On Evolutionary Spectral Clustering

YUN CHI
NEC Laboratories America
XIAODAN SONG
Google Inc.
DENGYONG ZHOU
Microsoft Research
KOJI HINO
NEC Laboratories America
and
BELLE L. TSENG
YAHOO! Inc.

Evolutionary clustering is an emerging research area essential to important applications such as clustering dynamic Web and blog contents and clustering data streams. In evolutionary clustering, a good clustering result should fit the current data well, while simultaneously not deviate too dramatically from the recent history. To fulfill this dual purpose, a measure of *temporal smoothness* is integrated in the overall measure of clustering quality. In this article, we propose two frameworks that incorporate temporal smoothness in evolutionary spectral clustering. For both frameworks, we start with intuitions gained from the well-known k-means clustering problem, and then propose and solve corresponding cost functions for the evolutionary spectral clustering problems. Our solutions to the evolutionary spectral clustering problems provide more stable and consistent clustering results that are less sensitive to short-term noises while at the same time are adaptive to long-term cluster drifts. Furthermore, we demonstrate that our methods provide the optimal solutions to the relaxed versions of the corresponding evolutionary k-means clustering problems. Performance experiments over a number of real and synthetic data sets illustrate our evolutionary spectral clustering methods provide more robust clustering results that are not sensitive to noise and can adapt to data drifts.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

General Terms: Algorithms, Experimentation, Measurement, Theory

Authors' addresses: Y. Chi and K. Hino, NEC Laboratories America, 10080 North Wolfe Road, SW3-350, Cupertino, CA 95014; X. Song, Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043; D. Zhou, Microsoft Research, One Microsoft Way, Redmond, WA 98052; B. L. Tseng, YAHOO! Inc., 2821 Mission College Blvd, Santa Clara, CA 95054; Contact email: ychi@sc.nec-labs.com.

DOI 10.1145/1631162.1631165 http://doi.acm.org/10.1145/1631162.1631165

Additional Key Words and Phrases: Evolutionary spectral clustering, temporal smoothness, preserving cluster quality, preserving cluster membership

ACM Reference Format:

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. 2009. On evolutionary spectral clustering. ACM Trans. Knowl. Discov. Data 3, 2, Article 17 (November 2009), 30 pages.
DOI = 10.1145/1631162.1631165 http://doi.acm.org/10.1145/1631162.1631165

1. INTRODUCTION

In many clustering applications, the characteristics of the objects to be clustered change over time. Very often, such characteristic change contains both long-term trend due to concept drift and short-term variation due to noise. For example, in the blogosphere where blog sites are to be clustered (e.g., for community detection), the overall interests of a blogger and the blogger's friendship network may drift slowly over time and simultaneously, short-term variation may be triggered by external events. As another example, in an ubiquitous computing environment, moving objects equipped with GPS sensors and wireless connections are to be clustered (e.g., for traffic jam prediction or for animal migration analysis). The coordinate of a moving object may follow a certain route in the long-term but its estimated coordinate at a given time may vary due to limitations on bandwidth and sensor accuracy.

These application scenarios, where the objects to be clustered evolve with time, raise new challenges to traditional clustering algorithms. In traditional clustering algorithms, the main target is to partition objects into groups so that objects within the same group are similar and those among different groups are dissimilar. In comparison, when clustering continuously evolving objects, some new considerations are needed. On one hand, the current clusters should depend mainly on the current data features—aggregating all historic data features makes little sense in non-stationary scenarios. On the other hand, the current clusters should not deviate too dramatically from the most recent history. This is because in most dynamic applications, we do not expect data to change too quickly and as a consequence, we expect certain levels of *temporal smoothness* between clusters in successive timesteps. We illustrate this point by using the following example. Assume we want to partition 5 blogs into 2 clusters. Figure 1 shows the relationship among the 5 blogs at time *t-1* and time *t*, where each node represents a blog and the numbers on the edges represent the similarities (e.g., the number of links) between blogs. Obviously, the blogs at time *t-1* should be clustered by Cut I. The clusters at time *t* are not so clear. Both Cut II and Cut III partition the blogs equally well. However, according to the principle of temporal smoothness, Cut III should be preferred because it is more consistent with recent history (time *t-1*). Similar ideas have long been used in time series analysis [Chatfield 2003] where moving averages are often used to smooth out short-term fluctuations. Because similar short-term variances also exist in clustering applications, either due to data noises or due to nonrobust behaviors of clustering algorithms (e.g., converging to different locally

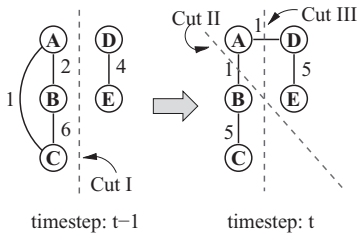

Fig. 1. An evolutionary clustering scenario.

suboptimal modes), new clustering techniques are needed to handle evolving objects and to obtain stable and consistent clustering results.

In this article, we propose two *evolutionary spectral clustering algorithms* in which the clustering cost functions contain terms that regularize temporal smoothness. Evolutionary clustering was first formulated by Chakrabarti et al. [2006] where they proposed heuristic solutions to evolutionary hierarchical clustering problems and evolutionary k-means clustering problems. In this article, we focus on evolutionary *spectral* clustering algorithms under a more rigorous framework. Spectral clustering [Bach and Jordan 2006; Chung 1997; Dhillon et al. 2004; Ding and He 2004; Shi and Malik 2000; Zha et al. 2001], which is based on top eigenvectors of matrices derived from the similarity matrix of the data points, is very appealing because the eigen-decomposition problem is a well understood topic in mathematics. In addition, spectral clustering algorithms have solid theory foundation [Chung 1997] and have shown very good performances. They have been successfully applied to many areas such as document clustering [Zha et al. 2001; Ji and Xu 2006], imagine segmentation [Shi and Malik 2000; Weiss 1999], and Web/blog clustering [Ding and He 2004; Ning et al. 2007]. Spectral clustering algorithms can be considered as solving certain graph partition problems, where different graph-based measures are to be optimized. Based on this observation, we define the cost functions in our evolutionary spectral clustering algorithms by using the graph-based measures and derive corresponding (relaxed) optimal solutions. At the same time, it has been shown that these graph partition problems have close connections to different variation of the k-means clustering problems. Through these connections, we demonstrate that our evolutionary spectral clustering algorithms provide solutions to the corresponding evolutionary k-means clustering problems as special cases.

In summary, our main contributions in this article can be summarized as the following:

(1) We propose two frameworks for evolutionary spectral clustering in which the temporal smoothness is incorporated into the overall clustering quality. To the best of our knowledge, our frameworks are the first evolutionary versions of the spectral clustering algorithms.
(2) We derive optimal solutions to the *relaxed* versions of the proposed evolutionary spectral clustering frameworks. Because the unrelaxed versions are shown to be NP-hard, our solutions provide both the practical ways of

obtaining the final clusters and the upper bounds on the performance of the algorithms.

(3) We also introduce extensions to our algorithms to handle the case where the number of clusters changes with time and the case where new data points are inserted and old ones are removed over time.

The rest of the article is organized as follows: We review related work in the rest of this Section. In Section 2, we introduce notations and necessary background. In Section 3, we present the two frameworks, PCQ and PCM, of our evolutionary spectral clustering algorithms. In Section 4, we present some extensions to our algorithms and discuss some unsolved issues. In Section 5, we provide experimental results and finally in Section 6 we give conclusions.

1.1 Related Work

As stated in Chakrabarti et al. [2006], evolutionary clustering is a fairly new topic formulated in 2006. However, it has close relationships with other research areas such as clustering data streams, incremental clustering, and constrained clustering.

In clustering data streams, large amount of data that arrive at high rate make it impractical to store all the data in memory or to scan them multiple times. Under such a new data model, many researchers have investigated issues such as how to efficiently cluster massive data set by using limited memory and by one-pass scanning of data [Guha et al. 2000], and how to cluster evolving data streams under multiple resolutions so that a user can query any historic time period with guaranteed accuracy [Aggarwal et al. 2003]. Clustering data stream is related to our work in that data in data streams evolve with time. However, instead of the scalability and one-pass-access issues, we focus on how to obtain clusters that evolve smoothly over time, an issue that has not been studied in the above works.

Incremental clustering algorithms are also related to our work. There exists a large research literature on incremental clustering algorithms, whose main task is to efficiently apply dynamic updates to the cluster centers [Gupta and Grossman 2004], medoids [Guha et al. 2000], or hierarchical trees [Charikar et al. 1997] when new data points arrive. However, in most of these studies, newly arrived data points have no direct relationship with existing data points, other than that they probably share similar statistical characteristics. In comparison, our study mainly focuses on the case when the similarity among *existing* data points varies with time, although we can also handle insertion and removal of data points over time. In Li et al. [2004], an algorithm is proposed to cluster moving objects based on a novel concept of micro-clustering. In Ningv [2007], an incremental spectral clustering algorithm is proposed to handles similarity changes among objects that evolve with time. However, the focus of both Li et al. [2004] and Ningv [2007] is to improve computation efficiency at the cost of lower cluster quality.

There is also a large body of work on constrained clustering. In these studies, either hard constraints such as *cannot links* and *must links* [Wagstaff et al. 2001] or soft constraints such as prior preferences [Ji and Xu 2006] are

incorporated in the clustering task. In comparison, in our work the constraints are not given a priori. Instead, we set our goal to optimize a cost function that incorporates temporal smoothness. As a consequence, some soft constraints are automatically implied when historic data and clusters are connected with current ones.

Another research that is closely related is Sarkar and Moore [2005], in which they proposed a dynamic method that embeds nodes into latent spaces where the locations of the nodes at consecutive timesteps are regularized so that dramatic change is unlikely. However, the work of Sarkar and Moore did not explicitly solve the clustering problem. Instead, their main focus is to preserve pairwise similarities among nodes.

One important application of clustering is to analyze communities in networked data. Recently, there exists a growing body of literature on analyzing communities and their evolutions in *dynamic networks*. Palla et al. [2007] analyzed a co-authorship network and a mobile phone network, where both networks are dynamic, by using the clique percolation method (CPM) to extract communities at each timestep and then match communities in consecutive timesteps to analyze community evolution. Toyoda and Kitsuregawa [2003] studied the evolution of Web communities from a series of Web achieves. Spiliopoulou et al. [2006] proposed a framework, MONIC, to model and monitor cluster transitions over time. Asur et al. [2007] introduced a family of events on both communities and individuals to characterize evolution of communities. Sun et al. [2007] proposed a parameter-free algorithm, GraphScope, to mine time-evolving graphs where the Minimum Description Length (MDL) principle is employed to extract communities and to detect community changes. Mei and Zhai [2005] extracted latent themes from text and used the evolution graph of themes for temporal text mining. All these studies, however, have a common weak point—community extraction and community evolution are analyzed in two separated stages. That is, when communities are extracted at a given timestep, historic community structure, which contains valuable information related to current community structure, is not taken into account. As a consequence, the evolutionary spectral clustering frameworks proposed in this article can provide a useful tool to the above studies that combines the community extraction and community evolution in a seamless way.

Our work is especially inspired by the work by Chakrabarti et al. [2006], in which they propose an evolutionary hierarchical clustering algorithm and an evolutionary k-means clustering algorithm. We mainly discuss the latter because of its connection to spectral clustering. Chakrabarti et al. [2006] proposed to measure the temporal smoothness by a distance between the clusters at time t and those at time $t-1$. Their cluster distance is defined by (1) pairing each centroid at t to its nearest peer at $t-1$ and (2) summing the distances between all pairs of centroids. We believe that such a distance has two weak points. First, the pairing procedure is based on heuristics and it could be unstable (a small perturbation on the centroids may change the pairing dramatically). Second, because it ignores the fact that the *same* data points are to be clustered in both t and $t-1$, this distance may be sensitive to the movement of data points such as shifts and rotations (e.g., consider a fleet of vehicles that move together while

the relative distances among them remain the same). A preliminary version of our work has been presented in Chi et al. [2007]. In this article, we provide more details such as some more in-depth discussions and the proofs for some of the claims.

2. NOTATIONS AND BACKGROUND

First, a word about notation. Capital letters, such as W and Z, represent matrices. Lower case letters in vector forms, such as $\vec{v}_i$ and $\vec{\mu}_l$, represent column vectors. Scripted letters, such as $\mathcal{V}$ and $\mathcal{V}_p$, represent sets. For easy presentation, for a given variable, such as W and $\vec{v}_i$, we attach a subscript t, that is, W_t and $\vec{v}_{i,t}$, to represent the value of the variable at time t. We use $Tr(W)$ to represent the *trace* of W where $Tr(W) = \sum_i W(i,i)$. In addition, for a matrix $X \in \mathcal{R}^{n\times k}$, we use $span(X)$ to represent the subspace spanned by the columns of X. For vector norms we use the Euclidian norm and for matrix norms we use the Frobenius norm, that is, $\|W\|^2 = \sum_{i,j} W(i,j)^2 = Tr(W^T W)$.

2.1 The Clustering Problem

We state the clustering problem in the following way. For a set $\mathcal{V}$ of n nodes, a clustering result is a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ of the nodes in $\mathcal{V}$ such that $\mathcal{V} = \cup_{l=1}^{k} \mathcal{V}_l$ and $\mathcal{V}_p \cap \mathcal{V}_q = \emptyset$ for $1 \le p, q \le k$, $p \ne q$. A partition (clustering result) can be equivalently represented as an n-by-k matrix $Z = [\vec{z}_1, \ldots, \vec{z}_k]$ whose elements are in $\{0, 1\}$ where $Z(i,j) = 1$ if only if node i belongs to cluster j. Obviously, $Z \cdot \vec{1}_k = \vec{1}_n$, where $\vec{1}_k$ and $\vec{1}_n$ are k-dimensional and n-dimensional vectors of all ones. In addition, we can see that the columns of Z are orthogonal. Furthermore, we normalize Z in the following way: we divide the lth column of Z by $\sqrt{|\mathcal{V}_l|}$ to get $\tilde{Z}$, where $|\mathcal{V}_l|$ is the size of $\mathcal{V}_l$. Note that the columns of $\tilde{Z}$ are orthonormal, that is, $\tilde{Z}^T \tilde{Z} = I_k$.

2.2 K-Means Clustering

The *k-means clustering problem* is one of the most widely studied clustering problems. Here we describe a very simple version of the k-means clustering problem that is based on the Euclidean distance. Assume the ith node in $\mathcal{V}$ can be represented by an m-dimensional feature vector $\vec{v}_i \in \mathcal{R}^m$, and the distance between the ith and jth nodes in $\mathcal{V}$ is $\|\vec{v}_i - \vec{v}_j\|$, the Euclidean distance. Then the k-means clustering problem is to find a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ that minimizes the following measure

$$KM = \sum_{l=1}^{k} \sum_{i\in\mathcal{V}_l} \|\vec{v}_i - \vec{\mu}_l\|^2, \tag{1}$$

where $\vec{\mu}_l$ is the centroid (mean) of the lth cluster, that is, $\vec{\mu}_l = \sum_{j\in\mathcal{V}_l} \vec{v}_j / |\mathcal{V}_l|$.

A well-known algorithm to the k-means clustering problem is the so called *k-means algorithm* in which after initially randomly picking k centroids, the following procedure is repeated until convergence: all the data points are assigned to the clusters whose centroids are nearest to them, and then the cluster centroids are updated by taking the average of the data points assigned to them.

A commonly used extension to the basic k-means measure is the *weighted* k-means measure

$$KM' = \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i - \vec{\mu}'_l\|^2, \tag{2}$$

where γ_i is the weight for the ith node and $\vec{\mu}'_l$ is the weighted centroid (mean) of the lth cluster, that is, $\vec{\mu}'_l = \sum_{j \in \mathcal{V}_l} \gamma_j \vec{v}_j / \sum_{i \in \mathcal{V}_l} \gamma_i$.

2.3 Spectral Clustering

The basic idea of spectral clustering is to cluster based on the eigenvectors of a (possibly normalized) *similarity matrix* W defined on the set of nodes in $\mathcal{V}$. Very often W is positive semi-definite. Commonly used similarities include the inner product of the feature vectors, $W(i, j) = \vec{v}_i^T \vec{v}_j$, the diagonally scaled Gaussian similarity, $W(i, j) = \exp(-(\vec{v}_i - \vec{v}_j)^T diag(\vec{\gamma})(\vec{v}_i - \vec{v}_j))$, and the affinity matrices of graphs.

Spectral clustering algorithms usually solve graph partitioning problems where different graph-based measures are to be optimized. Two popular measures are to maximize the *average association* and to minimize the *normalized cut* [Shi and Malik 2000]. For two subsets, $\mathcal{V}_p$ and $\mathcal{V}_q$, of the node set $\mathcal{V}$ (where $\mathcal{V}_p$ and $\mathcal{V}_q$ do not have to be disjoint), we first define the *association* between $\mathcal{V}_p$ and $\mathcal{V}_q$ as $assoc(\mathcal{V}_p, \mathcal{V}_q) = \sum_{i \in \mathcal{V}_p, j \in \mathcal{V}_q} W(i, j)$ Then we can write the k-way average association as

$$AA = \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V}_l)}{|\mathcal{V}_l|} \tag{3}$$

and the k-way normalized cut as

$$NC = \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V} \backslash \mathcal{V}_l)}{assoc(\mathcal{V}_l, \mathcal{V})}, \tag{4}$$

where $\mathcal{V} \backslash \mathcal{V}_l$ is the complement of $\mathcal{V}$, containing all data points that are not in $\mathcal{V}_l$. Notice that larger average association implies tighter within-cluster relations while smaller normalized cut implies both tighter within-cluster relations and looser inter-cluster relations. For consistency, we further define the *negated average association* as

$$NA = Tr(W) - AA = Tr(W) - \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V}_l)}{|\mathcal{V}_l|} \tag{5}$$

where, as will be shown later, NA is always non-negative if W is positive semi-definite. In the remaining of the article, instead of maximizing AA, we equivalently aim to minimize NA, and as a result, all the three objective functions, KM, NA and NC are to be minimized.

Finding the optimal partition Z for either the negated average association or the normalized cut is NP-hard [Shi and Malik 2000]. Therefore, in spectral clustering algorithms, usually a relaxed version of the optimization problem is

solved by (1) computing eigenvectors X (which is a relaxed version of $\tilde{Z}$ because $X \in \mathcal{R}^{n \times k}$ and $X^T X = I_k$) of some variations of the similarity matrix W, (2) projecting all data points to $span(X)$, and (3) applying the k-means algorithm to the projected data points to obtain the clustering result. While it may seem nonintuitive to apply spectral analysis and then again use the k-means algorithm, it has been shown that such procedures have many advantages such as they work well in the cases when the data points are not linearly separable [Ng et al. 2001]. The focus of our article is in step (1) in spectral clustering algorithms. That is, we define meaningful objective functions that incorporate temporal smoothness and then show how to optimize the objective functions by solving certain eigen-decomposition problems. After step (1) is done, for steps (2) and (3) we follow the standard procedures in traditional spectral clustering and thus will not give more details on them.

3. EVOLUTIONARY SPECTRAL CLUSTERING—TWO FRAMEWORKS

In this section we propose two frameworks for evolutionary spectral clustering. We first describe the basic idea.

3.1 Basic Idea

We define a general cost function to measure the quality of a clustering result on evolving data points. The function contains two costs. The first cost, *snapshot cost* (*CS*), only measures the snapshot quality of the current clustering result with respect to the *current* data features, where a higher snapshot cost means worse snapshot quality. The second cost, *temporal cost* (*CT*), measures the temporal smoothness in terms of the goodness-of-fit of the current clustering result with respect to either *historic* data features or *historic* clustering results, where a higher temporal cost means worse temporal smoothness. The overall cost function[1] is defined as a linear combination of these two costs:

$$Cost = \alpha \cdot CS + (1 - \alpha) \cdot CT \tag{6}$$

where $0 \leq \alpha \leq 1$ is a parameter assigned by the user to reflect the user's emphasis on the snapshot cost (versus that on the temporal cost).

In both frameworks that we propose, for a current partition (clustering result), the snapshot cost *CS* is measured by the clustering quality when the partition is applied to the current data. The two frameworks are different in how the temporal cost *CT* is defined. In the first framework, which we name PCQ for *preserving cluster quality*, the current partition is applied to *historic data* and the resulting cluster quality determines the temporal cost. In the second framework, which we name PCM for *preserving cluster membership*, the current partition is directly compared with the *historic partition* and the resulting difference determines the temporal cost.

In the discussion of both frameworks, we first use the k-means clustering problem, Eq. (1), as a motivational example and then formulate the

[1]Our general cost function is equivalent to the one defined in Chakrabarti et al. [2006], differing only by a constant factor and a negative sign.

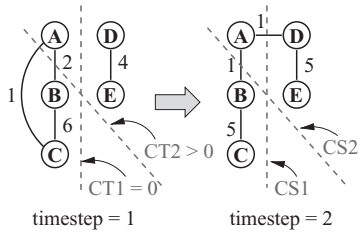

Fig. 2. Illustration of the PCQ framework: At time 2, the snapshot costs of the two cuts are the same; however, when we measure the temporal cost by applying the cuts to historic data at time 1, the temporal cost of cut 2 is greater than that of cut 1 because there are more edges crossing cut 2 at time 1.

corresponding evolutionary spectral clustering problems (both *NA* and *NC*). We also provide the optimal solutions to the relaxed versions of the evolutionary spectral clustering problems and show how they relate back to the evolutionary k-means clustering problem. In addition, in this section, we focus on a special case where the number of clusters does not change with time and neither does the number of nodes to be clustered. We will discuss the more general cases in the next section.

3.2 Preserving Cluster Quality (PCQ)

In the first framework, PCQ, the temporal cost is expressed as how well the current partition clusters historic data. We illustrate this through an example. Assume that two partitions, Z_t and Z'_t, cluster the current data at time t equally well. However, to cluster historic data at time *t-1*, the clustering performance using partition Z_t is better than using partition Z'_t. In such a case, Z_t is preferred over Z'_t because Z_t is *more consistent with historic data*. This basis idea is illustrated in Figure 2.

We formalize the PCQ framework for the k-means clustering problem using the following overall cost function

$$\begin{aligned} Cost_{KM} &= \alpha \cdot CS_{KM} + (1-\alpha) \cdot CT_{KM} \\ &= \alpha \cdot KM_t\big|_{Z_t} + (1-\alpha) \cdot KM_{t-1}\big|_{Z_t} \\ &= \alpha \cdot \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t} - \vec{\mu}_{l,t}\|^2 \\ &\quad + (1-\alpha) \cdot \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t-1} - \vec{\mu}_{l,t-1}\|^2 \end{aligned} \tag{7}$$

where $\big|_{Z_t}$ means "evaluated by the partition Z_t, where Z_t is computed at time t" and $\vec{\mu}_{l,t-1} = \sum_{j \in \mathcal{V}_{l,t}} \vec{v}_{j,t-1}/|\mathcal{V}_{l,t}|$. Note that in the formula of CT_{KM}, the inner summation is over all data points in $\mathcal{V}_{l,t}$, the clusters at time t. That is, although the feature values used in the summation are those at time *t-1* (i.e., $\vec{v}_{i,t-1}$'s), the partition used is that at time t (i.e., Z_t). As a result, this cost $CT_{KM} = KM_{t-1}\big|_{Z_t}$ penalizes those clustering results (at t) that do not fit well with recent historic data (at *t-1*) and therefore promotes temporal smoothness of clusters.

3.2.1 *Negated Average Association.* We now formulate the PCQ framework for evolutionary spectral clustering. We start with the case of negated average association. Following the idea of Eq. (7), at time t, for a given partition Z_t, a natural definition of the overall cost is

$$\begin{aligned} Cost_{NA} &= \alpha \cdot CS_{NA} + (1-\alpha) \cdot CT_{NA} \\ &= \alpha \cdot NA_t\big|_{Z_t} + (1-\alpha) \cdot NA_{t-1}\big|_{Z_t} \end{aligned} \tag{8}$$

The above cost function is almost identical to Eq. (7), except that the cluster quality is measured by the negated average association NA rather than the k-means KM.

Next, we derive a solution to minimizing $Cost_{NA}$. First, it can be easily shown that the negated average association defined in Eq. (5) can be equivalently written as

$$NA = Tr(W) - Tr(\tilde{Z}^T W \tilde{Z}) \tag{9}$$

Therefore[2] we write the overall cost (8) as

$$\begin{aligned} Cost_{NA} &= \alpha \cdot [Tr(W_t) - Tr(\tilde{Z}_t^T W_t \tilde{Z}_t)] \\ &\quad + (1-\alpha) \cdot [Tr(W_{t-1}) - Tr(\tilde{Z}_t^T W_{t-1} \tilde{Z}_t)] \\ &= Tr(\alpha W_t + (1-\alpha) W_{t-1}) - Tr\left[\tilde{Z}_t^T (\alpha W_t + (1-\alpha) W_{t-1}) \tilde{Z}_t\right] \end{aligned} \tag{10}$$

Notice that the first term $Tr(\alpha W_t + (1-\alpha) W_{t-1})$ is a constant independent of the clustering partitions and as a result, minimizing $Cost_{NA}$ is equivalent to maximizing the trace $Tr[\tilde{Z}_t^T (\alpha W_t + (1-\alpha) W_{t-1}) \tilde{Z}_t]$, subject to $\tilde{Z}_t$ being a normalized indicator matrix (cf Section 2.1). Because maximizing the average association is an NP-hard problem, finding the solution $\tilde{Z}_t$ that minimizes $Cost_{NA}$ is also NP-hard. So following most spectral clustering algorithms, we relax $\tilde{Z}_t$ to $X_t \in \mathcal{R}^{n \times k}$ with $X_t^T X_t = I_k$. It is well known [Golub and Loan 1996] that one solution to this relaxed optimization problem is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of matrix $\alpha W_t + (1-\alpha) W_{t-1}$. Therefore, after computing the solution X_t we can project the data points into $span(X_t)$ and then apply k-means to obtain a solution to the evolutionary spectral clustering problem under the measure of negated average association. In addition, the value $Tr(\alpha W_t + (1-\alpha) W_{t-1}) - Tr\left[X_t^T (\alpha W_t + (1-\alpha) W_{t-1}) X_t\right]$ provides a lower bound on the performance of the evolutionary clustering problem.

Moreover, Zha et al. [2001] have shown a close connection between the k-means clustering problem and spectral clustering algorithms—they proved (also see Xu et al. [2002]) that if we put the m-dimensional feature vectors

[2]Here we can show that NA is positive semi-definite: We have $\tilde{Z}^T \tilde{Z} = I_k$ and $Tr(W) = \sum_{i=1}^n \lambda_i$ where λ_i's are the eigenvalues of W ordered by decreasing magnitude. Therefore, by Fan's theorem [Fan 1949], which says that $\max_{X \in \mathcal{R}^{n \times k}, X^T X = I_k} Tr(X^T W X) = \sum_{j=1}^k \lambda_k$, we can derive from (9) that $NA \geq \sum_{j=k+1}^n \lambda_j \geq 0$ if W is positive semi-definite.

of the n data points in $\mathcal{V}$ into an m-by-n matrix $A = [\vec{v}_1, \ldots, \vec{v}_n]$, then

$$\begin{aligned} KM &= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \|\vec{v}_i - \vec{\mu}_l\|^2 \\ &= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \left(\|\vec{v}_i\|^2 + \|\vec{\mu}_l\|^2 - 2\vec{v}_i^T \vec{\mu}_l \right) \\ &= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \|\vec{v}_i\|^2 - \sum_{l=1}^{k} \frac{1}{|\mathcal{V}_l|} \sum_{i,j \in \mathcal{V}_l} \vec{v}_i^T \vec{v}_j \\ &= Tr(A^T A) - Tr(\tilde{Z}^T A^T A \tilde{Z}) \end{aligned} \tag{11}$$

Comparing Eq. (11) and (9), we can see that the k-means clustering problem is a special case of the negated average association spectral clustering problem, where the similarity matrix W is defined by the inner product $A^T A$. As a consequence, our solution to the *NA* evolutionary spectral clustering problem can also be applied to solve the k-means evolutionary clustering problem in the PCQ framework, i.e., under the cost function defined in Eq. (7).

3.2.2 *Normalized Cut.* For the normalized cut, we extend the idea of Eq. (7) similarly. By replacing the *KM* Eq. (7) with *NC*, we define the overall cost for evolutionary normalized cut to be

$$\begin{aligned} Cost_{NC} &= \alpha \cdot CS_{NC} + (1-\alpha) \cdot CT_{NC} \\ &= \alpha \cdot NC_t\big|_{Z_t} + (1-\alpha) \cdot NC_{t-1}\big|_{Z_t} \end{aligned} \tag{12}$$

Shi and Malik [2000] have proved that computing the optimal solution to minimize the normalized cut is NP-hard. As a result, finding an indicator matrix Z_t that minimizes $Cost_{NC}$ is also NP-hard. We now provide an optimal solution to a relaxed version of the problem. Bach and Jordan [2006] proved that (also see Xu et al. [2002]) for a given partition Z, the normalized cut can be equivalently written as

$$NC = k - Tr\left[Y^T \left(D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \right) Y \right], \tag{13}$$

where D is a diagonal matrix with $D(i,i) = \sum_{j=1}^{n} W(i,j)$ and Y is any matrix in $\mathcal{R}^{n \times k}$ that satisfies two conditions: (a) the columns of $D^{-1/2}Y$ are piecewise constant with respect to Z and (b) $Y^T Y = I_k$. We remove the constraint (a) to get a relaxed version for the optimization problem

$$\begin{aligned} Cost_{NC} \approx\ & \alpha \cdot k - \alpha \cdot Tr\left[X_t^T \left(D_t^{-\frac{1}{2}} W_t D_t^{-\frac{1}{2}} \right) X_t \right] \\ & + (1-\alpha) \cdot k - (1-\alpha) \cdot Tr\left[X_t^T \left(D_{t-1}^{-\frac{1}{2}} W_{t-1} D_{t-1}^{-\frac{1}{2}} \right) X_t \right] \\ =\ & k - Tr\left[X_t^T \left(\alpha D_t^{-\frac{1}{2}} W_t D_t^{-\frac{1}{2}} + (1-\alpha) D_{t-1}^{-\frac{1}{2}} W_{t-1} D_{t-1}^{-\frac{1}{2}} \right) X_t \right] \end{aligned} \tag{14}$$

for some $X_t \in \mathcal{R}^{n \times k}$ such that $X_t^T X_t = I_k$. Again we have a trace maximization problem and a solution is the matrix X_t whose columns are the k

eigenvectors associated with the top-k eigenvalues of matrix $\alpha D_t^{-\frac{1}{2}} W_t D_t^{-\frac{1}{2}} + (1-\alpha) D_{t-1}^{-\frac{1}{2}} W_{t-1} D_{t-1}^{-\frac{1}{2}}$. And again, after obtaining X_t, we can further project data points into $span(X_t)$ and then apply the k-means algorithm to obtain the final clusters.

Moreover, Xu et al. [2003] and Dhillon et al. [2004] have shown a close relationship between the normalized cut criterion and the weighted k-means criterion:

$$\begin{aligned}
KM' &= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i - \vec{\mu}'_l\|^2 \\
&= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} (\gamma_i \|\vec{v}_i\|^2 + \gamma_i \|\vec{\mu}'_l\|^2 - 2\gamma_i \vec{v}_i^T \vec{\mu}'_l) \\
&= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i\|^2 - \sum_{l=1}^{k} \frac{\left(\sum_{j \in \mathcal{V}_l} \gamma_j \vec{v}_j\right)^2}{\sum_{i \in \mathcal{V}_l} \gamma_i} \\
&= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i\|^2 - \sum_{l=1}^{k} \frac{\vec{z}_l^T \Gamma^{\frac{1}{2}} \left(\Gamma^{\frac{1}{2}} A^T A \Gamma^{\frac{1}{2}}\right) \Gamma^{\frac{1}{2}} \vec{z}_l}{\vec{z}_l^T \Gamma^{\frac{1}{2}} \Gamma^{\frac{1}{2}} \vec{z}_l} \\
&= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i\|^2 - \sum_{l=1}^{k} \vec{y}_l^T \left(\Gamma^{\frac{1}{2}} A^T A \Gamma^{\frac{1}{2}}\right) \vec{y}_l \\
&= \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \gamma_i \|\vec{v}_i\|^2 - Tr\left[Y^T \left(\Gamma^{\frac{1}{2}} A^T A \Gamma^{\frac{1}{2}}\right) Y\right] \qquad (15)
\end{aligned}$$

where $A = [\vec{v}_1, \ldots, \vec{v}_n]$, $Y = [\vec{y}_1, \ldots, \vec{y}_k]$, $\Gamma = diag(\gamma_1, \ldots, \gamma_n)$, and $\vec{y}_l = \Gamma^{1/2}\vec{z}_l / \|\Gamma^{1/2}\vec{z}_l\|$ (recall that $\vec{z}_l$ is the indicator vector for the lth cluster). Because the first term in Eq. (15) is a constant independent of the clusters, minimizing KM' is equivalent to maximizing $Tr[Y^T(\Gamma^{\frac{1}{2}} A^T A \Gamma^{\frac{1}{2}})Y]$. Compare Eq. (15) with Eq. (13), we can see the similarity between the weighted k-means criterion and the normalized cut criterion. In particular, if $\gamma_i = 1/d_{ii}$ where $D = diag(A^T A \cdot \vec{1}_n)$, then minimizing KM' in Eq. (15) turns out to be equivalent to minimizing NC in Eq. (13). Therefore our evolutionary spectral clustering algorithm can also be applied to solve the evolutionary version of the weighted k-means clustering problem.

3.2.3 *Discussion on the PCQ Framework.* The PCQ evolutionary clustering framework provides a data clustering technique similar to the moving average framework in time series analysis, in which the short-term fluctuation is expected to be smoothed out. The solutions to the PCQ framework turn out to be very intuitive—the historic similarity matrix is scaled and combined with current similarity matrix and the new combined similarity matrix is fed to traditional spectral clustering algorithms.

Notice that one assumption we have used in the above derivation is that the temporal cost is determined by data at time *t-1* only. However, the PCQ

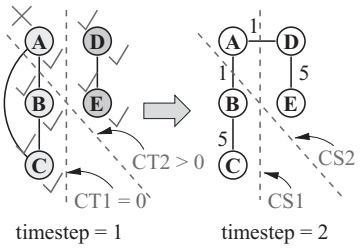

Fig. 3. Illustration of the PCM framework: At time 2, the snapshot costs of the two cuts are the same; however, when we measure the temporal cost by comparing the partition at time 2 with that at time 1, the temporal cost of cut 2 is greater because the partition resulting from cut 2 at time 2 is inconsistent the partition at time 1.

framework can be easily extended to cover longer historic data by including similarity matrices W's at older time, probably with different weights (e.g., scaled by an exponentially decaying factor to emphasize more recent history).

3.3 Preserving Cluster Membership (PCM)

The second framework of evolutionary spectral clustering, PCM, is different from the first framework, PCQ, in how the temporal cost is measured. In this second framework, the temporal cost is expressed as the difference between the current partition and the historic partition. We again illustrate this by an example. Assume that two partitions, Z_t and Z_t', cluster current data at time t equally well. However, when compared to the historic partition Z_{t-1}, Z_t is much more similar to Z_{t-1} than Z_t' is. In such a case, Z_t is preferred over Z_t' because Z_t is *more consistent with historic partition*. This basis idea is illustrated in Figure 3.

We first formalize the PCM framework for the evolutionary k-means problem. For convenience of discussion, we write the current partition as $Z_t = \{\mathcal{V}_{1,t}, \ldots, \mathcal{V}_{k,t}\}$ and the historic partition as $Z_{t-1} = \{\mathcal{V}_{1,t-1}, \ldots, \mathcal{V}_{k,t-1}\}$. Now we want to define a measure for the difference between Z_t and Z_{t-1}. Comparing two partitions has long been studied in the literatures of classification and clustering. Here we use the traditional chi-square statistics [Hubert and Arabie 1985] to represent the distance between two partitions

$$\chi^2(Z_t, Z_{t-1}) = n\left(\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|} - 1\right),$$

where $|\mathcal{V}_{ij}|$ is the number of nodes that are both in $\mathcal{V}_{i,t}$ (at time t) and in $\mathcal{V}_{j,t-1}$ (at time $t-1$). Actually, in the above definition, the number of clusters k does not have to be the same at time t and $t-1$, and we will come back to this point in the next section. By ignoring the constant shift of -1 and the constant scaling n, we define the temporal cost for the k-means clustering problem as

$$CT_{KM} = -\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|}, \tag{16}$$

where the negative sign is because we want to *minimize* CT_{KM}. The overall cost can be written as

$$\begin{aligned} Cost_{KM} &= \alpha \cdot CS_{KM} + (1-\alpha) \cdot CT_{KM} \\ &= \alpha \cdot \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t} - \vec{\mu}_{l,t}\|^2 - (1-\alpha) \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}| \cdot |\mathcal{V}_{j,t-1}|}. \end{aligned} \tag{17}$$

3.3.1 *Negated Average Association.* Recall that in the case of negated average association, we want to maximize $NA = Tr(\tilde{Z}^T W \tilde{Z})$ where $\tilde{Z}$ is further relaxed to continuous-valued X, whose columns are the k eigenvectors associated with the top-k eigenvalues of W. So in the PCM framework, we shall define a distance $dist(X_t, X_{t-1})$ between X_t, a set of eigenvectors at time t, and X_{t-1}, a set of eigenvectors at time *t-1*. However, there is a subtle point—for a solution $X \in \mathcal{R}^{n \times k}$ that maximizes $Tr(X^T W X)$, any $X' = XQ$ is also a solution, where $Q \in \mathcal{R}^{k \times k}$ is an arbitrary orthogonal matrix. This is because $Tr(X^T W X) = Tr(X^T W X Q Q^T) = Tr((XQ)^T W X Q) = Tr(X'^T W X')$. Therefore, we want a distance $dist(X_t, X_{t-1})$ that is invariant with respect to the rotation Q. One such solution, according to Golub and Loan [1996], is the norm of the difference between two projection matrices, that is,

$$dist(X_t, X_{t-1}) = \frac{1}{2} \left\| X_t X_t^T - X_{t-1} X_{t-1}^T \right\|^2, \tag{18}$$

which essentially measures the distance between $span(X_t)$ and $span(X_{t-1})$. Furthermore in Eq. (18), the number of columns in X_t does not have to be the same as that in X_{t-1} and we will discuss this in the next section.

By using this distance to quantify the temporal cost, we derive the total cost for the negated average association as

$$\begin{aligned} Cost_{NA} &= \alpha \cdot CS_{NA} + (1-\alpha) \cdot CT_{NA} \\ &= \alpha \cdot \left[Tr(W_t) - Tr\left(X_t^T W_t X_t\right) \right] + \frac{(1-\alpha)}{2} \cdot \left\| X_t X_t^T - X_{t-1} X_{t-1}^T \right\|^2 \\ &= \alpha \cdot \left[Tr(W_t) - Tr\left(X_t^T W_t X_t\right) \right] \\ &\quad + \frac{(1-\alpha)}{2} Tr\left(X_t X_t^T - X_{t-1} X_{t-1}^T \right)^T \left(X_t X_t^T - X_{t-1} X_{t-1}^T \right) \\ &= \alpha \cdot \left[Tr(W_t) - Tr\left(X_t^T W_t X_t\right) \right] \\ &\quad + \frac{(1-\alpha)}{2} Tr\left(X_t X_t^T X_t X_t^T - 2 X_t X_t^T X_{t-1} X_{t-1}^T + X_{t-1} X_{t-1}^T X_{t-1} X_{t-1}^T \right) \\ &= \alpha \cdot \left[Tr(W_t) - Tr\left(X_t^T W_t X_t\right) \right] + (1-\alpha) k - (1-\alpha) Tr\left(X_t^T X_{t-1} X_{t-1}^T X_t \right) \\ &= \alpha \cdot Tr(W_t) + (1-\alpha) \cdot k - Tr\left[X_t^T (\alpha W_t + (1-\alpha) X_{t-1} X_{t-1}^T) X_t \right] \end{aligned} \tag{19}$$

Therefore, an optimal solution that minimizes $Cost_{NA}$ is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of the matrix $\alpha W_t + (1-\alpha) X_{t-1} X_{t-1}^T$. After getting X_t, the following steps are the same as before.

Furthermore, it can be shown that the un-relaxed version of the distance defined in Eq. (18) for spectral clustering is equal to that defined in Eq. (17) for

k-means clustering by a constant shift. That is, it can be shown (cf. Bach and Jordan [2006]) that

$$\begin{aligned}\frac{1}{2}\left\|\tilde{Z}_t\tilde{Z}_t^T-\tilde{Z}_{t-1}\tilde{Z}_{t-1}^T\right\|^2 &= \frac{1}{2}\left[Tr\left(\tilde{Z}_t\tilde{Z}_t^T\tilde{Z}_t\tilde{Z}_t^T\right)+Tr\left(\tilde{Z}_{t-1}\tilde{Z}_{t-1}^T\tilde{Z}_{t-1}\tilde{Z}_{t-1}^T\right)\right.\\ &\quad \left.-2Tr\left(\tilde{Z}_t\tilde{Z}_t^T\tilde{Z}_{t-1}\tilde{Z}_{t-1}^T\right)\right]\\ &= \frac{1}{2}\left[k+k-2Tr\left(\tilde{Z}_{t-1}^T\tilde{Z}_t\tilde{Z}_t^T\tilde{Z}_{t-1}\right)\right]\\ &= k-\left\|\tilde{Z}_{t-1}^T\tilde{Z}_t\right\|^2\\ &= k-\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|}. \end{aligned} \tag{20}$$

As a result, the evolutionary spectral clustering based on negated average association in the PCM framework provides a relaxed solution to the evolutionary k-means clustering problem defined in the PCM framework, that is, Eq. (17).

3.3.2 *Normalized Cut.* It is straightforward to extend the PCM framework from the negated average association to normalized cut as

$$\begin{aligned}Cost_{NC} &= \alpha\cdot CS_{NC}+(1-\alpha)\cdot CT_{NC}\\ &= \alpha\cdot k-\alpha\cdot Tr\left[X_t^T\left(D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}}\right)X_t\right]\\ &+ \frac{(1-\alpha)}{2}\cdot\left\|X_tX_t^T-X_{t-1}X_{t-1}^T\right\|^2\\ &= k-Tr\left[X_t^T\left(\alpha D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}}+(1-\alpha)X_{t-1}X_{t-1}^T\right)X_t\right].\end{aligned} \tag{21}$$

Therefore, an optimal solution that minimizes $Cost_{NC}$ is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of the matrix $\alpha D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}}+(1-\alpha)X_{t-1}X_{t-1}^T$. After obtaining X_t, the subsequent steps are the same as before.

It is worth mentioning that in the PCM framework, $Cost_{NC}$ has an advantage over $Cost_{NA}$ in terms of the ease of selecting an appropriate α. In $Cost_{NA}$, the two terms CS_{NA} and CT_{NA} are of different scales—CS_{NA} measures a sum of variances and CT_{NA} measures some probability distribution. Consequently, this difference needs to be considered when choosing α. In contrast, for $Cost_{NC}$, because the CS_{NC} is normalized, both $D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}}$ and $X_{t-1}X_{t-1}^T$ have the same 2-norms scale, for both matrices have $\lambda_{max}=1$. Therefore, the two terms CS_{NC} and CT_{NC} are comparable and α can be selected in a straightforward way.

3.3.3 *Discussion on the PCM Framework.* In the PCM evolutionary clustering framework, all historic data are taken into consideration (with different weights)—X_t partly depends on X_{t-1}, which in turn partly depends on X_{t-2} and so on. Let us look at two extreme cases. When α approaches 1, the temporal cost will become unimportant and as a result, the clusters are computed at each time window independent of other time windows. On the other hand, when α approaches 0, the eigenvectors in all time windows are required to be

identical. Then the problem becomes a special case of the higher-order singular value decomposition problem [De Lathauwer et al. 2000], in which singular vectors are computed for the three modes (the rows of W, the columns of W, and the timeline) of a data tensor $\mathcal{W}$ where $\mathcal{W}$ is constructed by concatenating W_t's along the timeline.

In addition, if the similarity matrix W_t is positive semi-definite, then $\alpha D_t^{-\frac{1}{2}} W_t D_t^{-\frac{1}{2}} + (1-\alpha) X_{t-1} X_{t-1}^T$ is also *positive semi-definite* because both $D_t^{-\frac{1}{2}} W_t D_t^{-\frac{1}{2}}$ and $X_{t-1} X_{t-1}^T$ are positive semi-definite.

3.4 Comparing Frameworks PCQ and PCM

Now we compare the PCQ and PCM frameworks. For simplicity of discussion, we only consider time slots t and *t-1* and ignore older history.

In terms of the temporal cost, PCQ aims to maximize $Tr(X_t^T W_{t-1} X_t)$ while for PCM, $Tr(X_t^T X_{t-1} X_{t-1}^T X_t)$ is to be maximized. However, these two are closely connected. By applying the eigen-decomposition on W_{t-1}, we have

$$X_t^T W_{t-1} X_t = X_t^T \left(X_{t-1}, X_{t-1}^{\perp}\right) \Lambda_{t-1} \left(X_{t-1}, X_{t-1}^{\perp}\right)^T X_t,$$

where Λ_{t-1} is a diagonal matrix whose diagonal elements are the eigenvalues of W_{t-1} ordered by decreasing magnitude, and X_{t-1} and $X_{t-1}^{\perp}$ are the eigenvectors associated with the first k and the residual $n-k$ eigenvectors of W_{t-1}, respectively. It can be easily verified that both $Tr(X_t^T W_{t-1} X_t)$ and $Tr(X_t^T X_{t-1} X_{t-1}^T X_t)$ are maximized when $X_t = X_{t-1}$ (or more rigorously, when $span(X_t) = span(X_{t-1})$). The differences between PCQ and PCM are (a) if the eigenvectors associated with the smaller eigenvalues (other than the top k) are considered and (b) the level of penalty when X_t deviates from X_{t-1}. For PCQ, all the eigenvectors are considered and their deviations between time t and $t-1$ are penalized according to the corresponding eigenvalues. For PCM, rather than all eigenvectors, only the first k eigenvectors are considered and they are treated equally. In other words, in the PCM framework, other than the historic cluster membership, all details about historic data are ignored.

Although by keeping only historic cluster membership, PCM introduces more information loss, there may be benefits in other aspects. For example, the *CT* part in the PCM framework does not necessarily have to be temporal cost—it can represent any prior knowledge about cluster membership. For example, we can cluster blogs purely based on interlinks. However, other information such as the content of the blogs and the demographic data about the bloggers may provide valuable prior knowledge about cluster membership that can be incorporated into the clustering. The PCM framework can handle such information fusion easily.

4. EXTENSIONS

There are two assumptions in the PCQ and the PCM framework proposed in the last section. First, we assumed that the number of clusters remains the same over all time. Second, we assumed that the same set of nodes is to be clustered in all timesteps. Both assumptions are too restrictive in many applications.

In this section, we extend our frameworks to handle the issues of variation in cluster numbers and insertion/removal of nodes over time. In addition, we also discuss some other potential extensions and some unsolved issues.

4.1 Variation in Cluster Numbers

In our discussions so far, we have assumed that the number of clusters k does not change with time. However, keeping a fixed k over all time windows is a very strong restriction. Here we investigate what will happen if the cluster number k at time t is different from the cluster number k' at time $t-1$.

It turns out that both the PCQ and the PCM frameworks can handle variations in cluster number already. In the PCQ framework, the temporal cost is expressed by *historic data* themselves, not by *historic clusters* and therefore the computation at time t is independent of the cluster number k' at time $t-1$. In the PCM framework, as we have mentioned, the partition distance (Eq. 16) and the subspace distance (Eq. 18) can both be used without change when the two partitions have different numbers of clusters. As a result, both of our PCQ and PCM frameworks can handle variations in the cluster numbers.

4.2 Insertion and Removal of Nodes

Another assumption that we have been using is that the number of nodes in $\mathcal{V}$ does not change with time. However, in many applications the data points to be clustered may vary with time. In the blog example, very often there are old bloggers who stop blogging and new bloggers who just start. Here we propose some heuristic solutions to handle this issue.

4.2.1 *Node Insertion and Removal in PCQ.* For the PCQ framework, the key is $\alpha W_t + (1-\alpha)W_{t-1}$. When old nodes are removed, we can simply remove the corresponding rows and columns from W_{t-1} to get $\tilde{W}_{t-1}$ (assuming $\tilde{W}_{t-1}$ is $n_1 \times n_1$). However, when new nodes are inserted at time t, we need to add entries to $\tilde{W}_{t-1}$ and to extended it to $\hat{W}_{t-1}$, which has the same dimension as W_t (assuming W_t is $n_2 \times n_2$). Without lost of generality, we assume that the first n_1 rows and columns of W_t correspond to those nodes in $\tilde{W}_{t-1}$. We propose to achieve this by defining

$$\hat{W}_{t-1}=\begin{bmatrix} \tilde{W}_{t-1} & E_{t-1} \\ E_{t-1}^T & F_{t-1} \end{bmatrix} \text{for} \begin{cases} E_{t-1} = \frac{1}{n_1}\tilde{W}_{t-1}\vec{1}_{n_1}\vec{1}_{n_2-n_1}^T \\ F_{t-1} = \frac{1}{n_1^2}\vec{1}_{n_2-n_1}\vec{1}_{n_1}^T\tilde{W}_{t-1}\vec{1}_{n_1}\vec{1}_{n_2-n_1}^T \end{cases}$$

Such a heuristic has the following good properties.

PROPERTY 1. (1) *$\hat{W}_{t-1}$ is positive semi-definite if W_{t-1} is.* (2) *In $\hat{W}_{t-1}$, for each existing node v_{old}, each newly inserted node v_{new} looks like an average node in that the similarity between v_{new} and v_{old} is the same as the average similarity between any existing node and v_{old}.* (3) *In $\hat{W}_{t-1}$, the similarity between any pair of newly inserted nodes is the same as the average similarity among all pairs of existing nodes.*

PROOF. For claim (1), we first notice that if W_{t-1} is positive semi-definite then so is $\tilde{W}_{t-1}$. Furthermore, $\hat{W}_{t-1}$ can be written as

$$\hat{W}_{t-1} = \begin{bmatrix} I_{n_1 \times n_1} \\ B \end{bmatrix} [\tilde{W}_{t-1}] \begin{bmatrix} I_{n_1 \times n_1} \\ B \end{bmatrix}^T,$$

where $B = \frac{1}{n_1} \vec{1}_{n_2 - n_1} \vec{1}_{n_1}^T$ is an $(n_2 - n_1)$-by-n_1 matrix of all entries $\frac{1}{n_1}$. If $\tilde{W}_{t-1}$ is positive semi-definite, then we can write $\tilde{W}_{t-1} = CC^T$ (e.g., by the Cholesky decomposition). As a result, we have

$$\hat{W}_{t-1} = \left(\begin{bmatrix} I_{n_1 \times n_1} \\ B \end{bmatrix} [C] \right) \left(\begin{bmatrix} I_{n_1 \times n_1} \\ B \end{bmatrix} [C] \right)^T,$$

which implies that $\hat{W}_{t-1}$ is positive semi-definite as well.

For claim (2) and claim (3), we notice that the ijth element of E_{t-1} equals the average over the i-th row of $\tilde{W}_{t-1}$ and each element of F_{t-1} equals the average over all the elements of $\hat{W}_{t-1}$. □

We can see that these properties are appealing when no prior knowledge is given about the newly inserted nodes.

4.2.2 *Node Insertion and Removal in PCM.* For the PCM framework, when old nodes are removed, we remove the corresponding rows from X_{t-1} to get $\tilde{X}_{t-1}$ (assuming $\tilde{X}_{t-1}$ is $n_1 \times k$). When new nodes are inserted at time t, we extend $\tilde{X}_{t-1}$ to $\hat{X}_{t-1}$, which has the same dimension as X_t (assuming X_t is $n_2 \times k$) as follows

$$\hat{X}_{t-1} = \begin{bmatrix} \tilde{X}_{t-1} \\ G_{t-1} \end{bmatrix} \text{for } G_{t-1} = \frac{1}{n_1} \vec{1}_{n_2 - n_1} \vec{1}_{n_1}^T \tilde{X}_{t-1}. \tag{22}$$

That is, we insert new rows as the row average of $\tilde{X}_{t-1}$. After obtaining $\hat{X}_{t-1}$, we replace the term $(1-\alpha) X_{t-1} X_{t-1}^T$ with $(1-\alpha) \hat{X}_{t-1} (\hat{X}_{t-1}^T \hat{X}_{t-1})^{-1} \hat{X}_{t-1}^T$ in Eqs. (19) and (21).

Such a heuristic has the following good property.

PROPERTY 2. *Equation* (22) *corresponds to for each newly inserted nodes, assigning to it a prior clustering membership that is approximately proportional to the size of the clusters at time* $t - 1$.

PROOF. We start with an assumption

$$X_{t-1} Q^T \approx \tilde{Z}_{t-1}$$

where Q is a $k \times k$ orthogonal matrix. The intuition behind this assumption is that the eigenvectors is roughly inline with the normalized cluster membership indicator matrix. Weiss [1999] has shown that the above assumption is valid when the within-cluster and between-cluster similarities are approximately constant and has applied perturbation theories to extend it to general cases.

Under this assumption, we can write X_{t-1} as

$$X_{t-1} \approx \tilde{Z}_{t-1}Q = Z_{t-1}DQ$$

where D is a $k \times k$ diagonal matrix with $D(i,i) = 1/\sqrt{|\mathcal{V}_{i,t-1}|}$. Therefore, for a newly inserted nodes, for which we have no information at all, if we extend X_{t-1} by adding a row $\vec{x}^T = mean(X_{t-1})$, then equivalently on the righthand side, we need to extend Z_{t-1} by adding a row $\vec{z}^T = mean(Z_{t-1}) = (\frac{m_1}{n}, \ldots, \frac{m_k}{n})$ where $n = |\mathcal{V}_{t-1}|$ and $m_i = |\mathcal{V}_{i,t-1}|$. So instead of a hard indicator vector, $\vec{z}^T$ is a soft membership prior probability distribution where the probability of assigning the newly inserted nodes to cluster i is proportional to the size of $\mathcal{V}_{i,t-1}$, the size of cluster i at time $t-1$. The same property holds when multiple new nodes are inserted to get $\hat{X}_{t-1}$. □

The final step $\hat{X}_{t-1}(\hat{X}_{t-1}^T\hat{X}_{t-1})^{-1}\hat{X}_{t-1}^T$ is just to guarantee the result to be an orthogonal projection matrix.

4.3 Clustering Directed Graphs

Assume $\mathcal{V}$ are the nodes in a graph to be partitioned. In many applications, instead of *undirected* graphs, the graphs are *directed*. As a result, instead of a symmetric affinity matrix W, we have an asymmetric affinity matrix P. Assuming the graph represented by P represents an ergodic Markov chain, Zhou et al. [2005] extended the normalized cut criterion by defining the cut between two clustering as the probability of moving between the two clusters. Zhou et al. [2005] further proved that under the above extension, we can define the similarity matrix for the nodes in a directed graph by

$$\Theta = \frac{\Pi^{1/2}P\Pi^{-1/2} + \Pi^{-1/2}P^T\Pi^{1/2}}{2}$$

where Π is the stationary solution to the Markov chain defined by P. Then, by replacing all W's in our previous derivation by Θ (notice that Θ is symmetric), we can extend the evolutionary spectral clustering to the case of directed graphs.

4.4 Offline Algorithms

All the algorithms we have discussed so far are *online* algorithms, in which the system is causal and the historic data is used for temporal smoothness. However, there are applications in which an offline version is needed where both historic data and future data are used for temporal smoothness. For example, in image segmentation applications, there are cases in which the whole set of frames are obtained first and the segmentation is applied afterwards.

The PCQ framework can be easily extended to the offline scenario—when considering partitions at time t, we add weighted data at both time $t-1$ and time $t+1$. The PCM version is more complicated. For ease of discussion, we assume the total time windows are $1, 2, \ldots, T$ and we set $X_0 = X_1$ and $X_{T+1} = X_T$.

Then, we define the total objective function as

$$\begin{aligned} Total\_Cost &= \sum_{t=1}^{T} \left(\alpha \cdot CS_t + (1-\alpha) \cdot CT_t\right) \\ &= \sum_{t=1}^{T} \left\{ \alpha \cdot CS_t + \frac{(1-\alpha)}{4} \cdot \left\| X_{t-1} X_{t-1}^T - X_t X_t^T \right\|^2 \right. \\ &\qquad \left. + \frac{(1-\alpha)}{4} \cdot \left\| X_t X_t^T - X_{t+1} X_{t+1}^T \right\|^2 \right\}. \end{aligned} \tag{23}$$

In such as offline case, an iterative algorithm is needed to solve the globally optimal $X_1, \ldots, X_T$.

4.5 Determine k and α

In Section 4.1, we have discussed how our algorithms can handle the case where the cluster number k is varying over time. However, we did not discuss how k is determined in the first place. Determining the number of clusters is an important research problem in clustering and there are many effective methods for selecting appropriate cluster numbers. For example, one intuitive method is to look at the gap between consecutive eigenvalues and select the cluster number to be k where the gap between the kth and the $(k+1)$-th eigenvalues is large. As another example, some researchers define certain measures on clustering results (e.g., the *modularity* defined in Newman and Girvan [2004]) and then select the cluster number k that optimizes the defined measures. Our algorithms can adopt any of these methods for automatically choosing k because our algorithms only change the similarity among nodes but not the fundamental algorithms (i.e., we still use the same spectral clustering algorithms).

How to determine α is another challenging issue. Of course, when the ground truth is available, standard validation procedures can be used to select an optimal α. However, in many cases there is no ground truth and the clustering performance depends on the user's subjective preference (e.g., to what level the user cares about temporal smoothness). In this respect, through the parameter α, our algorithms provide the user a mechanism to push the clustering results toward his or her preferred outcomes. The problems of whether a "correct" α exists and how to automatically find the *best* α when there is no ground truth are beyond the scope of this paper.

5. EXPERIMENTAL STUDIES

In this section, we report experimental studies based on both synthetic data sets and a real blog data set.

5.1 Synthetic Data

We design three sets of experiments using different synthetic data. In the first two sets of experiments, we study some properties of our *NA*-based evolutionary spectral clustering algorithms on stationary and nonstationary data, respectively. In the third experiment, we study the *NC*-based evolutionary spectral

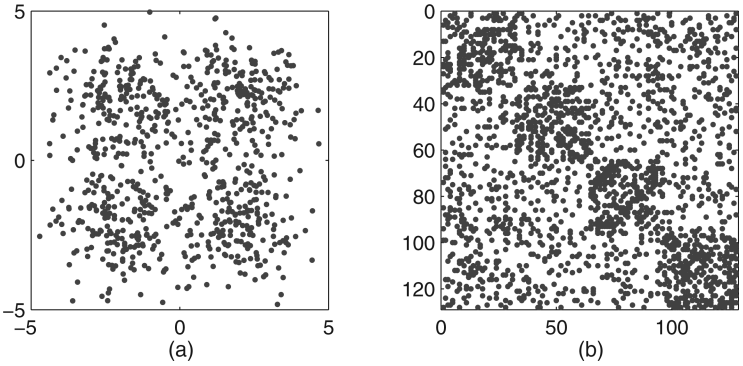

Fig. 4. (a) The initial positions for the data points in the stationary data set and (b) the initial adjacency matrix for the data points in the non-stationary data set.

clustering algorithm. In all the experiments, unless stated otherwise, we (somewhat arbitrarily) set α to be 0.9.

5.1.1 *Stationary Data.* In this section, we show several experimental studies using a stationary synthetic data set where data variation is due to a zero-mean noise. The data points to be clustered are generated in the following way. Eight hundred two-dimensional data points are initially positioned as described in Figure 4(a) at timestep 1. As can be seen, there are roughly four clusters (the data were actually generated by using four Gaussian distributions centered in the four quadrants). Then in timesteps 2 to 10, we perturb the initial positions of the data points by adding an i.i.d. Gaussian noise to the initial positions of the data points. We use this data to simulation a stationary situation where the concept is relatively stable but there exist short-term noises. We choose to use $W = A^T A$ as the similarity measure among different data points.

By using the *NA*-based spectral clustering algorithm, we design two baselines. The first baseline, which we call ACC, accumulates all historic data up to the current timestep t and applies the *NA*-based spectral clustering algorithm on the aggregated data. The second baseline, which we call IND, independently applies the *NA*-based spectral clustering algorithm on the data in only timestep t and ignores all historic data before t. For our algorithms, we use the *NA*-based PCQ and PCM algorithms. For performance comparison, we use the KM defined for the k-means clustering problem (i.e., Eq. (1)) as the measure, where a smaller KM value is better. Unless stated otherwise, all experiments are repeated 50 times with different random seeds and the average performances are reported.

In Figures 5(a) and 5(b), we report the snapshot cost CS_{KM} and the temporal cost CT_{KM} for the two baselines and for our algorithms from timesteps 1 to 10. For both costs, a lower value is better. As can be seen from the figure, the ACC baseline has low temporal smoothness but very high snapshot cost, whereas the IND baseline has the low snapshot cost but very high temporal cost. In comparison, our two algorithms show low temporal cost at the price of a little increase in snapshot cost. The overall cost $\alpha \cdot CS_{KM} + (1 - \alpha) \cdot CT_{KM}$ is

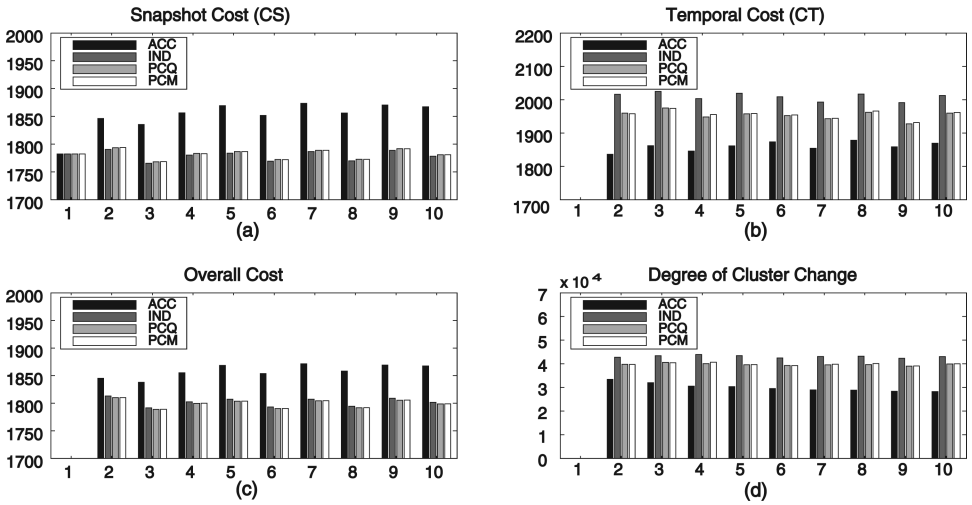

Fig. 5. The performance for the stationary synthetic data set, which shows that PCQ and PCM result in low temporal cost at a price of a small increase in snapshot cost.

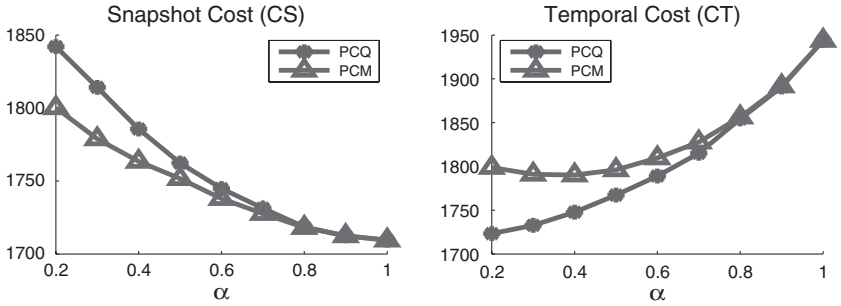

Fig. 6. The tradeoff between snapshot cost and temporal cost, which can be controlled by α.

given in Figure 5(c). As can be seen, the ACC baseline has the worst overall performance and our algorithms improve a little over the IND baseline. In addition, Figure 5(d) shows the degree of cluster change over time as defined in Eq. (20). We can see that as expected, the cluster membership change using our frameworks is less dramatic than that of the IND baseline, which takes no historic information into account.

Next, for the same data set, we let α increase from 0.2 to 1 with a step of 0.1. Figure 6 shows the average snapshot cost and the temporal cost over all 10 timesteps under different settings of α. As we expected, when α increases, to emphasize more on the snapshot cost, we get better snapshot quality at the price of worse temporal smoothness. This result demonstrates that our frameworks are able to control the tradeoff between the snapshot quality and the temporal smoothness.

In the third experiment, we show a case where the PCQ and PCM frameworks behave differently. We first generate data points using the procedure

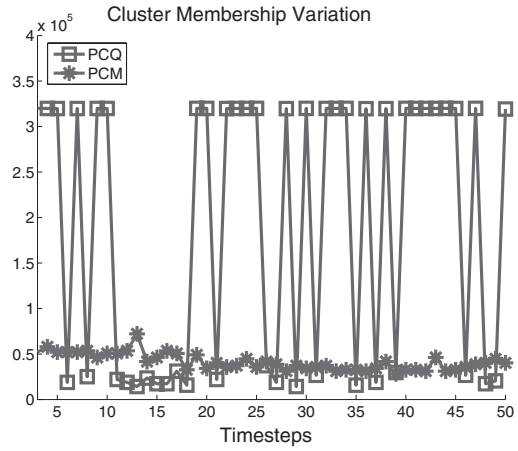

Fig. 7. A case where PCM is more robust vs PCQ.

described in the first experiment (the stationary scenario), except that this time we generate 50 timesteps for a better view. This time, instead of four clusters, we let the algorithms partition the data into two clusters. From Figure 4(a) we can see that there are obviously two possible partitions, a horizonal cut or a vertical cut at the center, that will give similar performance where the performance difference will mainly be due to short-term noises. Figure 7 shows the degree of cluster membership change over the 50 timesteps in one run (for obvious reasons, no averaging is taken in this experiment). As can be seen, the cluster membership of the PCM algorithm varies much lesser than that of the PCQ algorithm. The reason for this difference is that switching the partition from the horizontal cut to the vertical cut will introduce much higher penalty to PCM than to PCQ—PCM is directly penalized by the change of eigenvectors, which corresponds to the change of cluster membership; for PCQ, the penalty is indirectly acquired from historic data, not historic cluster membership.

5.1.2 *Nonstationary Data.* The nonstationary data set is generated in accordance with the description by Newman and Girvan [2004]. This data set contains 128 nodes, which are divided into 4 communities of 32 nodes each. Edges are added randomly with a higher probability p_{in} for within-community edges and a lower probability p_{out} for between-community edges. However, when the average degree for the nodes is fixed, a single parameter z, which represents the mean number of edges from a node to nodes in other communities, is enough to describe the data. In our experiments, we fix z to be 4. A sample adjacency matrix for the first timestep is shown in Figure 4(b). We generate data for 10 consecutive timesteps. In each timestep from 2 to 10, evolutions are introduced in the following way: from each community, we randomly select certain members to leave their original community and to join randomly the other three communities.

Because we have the ground truth for the community membership at each timestep, we directly study the accuracy of the community structure obtained

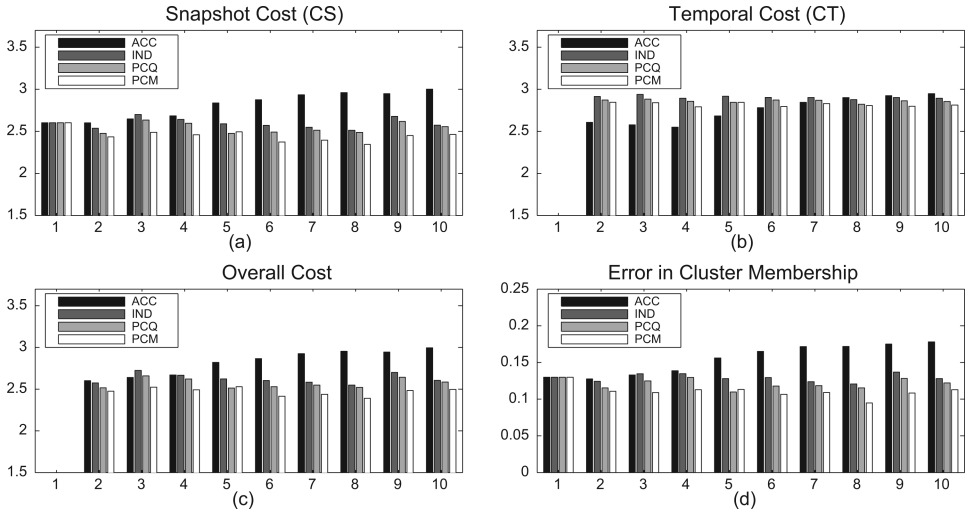

Fig. 8. The performance for the non-stationary synthetic data set, which shows that PCQ and PCM result in low overall cost and low cluster membership error.

by our algorithms. In Figure 8, we report the performance under the conditions that the average degree of each node is 16 and at each timestep, 30% nodes change their cluster membership. As can be seen from Figure 8(c), the evolutionary spectral clustering algorithms give lower overall costs at timesteps 2 to 10. In addition, Figure 8(d) shows that the evolutionary spectral clustering algorithms have lower error rates in cluster membership with respect to the ground truth.

We repeat the above experiment on a data set with longer time period (50 timesteps) and show the error rates for different algorithms in Figure 9(a). As can be seen, the IND baseline has flat error rates and the ACC baseline has increasing error rates over time. In comparison, the evolutionary spectral clustering algorithms have lower error rates at timesteps after the initial time.

However, Figure 9(b) show something unexpected—if we reduce the percentage of nodes who change their cluster membership at each timestep from 30% to 10%, the error rate of the ACC baseline actually decreases at the first few timesteps (to a level much lower than that of the evolutionary spectral clustering algorithms) before it increases. This result implies that for this special case, if instead of aggregating all history, we only aggregate data in the previous two or three timesteps, then we can get lower error rates than evolutionary spectral clustering algorithms. The reason for this unexpected result is due to the noise level in the data set: if the cluster structure cannot be clearly detected from data in one timestep, then aggregating data in several timesteps helps detect a better cluster structure; when the number of nodes who change cluster membership at each timestep is very low, then the benefit of such an aggregation outweighs the error it introduces. To verify this point, we increase the average degree from 16 to 18 while keeping z unchanged (this will result in less noisy data with clearer cluster structures). As shown in Figure 9(c), the benefit of

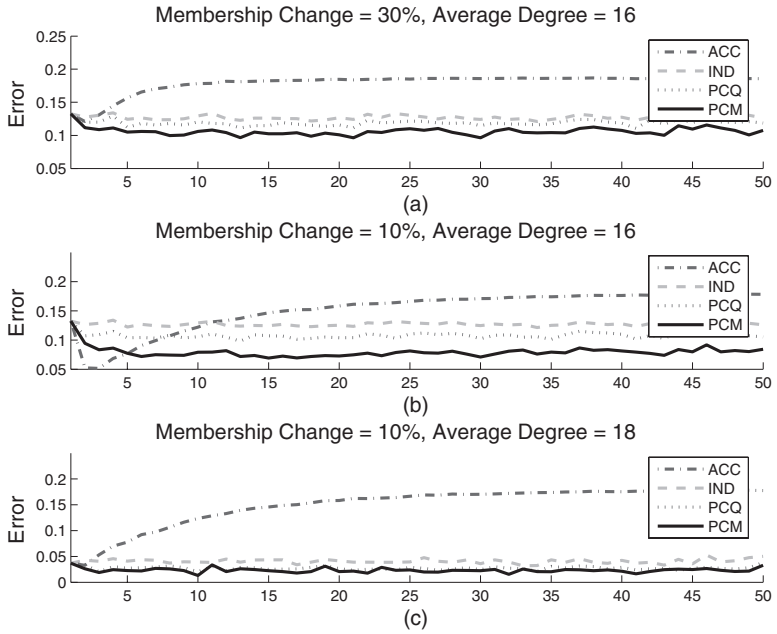

Fig. 9. The performance for the non-stationary synthetic data set, under different noise levels and evolution levels.

aggregating several timesteps disappears. This experimental study suggests that when the data is noisy and relatively stationary, the framework proposed in Section 3.2.3 that considers longer history may be more appropriate.

5.1.3 *NC-based Evolutionary Spectral Clustering.* For the case of *NC*-based evolutionary spectral clustering, we use a simple example to demonstrate that under certain circumstances, enforcing temporal smoothness has advantages. We generate data points in the 2-dimensional Euclidean space with only 4 timesteps (as shown in Figure 10) to compare the nonevolutionary version (upper panels, with $\alpha = 1$) and the evolutionary version (lower panels, with $\alpha = 0.9$) of the *NC*-based evolutionary spectral clustering algorithms. For this data set, we use the diagonally scaled Gaussian similarity $W(i, j) = \exp(-(\vec{v}_i - \vec{v}_j)^T \frac{1}{\sigma} I(\vec{v}_i - \vec{v}_j))$, where σ is set to be 20. Figure 10 gives the clustering results with the correct cluster numbers provided to the algorithm. As can be seen, for the non-evolutionary version, at timestep 2, the two letters "D"s are confused because they move too near to each other. At timestep 4, due to the change of cluster number, part of the newly introduced letter "0" is confused with the second "D". Neither happens to the evolutionary version, in which the temporal smoothness is taken into account.

As a conclusion, these experiments based on synthetic data sets demonstrate that compared to traditional clustering methods, our evolutionary spectral clustering algorithms can provide clustering results that are more stable and consistent, less sensitive to short-term noise, and adaptive to long-term trends.

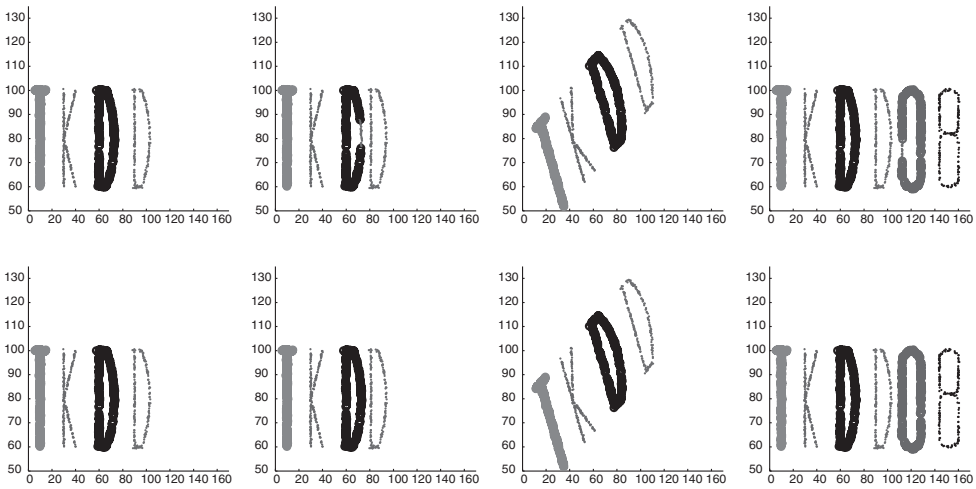

Fig. 10. A toy example demonstrates that in comparison to traditional spectral clustering (upper panels) our evolutionary spectral clustering (lower panels) is more robust and can handle changes of cluster number.

5.2 Real Blog Data

The real blog data was collected by an NEC in-house blog crawler. At the NEC Laboratories America, we have built a focused blog crawler centered on the topic of technology. Here we give a high-level description of the crawler. There are two databases used by the crawler. The first database contains a set of "seed blogs", which initially consist of some well-known blogs with technology focuses. For the seed blogs, the crawler continuously aggregates the RSS feeds and their corresponding entries. For each newly crawled entry, its content is analyzed and the hyperlinks embedded in the content are extracted. If an extracted hyperlink points to another entry and that entry belongs to a blog who is not a member of the seed blogs, then that entry and its blog are stored into the second database. The second database is checked regularly to see if any blog in the database meets the criteria to become a new seed blog (the criteria are based on the number of citations and trackbacks from current seed blogs) and if so, that blog is moved to the first database and starts to be crawled continuously.

This NEC blog data set contains 12,952 entry-to-entry links among 407 blogs crawled during 36 consecutive weeks, starting from July 2005. Figure 11 shows the blog graph for this data set, where the nodes are blogs and the edges are interlinks among blogs (obtained by aggregating all entry-to-entry links). It can be seen that the blogs roughly form 2 main clusters. It turns out that the larger cluster consists of blogs with technology focuses and the smaller cluster contains blogs with non-technical focuses (e.g., politics, international issues, digital libraries). Therefore, in the following studies, we set the number of clusters to be 2. In addition, because the edges are sparse, we take one month (actually 4 weeks) as a timestep and we aggregate all the edges in every month into an affinity matrix (i.e., the similarity matrix W) for that month (timestep). Figure 12 shows the number of edges in each month. As can be seen, the number

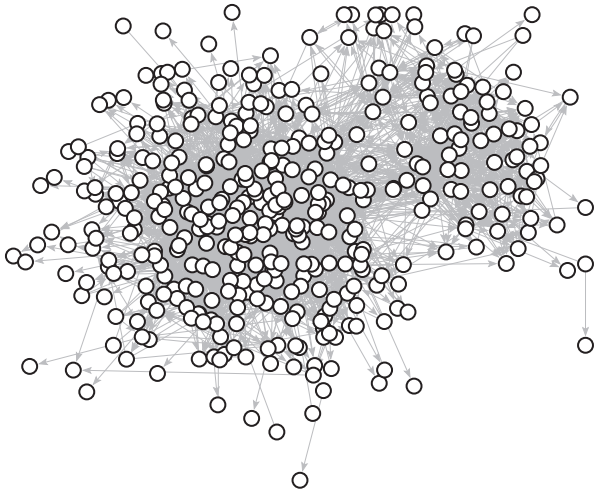

Fig. 11. The blog graph for the NEC data set.

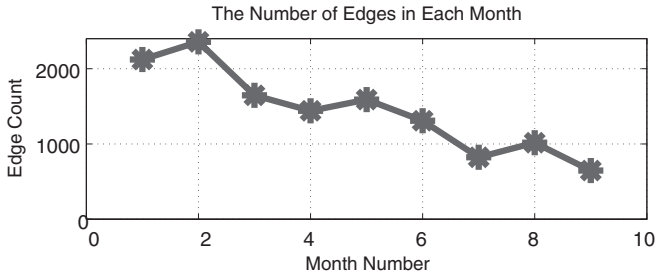

Fig. 12. The number of entry-to-entry links in each month of the NEC data set.

of edges decreases toward the end. This is because some blogs became less active over time.

In this study, for baselines we use ACC and IND with normalized cut and for our algorithms, we use the *NC*-based PCQ and PCM. First, in the first column of Table I, we report the overall costs by the four algorithms summed over months 2 to 9. As can be seen, both PCQ and PCM outperform the baselines while PCQ has the lowest overall cost.

Next, we test how well the cluster structures obtained by different algorithms can *predict* future edges. For this purpose, we count how many edges generated at time $t+1$ turn out to connect two blogs belonging to different clusters, whereas the cluster structures are computed at time t by the four algorithms. The reasoning behind this test is that a more reasonable cluster structure at time t should be the one that better explains the immediate future and therefore with fewer edges at time $t+1$ going between clusters. In the second column of Table I, we report the total number of edges at time $t+1$ that connect blogs belonging to different clusters at time t over months 2 to 9. In addition, to avoid any bias on cluster sizes (e.g., extremely unbalanced clusters), in the third column of Table I we report the total normalized cut values computed using edges at time

Table I. Performance on the Blog Data Set

	Overall Cost	Prediction Error (edge count)	Prediction Error (normalized cut)
ACC	1.2375	392	1.5018
IND	1.2612	374	1.3805
NC_PCQ	**1.0424**	363	1.3488
NC_PCM	1.1769	**359**	**1.3184**

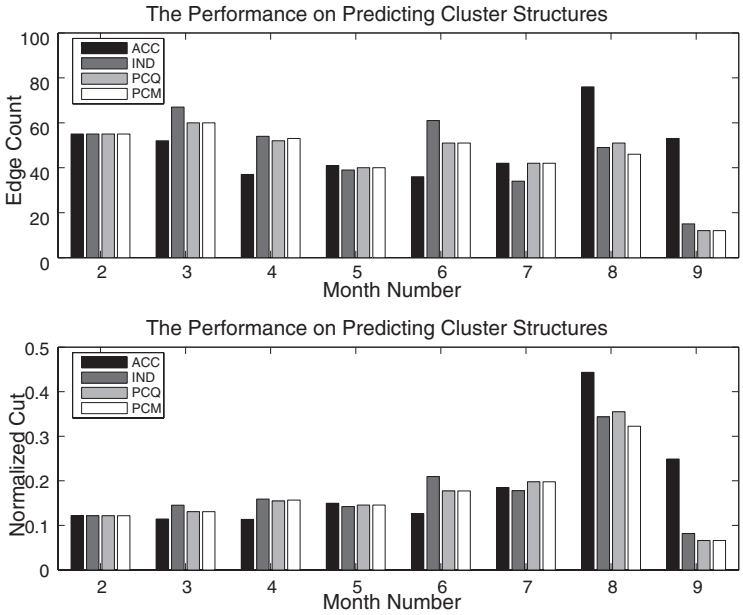

Fig. 13. The performance for the NEC data set, which shows that PCQ and PCM result in low prediction errors.

$t+1$ and cluster partitions at time t. As can be seen, in this measure both PCQ and PCM outperform the baselines with PCM having the best performance.

Finally, in Figure 13 we show the detailed predicting errors (in terms of edge count and in terms of normalized cut) over each month. From the figure, we have the following observations. First, the ACC baseline works very well at the very beginning. However, as time passes, because of the changes on the data characteristics (as reflected by the change of edge count over time shown in Figure 12), the aggregation model performs poorly toward the end. Second, the IND baseline actually performs pretty well over all the timesteps. This suggests that temporal smoothness does exist in this data set. Third, more often than not, the evolutionary clustering algorithms have better performances than the IND baseline. This good performance is due to the regularization introduced from time $t-1$ to time t, which makes the cluster structures more robust to the noise at time t. Such a regularization results in cluster structures at time t that can be better generalized to data at time $t+1$.

6. CONCLUSION

There are new challenges when traditional clustering techniques are applied to new data types, such as streaming data and Web/blog data, where the relationship among data evolves with time. On one hand, because of long-term concept drifts, a naive approach based on aggregation will not give satisfactory cluster results. On the other hand, short-term variations occur very often due to noise. Preferably the cluster results should not change dramatically over short time and should exhibit temporal smoothness. In this article, we propose two frameworks to incorporate temporal smoothness in evolutionary spectral clustering. In both frameworks, a cost function is defined where in addition to the traditional cluster quality cost, a second cost is introduced to regularize the temporal smoothness. We then derive the (relaxed) optimal solutions for solving the cost functions. The solutions turn out to have very intuitive interpretation and have forms analogous to traditional techniques used in time series analysis. Experimental studies demonstrate that these new frameworks provide cluster results that are both stable and consistent in the short-term and adaptive in the long run.

ACKNOWLEDGMENTS

We thank Shenghuo Zhu, Wei Xu, and Kai Yu for the inspiring discussions, and thank Junichi Tatemura for helping us prepare the data sets.

REFERENCES

AGGARWAL, C. C., HAN, J., WANG, J., AND YU, P. S. 2003. A framework for clustering evolving data streams. In *Proceedings of the 12th VLDB Conference*.

ASUR, S., PARTHASARATHY, S., AND UCAR, D. 2007. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *Proceedings of the 13th ACM SIGKDD Conference*. ACM, New York.

BACH, F. R. AND JORDAN, M. I. 2006. Learning spectral clustering, with application to speech separation. *J. Mach. Learn. Res. 7*.

CHAKRABARTI, D., KUMAR, R., AND TOMKINS, A. 2006. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD Conference*. ACM, New York.

CHARIKAR, M., CHEKURI, C., FEDER, T., AND MOTWANI, R. 1997. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th STOC Conference*. ACM, New York.

CHATFIELD, C. 2003. *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC.

CHI, Y., SONG, X., ZHOU, D., HINO, K., AND TSENG, B. L. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD Conference*. ACM, New York.

CHUNG, F. R. K. 1997. *Spectral Graph Theory*. American Mathematical Society.

DE LATHAUWER, L., DE MOOR, B., AND VANDEWALLE, J. 2000. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl. 21*, 4.

DHILLON, I. S., GUAN, Y., AND KULIS, B. 2004. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD Conference*.

DING, C. AND HE, X. 2004. K-means clustering via principal component analysis. In *Proceedings of the 21st ICML Conference*.

FAN, K. 1949. On a theorem of weyl concerning eigenvalues of linear transformations. In *Proceedings of the National Academy of Science*.

GOLUB, G. AND LOAN, C. V. 1996. *Matrix Computations*, third ed. Johns Hopkins University Press.

GUHA, S., MISHRA, N., MOTWANI, R., AND O'CALLAGHAN, L. 2000. Clustering data streams. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA.

GUPTA, C. AND GROSSMAN, R. 2004. Genic: A single pass generalized incremental algorithm for clustering. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, Philadelphia, PA.

HUBERT, L. J. AND ARABIE, P. 1985. Comparing partitions. *J. Classif. 2*.

JI, X. AND XU, W. 2006. Document clustering with prior knowledge. In *Proceedings of the SIGIR*. ACM, New York.

LI, Y., HAN, J., AND YANG, J. 2004. Clustering moving objects. In *Proceedings of the 10th ACM SIGKDD Conference*. ACM, New York.

MEI, Q. AND ZHAI, C. 2005. Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *Proceedings of the 11th ACM SIGKDD Conference*. ACM, New York.

NEWMAN, M. E. J. AND GIRVAN, M. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E*.

NG, A., JORDAN, M. AND WEISS, Y. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*.

NING, H., XU, W., CHI, Y., GONG, Y., AND HUANG, T. 2007. Incremental spectral clustering with application to monitoring of evolving blog communities. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, Philadelphia, PA.

PALLA, G., BARABASI, A.-L., AND VICSEK, T. 2007. Quantifying social group evolution. *Nature 446*.

SARKAR, P. AND MOORE, A. W. 2005. Dynamic social network analysis using latent space models. *SIGKDD Explor. Newsl. 7,* 2.

SHI, J. AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell. 22,* 8.

SPILIOPOULOU, M., NTOUTSI, I., THEODORIDIS, Y., AND SCHULT, R. 2006. Monic: Modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD Conference*. ACM, New York.

SUN, J., FALOUTSOS, C., PAPADIMITRIOU, S., AND YU, P. S. 2007. GraphScope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD Conference*. ACM, New York.

TOYODA, M. AND KITSUREGAWA, M. 2003. Extracting evolution of web communities from a series of web archives. In *HYPERTEXT '03: Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*. ACM, New York.

WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. 2001. Constrained K-means clustering with background knowledge. In *Proceedings of the 18th ICML Conference*.

WEISS, Y. 1999. Segmentation using eigenvectors: A unifying view. In *ICCV '99: Proceedings of the International Conference on Computer Vision Volume 2*.

XU, W., LIU, X., AND GONG, Y. 2002. Spectral text clustering. Tech. Rep. 2002-L011, NEC Laboratories America.

XU, W., LIU, X., AND GONG, Y. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th SIGIR Conference*. ACM, New York.

ZHA, H., HE, X., DING, C. H. Q., GU, M., AND SIMON, H. D. 2001. Spectral relaxation for k-means clustering. In *NIPS*.

ZHOU, D., HUANG, J., AND SCHÖLKOPF, B. 2005. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd ICML Conference*.

Received December 2007; revised September 2008; accepted November 2008