blogs because of the huge number of blogs and, in turn, of the high computational cost. For sparse similarity matrix, Lanczos method [8] may save much cost to solve the eigenvalue problem. But it is still impractical to recompute the solution from the scratch at each time instance the data set is updated, especially when the web-blogs are huge. On the contrary, our approach applied to the web-blog data achieves similar accuracy but with much lower computational cost, compared with recomputation by the standard spectral clustering.

It is worth to note that the core idea of our incremental clustering is dynamic updating of the (generalized) eigenvalue system. Actually it is a general algorithm that can also be applied to many other problems involving dynamic graphs. These problems require to solve the (generalized) eigenvalue system at each time the graph is updated. In Section 6, three related problems are stated and solved by this algorithm. The first problem is to choose edges from a candidate set to maximize the *algebraic connectivity* of a graph. *Algebraic connectivity* is the second smallest eigenvalue of the graph Laplacian that measures how well connected the graph is [9]. The second problem is to find the most significant edge of a graph. The last problem is related to linear embedding. These problems demonstrate the wide applicability of our algorithm.

This paper is an extension of our previous work [10]. Our previous work presented a basic algorithm to efficiently update the (generalized) eigenvalue system given a small change of the data set. It approximates the increments of eigenvalues and eigenvectors with first order error. Based on our previous work, this paper gives a second order approximation for the increments by alternately refining the eigenvalues and eigenvectors, respectively. Then more experiments are carried to show that the refinement algorithm achieves

significant improvement over our previous work. In this version, our algorithm is also applied to some other related problems involving dynamic graphs, which demonstrates the wide applicability of our incremental algorithm. Besides these, discussions on the number of clusters, more related work, and some other content are added in this paper to complement the previous version. The contributions of our work are summarized as follows:

1. We declare two kinds of dynamics existing in the evolving data: similarity change and insertion/deletion of data points. And then the incidence vector/matrix is introduced to represent the two dynamics so that our incremental clustering can handle them in the same framework.
2. Based on (but not limited to) normalized cut, the incremental spectral clustering is formulated as the problem of dynamically updating the eigen-system given a small similarity change. We give a closed-form solution to the eigenvalue increment with first order error and an approximate solution to the eigenvector increment.
3. To improve the accuracy of the increments, we propose an iterative algorithm that alternately refines the eigenvalues and eigenvectors. It approximates the increments with the second order error.
4. Our algorithm is also applied to solve some other related problems involving dynamic graphs. This demonstrates the wide applicability of our algorithm.
5. We carry intensive experiments on the real blog data set. The incremental approach can discover not only the stable blog communities but also the evolution of the individual multi-topic blogs, while the computational cost is very low.

This paper is organized as follows. In the next section we focus on related work. Section 3 describes the basic formulations. Section 4 presents the incremental algorithm for spectral clustering. Then the algorithm is discussed in Section 5. And it is also applied to some other related problems in Section 6. Section 7 gives the experimental results. The paper is concluded in Section 8.

## 2. Related work

To the best of our knowledge, our approach is the first work accomplishing the task of incremental spectral clustering that can handle not only insertion/removal of data points but also similarity changes. But there is still a large volume of literature related to our work, including topics on spectral methods, stream data, incremental PageRank, evolutionary clustering, and time series data.

The spectral method is where our work starts. Our work is based on normalized cut [2] but can be extended, without major modifications, to many other spectral methods involving solving eigenvalue systems. Spectral clustering evolved from the theory of spectral graph partitioning, an effective algorithm in high performance computing [11]. Recently there is a huge volume of literature on this topic. Ratio cut objective function [12,13] naturally captures both mincut and equipartition, the two traditional goals of partitioning. This function leads to eigenvalue decomposition of the Laplacian matrix. Shi and Malik [2] proposed a normalized cut criterion that measures both the total dissimilarity between the different groups as well as the total similarity within the groups. The criterion is equivalent to a generalized eigenvalue problem. Ding et al. [14] presented a min-max cut and claimed that this criterion always leads to a more balanced cut than the ratio cut and the normalized cut. Unlike the above approaches, Ng et al. [1] proposed a multi-way clustering method. The data points are mapped into a new space spanned by the first  $k$  eigenvectors of the normalized Laplacian. Clustering is then performed with traditional methods

(like  $k$ -means) in this new space. A detailed comparison of different spectral clustering methods is available in [15]. However, none of the above spectral algorithms are designed to incrementally cluster dynamic data. Valgren et al. [16] applied the spectral clustering to the affinity matrix after each row/column is added and made it possible to inspect the clusters as new data points are added. But it cannot handle the case of similarity changes.

The data stream model is motivated by the emerging applications of massive data sets such as customer click stream, retail transactions, telephone records, multimedia data, and so on [5]. The data arrive at very high rate that it is impossible to fit them in the memory or scan them for multiple times as conventional clustering does. Hence it demands for efficient incremental or online methods that cluster massive data by using limited memory and by one-pass scanning. Most of such algorithms dynamically update the cluster centers [4], medoids [5], or a hierarchical tree [6] when new data points are inserted, using limited memory. Recently there is a need to cluster the data streams under multiple resolutions so that the user can query the historic data at any time period (e.g., last month, last quarter, last year) in an interactive fashion. Aggarwal et al. [17] presented the *CluStream* framework to accomplish this task. *CluStream* includes an online component that periodically computes and stores detailed summary statistics and an off-line component which, based on only the summary statistics, responds to a wide variety of inputs (time horizon or number of clusters) and returns to the analyst a quick understanding of the broad clusters. More results on data stream can be found in [18–20]. However, these methods cannot be directly applied to the scenario with similarity changes, like the evolving web/blog data.

Recently, with the success of Google [21], incremental algorithms on web data gain more and more attention, of which the PageRank metric has gained enormous popularity. For example, Desikan et al. [21] and Langville et al. [22] exploit the underlying principle of the first order Markov model on which PageRank is based, to incrementally compute PageRank for the evolving Web graph. Unlike these methods focusing on incremental PageRank, our incremental algorithm efficiently clusters evolving data, like web/blog data, by dynamically updating the eigenvalue system.

Another related research area on clustering data over time is *evolutionary clustering* which was conceptualized and formulated by Chakrabarti et al. [23] in 2006. *Evolutionary clustering* simultaneously optimizes two potentially conflicting criteria, i.e., the clustering should fit the current data as much as possible, while should not deviate dramatically from the historic context. Chakrabarti et al. [23] presented a generic framework for this problem. Chi et al. [24] stepped further to evolutionary spectral clustering by incorporating temporal smoothness. Our work differs from these methods in that, the latter deals with the high level concept of clustering data over time, while our work is more focusing on low level techniques to trade-off accuracy and efficiency.

### 3. Basic formulations

First, all about the notations in this paper. Scripted letters, such as  $\mathcal{E}$  and  $\mathcal{V}$ , represent sets. Capital letters like  $W$  and  $L$  are matrixes. Bold lower case letters indicate column vectors, e.g.,  $\mathbf{q}$  and  $\mathbf{u}$ . Other lower case letters are scalars. We use subscripts to index the elements in vectors and matrixes unless specific declaration, for example,  $q_i$  is the  $i$ -th element of vector  $\mathbf{q}$  and  $w_{ij}$  is  $W(i, j)$ . The vector norm  $\|\cdot\|$  is the  $l_2$  norm, i.e.,  $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$ . The commonly used similarity between two data points includes the inner product of the feature vectors,  $w_{ij} = \mathbf{x}_i^T \mathbf{x}_j$ , the diagonally scaled Gaussian similarity,  $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \sigma^2)$ , and the affinity matrixes of the graph.

Given a weighted graph  $G = G(\mathcal{E}, \mathcal{V}, W)$  with node set  $\mathcal{V}$ , edge set  $\mathcal{E}$  and similarity matrix  $W$  where  $w_{ij}$  indicates the similarity between

node  $v_i$  and  $v_j$ , spectral clustering partitions the graph into two or more disjoint subgraphs. Since our approach is based on, but not limited to, the normalized cut [2], this algorithm is briefly reviewed using the notation in the tutorial [11].

#### 3.1. Normalized cut

In this paper, the similarity matrix  $W$  is assumed to be symmetric with  $w_{ij} = 1$  and  $0 \leq w_{ij} \leq 1$  when  $i \neq j$ . Denote the degree matrix as  $D = \text{diag}\{d_1, d_2, \dots, d_n\}$  where

$$d_i = \sum_j w_{ij}$$

is the degree of node  $v_i$  and Laplacian matrix as  $L = D - W$ . Let the similarity between the subgraphs  $\mathcal{A}$  and  $\mathcal{B}$  be

$$s(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{B}} w_{ij}.$$

The normalized cut aims to minimize the criterion function

$$J_{\text{Ncut}}(\mathcal{A}, \mathcal{B}) = \frac{s(\mathcal{A}, \mathcal{B})}{d_{\mathcal{A}}} + \frac{s(\mathcal{A}, \mathcal{B})}{d_{\mathcal{B}}},$$

where

$$d_{\mathcal{A}} = \sum_{i \in \mathcal{A}} d_i, \quad d_{\mathcal{B}} = \sum_{i \in \mathcal{B}} d_i.$$

After some manipulations, the criterion function can be rewritten as

$$\frac{\mathbf{q}^T L \mathbf{q}}{\mathbf{q}^T D \mathbf{q}}, \quad (1)$$

where  $\mathbf{q}$  is the cluster indicator and satisfies

$$\mathbf{q}^T D \mathbf{q} = 1, \quad \mathbf{q}^T D \mathbf{1} = 0, \quad (2)$$

and ideally

$$q_i = \begin{cases} \sqrt{d_{\mathcal{B}}/d_{\mathcal{A}}d} & \text{if } i \in \mathcal{A}, \\ -\sqrt{d_{\mathcal{A}}/d_{\mathcal{B}}d} & \text{if } i \in \mathcal{B}, \end{cases} \quad (3)$$

where  $d = \sum_{i \in \mathcal{V}} d_i$ . If  $\mathbf{q}$  is relaxed to take real value, we can minimize Eq. (1) by solving the generalized eigenvalue system

$$L \mathbf{q} = \lambda D \mathbf{q}. \quad (4)$$

Given a small change of the data set, a computationally wasteful solution to obtain the new cluster labels is resolving the generalized eigenvalue system. In this paper, we presents a much more efficient algorithm that computes the increments of the eigenvalues and eigenvectors based on the old results and dynamically updates the generalized eigenvalue system.

#### 3.2. Incidence vector/matrix

As we mentioned before, in many real applications both the data points and their similarities are dynamic. Then a question arises: How to represent the two kinds of dynamics in the same framework and how to feed them into the eigenvalue system without violating the original representation? We solve this problem by introducing the *incidence vector/matrix*.

**Definition 1.** An *incidence vector*  $\mathbf{r}_{ij}(w)$  is a column vector with only two nonzero elements:  $i$ -th element equal to  $\sqrt{w}$  and  $j$ -th element  $-\sqrt{w}$ , indicating data point  $i$  and  $j$  having a similarity  $w$ . The length of the vector is equal to the number of considered data points.

**Definition 2.** An incidence matrix  $R$  is a matrix whose columns are incidence vectors.

An incidence vector can be rewritten as  $\mathbf{r}_{ij}(w) = \sqrt{w}\mathbf{u}_{ij}$  where  $\mathbf{u}_{ij}$  (adopted from [25]) is a column vector with only two nonzero element:  $i$ -th element equal to 1 and  $j$ -th element  $-1$ .  $w$  can be negative if  $\sqrt{-1}$  is allowed. An incidence vector  $\mathbf{r}_{ij}(w)$  represents the similarity  $w$  between two data points  $i$  and  $j$  (or edge  $(i,j)$ ), and the incidence matrix is another representation of the similarity matrix. The definition of incidence vector/matrix in this paper partly differs from the traditional definition [7]. Firstly the incidence vector in this paper is allowed to represent a portion of the similarity, i.e., a similarity  $w_{ij}$  can be represented by two incidence vectors  $\mathbf{r}_{ij}(w_{ij}^{(1)})$  and  $\mathbf{r}_{ij}(w_{ij}^{(2)})$  where  $w_{ij} = w_{ij}^{(1)} + w_{ij}^{(2)}$ . Secondly, the incidence matrix in this paper allows multiple columns (incidence vectors) for each edge of the graph, while in [7] each edge corresponds to one and only one column. This difference is supported by Proposition 2 and enables that a similarity change can be incorporated into the incidence matrix by appending an incidence vector.

**Proposition 1.** If  $L = D - W \in \mathbb{R}^{n \times n}$  is a Laplacian matrix, then there exists an incidence matrix  $R$  such that  $L = RR^T$  [26]. In addition,  $R$  contains all the incidence vectors  $\mathbf{r}_{ij}(w_{ij})$ ,  $1 \leq i < j \leq n$ , that can be in any order.

**Proof.** Regardless of the order in which the incidence vectors stacked in  $R$ , the product

$$\begin{aligned} RR^T &= \sum_{1 \leq i < j \leq n} \mathbf{r}_{ij}(w_{ij})\mathbf{r}_{ij}(w_{ij})^T \\ &= \sum_{1 \leq i < j \leq n} w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T. \end{aligned} \quad (5)$$

But,

$$\mathbf{u}_{ij}\mathbf{u}_{ij}^T = \begin{pmatrix} \vdots & \vdots \\ \cdots & 1 & \cdots & -1 & \cdots \\ \vdots & \vdots \\ \cdots & -1 & \cdots & 1 & \cdots \\ \vdots & \vdots \end{pmatrix}. \quad (6)$$

Add all of the matrixes  $w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T$ ,  $1 \leq i < j \leq n$  together, and it follows that

$$\sum_{1 \leq i < j \leq n} w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T = L. \quad \square$$

**Proposition 2.** Assume  $L = RR^T$  where  $L$  is the graph Laplacian and  $R$  is an incidence matrix. If data points  $i$  and  $j$  have a similarity change  $\Delta w_{ij}$  corresponding to the incidence vector  $\mathbf{r}_{ij}(\Delta w_{ij})$ , the new graph Laplacian  $\tilde{L}$  can be decomposed as  $\tilde{L} = \tilde{R}\tilde{R}^T$  where  $\tilde{R} = [R, \mathbf{r}_{ij}(\Delta w_{ij})]$ .

**Proof.** According to the fact that  $\mathbf{r}_{ij}(w) = \sqrt{w}\mathbf{u}_{ij}$ ,

$$\begin{aligned} \tilde{R}\tilde{R}^T &= RR^T + \Delta w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T \\ &= L + \Delta w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T. \end{aligned} \quad (7)$$

Considering  $L = D - W$  and Eq. (6), the increment of  $L$  induced by the similarity change  $\Delta w_{ij}$  is exactly  $\Delta w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T$ . So

$$\tilde{L} = L + \Delta w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T = \tilde{R}\tilde{R}^T. \quad \square$$

Firstly we consider a single similarity change. According to Proposition 2, a similarity change  $\Delta w_{ij}$  can be incorporated into the incidence matrix  $R$  by appending the incidence vector  $\mathbf{r}_{ij}(\Delta w_{ij})$  to  $R$ .

Therefore, the increment of  $L$  with respect to  $\Delta w_{ij}$  can be easily deduced:

$$\Delta L = \tilde{L} - L = \Delta w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T. \quad (8)$$

By observation, the increment of the degree matrix  $D$  is

$$\Delta D = \Delta w_{ij} \text{diag}\{\mathbf{v}_{ij}\}, \quad (9)$$

where  $\mathbf{v}_{ij}$  is a column vector with  $i$ -th and  $j$ -th elements equal to 1 and other elements equal to 0.

As to insertion/deletion of a data point, it can be decomposed into a sequence of similarity changes (incidence vectors) appended to the original incidence matrix. For instance, when a data point  $i$ , which has similarities  $w_{ij_1}, w_{ij_2}, \dots, w_{ij_k}$ , is added, it is equivalent to a sequence of similarity changes of  $w_{ij_1}, w_{ij_2}, \dots, w_{ij_k}$  occurring in any order, corresponding to a sequence of incidence vectors  $\mathbf{r}_{ij_1}(w_{ij_1}), \mathbf{r}_{ij_2}(w_{ij_2}), \dots, \mathbf{r}_{ij_k}(w_{ij_k})$ .

#### 4. Incremental spectral clustering

In Section 3.2, any updating of the dynamic data is equivalent to a (or a sequence of) incidence vector(s)  $\mathbf{r}_{ij}(\Delta w_{ij})$  appended to the original incidence matrix  $R$ . Here we approximate the increments of the eigenvalues and eigenvectors in the spectral clustering, induced by the updating  $\mathbf{r}_{ij}(\Delta w_{ij})$ . The approximation is carried on, but not limited to, the generalized eigenvalue system of the normalized cut,  $L\mathbf{q} = \lambda D\mathbf{q}$  [2].

##### 4.1. Increment of eigenvalue $\Delta\lambda$

There is a closed-form solution to the eigenvalue increment of a symmetric generalized eigenvalue system.

**Proposition 3.** Suppose  $A\mathbf{x} = \lambda B\mathbf{x}$  is a generalized eigenvalue system where both  $A \in \mathbb{R}^{n \times n}$  and  $B \in \mathbb{R}^{n \times n}$  are symmetric, then the perturbation of  $\lambda$  in terms of the perturbations of  $A$  and  $B$  is

$$\Delta\lambda = \frac{\mathbf{x}^T(\Delta A - \lambda\Delta B)\mathbf{x}}{\mathbf{x}^T B \mathbf{x}}. \quad (10)$$

**Proof.** Differentiate both sides of the generalized eigenvalue system  $A\mathbf{x} = \lambda B\mathbf{x}$ ,

$$\Delta A\mathbf{x} + A\Delta\mathbf{x} = \Delta\lambda B\mathbf{x} + \lambda\Delta B\mathbf{x} + \lambda B\Delta\mathbf{x}. \quad (11)$$

Left multiply both sides by  $\mathbf{x}^T$  and obtain

$$\mathbf{x}^T \Delta A\mathbf{x} + \mathbf{x}^T A\Delta\mathbf{x} = \Delta\lambda \mathbf{x}^T B\mathbf{x} + \lambda \mathbf{x}^T \Delta B\mathbf{x} + \lambda \mathbf{x}^T B\Delta\mathbf{x}. \quad (12)$$

Since

$$\mathbf{x}^T A = \lambda \mathbf{x}^T B$$

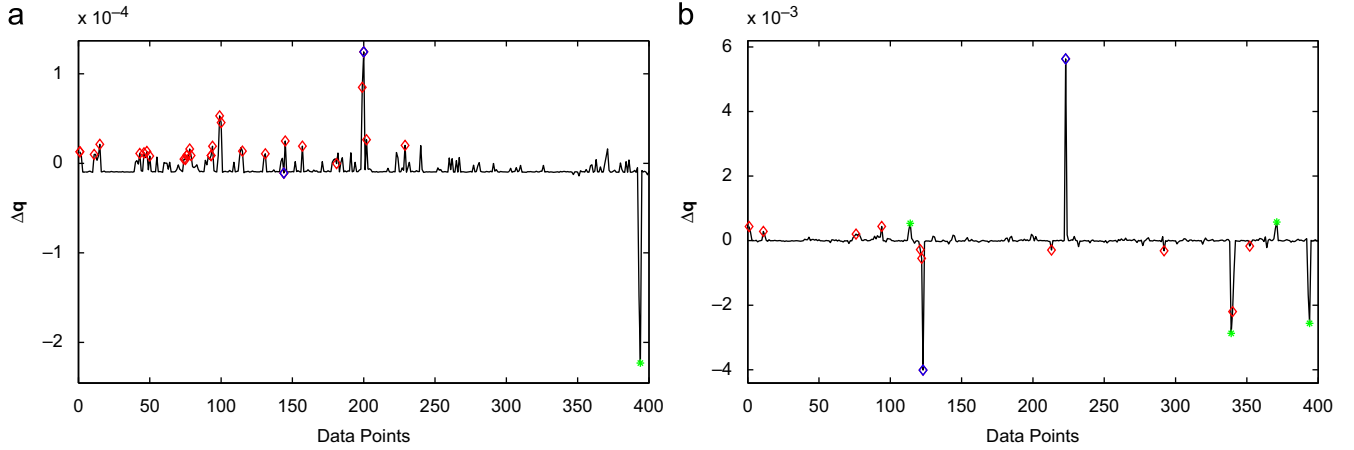
because  $A$  and  $B$  are symmetric, Eq. (12) can be rewritten as

$$\mathbf{x}^T \Delta A\mathbf{x} = \Delta\lambda \mathbf{x}^T B\mathbf{x} + \lambda \mathbf{x}^T \Delta B\mathbf{x}.$$

After a few manipulations, we obtain Eq. (10).  $\square$

Suppose the updating is the incidence vector  $\mathbf{r}_{ij}(\Delta w_{ij}) = \sqrt{\Delta w_{ij}}\mathbf{u}_{ij}$ . Substitute  $\Delta L$  in Eq. (8),  $\Delta D$  in Eq. (9), and  $\mathbf{q}$  for  $\Delta A$ ,  $\Delta B$ , and  $\mathbf{x}$  in Eq. (10), respectively, then  $\Delta\lambda$  of the generalized eigenvalue system  $L\mathbf{q} = \lambda D\mathbf{q}$  is

$$\Delta\lambda = \Delta w_{ij} \frac{\mathbf{q}^T (\mathbf{u}_{ij}\mathbf{u}_{ij}^T - \lambda \text{diag}\{\mathbf{v}_{ij}\}) \mathbf{q}}{\mathbf{q}^T D \mathbf{q}}.$$



**Fig. 2.**  $\Delta \mathbf{q}$  induced by a similarity change of two points (marked blue). A red mark indicates that the point is directly adjacent to the two points; and a green mark means a leaf in the graph. (a) Two points are in the same cluster. (b) Two points are in different clusters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

But

$$\mathbf{q}^T \mathbf{u}_{ij} \mathbf{u}_{ij}^T \mathbf{q} = (q_i - q_j)^2, \quad (13)$$

$$\mathbf{q}^T \text{diag}\{\mathbf{v}_{ij}\} \mathbf{q} = q_i^2 + q_j^2, \quad (14)$$

and

$$\mathbf{q}^T D \mathbf{q} = 1$$

because of the normalization in Eq. (2),  $\Delta \lambda$  can be expressed as

$$\Delta \lambda = \Delta w_{ij} ((q_i - q_j)^2 - \lambda (q_i^2 + q_j^2)). \quad (15)$$

$\Delta \lambda$  can be further simplified if we assume that  $\mathbf{q}$  is as ideal as in Eq. (3), two clusters have nearly the same degrees ( $d_{\mathcal{A}} \approx d_{\mathcal{B}}$  in Section 3.1), and  $\lambda \ll 1$  (it usually holds for the top smallest eigenvalues),

$$\Delta \lambda \approx \begin{cases} -2\lambda \frac{\Delta w_{ij}}{d}, & i, j \text{ in the same cluster,} \\ 4\lambda \frac{\Delta w_{ij}}{d}, & i, j \text{ in different clusters.} \end{cases} \quad (16)$$

From the approximation, increasing the similarity of two points in the same cluster decreases the eigenvalue  $\lambda$ , and  $\lambda$  increases if they are in different clusters. In addition, the increase is much greater in magnitude than the decrease because usually the top smallest eigenvalues  $\lambda \ll 1$ .

#### 4.2. Increment of eigenvector $\Delta \mathbf{q}$

Generally the increment of a eigenvector  $\Delta \mathbf{q}$  can be solved by power iteration or Lanczos method [8]. These methods run very fast on sparse matrixes, but it is still impractical for huge matrixes such as web data (see discussions in Section 5). Fortunately, clustering requires only discrete cluster labels and precise eigenvectors are not necessary. Thus we adopt an approximate approach to fast compute  $\Delta \mathbf{q}$  so it can be applied to huge and evolving data, partly at the expense of accuracy.

Substitute  $\Delta L$  in Eq. (8),  $\Delta D$  in Eq. (9), and  $\mathbf{q}$  for  $\Delta A$ ,  $\Delta B$ , and  $\mathbf{x}$  in Eq. (11), respectively, and after some manipulations, we obtain a linear equation for  $\Delta \mathbf{q}$

$$K \Delta \mathbf{q} = \mathbf{h}, \quad (17)$$

where

$$K = L - \lambda D, \quad (18)$$

$$\mathbf{h} = (\Delta \lambda D + \lambda \Delta D - \Delta L) \mathbf{q}. \quad (19)$$

Since  $\Delta L$ ,  $\Delta D$ , and  $\Delta \lambda$  are known according to Eqs. (8), (9) and (15),  $\Delta \mathbf{q}$  can be solved by

$$\Delta \mathbf{q} = (K^T K)^{-1} K^T \mathbf{h} \quad (20)$$

if  $K^T K$  is nonsingular.

However, it is impractical to directly apply Eq. (17) or (20) to solve  $\Delta \mathbf{q}$ . Firstly,  $K$  and  $K^T K$  are singular because  $L \mathbf{q} = \lambda D \mathbf{q}$ , i.e.,  $K \mathbf{q} = 0$ . Secondly, and more importantly, the size of  $K$  and  $K^T K$  is huge for large data sets, which means that solving Eq. (17) requires very high computational cost, even higher than that of the Lanczos method [8].

Therefore, we adopt an approximate approach by fully exploiting the properties of the application, and even partly at the expense of the accuracy of  $\Delta \mathbf{q}$  since only discrete cluster labels are needed. As we know, a similarity change of two data points  $i$  and  $j$  has little influence on the data points far from  $i$  and  $j$ . In other words, to maintain the cluster structure of the graph, only  $i$  and  $j$  and a few points close to them need to be considered. Fig. 2 gives two examples of  $\Delta \mathbf{q}$  induced by a similarity change on the data set of web-blogs (see Section 7). Here the eigenvectors are solved by Lanczos method [8] and  $\Delta \mathbf{q}$  by Eq. (A.2) in Appendix A. In Fig. 2, points  $i$  and  $j$  are marked blue, a red mark indicates that the point is directly adjacent to  $i$  or  $j$ , and a green mark means a leaf in the graph. It shows that most of the spikes are either adjacent to  $i$  or  $j$  or leaves. The issue of leaves will be discussed in Section 5.

Given a similarity change  $r_{ij}(\Delta w_{ij})$ , let  $\mathcal{N}_{ij} = \{k | w_{ik} > \tau \text{ or } w_{jk} > \tau\}$  be the spatial neighborhood of both  $i$  and  $j$ , where  $\tau$  is a predefined threshold which can take 0 for sparse data set. In accordance with the above observations, we assume  $\Delta q_k = 0$  if  $k \notin \mathcal{N}_{ij}$  and eliminate them from  $\Delta \mathbf{q}$ , and accordingly the corresponding columns in  $K$  are also eliminated. After elimination of the elements (of both  $\Delta \mathbf{q}$  and  $K$ ) that are not in the spatial neighborhood  $\mathcal{N}_{ij}$ , the left elements form  $\Delta \mathbf{q}_{ij}$  and  $K_{\mathcal{N}_{ij}}$ . Thus we obtain

$$\Delta \mathbf{q}_{ij} = (K_{\mathcal{N}_{ij}}^T K_{\mathcal{N}_{ij}})^{-1} K_{\mathcal{N}_{ij}}^T \mathbf{h}. \quad (21)$$

Since the number of the columns of  $K_{\mathcal{N}_{ij}}$  is very small compared with the size of the data set, the computational cost of the above equation is very low.

#### 4.3. Iterative refinement of $\Delta \lambda$ and $\Delta \mathbf{q}$

Eq. (10) or (15) is the first order approximation of the eigenvalue increment and ignores the second order error. But in some cases the



second order error is not neglectable, so here we give a more accurate approximation of the eigenvalue increment by considering the second and third order error and by using the existing eigenvector increment. And, in turn, the eigenvector increment can be further refined by using the more accurate eigenvalue increment. Alternately repeating this procedure, we come up with an iterative algorithm to refine  $\Delta\lambda$  and  $\Delta\mathbf{q}$ .

**Proposition 4.** Suppose  $\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x}$  is a generalized eigenvalue system where both  $\mathbf{A} \in \mathbb{R}^{n \times n}$  and  $\mathbf{B} \in \mathbb{R}^{n \times n}$  are symmetric, then the second order approximation of the perturbation  $\Delta\lambda$  in terms of the perturbations of  $\mathbf{A}$  and  $\mathbf{B}$  is

$$\Delta\lambda = \frac{\mathbf{x}^T(\Delta\mathbf{A} - \lambda\Delta\mathbf{B})(\mathbf{x} + \Delta\mathbf{x})}{\mathbf{x}^T(\mathbf{B} + \Delta\mathbf{B})(\mathbf{x} + \Delta\mathbf{x})}. \quad (22)$$

**Proof.** Differentiate both sides of the generalized eigenvalue system  $\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x}$  and keep the second and third order error items,

$$\Delta\mathbf{A}\mathbf{x} + \mathbf{A}\Delta\mathbf{x} + \Delta\mathbf{A}\Delta\mathbf{x} = \Delta\lambda\mathbf{B}\mathbf{x} + \lambda\Delta\mathbf{B}\mathbf{x} + \lambda\mathbf{B}\Delta\mathbf{x} + \Delta\lambda\Delta\mathbf{B}\mathbf{x} + \lambda\Delta\mathbf{B}\Delta\mathbf{x} + \Delta\lambda\Delta\mathbf{B}\Delta\mathbf{x}. \quad (23)$$

Left multiply both sides by  $\mathbf{x}^T$  and obtain

$$\mathbf{x}^T\Delta\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}\Delta\mathbf{x} + \mathbf{x}^T\Delta\mathbf{A}\Delta\mathbf{x} = \Delta\lambda\mathbf{x}^T\mathbf{B}\mathbf{x} + \lambda\mathbf{x}^T\Delta\mathbf{B}\mathbf{x} + \lambda\mathbf{x}^T\mathbf{B}\Delta\mathbf{x} + \Delta\lambda\mathbf{x}^T\Delta\mathbf{B}\mathbf{x} + \lambda\mathbf{x}^T\Delta\mathbf{B}\Delta\mathbf{x} + \Delta\lambda\mathbf{x}^T\Delta\mathbf{B}\Delta\mathbf{x}. \quad (24)$$

Since

$$\mathbf{x}^T\mathbf{A} = \lambda\mathbf{x}^T\mathbf{B}$$

because  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric,  $\mathbf{x}^T\mathbf{A}\Delta\mathbf{x}$  and  $\lambda\mathbf{x}^T\mathbf{B}\Delta\mathbf{x}$  can be cancelled from both sides of Eq. (24),

$$\mathbf{x}^T\Delta\mathbf{A}\mathbf{x} + \mathbf{x}^T\Delta\mathbf{A}\Delta\mathbf{x} = \Delta\lambda\mathbf{x}^T\mathbf{B}\mathbf{x} + \lambda\mathbf{x}^T\Delta\mathbf{B}\mathbf{x} + \Delta\lambda\mathbf{x}^T\Delta\mathbf{B}\mathbf{x} + \lambda\mathbf{x}^T\Delta\mathbf{B}\Delta\mathbf{x} + \Delta\lambda\mathbf{x}^T\Delta\mathbf{B}\Delta\mathbf{x}.$$

After a few manipulations, we obtain Eq. (22).  $\square$

Suppose the similarity change is the incidence vector  $\mathbf{r}_{ij}(\Delta w_{ij}) = \sqrt{\Delta w_{ij}}\mathbf{u}_{ij}$ . Substitute  $\Delta\mathbf{L}$  in Eq. (8),  $\Delta\mathbf{D}$  in Eq. (9), and  $\mathbf{q}$  for  $\Delta\mathbf{A}$ ,  $\Delta\mathbf{B}$ , and  $\mathbf{x}$  in Eq. (22), respectively, then  $\Delta\lambda$  of the generalized eigenvalue system  $\mathbf{L}\mathbf{q} = \lambda\mathbf{D}\mathbf{q}$  is

$$\Delta\lambda = \Delta w_{ij} \frac{\mathbf{q}^T(\mathbf{u}_{ij}\mathbf{u}_{ij}^T - \lambda \text{diag}\{\mathbf{v}_{ij}\})(\mathbf{q} + \Delta\mathbf{q})}{\mathbf{q}^T(\mathbf{D} + \Delta w_{ij} \text{diag}\{\mathbf{v}_{ij}\})(\mathbf{q} + \Delta\mathbf{q})}.$$

Like the deduction of Eq. (15), the second order approximation of  $\Delta\lambda$  can be expressed as

$$\Delta\lambda = \Delta w_{ij} \frac{a + b}{1 + c + d + e},$$

where

$$a = (q_i - q_j)^2 - \lambda(q_i^2 + q_j^2),$$

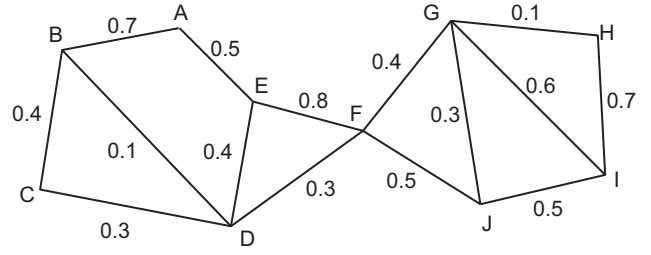
$$b = (q_i - q_j)(\Delta q_i - \Delta q_j) - \lambda(q_i\Delta q_i + q_j\Delta q_j),$$

$$c = \Delta w_{ij}(q_i^2 + q_j^2),$$

$$d = \sum_{k \in \mathcal{N}_{ij}} q_k d_k \Delta q_k,$$

$$e = \Delta w_{ij}(q_i\Delta q_i + q_j\Delta q_j).$$

Here  $\sum_{k \in \mathcal{N}_{ij}} q_k d_k \Delta q_k$  is an approximation of  $\mathbf{q}^T \mathbf{D} \Delta \mathbf{q}$  with the assumption that the impact of the similarity change  $\Delta w_{ij}$  is within the



**Fig. 3.** A graph example with 10 nodes labelled from A to J. The similarity associated with each edge is also marked on the graph.

spatial neighborhood  $\mathcal{N}_{ij}$ .  $e$  can be ignored since  $e \ll 1$  and  $e \ll c$ , so  $\Delta\lambda$  is simplified as

$$\Delta\lambda = \Delta w_{ij} \frac{a + b}{1 + c + d}. \quad (25)$$

However, the second order approximation of  $\Delta\lambda$  in Eq. (25) depends on the eigenvector increment  $\Delta\mathbf{q}$  in Eq. (21), and  $\Delta\mathbf{q}$  also depends on  $\Delta\lambda$ . Fortunately we can initialize  $\Delta\lambda$  by Eq. (15) and then refine  $\Delta\lambda$  and  $\Delta\mathbf{q}$  alternately. The iterative refinement algorithm is summarized in Algorithm 1.

**Algorithm 1.** Iterative refinement of  $\Delta\lambda$  and  $\Delta\mathbf{q}$ .

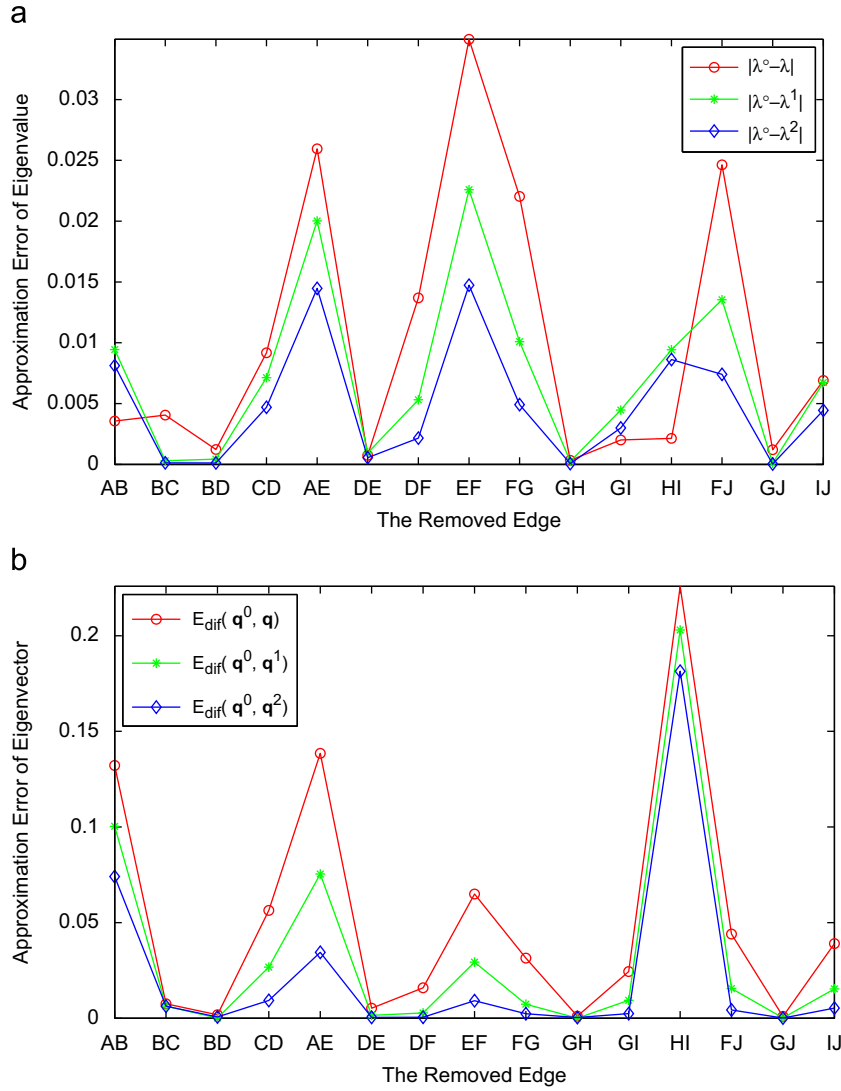
- 1: Set  $\Delta\mathbf{q} = \mathbf{0}$ .
- 2: Compute  $\Delta\lambda$  by Eq. (25), using the existing  $\Delta\mathbf{q}$ .
- 3: Compute  $\Delta\mathbf{q}$  by Eq. (21), using the existing  $\Delta\lambda$ .
- 4: Repeat steps 2 and 3 until  $\Delta\lambda$  and  $\Delta\mathbf{q}$  have no significant change or for at most  $n$  iterations.

However, the convergence property of Algorithm 1 is still unknown. Large experiments show that the approximation error of both eigenvalues and eigenvectors basically decrease when the iteration times  $n \leq 2$ . Here we use a simple example to illustrate it. Fig. 3 is a small graph and its second smallest eigenvalue is  $\lambda$  and the associated eigenvector is  $\mathbf{q}$ . Remove one edge in the graph and compute the new eigenvalue and eigenvector by three methods: SVD decomposition, Algorithm 1 with one iteration, and with two iterations. The results are denoted as  $\lambda^0, \mathbf{q}^0, \lambda^1, \mathbf{q}^1, \lambda^2, \mathbf{q}^2$ , respectively. And  $\lambda^0$  and  $\mathbf{q}^0$  serve as ground truth. Fig. 4 shows the approximation error of the second smallest eigenvalue and the associated eigenvector after removing only one edge (see the ticks on the  $x$ -axis). Fig. 4(a) gives the eigenvalue error without approximation  $|\lambda^0 - \lambda|$ , after one iteration  $|\lambda^0 - \lambda^1|$ , and after two iterations  $|\lambda^0 - \lambda^2|$ , respectively. Fig. 4(b) gives the eigenvector error without approximation  $E_{dif}(\mathbf{q}^0, \mathbf{q})$ , after one iteration  $E_{dif}(\mathbf{q}^0, \mathbf{q}^1)$ , and after two iterations  $E_{dif}(\mathbf{q}^0, \mathbf{q}^2)$ , respectively. Basically the error is decreasing from no approximation to two iterations with a few exceptions. The error between two eigenvectors  $E_{dif}(\cdot, \cdot)$  is computed by Eq. (A.1) in Appendix A.

Summarize Sections 4.1–4.3, and the incremental spectral clustering is briefed in Algorithm 2.

**Algorithm 2.** Incremental spectral clustering.

- 1: Suppose at time  $t$ , the data set grows large enough and the similarity matrix  $W$ , degree matrix  $D$ , and Laplacian matrix  $L$  are available.
- 2: Solve Eq. (4) as standard spectral clustering does for eigenvectors with the smallest eigenvalues. This solution and the matrixes ( $D$  and  $L$ ) serve as the initialization.



**Fig. 4.** Approximation error of the second smallest eigenvalue and the associated eigenvector after removing only one edge (see the ticks on the x-axis). (a) Eigenvalue error without approximation  $|\lambda^0 - \lambda|$ , after one iteration  $|\lambda^0 - \lambda^1|$ , and after two iterations  $|\lambda^0 - \lambda^2|$ . (b) Eigenvector error without approximation  $E_{diff}(\mathbf{q}^0, \mathbf{q})$ , after one iteration  $E_{diff}(\mathbf{q}^0, \mathbf{q}^1)$ , and after two iterations  $E_{diff}(\mathbf{q}^0, \mathbf{q}^2)$ .

- 3: From then on, when a similarity change occurs, use Algorithm 1 to update the eigenvalues and eigenvectors and Eqs. (8) and (9) to update  $D$  and  $L$ .
- 4: If a data point is added or deleted, it is decomposed into a sequence of similarity changes and step 3 is repeatedly conducted.
- 5: After  $T$  similarity changes occur, re-initialize the spectral clustering by repeating from step 2, so as to avoid large accumulated errors.

### 5. Discussions on the incremental algorithm

Firstly, the time complexity is discussed. The generalized eigenvalue system can be transformed into a standard eigenvalue problem of  $D^{-1/2}LD^{-1/2}\mathbf{q} = \lambda\mathbf{q}$ , and solving a standard eigenvalue problem takes  $O(n^3)$  operations [2], where  $n$  is the number of data points. When  $n$  is as large as in the Web applications, this is impractical. Fortunately, Lanczos method [8] can greatly reduce the computational cost if the similarity matrix  $W$  is sparse. And Web application is such an example. The time complexity of the Lanczos algorithm

is  $O(n^{3/2})$  if the special properties of  $W$  is fully exploited [2]. This is also the running time of the baseline system in Section 7.

However, the computational cost is still very high if the data set is large and undergoes frequent evolution as the web-blogs be. In this case, it is hard for a standard spectral clustering to update the cluster labels immediately after the data set is updated. On the contrary, our incremental approach may success. It needs constant running time to compute  $\Delta\lambda$  and  $O(\bar{N}^2 n) + O(\bar{N}^3) + O(\bar{N}n) + O(\bar{N}^2)$  to compute  $\Delta\mathbf{q}$ , where  $\bar{N}$  is the average size of the spatial neighborhood of a node,  $O(\bar{N}^2 n)$  is needed to compute  $A = K_{N_{ij}}^T K_{N_{ij}}$  in Eq. (21),  $O(\bar{N}^3)$  for inversion of  $A$ ,  $O(\bar{N}n)$  for  $b = K_N^T \mathbf{h}$ , and  $O(\bar{N}^2)$  for  $A^{-1}b$ . In the Web applications,  $\bar{N}$  is usually constant and small, so the running time of the incremental approach is  $O(n)$ . The  $\bar{N}$  can be adjusted by tuning the threshold  $\tau$ , so the time complexity is tunable to some extent at the expense of accuracy.

Secondly, the influence of the leaves on the eigenvectors need to be explained. Suppose that node  $i$  is a leaf that is connected to node  $j$  and  $j$  is only connected to node  $k$  and  $i$ , i.e.,  $i - j - k - \dots$ .

As discussed in Appendix B,  $q_i$ ,  $q_j$ ,  $\Delta q_i$  and  $\Delta q_j$  may be spikes (see Fig. 2). However, their influence on other nodes may decay greatly because they have only one edge  $(j, k)$  connected the other part of the graph. Furthermore, since the leaves usually do not lie on the boundaries between the clusters, i.e., they are usually not supporting vectors, they should be trivial in graph partitioning. Therefore, we ignore their influence in Eq. (21).

In our discussions so far, we have assumed that the number of clusters is manually set and does not change with time. However, it is very likely some old clusters disappear and some new ones show up over a period. How to monitor such live/death information could be interesting to many applications on evolving data. This can be decomposed into two sub-problems: determination of the number of clusters and tracking of the clusters. The latter is not a focus of this paper. The interested readers are referred to [27]. Automatic determination of the number of clusters is an important and tough research problem in clustering. In spectral clustering, a tricky approach is to search for a drastic drop (where gradient is greater than a pre-defined threshold) in the magnitude of the eigenvalues [28]. Zelnik-Manor et al. [29] proposed an alternative approach with higher accuracy which relies on analyzing the structure of the eigenvectors. But this approach requires more computational cost. This paper uses the first approach to select and monitor the number of clusters.

It is worth to mention a technical problem that some similarity changes may change the size of the matrixes  $W$ ,  $D$ , and  $L$  because these similarity changes involve insertion/deletion of data points. Fortunately, this can be handled in our framework. When a similarity change induces insertion of a new data point, the matrixes  $W$ ,  $D$ , and  $L$  are expanded by padding one row and one column of zeros, and then the eigenvalues and eigenvectors are updated in the regular way. When a similarity change induces deletion of an old data point, first update the eigenvalues and eigenvectors, and then remove the corresponding row and column from the matrixes  $W$ ,  $D$ , and  $L$ .

Finally, it is important to point out the limitations of our incremental approach and their possible solutions. (1) The error is accumulating though growing slowly. This is also a critical problem in many other incremental algorithms. We use re-initialization to avoid a collapse. (2) When the data set grows larger or evolves more frequently, e.g., if all the blogs on the internet are considered, the  $O(n)$  complexity probably makes our incremental algorithm fail. One possible  $O(1)$  solution is to propagate the influence of a similarity change to its spatial neighborhood. And then the eigenvector is updated according to the received influence. This is part of our future work.

## 6. Relation to other problems

The essential idea of our incremental clustering algorithm is the dynamic updating of the (generalized) eigenvalue system given a small change of the graph. This idea can also be applied to many other related problems involving graph evolution. Three related problems are described as follows.

### 6.1. Algebraic connectivity

The algebraic connectivity of a graph  $G$  is the second smallest eigenvalue (denoted as  $\lambda_2$ ) of its Laplacian matrix  $L$  (Note that here  $\lambda_2$  is the eigenvalue of  $L\mathbf{q} = \lambda\mathbf{q}$  but not  $L\mathbf{q} = \lambda D\mathbf{q}$ ) and is a measure of how well connected the graph is [9]. It has direct connection to the number of connected components:  $\lambda_2 > 0$  if and only if the graph is connected [9]. It also reveals the sparsity of cuts in the graph: a graph with large  $\lambda_2$  cannot have sparse cuts and conversely a small  $\lambda_2$  means sparse cuts [30]. The algebraic connectivity has also been used as an important parameter in many system problems, especially

as a measure of stability and robustness of the networked dynamic systems [31].

A well-known problem with algebraic connectivity is adding edges from a set of candidate edges to a graph so as to maximize its algebraic connectivity. The problem can be formulated as follows [30]. Given a base graph  $G_{base} = (\mathcal{V}, \mathcal{E}_{base}, W)$ , a set of  $m$  candidate edges  $\mathcal{E}_{cand}$  on  $\mathcal{V}$ ,  $\mathcal{E}_{base} \cap \mathcal{E}_{cand} = \emptyset$ , its corresponding similarity matrix  $W_{cand}$ , and a number  $k, 0 \leq k < m$ , choose at most  $k$  edges from  $\mathcal{E}_{cand}$  to  $G$  so as to maximize the algebraic connectivity. That is to solve the optimization problem

$$\begin{aligned} & \text{maximize} && \lambda_2(L(\mathcal{E}_{base} \cup \mathcal{E})) \\ & \text{subject to} && |\mathcal{E}| \leq k, \\ & && \mathcal{E} \subseteq \mathcal{E}_{cand}, \end{aligned} \quad (26)$$

This problem is combinational and can be solved by exhaustive search over all  $\binom{m}{k}$  subsets of  $\mathcal{E}_{cand}$ , i.e., by computing  $\lambda_2$  for  $\binom{m}{k}$  Laplacian matrices. However, this is impractical for large  $m$  and  $k$ . Here we come up with a greedy search solution. As discussed in Section 4, adding an edge  $e_{ij} \in \mathcal{E}_{cand}$  (equivalent to a similarity change  $w_{ij}$ ) induces increments on eigenvalues and eigenvectors. But in this case, the degree matrix  $D$  is ignored, and the eigenvalue system  $L\mathbf{q} = \lambda\mathbf{q}$  is considered. And the increment of the second smallest eigenvalue and the associated eigenvector can be approximated in a similar way as Eqs. (15) and (21)

$$\Delta\lambda = w_{ij}(q_i - q_j)^2, \quad (27)$$

$$\Delta\mathbf{q}_{ij} = (K_{\mathcal{N}_{ij}}^T K_{\mathcal{N}_{ij}})^{-1} K_{\mathcal{N}_{ij}}^T \mathbf{h}, \quad (28)$$

where  $\mathbf{q}^T \mathbf{q} = 1$ ,  $K = L - \lambda\mathbf{I}$  and  $\mathbf{h} = (\Delta\lambda\mathbf{I} - \Delta L)\mathbf{q}$ .

The greedy search solution picks one edge at a time, each time selecting the edge that induces the largest increment of the second smallest eigenvalue  $\Delta\lambda_2$ . The solution is described in Algorithm 3 step by step.

**Algorithm 3.** Maximizing algebraic connectivity.

- 1: Find  $\lambda_2(L)$  and the associated eigenvector  $\mathbf{q}$  (in unit length) of the Laplacian  $L$  of the base graph.
- 2: From the remaining candidate edges, select the edge  $(i, j)$  with the largest  $\Delta\lambda_2$  and add to the graph.
- 3: Update  $\lambda_2$  and  $\mathbf{q}$  by Eqs. (27) and (28).
- 4: Repeat from step 2 until  $k$  edges are added.

This solution is similar to that in [30] in the aspect of greedy search. But the latter requires to recompute the eigenvector corresponding to  $\lambda_2(L)$  each time adding an edge. While our solution incrementally updates the eigenvector so it requires much lower computational cost.

### 6.2. The most significant edge

A similar problem is finding the most significant edge  $e_s$  in a graph  $G = (\mathcal{V}, \mathcal{E}, W)$ . The most significant edge has the greatest impact on the graph structure. It has wide applications in computer network, highway layout, city planning, and so on. Removing the most significant edge  $e_s$  results in heavy traffic in the computer network or on the highway.  $e_s$  can be defined via the second smallest eigenvalue of the generalized eigenvalue system  $L\mathbf{q} = \lambda D\mathbf{q}$ ,

$$e_s = \underset{e}{\operatorname{argmin}} \lambda_2(\mathcal{E} - e), \quad (29)$$

where  $\mathcal{E} - e$  means removing edge  $e$  from edge set  $\mathcal{E}$ . A basic solution is to find the maximum decrement of  $\lambda_2$  after removing one edge. With Eq. (15) or Algorithm 1, this can be done in time complexity  $O(m)$  where  $m$  is the number of edges. We apply this method to the



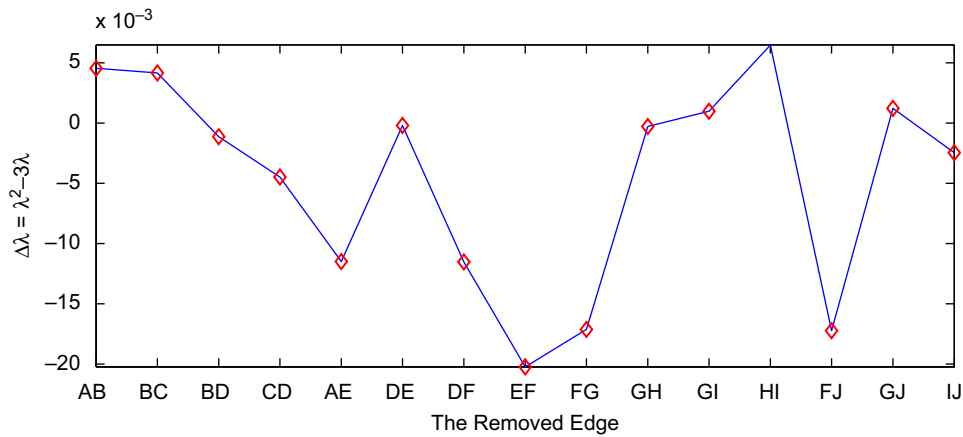


Fig. 5. The decrements of  $\lambda_2$  after removing one and only one edge in the graph in Fig. 3.

graph in Fig. 3 and obtain the decrements of  $\lambda_2$  illustrated in Fig. 5. It shows that *EF* is the most significant edge and *FJ* is the second.

### 6.3. Linear embedding

The problem of embedding a graph  $G = (\mathcal{V}, \mathcal{E}, W)$  into a real line is defined as: find a real vector  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  with zero mean and unit variance for  $n$  nodes  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ , such that the sum of the weighted square distances between adjacent nodes is minimized, i.e., the linear embedding problem is

$$\begin{aligned} & \text{maximize} && \sum_{1 \leq i, j \leq n} w_{ij} (x_i - x_j)^2 \\ & \text{subject to} && \mathbf{1}^T \mathbf{x} = 0, \\ & && \mathbf{x}^T \mathbf{x} = 1. \end{aligned} \quad (30)$$

The objective function in Eq. (30) has a matrix formulation  $\mathbf{x}^T L \mathbf{x}$ . It is minimized at the stationary point of the Lagrangian:  $\mathbf{x}^T L \mathbf{x} - \lambda(\|\mathbf{x}\| - 1)$  where  $\lambda$  is the Lagrangian multiplier. In other words, the Lagrangian has zero gradient at the optimal solution:  $2L\mathbf{x} - 2\lambda\mathbf{x} = 0$ . So the optimal value is the second smallest eigenvalue  $\lambda_2(L)$ , and the optimal solution is the associated eigenvector. A challenging problem in linear embedding is: given a set of  $m$  candidate edges  $\mathcal{E}_{cand}$  on  $\mathcal{V}$ , choose  $k < m$  edges from  $\mathcal{E}_{cand}$  and add to the graph, such that the objective function in Eq. (30) leads to smallest increase. Algorithm 3 can also be applied to this problem, except that the edge with the smallest  $\Delta\lambda_2$  is chosen at step 2.

## 7. Experiments

Our algorithm is designed to handle the sparse (or nearly sparse) data sets with two kinds of dynamics in the same framework. Web-blogs are such kind of data. Recently, web-blogs have become a prominent media on the internet that enable the users (bloggers) to publish, update, and access the personalized content. Bloggers frequently interact with each other by trackback, adding comments to, or by referring to the entries in other blogs. The reference links, comments, and trackback grow continuously, new blogs may be created, and old blogs may be deleted. Therefore web-blogs are evolving with the time. Simultaneously, the virtual blog communities may emerge through the bloggers' interactions and evolve as the web-blogs updating.

There is a volume of literature on the research of virtual community [32,33]. Rheingold [32] defined virtual community as "social aggregations that emerge from the Net when enough people carry on

those public discussions long enough, with sufficient human feeling, to form webs of personal relationship in cyberspace". It should involve long-term and meaningful conversations in cyberspace, which suggests sustained membership [33]. In other words, many factors, such as time decaying, direction of reference, awareness of trackback, and so on, should be considered when measuring the relationship between two blogs. To save our approach from being overwhelmed by the details of blogs, we simply measure the similarity between two blogs  $i$  and  $j$  by

$$w_{ij} = e^{-1/\beta l_{ij}}, \quad (31)$$

where any interaction between  $i$  and  $j$  serves as an endorsement to the similarity,  $l_{ij}$  measures the number of interactions or links, and  $\beta$  controls the marginal effect of the endorsement when more interactions occur. When further interactions occur, the similarity increases more slowly and finally approaches 1. We aim to discover the blog communities and their evolution through the incremental spectral clustering.

### 7.1. Data description

The blog data were collected by the NEC laboratories American, using a blog crawler. Starting from some seed blogs, the crawler continuously crawls their RSS feeds and then the corresponding entries [33]. The blog entries are analyzed, and the hyperlinks embedded in the content are extracted. Some "hot" blogs become new seed blogs when they meet some criteria.

The data were crawled from July 10th 2005 to April 30th 2006, for 42 consecutive weeks. We use the subset written in English that consists of 406 blogs and totally 75,614 links. The self-reference links are removed since they do not contribute to the interactions in the communities. And 14,510 links remain effective, averagely 30 effective links for each blog. Fig. 6 shows the number of remaining links created in each week.

### 7.2. Comparison with the baseline

We use the standard normalized cut [2] as a baseline that is implemented using the Lanczos method [8]. We start from the 23th week and the similarity matrix  $W$ , degree matrix  $D$ , and Laplacian matrix  $L$  are built on the links of the previous 22 weeks (10,604 links in total). Then Algorithm 2 is applied to the data set, and the clusters are continuously updated as more links are added from Week 23 to 42 (3906 links in total). Each time the eigenvalues and eigenvectors are refined for one or two iterations. The cluster number is

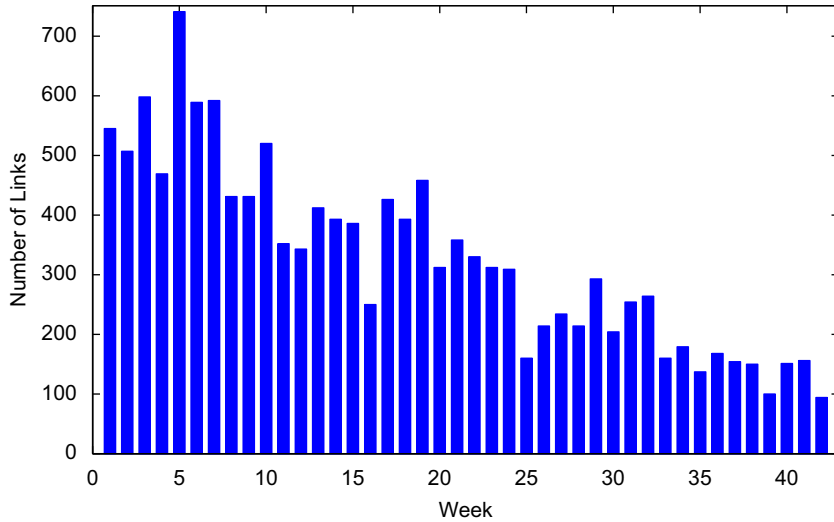


Fig. 6. The number of effective links in each week in NEC Blog Data.

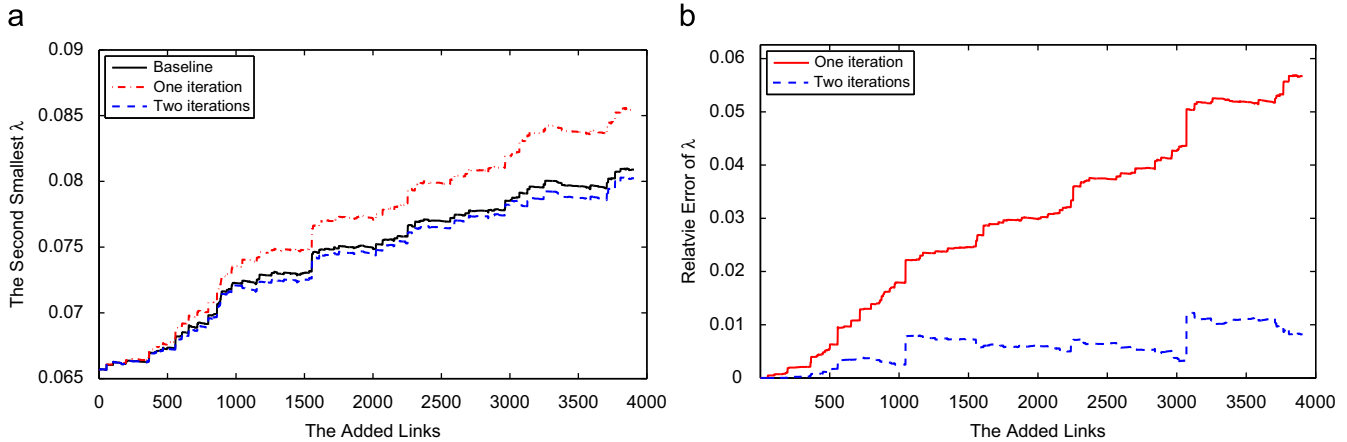


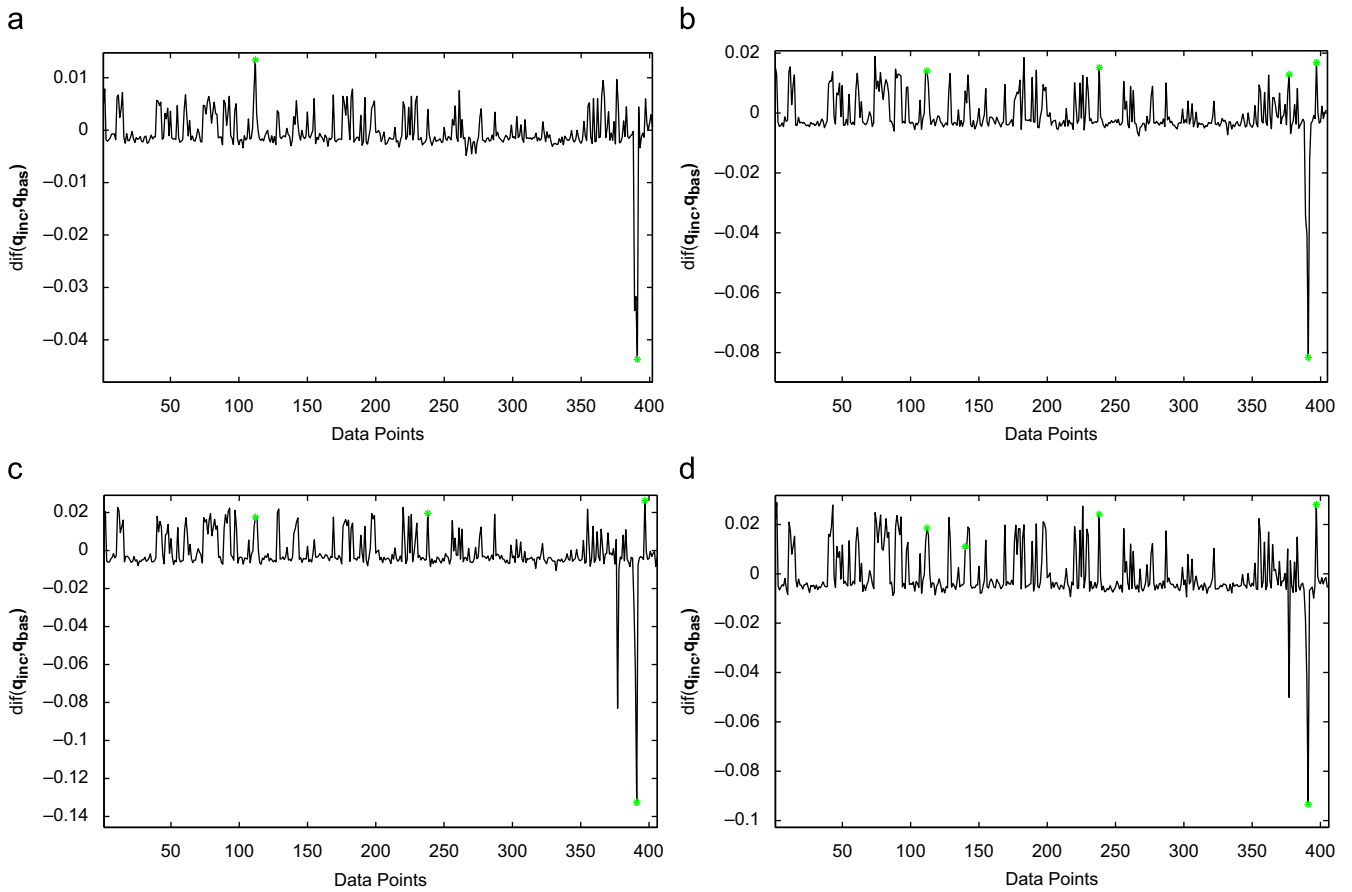
Fig. 7. (a) The changing of the second smallest eigenvalue by the standard normalized cut, our incremental approach after one iteration, and after two iterations. (b) The relative error of the eigenvalue  $\lambda$  by the incremental approach with one or two iterations.

chosen as 3 after close inspection of the initial eigenvalues. To make comparison, the baseline—standard normalized cut—recomputes the generalized eigenvalue system at each time instance when a link is added ( $L$  and  $D$  are updated by Eqs. (8) and (9)).

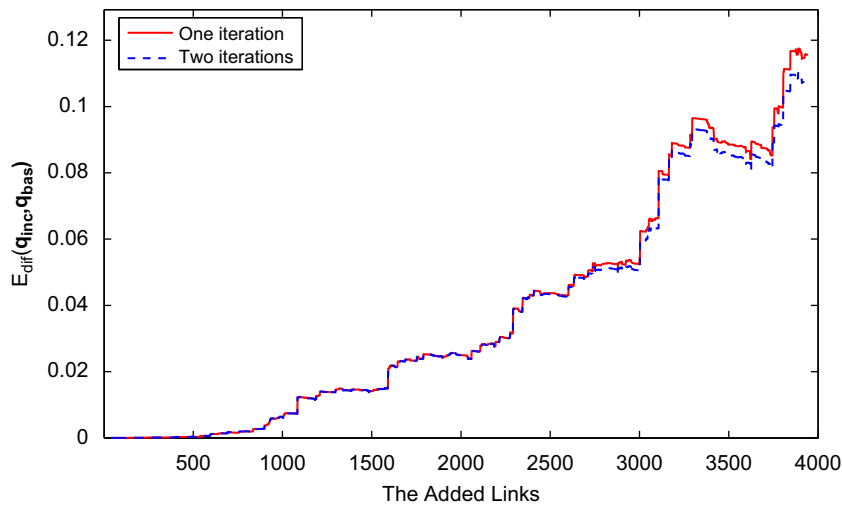
Firstly the approximation error of the eigenvalues is examined. Fig. 7(a) shows the changing of the second smallest eigenvalue generated by the baseline, our incremental approach after one iteration, and after two iterations. Basically the eigenvalue is increasing when more links are added. This is consistent with the property of algebraic connectivity, i.e., the more links are added, the more connected the graph is. The local decrements on the curve are due to that the generalized eigenvalue system, instead of  $L\mathbf{q} = \lambda\mathbf{q}$ , is used in this case. In the view of normalized cut, adding some links may contribute to better graph partitioning and hence decrease the second smallest eigenvalue (also see Eq. (16)). The first iteration of our incremental approach often overestimates the eigenvalue and the second underestimates but with much smaller error. This can be explained by Eqs. (15) and (25). In Eq. (25)  $b$  and  $d$  are very small compared with  $a$  and  $c$ , and  $c$  is always nonnegative when adding links. Therefore ignoring  $c$  in the first iteration results in overestimation. Fig. 7(b) shows the relative approximation error of the second

smallest eigenvalue by our incremental approach. The eigenvalue computed by the baseline serves as “true measurement”, and the relative error is defined by  $|\lambda_i - \lambda_b|/|\lambda_b|$  where  $\lambda_i$  and  $\lambda_b$  are eigenvalues by incremental approach and baseline, respectively. The relative error is 3.0% and 0.59% on average for one and two iterations, respectively. The relative error accumulates quickly and reaches maximum 5.70% for one iteration refinement. It accumulates very slowly if it iterates one more time at the expense of double time cost.

We use Eqs. (A.2) and (A.3) in Appendix A to compute the error  $E_{dif}(\mathbf{q}_{inc}, \mathbf{q}_{bas})$  and difference  $dif(\mathbf{q}_{inc}, \mathbf{q}_{bas})$  between the two sets of eigenvectors generated by the incremental approach ( $\mathbf{q}_{inc}$ ) and by the baseline ( $\mathbf{q}_{bas}$ ), respectively, after each link is added. Fig. 8 shows the differences of the eigenvectors with the second smallest eigenvalue (the smallest is 0), right after Week 27, 32, 37, and 42. The points marked green are leaves which cover almost all of the salient spikes. Fig. 9 shows the error  $E_{dif}$  of the corresponding eigenvectors. The error is accumulating to about 0.11 as more links are added, and averagely equal to about 0.036. Considering that the eigenvectors are normalized to one before the errors are computed, these errors are actually very small. Note that, at the beginning, the two-iteration refinement achieves a performance very similar to that by one



**Fig. 8.** The difference  $\text{dif}(\mathbf{q}_{inc}, \mathbf{q}_{bas})$  of the eigenvectors with the second smallest eigenvalue after (a) 27, (b) 32, (c) 37, (d) 42 weeks, computed after one iteration. Green marks indicate leaf nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**Fig. 9.** The error  $E_{\text{dif}}(\mathbf{q}_{inc}, \mathbf{q}_{bas})$  of the eigenvectors associated with the second smallest eigenvalue.

iteration, while it has significant improvement after more links are added, i.e., when the accumulating error grows larger. This suggests a trade-off between accuracy and computational cost: the eigenvector is updated for one iteration at the beginning and is refined for two iterations after more links are added. But in any case, the

eigenvalue can be refined for two iterations since it requires only  $O(1)$  computational cost.

The above comparison only reveals that our incremental solution to the generalized eigenvalue system closely approximates the solution by the baseline. However, it is important to compare the

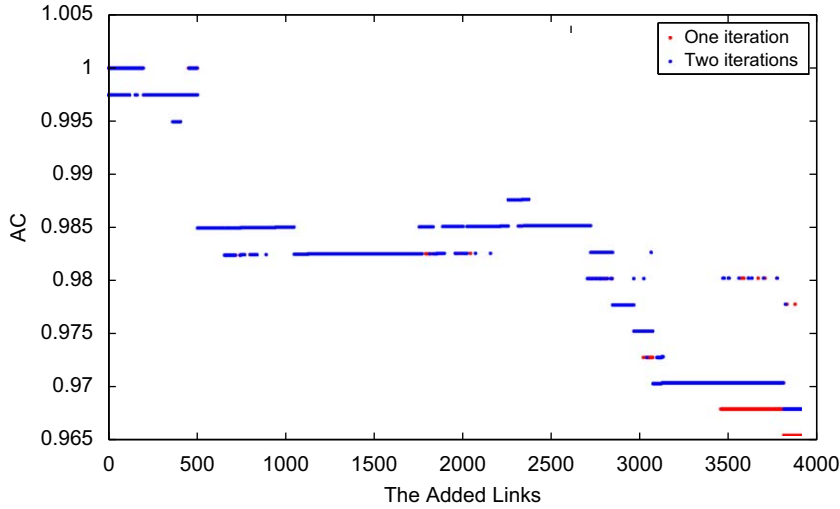


Fig. 10. The accuracy corresponding to each added link.

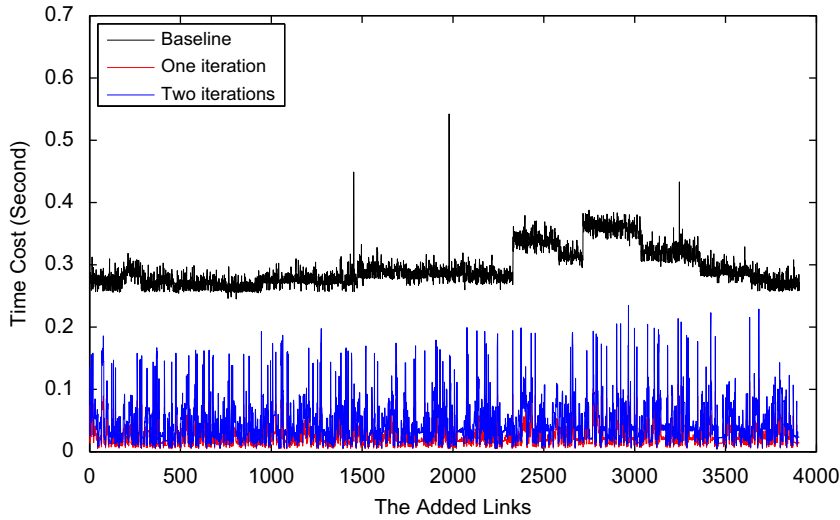


Fig. 11. Time cost of adding each link for both the incremental approach and the baseline.

clustering results. Cluster labels are obtained by discretizing the eigenvectors as [2] did. Given a data point (blog)  $i$ , let  $\hat{l}_i$  and  $\hat{l}_i$  be the labels generated by the baseline and the incremental approach, respectively. The following metric [34] is defined to measure the accuracy,

$$AC = \max_{map} \frac{\sum_i^n \delta(l_i, map(\hat{l}_i))}{n},$$

where  $n$  denotes the total number of data points,  $\delta(x, y)$  is the delta function that gives one when  $x = y$  otherwise zero, and  $map(\hat{l}_i)$  is a mapping function of labels. The optimized mapping function can be found by the Kuhn–Munkres algorithm [35] in polynomial time. Fig. 10 shows the accuracy corresponding to each added link. It is about 98.22% on average. The accuracy drops slowly as more links are added and reaches the minimum about 96.55%, i.e., about 8 blogs are clustered differently from the baseline. Compared with one iteration, two-iteration refinement does not make big difference if the first portion of the links are averagely considered. But it has significant improvement over one iteration when more links are added. From

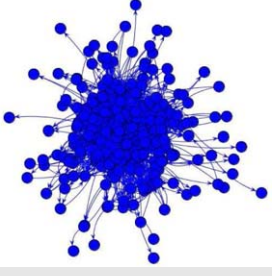
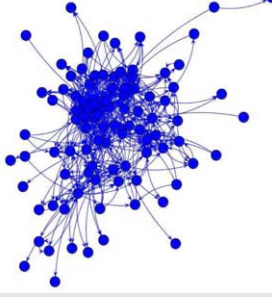
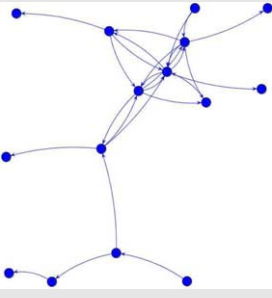
the 3400-th link to the last, the clustering error by two-iteration refinement is about relatively 7% smaller than that by one iteration.

Besides the accuracy, the computational cost is also compared. Both the incremental approach and the baseline are implemented in MATLAB. The time cost of adding each link is recorded for both systems, which is plotted in Fig. 11. It shows that the computational cost for the baseline is much higher than that of the incremental approach. Averagely it is 0.2942 s for the baseline, 0.0243 for one-iteration refinement, and 0.0463 for two iterations. It is expected that the difference is bigger for larger data sets, because the computational cost for the incremental approach is  $O(n)$  while it is  $O(n^{3/2})$  for the baseline.

### 7.3. Blog communities

The incremental spectral clustering is applied to the NEC web-blog data set after Week 22, with the output of the baseline as initialization. The method discovers three main communities. The communities are basically stable from Week 23 to Week 42,

**Table 1**  
Discovered communities.

Subgraph	Keywords
	fleshbot, betanews, sfist, torrone, lifehacker, threaded, middot, vonage, cingular, adwords, username, engadget, sprint, psp, bluetooth, nokia, phillip, powerbook, macromedia, verizon, usb, adsense, lcd, widget, tivo, tuaw, sms, voip, cellphones, businessweek, myspace, samsung, aol, feedburner, plasma, wifi, wireless, logged, hd, ndash, skype, xbox, apis, api, ipod, shuffle, nano, yahoo, os, gps, newsgator, cellphone, mozilla, sf, mobile, flash, dept, inch, blackberry, apps, mac, icio, gadget, linking, keyboard, ads, phones, beta, subscribers, interface, bookmarks, apple, motorola, startup, ebay, itunes, opml, advertisers, google, enhanced, newest, ajax, porn, firefox, messenger, messaging, offerings, desktop, hp, portable, acquisition, publishers, password, networking, xp, seth, robot, silicon, gadgets, broadband
	shanghaiist, ebfed, atta, uzbekistan, uranium, irs, niger, shanghai, liberals, islam, saudi, iranian, loan, sheehan, tax, gop, beijing, opposition, fitzgerald, elections, reform, cia, arab, valerie, plame, muslims, arabia, danger, muslim, islamic, hong, novak, democrats, pakistan, iran, partisan, palestinian, liberal, immigration, conservatives, abortion, democrat, corruption, cindy, saddam, democratic, rove, msm, indictment, wilson, libby, constitutional, terror, ambassador, democracy, syria, withdrawal, regime, republicans, presidential, congressional, propaganda, corps, sunni, hussein, voters, israeli, deputy, kong, taiwan, russia, hearings, iraq, terrorism, miers, republican, election, constitution, afghanistan, pentagon, clinton, scandal, taxes, iraqis, troops, liberty, senate, israel, fema, conservative, minister, fbi, civil, parliament, nuclear, foreign, kerry, nomination, administration, china
	libraries, library, gaming, copyright, stephen, academic, wiki, spyware, lawsuit, circuit, chicago, learning, brands, games, teach, skills, cultural, classes, complaint, teaching, staff, dance, presentation, vendors, brand, anybody, commons, amp, conferences, marketers, culture, colleagues, corporation, print, survey, students, eric, distance, game, consumers, desk, contract, access, computing, collection, books, titles, flickr, knowledge, conference, searching, student, authors, tech, county, trends, permission, registration, activities, java, learned, fair, buttons, letters, cases, deeply, amendment, engines, keyword, drm, privacy, copies, collaboration, practices, speakers, school, celebrity, taught, resources, practical, audience, seth, marketing, context, training, motion, xml, websites, boss, courts, define, studies, job, communities, database, fiction, community, association, chat, players

i.e., both the membership and topic are roughly sustained during this period. However, some individual blogs may jump among the communities as the data are evolving. The incremental algorithm can capture the evolution.

First we use the content of the blog entries to validate the discovered communities. Since the algorithm simply depends on the structural (hyperlink) information, the content validation is meaningful. We extract the keywords with top relative frequency from each community to validate whether they form a consistent cluster [33]. Here the relative frequency  $f_r(w)$  of a keyword  $w$  is the ratio of its frequency  $f'(w)$  in the content of its community to its frequency  $f(w)$  in the entire data set, i.e.,  $f_r(w) = f'(w)/f(w)$ . Unlike the absolute frequency  $f'(w)$  in a community, relative frequency can reduce the influence of the internal language frequency of a keyword. The validation is done every three weeks and the top keywords basically remain stable. Table 1 shows the subgraph of each community and its corresponding keywords at Week 28. The topics can be clearly identified from the keywords. The three communities focus on technology&business, politics, and culture, respectively. The technology&business community is mainly discussing high technologies and their commercials that are often talked at the same time on the Internet, and it is hard to separate them. More interestingly, we find that the third community is mainly discussing a small topic of culture—library, after carefully checking the content of its blogs.

**Table 2**  
List of multi-topic blogs.

No.	URL	Topic
1	<a href="http://rconversation.blogs.com/rconversation/">http://rconversation.blogs.com/rconversation/</a>	1, 2
2	<a href="http://blog.ericgoldman.org/">http://blog.ericgoldman.org/</a>	1, 3
3	<a href="http://www.josalmon.co.uk">http://www.josalmon.co.uk</a>	1, 2
4	<a href="http://www.davidmattison.ca/wordpress">http://www.davidmattison.ca/wordpress</a>	1, 3
5	<a href="http://www.cultureby.com/trilogy/">http://www.cultureby.com/trilogy/</a>	1, 3
6	<a href="http://www.joegratz.net">http://www.joegratz.net</a>	1, 3

Topic 1: tech&business; Topic 2: politics; Topic 3: culture.

Although the topics of the three communities are roughly stable, some individual blogs may bounce among them. We select out such blogs and carefully read their contents. We found that they usually cover more than one topics or change their topics at some time. So we call them “multi-topic blogs”. The URLs and the discovered topics of some multi-topic blogs are listed in Table 2. Among them, the first blog in Table 2 was created by Rebecca MacKinnon, a research Fellow at the Law School’s Berkman Center for Internet and Society, who focuses on three main subjects: the future of media in the Internet age, freedom of speech online, and the Internet in China. These subjects can be labelled as high technology and politics. The 4-th blog in Table 2, created by David Mattison, focuses



on digital libraries, digital collections, and digital preservations. So it is assigned to the culture community (library subtopic) or the high technology community. When the multi-topic blogs refer to or are referred by more blogs in one community during a specific period, they are prone to be clustered into that community. The incremental approach can basically capture this evolution. The baseline may also obtain it by recomputing the eigenvalue system for each updating, but our approach is much more efficient. Note that the identification of multi-topic blogs is a byproduct of our approach.

## 8. Conclusions

This paper presented an incremental approach for spectral clustering to handle dynamic data. Two kinds of dynamics, insertion/deletion of data points and similarity change of existing data points, are incorporated in the same framework by representing them with *incidence vector/matrix*. The incremental algorithm, initialized by a standard spectral clustering, continuously and efficiently updates the eigenvalue system and generates instant cluster labels, as the data set is evolving. The algorithm is applied to the web-blogs data. Compared with recomputation by standard spectral clustering, it achieves similar accuracy but with much lower computational cost. Close inspection into the blog content shows that the incremental approach can discover not only the stable blog communities but also the evolution of the individual multi-topic blogs. Our algorithm can also solve some other related problems involving dynamic graphs. This demonstrates the wide applicability of our algorithm.

## Appendix A. Difference of two eigenvectors

An eigenvector is subject to a scalar, i.e., if  $\mathbf{q}$  is an eigenvector, then  $c\mathbf{q}$  is also an eigenvector for any constant  $c \neq 0$ . Therefore,  $\mathbf{q}_1 - \mathbf{q}_2$  cannot be a proper difference. We define an *error* that is invariant to scale and measures the difference of the two eigenvectors,

$$E_{dif}(\mathbf{q}_1, \mathbf{q}_2) = \min_{\gamma} \left\| \gamma \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|} - \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|} \right\|^2, \quad (\text{A.1})$$

and define the difference as

$$dif(\mathbf{q}_1, \mathbf{q}_2) = \gamma \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|} - \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|}.$$

Differentiate the right side of Eq. (A.1) and set the differentiation to 0, and then  $\gamma$  can be solved:

$$\gamma = \frac{\mathbf{q}_1^T \mathbf{q}_2}{\|\mathbf{q}_1\| \|\mathbf{q}_2\|}.$$

Then, the difference and the *error* can be rewritten as

$$dif(\mathbf{q}_1, \mathbf{q}_2) = \frac{(\mathbf{q}_1^T \mathbf{q}_2) \mathbf{q}_1}{\|\mathbf{q}_1\|^2 \|\mathbf{q}_2\|} - \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|}, \quad (\text{A.2})$$

$$E_{dif}(\mathbf{q}_1, \mathbf{q}_2) = 1 - \frac{(\mathbf{q}_1^T \mathbf{q}_2)^2}{\|\mathbf{q}_1\|^2 \|\mathbf{q}_2\|^2}. \quad (\text{A.3})$$

The error is symmetric, i.e.,  $E_{dif}(\mathbf{q}_1, \mathbf{q}_2) = E_{dif}(\mathbf{q}_2, \mathbf{q}_1)$ , but the difference is not.

## Appendix B. Influence of the leaves

Suppose that node  $i$  is a leaf that is connected to node  $j$  and  $j$  is only connected to node  $k$  and  $i$ , i.e.,  $i-j-k-\dots$ . The  $i$ - and  $j$ -th rows of Eq. (4) give the relation of  $q_i$ ,  $q_j$ , and  $q_k$ ,

$$A \begin{bmatrix} q_i \\ q_j \end{bmatrix} = q_k b,$$

**Table 3**  
Influence of  $\lambda$  on  $\eta$ .

$\lambda$	0	0.020	0.040	0.060	0.080	0.100
$\eta_i$	1.000	1.309	1.845	2.995	7.1483	-24.972
$\eta_j$	1.000	1.211	1.571	2.327	5.022	-15.686

where

$$A = \begin{bmatrix} w_{ij} - \lambda(1 + w_{ij}) & -w_{ij} \\ -w_{ij} & (w_{ij} + w_{jk})(1 - \lambda) - \lambda \end{bmatrix}$$

and

$$b = \begin{bmatrix} 0 \\ w_{jk} \end{bmatrix}.$$

If  $A$  is nonsingular, the above equation is solved,

$$\begin{cases} q_i = \eta_i q_k, \\ q_j = \eta_j q_k. \end{cases}$$

For some combinations of  $w_{ij}$ ,  $w_{jk}$ , and  $\lambda$ ,  $\eta_i$  and  $\eta_j$  may be very large. Table 3 is an example where we assume  $w_{ij} = w_{jk} = e^{-1}$  and  $\lambda$  is varying from 0 to 0.1. The magnitudes of  $\eta_i$  and  $\eta_j$  grow to more than 15. Consequently, a small change of  $\Delta q_k$  may correspond to a large  $\Delta q_i$  and  $\Delta q_j$ .

## References

- [1] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Proceedings of the Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2002, pp. 849–856.
- [2] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.
- [3] X. Zhu, Semi-supervised learning literature survey (1530). URL ([http://www.cs.wisc.edu/~jerryzhu/pub/ssl\\_survey.pdf](http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf)).
- [4] C. Gupta, R. Grossman, Genic: a single pass generalized incremental algorithm for clustering, in: Proceedings of the SIAM International Conference on Data Mining, 2004, pp. 137–153.
- [5] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams, in: Proceedings of the IEEE Annual Symposium on Foundations of Computer Science, 2000, p. 359.
- [6] M. Charikar, C. Chekuri, T. Feder, R. Motwani, Incremental clustering and dynamic information retrieval, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 626–635.
- [7] B. Bollobas, Modern Graph Theory, Springer, New York, 1998.
- [8] G.H. Golub, C.F.V. Loan, Matrix Computations, John Hopkins Press, Baltimore, MD, 1989.
- [9] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak Mathematical Journal 23 (98) (1973) 298–305.
- [10] H. Ning, W. Xu, Y. Chi, Y. Gong, T. Huang, Incremental spectral clustering with application to monitoring of evolving blog communities, in: Proceedings of the SIAM International Conference on Data Mining, 2007, pp. 261–272.
- [11] C. Ding, A tutorial on spectral clustering, in: Proceedings of the International Conference on Machine Learning, 2004.
- [12] Y.C. Wei, C.K. Cheng, Towards efficient hierarchical designs by ratio cut partitioning, in: Proceedings of the International Conference on Computer Aided Design, 1989, pp. 298–301.
- [13] L. Hagen, A.B. Kahng, New spectral methods for ratio cut partitioning and clustering, IEEE Transactions on Computer-Aided Design 11 (9) (1992) 1074–1085.
- [14] C. Ding, X. He, H. Zha, M. Gu, H. Simon, A min-max cut algorithm for graph partitioning and data clustering, in: Proceedings of the IEEE International Conference on Data Mining, 2001, pp. 107–114.
- [15] D. Verma, M. Meila, A comparison of spectral clustering algorithms, Technical Report UW-CSE-03-05-01, University of Washington, Seattle, WA, 2003.
- [16] C. Valgren, T. Duckett, A. Lilienthal, Incremental spectral clustering and its application to topological mapping, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2007, pp. 4283–4288.
- [17] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th International Conference on Very Large Data Bases, VLDB Endowment, 2003, pp. 81–92.
- [18] G.S. Manku, S. Rajagopalan, B.G. Lindsay, Approximate medians and other quantiles in one pass and with limited memory, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 426–435.

- [19] G.S. Manku, S. Rajagopalan, B.G. Lindsay, Random sampling techniques for space efficient online computation of order statistics of large datasets, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1999, pp. 251–262.
- [20] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, Streaming-data algorithms for high-quality clustering, in: Proceedings of the International Conference on Data Engineering, 2002, pp. 685–694.
- [21] P. Desikan, N. Pathak, J. Srivastava, V. Kumar, Incremental page rank computation on evolving graphs, in: Proceedings of the International World Wide Web Conference, 2005, pp. 1094–1095.
- [22] A.N. Langville, C.D. Meyer, Updating pagerank with iterative aggregation, in: Proceedings of the International World Wide Web Conference, 2004, pp. 392–393.
- [23] D. Chakrabarti, R. Kumar, A. Tomkins, Evolutionary clustering, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 554–560.
- [24] Y. Chi, X. Song, D. Zhou, K. Hino, B. Tseng, Evolutionary spectral clustering by incorporating temporal smoothness, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 153–162.
- [25] S.X. Yu, J. Shi, Segmentation given partial grouping constraints, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 173–183.
- [26] F.R.K. Chung, Spectral Graph Theory, in: CBMS Regional Conference Series in Mathematics, vol. 92, American Mathematical Society, Providence, RI, 1997.
- [27] A. Zhou, F. Cao, W. Qian, C. Jin, Tracking clusters in evolving data streams over sliding windows, Knowledge and Information Systems 15 (2) (2008) 181–214.
- [28] H. Ning, M. Liu, H. Tang, T. Huang, A spectral clustering approach to speaker diarization, in: Proceedings of the International Conference on Spoken Language Processing, 2006.
- [29] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Proceedings of the Advances in Neural Information Processing Systems, 2005, pp. 1601–1608.
- [30] A. Ghosh, S. Boyd, Growing well-connected graphs, in: Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 6605–6611.
- [31] Y. Kim, M. Mesbahi, On maximizing the second smallest eigenvalue of a state dependent graph Laplacian, IEEE Transactions on Automatic Control 51 (1) (2006) 116–120.
- [32] H. Rheingold, The Virtual Community: Homesteading on the Electronic Frontier, The MIT Press, Cambridge, MA, 2000.
- [33] Y.-R. Lin, H. Sundaram, Y. Chi, J. Tatemura, B. Tseng, Discovery of blog communities based on mutual awareness, in: Proceedings of the 3rd Annual Workshop on the Weblogging Ecosystem, 2006.
- [34] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of the ACM SIGIR Special Interest Group on Information Retrieval, 2003, pp. 267–273.
- [35] L. Lovasz, M. Plummer, Matching Theory, Akademiai Kiado, North-Holland, Budapest, 1986.

**About the Author**—HUAZHONG NING received the B.Sc. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2000, and received the M.Sc. degree in pattern recognition and intelligence systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2003, and is currently working toward the Ph.D. degree in electrical engineering at the University of Illinois at Urbana-Champaign. He has worked as a 3G Software Engineer in Alcatel Shanghai Bell, China, and as summer interns in NEC American Labs, USA. His current research interests include video/image processing, machine learning, clustering, audio analysis, data mining, etc.

**About the Author**—WEI XU received his B.S. degree from Tsinghua University, Beijing, China, in 1998, and M.S. degree from Carnegie Mellon University (CMU), Pittsburgh, USA, in 2000. From 1998 to 2001, he was a research assistant at the Language Technology Institute at CMU. In 2001, he joined NEC Laboratories America working on intelligent video analysis. His research interests include computer vision, image and video understanding, machine learning and data mining.

**About the Author**—YUN CHI has been a research staff member in NEC Laboratories America, Inc. (Cupertino, CA) since 2005. He received an M.S. degree in electrical engineering from the University of Notre Dame in 2000, an M.S. degree and a Ph.D. degree, both in Computer Science, from the University of California, Los Angeles in 2001 and 2005, respectively. His primary research interests include data mining, machine learning, information retrieval, and databases.

**About the Author**—YIHONG GONG received his B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo in 1987, 1989, and 1992, respectively. He then joined the Nanyang Technological University of Singapore, where he worked as an assistant professor in the School of Electrical and Electronic Engineering for four years. From 1996 to 1998, he worked for the Robotics Institute, Carnegie Mellon University as a project scientist. He was a principal investigator for both the Informedia Digital Video Library project and the Experience-On-Demand project funded in multi-million dollars by NSF, DARPA, NASA, and other government agencies. In 1999, he joined NEC Laboratories America, and has been leading the Multimedia Processing group since then. In 2006, he became the site manager to lead the Cupertino branch of the labs. His research interests include multimedia content analysis and machine learning applications. The major research achievements from his group include news video summarization, sports highlight detection, data clustering, and SmartCatch video surveillance that led to a successful spin-off.

**About the Author**—THOMAS S. HUANG (S'61–M'63–SM'76–F'79–LF-01) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., and the M.S. and D.Sc. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge. He was on the Faculty of the Department of Electrical Engineering at MIT from 1963 to 1973 and the School of Electrical Engineering and Director of its Laboratory for Information and Signal Processing at Purdue University, West Lafayette, IN, from 1973 to 1980. In 1980, he joined the University of Illinois at Urbana-Champaign, Urbana, where he is now the William L. Everitt Distinguished Professor of Electrical and Computer Engineering, a Research Professor at the Coordinated Science Laboratory, and Head of the Image Formation and Processing Group at the Beckman Institute for Advanced Science and Technology and Co-Chair of the Institute's major research theme Human Computer Intelligent Interaction. He has published 20 books, and over 500 papers in network theory, digital filtering, image processing, and computer vision. His professional interests lie in the broad area of information technology, especially the transmission and processing of multidimensional signals.