# Co-Clustering-Based Clustering and Segmentation for Pattern Discovery from Time Course Data

Hyuk Cho and Min Kyung An

***Abstract*—Time course data may inherit critical temporal ordering in contiguous (i.e., neighboring) time slot. Traditional one-way *k*-means clustering algorithms handle time points independently, ignoring the internal time locality. Although co-clustering algorithms can discover latent local patterns, the discovered patterns are not necessary to be in a continuous time order. Therefore, this paper targets to extend an existing co-clustering framework to be applicable to time course data so that time-dependent local segment patterns over specific intervals can be captured. While following the general co-clustering framework of the alternating optimization process, the proposed algorithms employ clustering on instance dimension and segmentation on time dimension. Both batch and incremental updates at boundary time points are proposed to search for a sequence of time segments. Eight time course datasets and two specific data normalization schemes are considered in the experimental study. Clustering similarity performance among *k*-means, one existing co-clustering, and the two proposed clustering segmentation algorithms is compared.**

***Index Terms*—Co-clustering, segmentation, pattern discovery, time-course data.**

## I. Introduction

The common characteristic of time-course datasets such as cell cycle data, sensor network data, and weather data is that there may exist critical continuity latent between neighboring time periods (i.e., time order or time locality). Accordingly, the main task in time course data analysis is to discover data points that express similar profiles in a certain contiguous sub-interval of the given time-course. Traditional one-way clustering algorithms are not appropriate for this purpose, because they treat the sampling at each time point as obtained under an independent experimental condition, thereby ignoring the internal sequential continuous relationship hidden between time points.

In addition, the focus of most existing co-clustering algorithms (see [1]) is to discover latent local patterns, not necessarily required to be contiguous over time. Finding latent local pattern is considered to be the main desirable characteristic of co-clustering. However, critical latent local patterns existing over continuous local time intervals may not be captured using traditional co-clustering approaches, yet. To address this, a couple of ideas were proposed to equip co-clustering algorithms with the functionality of discovery of the local patterns in gene expression datasets:

**CC-TSB (CC Time-Series Biclustering):** Based on Cheng and Church [2], Zhang *et al.* [3] proposed a deletion-based biclustering algorithm to coregulated genes showing similar expression profiles in certain sub-interval of the time course. The time locality is preserved through constraining the set of time points eligible for deletion.

**CCC-Biclustering algorithms:** Madeira *et al.* [1], [4] proposed a linear time algorithm that uses a discretized matrix and efficient string processing techniques based on suffix trees. Also, *e*-CCC-Biclustering was proposed to find CCC-Biclusters with up to a given number of errors per gene in their expression pattern [5].

These approaches are of simple generalization of the biclustering algorithm by Cheng and Church [2], and hence the applicability of their proposed approaches is limited. They share a common idea of keeping track of boundary time points so as to preserve time locality. In this paper, we employ this idea to the general co-clustering framework and develop Clustering Segmentation algorithm (Algorithm 1) that performs row (or column) clustering followed by column (or row) segmentation for a given time-course data. The proposed algorithm preserves the time locality by keeping track of the border time points in each co-cluster by efficiently updating cluster labels of the boundary time points through the local search strategy, originated from [6], [7]. Furthermore, this paper considers the effect of different data transformations of time-course datasets. Unlike the traditional co-clustering algorithms that target to cluster both dimensions, the proposed approach is to perform segmentation of time dimension, while clustering the other dimension. The proposed approach is generic; thus, it can be applicable to existing co-clustering algorithms, including all the algorithms in the Bregman Co-clustering (BCC) framework [8].

The rest of this paper is organized as follows: In section II, we define notations and the general co-clustering framework used in the paper. In Section III, we discuss detailed steps of the proposed algorithm, where two variations of the algorithm are explained. In Section III, we compare the experimental results of *k*-means, traditional co-clustering, and the proposed approach on the benchmark time course datasets. In Section V, we conclude with summary and remark.

## II. Background

### A. Notations

Upper-case boldfaced letters such as $\boldsymbol{X}$ and $\boldsymbol{A}$ denote matrices. $\boldsymbol{X}_{i\cdot}$ and $\boldsymbol{X}_{\cdot j}$ denote row $i$ and column $j$ of matrix $\boldsymbol{X}$, respectively, and $\boldsymbol{X}_{ij}$ (or $x_{ij}$) denotes the $(i,j)$-th element of matrix $\boldsymbol{X}$. Upper-case letters $I$ and $J$ (or otherwise subscripted)

Manuscript received April 15, 2014; revised June 15, 2014.

H. Cho and M. K. An are with the Department of Computer Science, Sam Houston State University, Huntsville, Texas, USA (e-mail: hyukcho@shsu.com, an@shsu.com).

denote row and column index sets of a co-cluster $\boldsymbol{A}_{IJ}$, and $|\boldsymbol{I}|$ and $|J|$ denote the cardinality of index set $I$ and index set $J$, respectively. The norm $\|\boldsymbol{X}\|$ denotes the Frobenius norm of matrix $\boldsymbol{X}$, i.e., $\|\boldsymbol{X}\|^2 = \sum_{ij} |x_{ij}^2|$. The symbols $\mathbb{R}$, $\mathbb{R}^d$, and $\mathbb{R}^{m\times n}$ denote the set of reals, the $d$-dimensional real vector space, and the $m\times n$ real matrix space, respectively.

The data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, whose $(i,j)$-th element is denoted by $a_{ij}$, is defined as follows:

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

We partition $\boldsymbol{A}$ into $k$ row clusters and $\ell$ column clusters defined by the following functions ,

$$\rho : \{1,2,\cdots,m\} \to \{1,2,\cdots,k\}, \tag{1}$$

$$\gamma : \{1,2,\cdots,n\} \to \{1,2,\cdots,\ell\}, \tag{2}$$

where $\rho(i) = r$ implies that row $i$ is in row cluster $r$ and similarly $\gamma(j) = c$ implies that column $j$ is in column cluster $c$. As defined previously, let $I$ denote the set of indices of rows in a row cluster and $J$ denote the set of indices of columns in a column cluster. The submatrix of $\boldsymbol{A}$ determined by $I$ and $J$, denoted as $\boldsymbol{A}_{IJ}$, is called a *co-cluster*.

## B. Definitions

1) *Cluster Indicator Matrix*: Assume $m_r$ rows belong to row-cluster $\gamma$ $(1 \le r \le k)$, so that $\sum_r m_r = m$. Similarly, $n_c$ rows belong to column-cluster $c$ $(1 \le c \le \ell)$, so that $\sum_c n_c = n$. Then, we define a row cluster indicator matrix, $\boldsymbol{R} \in \mathbb{R}^{m\times k}$ and a column cluster indicator matrix, $\boldsymbol{C} \in \mathbb{R}^{m\times \ell}$ as follows: column $r$ of $\boldsymbol{R}$ has $m_r$ non-zeros, each of which equals $m_r^{-1/2}$, the non-zeros of $\boldsymbol{C}$ are defined similarly. Without loss of generality, we assume that the rows that belong to a particular cluster are contiguous and so are the columns. Then the matrix $\boldsymbol{R}$ may have the form,

$$\boldsymbol{R} = \begin{pmatrix} m_1^{-1/2} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & m_k^{-1/2} \\ 0 & m_2^{-1/2} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & m_2^{-1/2} & \cdots & 0 \\ m_1^{-1/2} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & m_k^{-1/2} \end{pmatrix},$$

where the first column has $m_1$ non-zeros, the second column has $m_2$ non-zeros, and the last $k$-th column has $m_k$ non-zeros, which can be either consecutive or not. Matrix $\boldsymbol{C}$ has a similar structure. Therefore, $\left\|\boldsymbol{R}_{\cdot r}\right\|_1^2 = m_r$ and $\left\|\boldsymbol{C}_{\cdot c}\right\|_1^2 = n_c$. Note that $\boldsymbol{R}$ and $\boldsymbol{C}$ are column orthonormal matrices since the columns of $\boldsymbol{R}$ and $\boldsymbol{C}$ are clearly orthogonal and $\|\boldsymbol{R}_{\cdot r}\|_2 = 1$ and $\|\boldsymbol{C}_{\cdot c}\|_2 = 1$. Using these definitions of $\boldsymbol{R}$ and $\boldsymbol{C}$, we can write both the residues compactly in the matrix form as follows.

2) *Residue*: In order to evaluate the quality of such a co-cluster, two *measures* have been considered, each of which targets to capture either "homogeneity" or "trend" of data as discussed in [6], [9].
   - The sum of squared differences between each entry in the co-cluster and the mean of the co-cluster.
   - The sum of squared differences between each entry in the co-cluster and the corresponding row mean and the column mean. The co-cluster mean is to be added to retain symmetry.

We define the residue of an element $a_{ij}$ in the co-cluster determined by index sets $I$ and $J$ to be

$$h_{ij} = a_{ij} - a_{IJ} \qquad \text{for basis 2,} \tag{3}$$

$$h_{ij} = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \quad \text{for basis 6,} \tag{4}$$

where $a_{iJ} = \Sigma_{j\in J}\, a_{ij}/|J|$ is the mean of the entries in row $i$ whose column indices are in $J$, $a_{Ij} = \Sigma_{i\in I}\, a_{Ij} / |I|$ is the mean of the entries in column j whose row indices are in $I$, and $a_{IJ} = \Sigma_{i\in I, j\in J}\, a_{IJ} / |I||J|$ is the mean of all the entries in the co-cluster, where $|I|$ and $|J|$ denote the cardinality of $I$ and $J$. (3) was the measure used by Hartigan [10]. It is also related to the first residue in [6], [9], and basis 2 in [8]. (4) was used by Cheng and Church [2]. It is also related to the second residue in [6], [9] and basis 6 in [8].

3) *Residue Matrix*: Suppose $\boldsymbol{H} = [h_{ij}]$, where $h_{ij}$ is defined in (3) or (4), and $\boldsymbol{R}$ and $\boldsymbol{C}$ are the cluster indicator matrices as defined above. Then, we have

$$\boldsymbol{H} = \boldsymbol{A} - \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T \qquad \text{for (3),} \tag{5}$$

$$\boldsymbol{H} = \left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^T\right)\boldsymbol{A}\left(\boldsymbol{I} - \boldsymbol{C}\boldsymbol{C}^T\right) \quad \text{for (4).} \tag{6}$$

As shown in [6], [9], the rows of $\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}$ give the row cluster mean vectors. In a similar fashion we have that $(\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T)_{ij} = a_{iJ}$ and $(\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T)_{ij} = a_{IJ}$, where $i \in I$ and $j \in J$.

## C. Minimum-Sum Squared Residue Co-Clustering

The residue matrix $\boldsymbol{H}$ leads to the following objective function for minimizing squared residues: find both row and column clusters simultaneously such that $\|\boldsymbol{H}\|^2 = \Sigma_{I,J}\, \|\boldsymbol{H}_{IJ}\|^2$ is minimized. In other words, our optimization problem is to minimize the total squared residue of the objective function,

$$\|\boldsymbol{H}\|^2 = \sum_{I,J} \|\boldsymbol{H}_{IJ}\|^2 = \sum_{I,J} \sum_{i\in I, j\in J} \left|h_{ij}\right|^2,$$

where $\boldsymbol{H}_{IJ}$ is the co-cluster induced by $I$ and $J$. The following toy example provides some insight into the different residue measures defined in (3) and (4).

For each definition of $\boldsymbol{H}$, we get a corresponding residue minimization problem, called Minimum-Sum Squared Residue Co-clustering (MSSRCC) [6], [9], [11]. We refer to these minimization problems as our first and second problems, respectively. When $\boldsymbol{R}$ and $\boldsymbol{C}$ are constrained to be cluster indicator matrices as in our case, the problem of obtaining the global minimum for $\|\boldsymbol{H}\|$ is NP-hard. Therefore, we resort to iterative algorithms that monotonically decrease the objective functions and converge to a local minimum.

## III. Clustering Segmentation

Let us consider the data matrix whose rows and columns consist of objects and time-courses, respectively. Therefore, we do not claim rows in row cluster $r$ (i.e., $r$-th column in $\boldsymbol{R}$ defined in (II-B1)) to be consecutive, while requiring columns in column cluster $c$ (i.e., $c$-th column in $\boldsymbol{C}$) to be consecutive in order to ensure the time locality of time-courses. Therefore, different from $\boldsymbol{R}$, $\boldsymbol{C}$ should have the following form,

$$\boldsymbol{C} = \begin{pmatrix} n_1^{-1/2} & 0 & \cdots & 0 \\ n_1^{-1/2} & 0 & \cdots & 0 \\ n_1^{-1/2} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & n_\ell^{-1/2} \\ 0 & 0 & \cdots & n_\ell^{-1/2} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & n_2^{-1/2} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & n_2^{-1/2} & \cdots & 0 \end{pmatrix},$$

where each co-cluster consists of only one group of consecutive rows. As before, $\|\boldsymbol{R}_{\cdot r}\|_1^2 = m_r$ and $\|\boldsymbol{C}_{\cdot c}\|_1^2 = n_c$, and $\|\boldsymbol{R}_{\cdot r}\|_2 = 1$ and $\|\boldsymbol{C}_{\cdot c}\|_2 = 1$. In summary, both $\boldsymbol{R}$ and $\boldsymbol{C}$ are column orthonormal matrices, however non-zeros in $\boldsymbol{R}$ are not necessarily consecutive but those in $\boldsymbol{C}$ are required to be consecutive, which will guarantee to preserve the time locality of the considered time course (i.e., consecutive columns in the example).

As defined in (1) and (2), $\rho$ is a mapping from the $m$ rows to the $k$ row clusters and $\gamma$ is a mapping from the $n$ columns to the $\ell$ column segments. Note that there exist no restrictions on $\rho$, while $\gamma$ is constrained to ensure that the ordering of the time intervals is retained as in the column segment indicator matrix $\boldsymbol{C}$. Accordingly, $\gamma$ should be of the form, $\gamma(j) = \ell'$ that satisfies $1 \le j \le n$, $1 \le \ell' \le \ell$, and $first(\ell') \le j \le last(\ell')$, where $first(\ell')$ and $last(\ell')$ return the first column index and the last column index of column cluster $\ell'$, respectively.

**Algorithm 1:** Bregman Clustering Segmentation (BCS)

```
BCS(A, k, ℓ, ρ, γ)
Input: Data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, number of row clusters $k$,
    number of column segments $\ell$, and cluster indicator
    vectors $\rho \in \{1, \cdots, k\}$ and $\gamma \in \{1, \cdots, \ell\}^{n\times 1}$
Output: Cluster indicator vectors $\rho$ and $\gamma$
begin
    Initialize cluster assignment of $\rho$ and $\gamma$;
    $\boldsymbol{R} \leftarrow$ Update using initial $\rho$;
    $\boldsymbol{C} \leftarrow$ Update using initial $\rho$;
    $\tau \leftarrow 10^{-3}\|\boldsymbol{A}\|^2$;
    $newobj \leftarrow$ Update the target objective function;
    $oldobj \leftarrow newobj + \tau + 1$;
    while $|oldobj - newobj| > \tau$ do
        RowClusterUpdate ($\boldsymbol{A}$, $k$, $\ell$, $\rho$, $\boldsymbol{R}$, $\boldsymbol{C}$);
        $\boldsymbol{R} \leftarrow$ Update using new $\rho$;
        ColSegmentUpdate ($\boldsymbol{A}$, $k$, $\ell$, $\gamma$, $\boldsymbol{R}$, $\boldsymbol{C}$);
        $\boldsymbol{C} \leftarrow$ Update using new $\rho$;
        $oldobj \leftarrow newobj$;
        $newobj \leftarrow$ Update the target objective function;
    end
end
```

**Algorithm 2:** Column Segment Update (Batch)

```
ColSegmentUpdate ($\boldsymbol{A}$, $k$, $\ell$, $\gamma$, $\boldsymbol{R}$, $\boldsymbol{C}$)
Input: Data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, number of row clusters $k$,
    number of column segments $\ell$, column cluster
    indicator vector $\gamma \in \{1, \cdots, \ell\}^{n\times 1}$, row cluster
    indicator matrix $\boldsymbol{R} \in \mathbb{R}^{m\times k}$, and column cluster indicator
    matrix $\boldsymbol{C} \in \mathbb{R}^{n\times \ell}$
Output: Column cluster indicator vector $\gamma$
begin
    $\tau \leftarrow 10^{-5}\|\boldsymbol{A}\|^2$;                                /* Adjustable */
    /* Gain of moving first to adjacent column cluster */
    for $b \leftarrow 2$ to $\ell$ do
        $c' \leftarrow \gamma(first(b)-1)$;
        $\delta_{first(b)}(c') \leftarrow$ Compute gain;                  /* FIRST */
    end
    /* Gain of moving last to adjacent column cluster */
    for $b \leftarrow 1$ to $\ell-1$ do
        $c' \leftarrow \gamma(last(b)+1)$;
        $\delta_{last(b)}(c') \leftarrow$ Compute gain;                   /* LAST */
    end
    /* Find the best column to move */
    $(j^*, c^*) \leftarrow \arg\max_{(j,c)} \delta_j(c)$;                  /* BEST */
    if $\delta_{j^*}(c^*) > \tau$ then
        $\gamma(j^*) \leftarrow c^*$;
    end
end
```

**Algorithm 3:** Column Segment Update (Incremental)

```
ColSegmentUpdate ($\boldsymbol{A}$, $k$, $\ell$, $\rho$, $\boldsymbol{R}$, $\boldsymbol{C}$)
Input: Data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, number of row clusters $k$,
    number of column segments $\ell$, column cluster
    indicator vector $\gamma \in \{1, \cdots, \ell\}^{n\times 1}$, row cluster
    indicator matrix $\boldsymbol{R} \in \mathbb{R}^{m\times k}$, and column cluster indicator
    matrix $\boldsymbol{C} \in \mathbb{R}^{n\times \ell}$
Output: Column cluster indicator vector $\gamma$
begin
    $\tau \leftarrow 10^{-5}\|\boldsymbol{A}\|^2$;                                /* Adjustable */
    /* Gain of moving first and last to adjacent column cluster
*/
    for $b \leftarrow 1$ to $\ell-1$ do
        $c'' \leftarrow \gamma(first(b+1)-1)$;
        $\delta_{first(b+1)}(c'') \leftarrow$ Compute gain;              /* FIRST */
        $c' \leftarrow \gamma(last(b)+1)$;
        $\delta_{last(b)}(c') \leftarrow$ Compute gain;                   /* LAST */
        if $\delta_{last(b)}(c') < \delta_{first(b+1)}(c'')$ then
            $j^* \leftarrow first(b+1)$;                     /*FIRST move is
    better.*/
            $c^* \leftarrow c''$;
        end
        if $\delta_{last(b)}(c') > \delta_{first(b+1)}(c'')$ then
            $j^* \leftarrow last(b)$;                        /*LAST move is better.*/
            $c^* \leftarrow c'$;
        end
        if $\delta_{j^*}(c^*) > \tau$ then
            $\gamma(j^*) \leftarrow c^*$;
        end
    end
end
```

The overall process of the proposed Bregman Clustering Segmentation (BCS) described in Algorithm 1 resembles the general framework of both the MSSRCC [6] and the BCC [8]. The BCS algorithm begins out with some initialization of both the indicator matrices, ***R*** and ***C***, and then it iterates till the decrease in the objective function value becomes small as governed by the tolerance factor $\tau$. Each iteration of the BCS consists of the alternating optimization process between clustering of all rows of data matrix and segmentation of only columns at segment boundaries. It is worth noting that `RowClusterUpdate`() of Algorithm 1 is equivalent to the batch greedy assignment/update step in both the MSSRCC [6] and the BCC [8]. `RowClusterUpdate`() updates the cluster indicator vector ρ for all the rows so that every row is assigned to its closest row cluster. Such greedy row cluster update decreases the objective function at each iteration and improves the row clustering as proved in [6], [8].

In contrast, `ColSegmentUpdate`() cannot simply assign every column to the closest column segment, since we want to ensure that the time points (i.e., columns) are not permuted. It can start with the simple initialization that randomly selects $\ell$–1 time points in order to divide the columns into $\ell$ segments and then only the time points (i.e., columns) at the segment boundaries are investigated by employing the similar greedy local search strategy with the gain value defined in [7], [9]. Note that the gain is defined as the change of objective function value by moving one boundary point into one of the two neighboring segments. In this paper, we develop two strategies for such column segmentation update. The first strategy is the batch update described in Algorithm 2, where gains of all the boundary points are computed and the segment assignment of the best column is updated, if the best gain is above the specified threshold. The second strategy is the incremental update described in Algorithm 3, where gains of every boundary data point are computed and its segment is incrementally updated to one of its neighboring segment, if the gain of its move is above the specified threshold.

Notice that *first*(1) has no neighboring left column cluster 0 to move to and *last*($\ell$) has no right column cluster $\ell + 1$ to move to. After computing all the possible $2(\ell - 1)$ gains, Algorithm 2 updates the segment assignment only for the one move that leads to the best (i.e., maximum) gain out of $2(\ell - 1)$ gains. However, one call of Algorithm 3 can update up to $\ell$ segment assignments, since either left or right move can be incrementally selected for each column segment.

Both Algorithms 2 and 3 can be applicable to all the algorithms in the BCC framework (see [8]), if we can efficiently compute the gain of moving a data point to another column segment (i.e., `/*FIRST*/` and `/*LAST*/`) in Algorithm 2 and "Compute gain" step in Algorithm 3. Furthermore, both can adopt the chain of the incremental local search strategy in [7], [9] to escape from poor local minima.

## IV. Experimental Results

### A. Time Course Data

Time course data consists of observations on variable(s) over time (i.e., stochastic process) and has particular characteristics: (1) Temporal ordering is important and thus past can affect the future, but not vice versa; (2) Observations can rarely be assumed to be independent over time; and (3) A particular time course data is one possible outcome of the stochastic process. Therefore, keeping temporal locality has an importance issue in time course data analysis. Because of this requirement, time course data analysis is not a trivial problem. There exist two goals in time course data analysis: (1) identifying the nature of the phenomenon represented by the sequence of observations over time and (2) predicting future values of the time course variable. In this paper, we are interested in identifying the time-dependent latent local patterns from a given time course data

The eight time-course datasets used as benchmark time course datasets are summarized in Table I. The datasets were collected from the UCR Time Series Classification/Clustering page at http://www.cs.ucr.edu/~eamonn/time_series_data/. Note that each dataset was split into training and test sets because the original purpose of the datasets was to do classification. However, for our experimental study, we combine them together for the purpose of *k*-means clustering, co-clustering, and BCS.

TABLE I: Time Course Datasets Used in Our Experiments

| Dataset | Classes | Size | Length | Segments |
|---|---|---|---|---|
| Synthetic Control | 6 | 600 | 60 | 10 |
| Gun Point | 2 | 200 | 150 | 2 |
| CBF | 3 | 930 | 128 | 5 |
| Face All | 14 | 2250 | 131 | 20 |
| Trace | 4 | 200 | 275 | 3 |
| Face Four | 4 | 112 | 350 | 20 |
| Lighting2 | 2 | 121 | 637 | 2 |
| Lighting7 | 7 | 143 | 319 | 5 |

### B. Data Normalization

In our previous research [11], [12], we analyzed the effect of data transformations on the MSSRCC and BCC frameworks and also showed the empirical result that supports the analysis. Therefore, following the suggestion in [11], [12], we apply the two specific data transformations, Z-score transformation (ZT) and bispherical normalization (NBIN), to every dataset in Table I. The two data normalization are summarized below.

1) *(Column/Row) Z-score Transformation (ZT)*: Column standardization is defined as $a_{ij} = (a_{ij} - a_{\cdot j}) / \sigma_{\cdot j}$ for $i = 1, \cdots, m$ and $j = 1, \cdots, n$. Row standardization is defined similarly with $a_{i\cdot}$ and $\sigma_{i\cdot}$. It is also called "autoscaling", where the measurements are scaled so that each column/row has a zero mean and a unit variance [13]. Through ZT, the relative variation in intensity is emphasized, since ZT is a linear transformation, which

keeps the relative positions of observations and the shape of the original distribution.

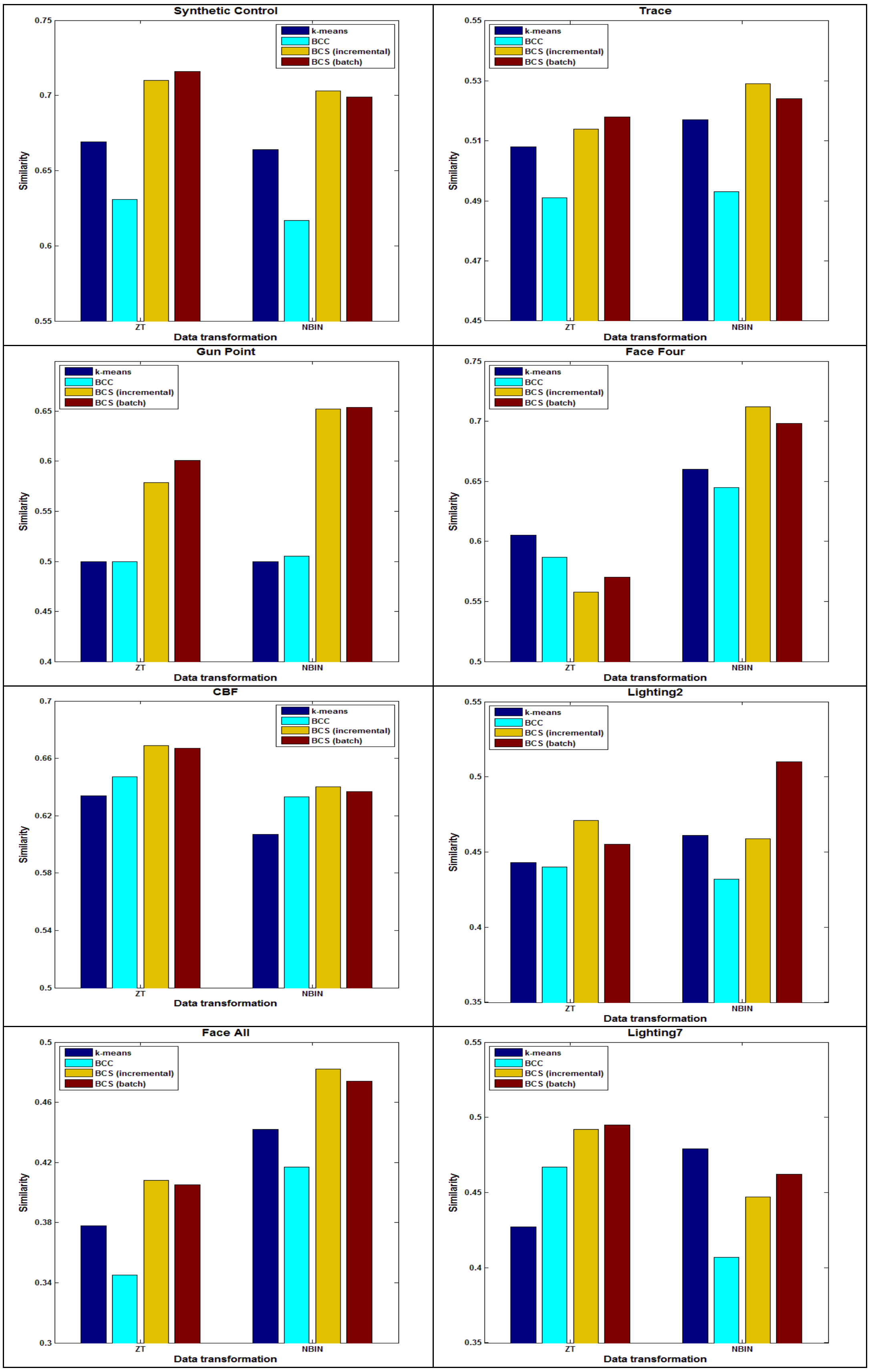

Fig. 1. Comparison of similarity performance with different data transformations and clustering algorithms. Two data transformations (ZT for (column) Z-score transformation and NBIN for bispherical normalization) and four clustering algorithms (k-means, BCC for Bregman Co-clustering, BCS (incremental) and BCS (batch) for Bregman Clustering Segmentation with incremental update and with batch update, respectively) are considered. Similarity values are averaged over 20 random runs.

2) *Binormalization (NBIN)*: Livne and Golub [14] presented iterative algorithms (called BIN for a square matrix and NBIN for a rectangle matrix) for scaling all the rows and columns of a square (*rectangle*) matrix to

have unit (same) *L2*-norm. Binormalization of a rectangular matrix through NBIN results in $\Sigma_j a_{ij}^2 = n$ for $i = 1, \cdots, m$, and $\Sigma_i a_{ij}^2 = m$ for $j = 1, \cdots, n$. The effect is a bi-sphericalization, where the rows as well as the columns are forced to lie on hyperspheres with radii $\sqrt{n}$ and $\sqrt{m}$ , respectively.

Note that we apply column ZT to make each data point have mean 0 and variance 1 over the considered time points (see [11], [12] for details of data transformations). Note basis 2 is employed for BCC and BCS algorithms, however other bases in [8] can be directly applicable. Fixing number of row clusters (i.e. *k*) to the size of true class labels for each dataset, we vary number of column segmentations (i.e., $\ell$), until it leads to comparable performance with *k*-means. In Table I, we specify the number of column clusters resulted from the above mentioned search process.

### C. Similarity Measure

We compare the performance of our BCS algorithms (i.e., Algorithms 2 and 3) with those of typical one-way *k*-means and the BCC algorithm [8]. Since all the considered time course datasets have their class labels (i.e., "ground-truth"), external evaluation measures can be applicable. In our experiments, we use similarity proposed by Gavrilov *et al.* [15] as follows: Given the given clustering $C = C_1 C_2 \cdots C_k$ (i.e., "ground-truth") and the clustering $C' = C'_1 C'_2 \cdots C'_k$ (i.e., cluster labels),

$$similarity(C, C') = \frac{1}{k} \sum_i \max_j similarity(C_i, C'_j),$$

where $similarity(C_i, C'_j) = 2|C_i \cap C'_j| / (|C_i| + |C'_j|)$. Note that $similarity(C, C')$ will return 0 if both the given clustering and the obtained one are completely different, and 1 if they are identical. Since the measure is not symmetric, we always use the "ground-truth" clustering as the first parameter.

### D. Observation and Summary

From our experiments, we observe the following characteristics of the proposed BCS algorithms:

- For some datasets, including Synthetic Control (with both ZT and NBIN), Face All (with ZT), Trace (with both ZT and NBIN), Lighting2 (with both ZT and NBIN), and Lighting7 (with NBIN), BCC leads to the worst similarity. We attribute the deteriorated performance with BCC to the locality of the considered time course datasets. Note that BCC permutes the order of both data points and features through the co-clustering process and hence it might break critical latent temporal ordering existing over continuous local time intervals of some of the datasets. Note that normal k-means is not sensitive to the order of time points in nature and the proposed BCS algorithms preserve the temporal order on purpose.
- For all the considered time course datasets, BCS with ZT and BCS with NBIN perform better. However, much difference in performance between incremental and batch versions of BCS is not observed.
- Interestingly, Face Four (with ZT) and Lighting7 (with NBIN) perform best with k-means, while Face Four (with NBIN) and Lighting7 (with ZT) perform best with the BCS algorithms.
- Except Synthetic Control, CBF, and Lighting7, BCS with NBIN performs best. It was also shown in [11], [12] that BCC worked well with NBIN. Similarly, BCS matches reasonably well with NBIN in our experiments.

These experimental results support that the proposed BCS algorithms can discover latent local patterns existing over continuous local time intervals and data transformations also affect the overall performance of the BCS algorithms, as for the BCC algorithms in our previous study (see [11], [12]).

## V. Conclusion and Remarks

We presented the clustering and segment performance of BBC and BCS with basis 2 in the MSSRCC and the BCS frameworks in Algorithm 1. In our preliminary research, we also witnessed that the proposed algorithms with other bases in [8] performed well with various parameter settings (not reported in this paper). Hence, it would be worthwhile to further investigate detailed performance with other bases.

We demonstrated the effect of the two data normalizations, ZT and NBIN on the similarity performance of the considered algorithms. Therefore, it would be desirable to include more data preprocessing techniques, e.g., [16], [17], [18], and more external evaluation measures.

Visual inspection of the segments may be interesting to pursue, as it is a simple way to verify whether the resulted segments preserve the inherent characteristics of the considered bases. For example, as shown in [6], [11], if segments contain uniform valued patterns, basis 2 should discover these uniform patterns; if segments express some trends, basis 6 should successfully capture these trends.

Note that the performance reported here is not optimal since we did not apply any sophisticated methods to find optimal parameter values. Therefore, with tailored model selection approaches, the clustering and segment performance may be further improved.

## References

[1] S. C. Madeira, M. C. Teixeira, I. Sá-Correia, and A. L. Oliveira, "Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 1, pp. 153–165, January-March 2010.

[2] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proc. the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, vol. 8, 2000, pp. 93–103.

[3] Y. Zhang, H. Zha, and C. Chu, "A time-series biclustering algorithm for revealing co-regulated genes," in *Proc. International Conference on Information Technology: Coding and Computation (ITCC'05)*, 2005, pp. 32–37.

[4] S. C. Madeira and A. L. Oliveira, "A linear time biclustering algorithm for time series gene expression data," in *Proc. the 5th Workshop on Algorithms in Bioinformatics (WABI'05)*, 2005, pp. 39–52.

[5] S. C. Madeira and A. L. Oliveira, "An efficient biclustering algorithm for finding genes with similar patterns in time-series expression data," in *Proc. the 5th Asia-Pacific Bioinformatics Conference (APBC'07)*, 2007, pp. 67–80.

[6] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra, "Minimum sum squared residue based coclustering of gene expression data," in *Proc. the 4th SIAM International Conference on Data Mining (SDM'04)*, 2004, pp. 114–125.

[7] I. S. Dhillon, Y. Guan, and J. Kogan, "Iterative clustering of high dimensional text data augmented by local search," in *Proc. the 2nd*

*IEEE International Conference on Data Mining (ICDM'02)*, 2002, pp. 131–138.
[8] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," *Journal of Machine Learning Research*, vol. 8, pp. 1919–1986, 2007.
[9] H. Cho, "Co-clustering algorithms: Extensions and applications," Ph.D. dissertation, The University of Texas at Austin, August 2008.
[10] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.
[11] H. Cho and I. S. Dhillon, "Co-clustering of human cancer microarrays using minimum sum-squared residue co- clustering," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 5, no. 3, pp. 385–400, 2008.
[12] H. Cho, "Data transformation for sum squared residue," in *Proc. the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10)*, Part I, *Lecture Notes in Artificial Intelligence 6118*, pp. 48-55, 2010.
[13] B. R. Kowalski and C. F. Bender, "Pattern recognition: A powerful approach to interpreting chemical data," *Journal of the American Chemical Society*, vol. 94, no. 16, pp. 5632–5639, 1972.
[14] O. E. Livne and G. H. Golub, "Scaling by binormalization," *Numerical Algorithms*, vol. 35, no. 1, pp. 97–120, 2004.
[15] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani, "Mining the stock market: Which measure is best?" in *Proc. the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'00)*, 2000, pp. 487–496.
[16] K. Hakamada, M. Okamoto, and T. Hanai, "Novel technique for preprocessing high dimensional time-course data from DNA microarray: mathematical model-based clustering," *Bioinformatics*, vol. 22, no. 7, pp. 843-848, 2006.
[17] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.
[18] M. Gavrilescu, G. W. Stuart, S. Rossell, K. Henshall, C. McKay, A. A. Sergejew, D. Copolov, and G. F. Egan, "Functional connectivity estimation in fMRI data: Influence of preprocessing and time course selection," *Human Brain Mapping*, vol. 29, pp. 1040–1052, 2008.

**Hyuk Cho** received the B.E. degree in computer engineering from Chonbuk National University, Korea, the M.A. degree in computer science from Korea University, Korea, and both the M.S. and the Ph.D. degrees in computer sciences from the University of Texas at Austin.

He is an assistant professor in the Department of Computer Science at Sam Houston State University, Huntsville, Texas, USA. He is known for his work on co-clustering algorithms, their extensions, and their applications to various practical tasks in real world problems. His research interests include data mining, statistical pattern recognition, machine learning, pervasive computing, bioinformatics, and data science. Previously he worked on linear matrix inequality and soft computing, including neural networks, evolutionary computation, genetic algorithm, and fuzzy/rough set theory.

Prof. Cho is a member of IEEE/ACM and SIAM.

**Min Kyung An** received her Ph.D. in computer science from the University of Texas at Dallas in August 2013, and her M.S. in computer science from the University of Texas at Arlington in August 2007. During her M.S. studies, she received the Graduate Studies Abroad Program Scholarship funded by the Korean government.

She is currently an assistant professor in the Department of Computer Science at Sam Houston State University, Huntsville, Texas, USA. Her major research areas include wireless ad hoc and sensor networks, social networks, design and analysis of approximation algorithms, graph theory, and program analysis.

Prof. An is a member of IEEE/ACM.