Minimum Sum-Squared Residue Co-clustering of Gene Expression Data *

Hyuk Cho †, Inderjit S. Dhillon †, Yuqiang Guan †, Suvrit Sra †

Abstract

Microarray experiments have been extensively used for simultaneously measuring DNA expression levels of thousands of genes in genome research. A key step in the analysis of gene expression data is the clustering of genes into groups that show similar expression values over a range of conditions. Since only a small subset of the genes participate in any cellular process of interest, by focusing on subsets of genes and conditions, we can lower the noise induced by other genes and conditions — a co-cluster characterizes such a subset of interest. Cheng and Church [3] introduced an effective measure of co-cluster quality based on mean squared residue. In this paper, we use two similar squared residue measures and propose two fast k-means like co-clustering algorithms corresponding to the two residue measures. Our algorithms discover k row clusters and l column clusters *simultaneously* while monotonically decreasing the respective squared residues. Our co-clustering algorithms inherit the simplicity, efficiency and wide applicability of the k-means algorithm. Minimizing the residues may also be formulated as trace optimization problems that allow us to obtain a spectral relaxation that we use for a principled initialization for our iterative algorithms. We further enhance our algorithms by an incremental local search strategy that helps avoid empty clusters and escape poor local minima. We illustrate co-clustering results on a yeast cell cycle dataset and a human B-cell lymphoma dataset. Our experiments show that our co-clustering algorithms are efficient and are able to discover coherent co-clusters.

Keywords: Gene-expression, co-clustering, biclustering, residue, spectral relaxation

1 Introduction

Microarrays simultaneously measure the expression levels of thousands of genes in a single experiment [5]. The results of a microarray experiment are often organized as gene expression matrices whose rows represent genes, and columns represent various environmental conditions or samples such as tissues. The entries of these matrices give a numeric representation of the expression/activity of a particular gene under a given experimental condition. Applications of microarrays range from the study of gene expression in yeast under different environmental stress conditions to the comparisons of gene expression profiles for tumors from cancer patients. In addition to the enormous scientific potential of DNA microarrays to help in understanding gene regulation and interactions, microarrays have important applications in pharmaceutical and clinical research. By comparing gene expression in normal and disease cells, microarrays may be used to identify disease genes and targets for therapeutic drugs.

Clustering algorithms have proved useful for grouping together genes with similar functions based on gene expression patterns under various conditions or across different tissue samples. Expanding functional families of genes with known function together with poorly characterized genes can help in understanding the functions of many genes for which such information is not yet available.

A wealth of work in cluster analysis of genes has been done, for example hierarchical clustering, self-organizing maps, graph-based algorithms, algorithms based on mixture models, neural networks, simulated annealing and algorithms based on principal components analysis. For a survey, see [12]. Dhillon et al. [9] present diametric clustering for identifying anti-correlated gene clusters for it is observed that genes that are functionally related may demonstrate strong anti-correlation in their expression levels. All of the above work is focussed on clustering genes using conditions as features.

Cheng and Church [3] proposed co-clustering (biclustering) of gene expression data and advocated the importance of such simultaneous clustering of genes and conditions for discovering more coherent and meaningful clusters. They formulated their problem of co-clustering by proposing a *mean squared residue* score for measuring cluster quality. Co-clustering may prove useful in practice since it is widely believed (as one may ascertain by scanning current bioinformatics research articles) that only a small subset of the genes participate in any cellular process of interest that takes place in only a subset of the conditions. By focusing on subsets of genes and conditions, we lower the bias exerted by other genes and conditions. Evidently co-clusters appear to be natural candidates for obtaining such coherent subsets of genes and conditions.

The development of our work is largely motivated by that of [3]. We formulate objective functions based on min-

*Supported by NSF CAREER Award No. ACI-0093404 and Texas Advanced Research Program grant 003658-0431-2001.

†Department of Computer Sciences, University of Texas, Austin, TX 78712-1188, USA

imizing two measures of squared residue that are similar to those used by Cheng and Church [3] and Hartigan [11]. Our co-clustering model is the partitioning model proposed by Hartigan [11], who also proposed hierarchical co-clustering models. Our formulation is thereby slightly different from [3] but this difference is necessitated by our ability to find $k \times l$ co-clusters simultaneously as opposed to finding a single co-cluster at a time like Cheng and Church. We propose iterative algorithms that have the benefits of simplicity and speed while directly optimizing the squared residues. Our co-clustering algorithms inherit the simplicity, efficiency and broad applicability of the k-means algorithm.

Minimizing the squared residues can be viewed as certain constrained trace maximization problems [19]. A relaxation of the constraints of these maximization problems makes them much easier to solve and leads to a spectral relaxation. We exploit this relaxation to develop a principled method for initializing our iterative algorithms. Not surprisingly, algorithms as the ones proposed herein, suffer from being trapped in local minima. To escape poor local minima in certain situations, we use a local search strategy [7, 21], that incrementally moves rows and columns among clusters if that leads to an improvement in the objective function. These incremental algorithms also provide benefits against empty clusters as shall become evident later in this paper. We observe at this point that though our work is largely motivated by Cheng and Church [3] and is focused on gene expression data, the techniques discussed herein are simple and as widely applicable as k-means.

The remainder of this paper is organized as follows. Section 2 gives a brief survey of related work and Section 3 describes the residue measures and resulting objective functions. In Section 4 we present our co-clustering algorithms that monotonically decrease the proposed objective functions along with proofs of monotonicity. To alleviate the problems of getting stuck in poor local minima, we describe local search procedures in Section 4.2. We propose a principled initialization method based on spectral techniques in Section 4.3. Detailed empirical results substantiating the usefulness of co-clustering are provided in Section 5. Finally we conclude with a brief summary and directions of future work in Section 6.

Notation: Upper-case boldfaced letters such as $\boldsymbol{X}$, $\boldsymbol{A}$ denote matrices while lower-case boldfaced letters like $\boldsymbol{x}$ denote column vectors. Column j and row i of matrix $\boldsymbol{X}$ are denoted $\boldsymbol{X}_{\cdot j}$ and $\boldsymbol{X}_{i\cdot}$ respectively, while X_{ij} or x_{ij} denotes the (i, j)-th element of $\boldsymbol{X}$. Upper-case letters I and J (subscripted or otherwise) respectively denote row and column index sets of a co-cluster. The norm $\|\boldsymbol{X}\|$ denotes the Frobenius norm of matrix $\boldsymbol{X}$, i.e., $\|\boldsymbol{X}\|^2 = \sum_{i,j} X_{ij}^2$.

2 Related work

One of the earliest co-clustering formulations, block clustering was introduced by Hartigan who called it "direct clustering" [11, 14]. Hartigan introduced various co-clustering quality measures and models including the partitional model used in this paper. However, [11] only gives a greedy algorithm for a hierarchical co-clustering model. This algorithm begins with the entire data in a single block and then at each stage finds the row or column split of every block into two pieces, choosing the one that produces largest reduction in the total within block variance. The splitting is continued till the reduction of within block variance due to further splitting is less than a given threshold. During the whole process, if there exist row splits that intersect blocks, one of them shall be used for the next row split, called a "fixed split". The same is done for columns. Otherwise, all split points are tried. By restricting the splits to fixed splits, it is ensured that: 1) the overall partition can be displayed as a contiguous representation, with a re-ordering of rows and columns; 2) the partitions of row and columns can be described in a hierarchical manner by trees. In contrast, our co-clustering algorithms are partitional algorithms and optimize *global* objective functions.

Baier et al. [2] propose overlapping and non-overlapping two-mode partitioning algorithms, of which the non-overlapping two-mode algorithm tries to minimize the same objective function as our Algorithm 4.1. The main difference between their non-overlapping two-mode partitioning algorithm and Algorithm 4.1 is in our intermediate updates of cluster prototypes.

Cheng and Church [3] propose a co-clustering algorithm for gene expression data using mean squared residue as the measure of the coherence of the genes and conditions. The algorithm produces one co-cluster at a time — a low mean squared residue plus a large variation from the constant gives a good criterion for identifying a co-cluster. A sequence of node (i.e. row or column) deletions and additions is applied to the gene condition matrix, while the mean squared residue of the co-cluster is kept under a given threshold. After each co-cluster is produced, the elements of the co-cluster are replaced with random numbers and then the same procedure is applied on the modified gene condition matrix to generate another, possibly overlapping, co-cluster till the required number of co-clusters is found. Their method finds one co-cluster at a time whereas our algorithms find $k \times l$ co-clusters simultaneously. In our co-clustering algorithms, as in the algorithm of [3], the row and column clustering depend on each other as opposed to some simple two-way clustering schemes that cluster rows and columns independently, see [17] for a discussion.

Yang et al. [18] point out that random numbers used as replacements in [3] can interfere with the future discovery of co-clusters, especially ones that have overlap with the discovered ones. They present an algorithm called FLOC (FLexible Overlapped biClustering) that simultaneously produces k co-clusters whose mean residues are all less than a pre-defined constant r. FLOC incrementally moves a row or column out of or into a co-cluster depending on whether the row or column is already included in that co-cluster or not, which is called an *action*. Then the best (one that gives the highest *gain*) action for a row or column, which is used to evaluate the relative reduction of the co-cluster's residue and the relative enlargement of the co-cluster's size, is performed. This is done for every row and column sequentially so $M + N$ co-clusterings are produced since there are M rows and N columns. The co-clustering with minimum mean residue is stored and the whole process is repeated. The idea of *action* is very similar to our incremental co-clustering algorithms (see Section 4.2).

Kluger et al. [13] apply a spectral co-clustering algorithm similar to the one proposed by Dhillon [6] on gene expression data to produce "checkerboard" structure. The largest several left and right singular vectors of the normalized gene expression matrix are computed and then a final clustering step using k-means and normalized cuts [15] is applied to the data projected to the topmost singular vectors. Different normalizations of genes and conditions are compared in [13]; however, the algorithms in [13] and [6, 20] model the gene expression matrix as a bipartite graph with non-negative edge weights where the quality of a co-cluster by the normalized cut criterion of [15]. Hence they are restricted to non-negative matrices. On the other hand, we used squared residue as a measure of the quality of a co-cluster, which is not restricted to non-negative matrices.

Dhillon et al. [8] propose an information-theoretic co-clustering algorithm that views a non-negative matrix as an empirical joint probability distribution of two discrete random variables and poses the co-clustering problem as an optimization problem in information theory: the optimal co-clustering maximizes the mutual information between the clustered random variables subject to constraints on the number of row and column clusters. Again, the restriction is to non-negative matrices but the algorithm is similar to our batch co-clustering algorithms with main difference of distance measure.

3 Squared residue

In this section we define *residue* and two different objective functions based on different squared residue measures.

Consider the data matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, whose (i,j)-th element is given by a_{ij}. We partition $\boldsymbol{A}$ into k row clusters and l column clusters defined by the following functions,

$$\begin{aligned} \rho : \{1,2,\ldots,m\} &\rightarrow \{1,2,\ldots,k\} \\ \gamma : \{1,2,\ldots,n\} &\rightarrow \{1,2,\ldots,l\}, \end{aligned}$$

where $\rho(i) = r$ implies that row i is in row cluster r; $\gamma(j) = s$ implies that column j is in column cluster s. Let I denote the set of indices of the rows in a row cluster and J denote the set of indices of the columns in a column cluster. The submatrix of $\boldsymbol{A}$ determined by I and J is called a *co-cluster*. In order to evaluate the homogeneity of such a co-cluster, we consider two measures:

1. The sum of squared differences between each entry in the co-cluster and the mean of the co-cluster.

2. 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.

These two considerations lead to two different measures of residue. We define the *residue* of an element a_{ij} in the co-cluster determined by index sets I and J to be

$$h_{ij} = a_{ij} - a_{IJ} \quad \text{for the first case,} \tag{3.1}$$
$$h_{ij} = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \quad \text{for the second case,} \tag{3.2}$$

where $a_{IJ} = \frac{\sum_{i \in I, j \in J} a_{ij}}{|I| \cdot |J|}$ is the mean of all the entries in the co-cluster, $a_{iJ} = \frac{\sum_{j \in J} a_{ij}}{|J|}$ is the mean of the entries in row i whose column indices are in J, and $a_{Ij} = \frac{\sum_{i \in I} a_{ij}}{|I|}$ is the mean of the entries in column j whose row indices are in I, where $|I|$ and $|J|$ denote the cardinality of I and J. Case (3.1) was the measure used by Hartigan [11] while case (3.2) in the context of gene expression data was used by Cheng and Church [3]. Let $\boldsymbol{H} = [h_{ij}]_{m \times n}$ be the residue matrix whose entries are described by either (3.1) or (3.2). Our optimization problems are to minimize the total squared residue and these result in the objective function

$$\|\boldsymbol{H}\|^2 = \sum_{I,J} \|\boldsymbol{H}_{IJ}\|^2 = \sum_{I,J} \sum_{i \in I, j \in J} h_{ij}^2, \tag{3.3}$$

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 (3.1) and (3.2). Consider the two different matrices,

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 2 & 3 & 4 \end{bmatrix}.$$

For both these matrices we would prefer the clustering (1122) for rows and (111222) for columns. This "desirable" clustering leads to zero residue by both measures for $\boldsymbol{A}_1$, but a first residue of 3.317 for $\boldsymbol{A}_2$. We might thus incline towards the second residue as the measure of choice, but we observe that for $\boldsymbol{A}_1$, even less desirable clusterings such as (1222) for rows and (111222) for columns, have zero second residue. In fact many more such "uninteresting" clusters give zero second residue for $\boldsymbol{A}_1$. Our experiments suggest that the second residue is a better measure for clustering gene expression data since it better captures the "trend" of the data, but other types of data could still benefit from the first measure.

Consider h_{ij} as defined by (3.1). We note that $\|\boldsymbol{H}_{IJ}\|^2 = 0$ if and only if all the entries in $\boldsymbol{H}_{IJ}$ are the same or $\boldsymbol{H}_{IJ}$ is trivial, i.e., it has one or no entry. If $k = m$, i.e., each row is in a cluster by itself, then $\|\boldsymbol{H}\|^2$ is the sum of squared Euclidean distance of every column vector to its column cluster mean vector, which is exactly what the k-means algorithm tries to minimize for column clustering. Now consider h_{ij} as defined by (3.2). We note that $\|\boldsymbol{H}_{IJ}\|^2 = 0$ if and only if the submatrix described by I and J is of the form $\boldsymbol{x}\boldsymbol{e}^T + \boldsymbol{e}\boldsymbol{y}^T$ where $\boldsymbol{e} = [1\ 1\ \ldots\ 1]^T$, $\boldsymbol{x}$ and $\boldsymbol{y}$ are arbitrary vectors. As seen below, if instead of the matrix $\boldsymbol{A}$ we consider the projected matrix $(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}$ then (3.2) gives the k-means objective function for this modified matrix. Thus, both residue measures may be viewed as generalizations of the one-dimensional k-means clustering objective to the co-clustering case.

Assume row-cluster r $(1 \le r \le k)$ has m_r rows, so that $m_1 + m_2 + \cdots + m_k = m$. Similarly, column-cluster c $(1 \le c \le l)$ has n_c columns, so that $n_1 + n_2 + \cdots + n_l = n$. Then, we define a *row cluster indicator matrix*, $\boldsymbol{R} \in \mathbb{R}^{m\times k}$ and a *column cluster indicator matrix*, $\boldsymbol{C} \in \mathbb{R}^{n\times l}$ 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 $\boldsymbol{R}$ has the form,

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

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. Matrix $\boldsymbol{C}$ has a similar structure. Therefore, $\|\boldsymbol{R}_{\cdot r}\|_1^2 = m_r$, and $\|\boldsymbol{C}_{\cdot c}\|_1^2 = n_c$. Note that $\boldsymbol{R}$ and $\boldsymbol{C}$ are column orthonormal matrices since the columns of $\boldsymbol{R}$ and $\boldsymbol{C}$ are clearly orthogonal and $\|\boldsymbol{R}_{\cdot r}\|_2 = 1, \|\boldsymbol{C}_{\cdot c}\|_2 = 1$. Using these definitions of $\boldsymbol{R}$ and $\boldsymbol{C}$ we can write both the residues compactly.

LEMMA 3.1. (RESIDUE MATRIX) *Suppose* $\boldsymbol{H} = [h_{ij}]$, *where* h_{ij} *is defined by* (3.1) *or* (3.2), *and* $\boldsymbol{R}$ *and* $\boldsymbol{C}$ *are the cluster indicator matrices as defined above. Then,*

$$\boldsymbol{H} = \boldsymbol{A} - \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T \quad \textit{for } (3.1), \tag{3.4}$$

$$\boldsymbol{H} = (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{C}\boldsymbol{C}^T) \quad \textit{for } (3.2). \tag{3.5}$$

Proof. Since there are k row clusters and l column clusters, we will write I_r and J_c to refer to the appropriate index sets when necessary. Consider,

$$\begin{aligned}
(\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A})_{ij} &= \sum_{r=1}^{k} R_{ir}(\boldsymbol{R}^T\boldsymbol{A})_{rj} \\
&= \sum_{r=1}^{k} R_{ir} \sum_{l=1}^{m} R_{lr}A_{lj} \\
&= \sum_{r=1}^{k} R_{ir} \sum_{l\in I_r} m_r^{-1/2}A_{lj}, \quad \big(R_{lr} = 0 \text{ for } l \notin I_r\big) \\
&= \sum_{r=1}^{k} R_{ir}m_r^{-1/2}m_rA_{I_rj}, \ \Big(|I_r| = m_r \text{ and } A_{I_rj} = \frac{1}{m_r}\sum_{l\in I_r} A_{lj}\Big) \\
&= m_t^{-1/2}m_t^{-1/2}m_t a_{I_tj}, \ \big(R_{ir} = 0 \text{ for all but one } r, \text{ say } r = t\big) \\
&= a_{I_tj}.
\end{aligned}$$

Thus the rows of $\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}$ give the row cluster mean vectors. In a similar fashion we conclude 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 column $j \in J$. Thus (3.4) and (3.5) follow. □

4 Algorithms

The residue matrix $\boldsymbol{H}$ leads to objective functions for minimizing squared residues: find row clusters I and column clusters J such that $\|\boldsymbol{H}\|^2 = \sum_{I,J}\|\boldsymbol{H}_{I,J}\|^2$ is minimized. For each definition of $\boldsymbol{H}$ we get a corresponding residue minimization problem. We refer to these minimization problems as our first and second problem 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. So we resort to iterative algorithms that monotonically decrease the objective functions and converge to a local minimum.

4.1 Batch iteration We first present batch iterative algorithms for our clustering problems. The algorithms operate in a batch fashion in the sense that at each iteration the column clustering $\boldsymbol{C}$ is updated only after determining the nearest column cluster for *every* column of $\boldsymbol{A}$ (likewise for rows). Define $\boldsymbol{A}^C = \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}$ and $\boldsymbol{A}^R = \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T$. Defining $\hat{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T = \boldsymbol{A}^C\boldsymbol{C}^T$, we can express $\|\boldsymbol{H}\|^2$

of (3.4) as

$$\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2 = \sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-\hat{\boldsymbol{A}}_{\cdot j}\|^2 \tag{4.6a}$$

$$= \sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-(\boldsymbol{A}^C\boldsymbol{C}^T)_{\cdot j}\|^2 \tag{4.6b}$$

$$= \sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-n_c^{-1/2}\boldsymbol{A}^C_{\cdot c}\|^2. \tag{4.6c}$$

Similarly we can decompose the objective function in terms of rows to obtain

$$\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2 = \sum_{r=1}^{k}\sum_{i\in I_r}\|\boldsymbol{A}_{i\cdot}-m_r^{-1/2}\boldsymbol{A}^R_{r\cdot}\|^2.$$

These simplifications lead to Algorithm 4.1. Notice that the columns and rows of the matrices $\boldsymbol{A}^C$ and $\boldsymbol{A}^R$ play the roles of column cluster prototypes and row cluster prototypes respectively. The algorithm begins out with some initialization (see Section 4.3 for more information) of $\boldsymbol{R}$ and $\boldsymbol{C}$. Each iteration involves finding the closest column (row) cluster prototype, given by a column (row) of $\boldsymbol{A}^C$ ($\boldsymbol{A}^R$), for each column (row) of $\boldsymbol{A}$ and setting its column (row) cluster accordingly. The algorithm iterates till the decrease in objective function becomes small as governed by the tolerance factor τ.

ALGORITHM 4.1: Co-clustering problem 1.

COCLUS_H1($\boldsymbol{A}$, k, l)
Input: Data matrix $\boldsymbol{A}$ and k, l
Output: Clustering matrices $\boldsymbol{R}$ and $\boldsymbol{C}$
Initialize $\boldsymbol{R}$ and $\boldsymbol{C}$
objval $\leftarrow \|\boldsymbol{A}-\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T\|^2$
$\Delta \leftarrow 1$; $\tau \leftarrow 10^{-2}\|\boldsymbol{A}\|^2$; {Adjustable parameter}
while $\Delta > \tau$
  $\boldsymbol{A}^C \leftarrow \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}$
  foreach $1 \le j \le n$
    $\gamma(j) \leftarrow \underset{1\le c\le l}{\operatorname{argmin}}\|\boldsymbol{A}_{\cdot j}-n_c^{-1/2}\boldsymbol{A}^C_{\cdot c}\|^2$ $(\star)$
  $\boldsymbol{C} \leftarrow$ Update using γ
  $\boldsymbol{A}^R \leftarrow \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T$
  foreach $1 \le i \le m$
    $\rho(i) \leftarrow \underset{1\le r\le k}{\operatorname{argmin}}\|\boldsymbol{A}_{i\cdot}-m_r^{-1/2}\boldsymbol{A}^R_{r\cdot}\|^2$ $(\star\star)$
  $\boldsymbol{R} \leftarrow$ Update using ρ
  oldobj $\leftarrow$ objval; objval $\leftarrow \|\boldsymbol{A}-\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T\|^2$
  $\Delta \leftarrow |$oldobj $-$ objval$|$

Before providing a proof of convergence of Algorithm 4.1, we need the following simple Lemma.

ALGORITHM 4.2: Co-clustering problem 2

COCLUS_H2($\boldsymbol{A}$, k, l)
Input: Data matrix $\boldsymbol{A}$ and k, l
Output: Clustering matrices $\boldsymbol{R}$ and $\boldsymbol{C}$
Initialize $\boldsymbol{R}$ and $\boldsymbol{C}$
objval $\leftarrow \|(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^T)\|^2$
$\Delta \leftarrow 1$; $\tau \leftarrow 10^{-2}\|\boldsymbol{A}\|^2$; {Adjustable parameter}
while $\Delta > \tau$
  $\boldsymbol{A}^C \leftarrow (\boldsymbol{I}-\boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}\boldsymbol{C}$
  $\boldsymbol{A}^P \leftarrow (\boldsymbol{I}-\boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}$
  foreach $1 \le j \le n$
    $\gamma(j) \leftarrow \underset{1\le c\le l}{\operatorname{argmin}}\|\boldsymbol{A}^P_{\cdot j}-n_c^{-1/2}\boldsymbol{A}^C_{\cdot c}\|^2$ $(\star)$
  $\boldsymbol{C} \leftarrow$ Update using γ
  $\boldsymbol{A}^R \leftarrow \boldsymbol{R}^T\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^T)$
  $\boldsymbol{A}^P \leftarrow \boldsymbol{A}(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^T)$
  foreach $1 \le i \le m$
    $\rho(i) \leftarrow \underset{1\le r\le k}{\operatorname{argmin}}\|\boldsymbol{A}^P_{i\cdot}-m_r^{-1/2}\boldsymbol{A}^R_{r\cdot}\|^2$ $(\star\star)$
  $\boldsymbol{R} \leftarrow$ Update using ρ
  oldobj $\leftarrow$ objval
  objval $\leftarrow \|(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{R}^T)\boldsymbol{A}(\boldsymbol{I}-\boldsymbol{C}\boldsymbol{C}^T)\|^2$
  $\Delta \leftarrow |$oldobj $-$ objval$|$

LEMMA 4.1. *Consider the function*

$$f(\boldsymbol{z}) = \sum_i \pi_i\|\boldsymbol{a}_i-\boldsymbol{M}\boldsymbol{z}\|^2, \quad \pi_i \ge 0, \tag{4.7}$$

$\boldsymbol{a}_i$*,* $\boldsymbol{z}$ *are vectors and* $\boldsymbol{M}$ *is a matrix of appropriate dimensions. Then* $f(\boldsymbol{z})$ *is minimized by* $\boldsymbol{z}^\star$ *that satisfies* $\pi\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{z}^\star = \boldsymbol{M}^T\boldsymbol{a}$*, where* $\pi = \sum_i \pi_i$ *and* $\boldsymbol{a} = \sum_i \pi_i\boldsymbol{a}_i$*.*

Proof. Expanding (4.7) we get

$$f(\boldsymbol{z}) = \sum_i \pi_i(\boldsymbol{a}_i^T\boldsymbol{a}_i - 2\boldsymbol{z}^T\boldsymbol{M}^T\boldsymbol{a}_i + \boldsymbol{z}^T\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{z}),$$

thus,

$$\frac{\partial f}{\partial \boldsymbol{z}} = \sum_i \pi_i(2\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{z} - 2\boldsymbol{M}^T\boldsymbol{a}_i).$$

On setting this gradient to zero we find that a minimizing $\boldsymbol{z}^\star$ must satisfy

$$\boldsymbol{M}^T\left(\sum_i \pi_i\boldsymbol{a}_i\right) = \left(\sum_i \pi_i\right)\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{z}. \qquad \square \tag{4.8}$$

Lemma 4.1 leads to the following corollary that we employ in our convergence proofs.

COROLLARY 4.1. *The* $\boldsymbol{z}^\star$ *minimizing* $f(\boldsymbol{z})$ *is given by*

$$\pi\boldsymbol{z}^\star = \boldsymbol{M}^T\boldsymbol{a}, \quad \textit{if } \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}, \tag{4.9}$$

$$\pi\boldsymbol{M}\boldsymbol{z} = \boldsymbol{M}^T\boldsymbol{a}, \quad \textit{if } \boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{M}. \tag{4.10}$$

THEOREM 4.1. (CONVERGENCE OF ALGORITHM 4.1) *Co-clustering Algorithm 4.1 decreases the objective function value* $\|\boldsymbol{H}\|^2$ *monotonically, where* $\boldsymbol{H}$ *is given by* (3.4).

Proof. Let the current approximation to $\boldsymbol{A}$ be denoted by $\hat{\boldsymbol{A}}$ and the approximation obtained after the greedy column assignments in step ($\star$) of Algorithm 4.1 be denoted by $\widetilde{\boldsymbol{A}}$. Denote the current column clustering by $\boldsymbol{C}$ and the new clustering obtained after the greedy step by $\widetilde{\boldsymbol{C}}$. The current and new column indices for column cluster c are denoted by J_c and $\widetilde{J}_c$ respectively. We have

$$
\begin{aligned}
&\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2\\
&=\sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-(\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T)_{\cdot j}\|^2,\\
&=\sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-n_c^{-1/2}\boldsymbol{A}^C_{\cdot c}\|^2\\
&\qquad\{\text{using (4.6a)–(4.6c)}\},\\
&\geq\sum_{c=1}^{l}\sum_{j\in J_c}\|\boldsymbol{A}_{\cdot j}-\boldsymbol{R}n_{\tilde{c}}^{-1/2}(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C})_{\cdot\tilde{c}}\|^2\\
&\qquad\{\text{from step }(\star)\text{ of Algorithm 4.1, }\tilde{c}=\gamma(j)\},\\
&\geq\sum_{c=1}^{l}\sum_{j\in\widetilde{J}_c}\left\|\boldsymbol{A}_{\cdot j}-\boldsymbol{R}\boldsymbol{R}^T\frac{1}{n_c}\sum_{t\in\widetilde{J}_c}\boldsymbol{A}_{\cdot t}\right\|^2\\
&\qquad\{\text{rearranging sum and (4.9) with }\boldsymbol{M}=\boldsymbol{R}\},\\
&=\sum_{c=1}^{l}\sum_{j\in\widetilde{J}_c}\|\boldsymbol{A}_{\cdot j}-n_c^{-1/2}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2\\
&=\sum_{c=1}^{l}\sum_{j\in\widetilde{J}_c}\|\boldsymbol{A}_{\cdot j}-n_c^{-1/2}\widetilde{\boldsymbol{A}}^C_{\cdot c}\|^2\\
&=\|\boldsymbol{A}-\widetilde{\boldsymbol{A}}\|^2.
\end{aligned}
$$

Thus the objective function is non-increasing under the column cluster updates. Similarly we can prove that the objective function is non-increasing under the row cluster updates (step ($\star\star$) of Algorithm 4.1). □

The batch iterative algorithm for the second problem is displayed as Algorithm 4.2. The algorithm is similar to the first problem except that $\boldsymbol{A}^C$ and $\boldsymbol{A}^R$ are now given differently due to the different objective function — the remaining structure of the procedure is unchanged[1].

THEOREM 4.2. (CONVERGENCE OF ALGORITHM 4.2) *Co-clustering Algorithm 4.2 decreases the objective function value* $\|\boldsymbol{H}\|^2$ *monotonically, where* $\boldsymbol{H}$ *is as in* (3.5).

Proof. The proof is similar to that of Theorem 4.1 ((4.10) is used with $\boldsymbol{M}=\boldsymbol{I}-\boldsymbol{R}\boldsymbol{R}^T$) and is omitted for brevity.

We would like to emphasize that Algorithms 4.1 and 4.2 can only guarantee convergence to a local minimum of the objective function value. In practice it has been observed that batch clustering algorithms make large changes to the objective function in their initial few iterations, thereafter effecting little changes. Once the batch algorithm converges it might suffer from two problems: 1) a poor local minimum, 2) the presence of empty clusters. In the next section we present a local search strategy that moves a single point (or in general a subset of points) from a given cluster to another if the move leads to a decrease in the objective function. Such a local search strategy has been shown to be effective in escaping poor local minima and avoiding empty clusters [7].

4.2 Incremental algorithms We now formulate incremental schemes for moving columns (rows) between column (row) clusters if such a move leads to decrease in the objective function. Each invocation of the incremental procedures tries to perform such a move for each row and column of the data matrix. Since moving a row or column from its current cluster to an empty cluster always leads to a decrease in the objective function (assuming non-degeneracy) such a move will always be made guaranteeing that no cluster is empty.

To aid the derivation of an efficient incremental update scheme we decompose the residue of (3.4) as follows.

$$
\begin{aligned}
\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2 &= \operatorname{Tr}((\boldsymbol{A}-\hat{\boldsymbol{A}})^T(\boldsymbol{A}-\hat{\boldsymbol{A}}))\\
&= \operatorname{Tr}(\boldsymbol{A}^T\boldsymbol{A})-2\operatorname{Tr}(\boldsymbol{A}^T\hat{\boldsymbol{A}})+\operatorname{Tr}(\hat{\boldsymbol{A}}^T\hat{\boldsymbol{A}})\\
&= \|\boldsymbol{A}\|^2-2\operatorname{Tr}(\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T)+\\
&\qquad \operatorname{Tr}(\boldsymbol{C}\boldsymbol{C}^T\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T)\\
&= \|\boldsymbol{A}\|^2-\|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2. \qquad (4.11)
\end{aligned}
$$

In the above derivation, we used the properties $\|\boldsymbol{X}\|^2=\operatorname{Tr}(\boldsymbol{X}^T\boldsymbol{X})$, $\operatorname{Tr}(\boldsymbol{A}+\boldsymbol{B})=\operatorname{Tr}(\boldsymbol{A})+\operatorname{Tr}(\boldsymbol{B})$, $\operatorname{Tr}(\boldsymbol{A}\boldsymbol{B})=\operatorname{Tr}(\boldsymbol{B}\boldsymbol{A})$ and the fact that $\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{I}$ and $\boldsymbol{C}^T\boldsymbol{C}=\boldsymbol{I}$. Similarly we can decompose the residue in (3.5) to yield

$$
\begin{aligned}
&\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2\\
&=\|\boldsymbol{A}\|^2-\|\boldsymbol{R}^T\boldsymbol{A}\|^2-\|\boldsymbol{A}\boldsymbol{C}\|^2+\|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2. \qquad (4.12)
\end{aligned}
$$

From (4.11) we see that minimizing $\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|^2$ is equivalent to maximizing $\|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2$. We can try to perform this maximization in the following way:

- Fix $\boldsymbol{R}$ and solve $\max_{\boldsymbol{C}}\|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2$.
- Fix $\boldsymbol{C}$ and solve $\max_{\boldsymbol{R}}\|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2$.

Let the current column clustering be given by $\boldsymbol{C}$ and the new clustering (that could be obtained by moving some

[1] In fact the same algorithm structure could be employed for co-clustering using any distortion measure that can be split up over rows and columns.

column of $\boldsymbol{A}$ to another cluster) be represented by $\widetilde{\boldsymbol{C}}$. Our aim to is maximize $\|\boldsymbol{R}^T\boldsymbol{A}\widetilde{\boldsymbol{C}}\|^2 - \|\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\|^2$ over $\widetilde{\boldsymbol{C}}$ that can arise from single moves. For notational convenience let us denote $\boldsymbol{R}^T\boldsymbol{A}$ by $\bar{\boldsymbol{A}}$. Suppose that a column j is moved from its current cluster c to cluster c'. Then, $\boldsymbol{C}$ and $\widetilde{\boldsymbol{C}}$ differ only in their columns c and c'. We find the difference in objective function values (as determined by $\boldsymbol{C}$ and $\widetilde{\boldsymbol{C}}$) to be,

$$\begin{aligned}&\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}\|^2 - \|\bar{\boldsymbol{A}}\boldsymbol{C}\|^2 = \\ &\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c'}\|^2 - \|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c'}\|^2 + \|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2 - \|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c}\|^2.\end{aligned} \tag{4.13}$$

Note that $\widetilde{\boldsymbol{C}}_{\cdot c}$ has $n_c - 1$ entries each of which equals $(n_c - 1)^{-1/2}$. Similarly each of the entries in $\boldsymbol{C}_{\cdot c'}$ equals $(n_{c'} + 1)^{-1/2}$.

A procedure that incrementally assigns columns to their closest column cluster is described below as Algorithm 4.3.

ALGORITHM 4.3: Local Search Step.

```
ColIncr_H1(R, A, l, γ)
Input: R, A, l, γ
Output: C
τ ← 10^{-5}||A||^2; {Adjustable parameter}
Ā ← R^T A
{Optimizing over column clusters}
for j = 1 to n
    for c' = 1 to l, c' ≠ γ(j) = c
        δ_j(c') ← ||Ā C̃_{·c'}||^2 − ||Ā C_{·c'}||^2 + ||Ā C̃_{·c}||^2 −
        ||Ā C_{·c}||^2                                          (⋆)
{Find best column to move along with best cluster}
(j⋆, c⋆) ← argmax_{(j,c)} δ_j(c)
if δ_{j⋆}(c⋆) > τ
    γ(j⋆) ← c⋆                                                  (⋆⋆)
{Update the cluster description matrix}
C ← Update using γ
```

The procedure for incrementally assigning rows to row clusters is similar. Notice that the algorithm ensures the change in objective function is monotonic. The algorithm above just makes one move at a time. One could also make a chain of moves (see for e.g. [7]) for obtaining better local minima. Variants of the algorithm perform any move that leads to a decrease in objective, and not insist on the best possible move.

Following exactly the same derivation we deduce that for the second problem the change in objective function on moving a column to another cluster is given by

$$\begin{aligned}&\|\boldsymbol{A}\widetilde{\boldsymbol{C}}_{\cdot c'}\|^2 - \|\boldsymbol{A}\boldsymbol{C}_{\cdot c'}\|^2 + \|\boldsymbol{A}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2 - \|\boldsymbol{A}\boldsymbol{C}_{\cdot c}\|^2 - \\ &\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c'}\|^2 + \|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c'}\|^2 - \|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2 + \|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c}\|^2.\end{aligned} \tag{4.14}$$

If we replace step $(\star)$ of Algorithm 4.3 by formula (4.14) then we obtain an incremental algorithm for the second problem.

Incremental algorithms such as the one described above often tend to be slow but sometimes we can at least speed up each iteration by performing the computations in a different manner. We now briefly look at simplifications that can enable us to greatly reduce the time of each iteration (at the expense of additional storage). Consider

$$\begin{aligned}\|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c}\|^2 &= \frac{1}{n_c}\Big(\sum_{j'\in J_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\Big)\Big(\sum_{j'\in J_c}\bar{\boldsymbol{A}}_{\cdot j'}\Big),\\ \|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2 &= \frac{1}{n_c-1}\Big(\sum_{j'\in \tilde{J}_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\Big)\Big(\sum_{j'\in \tilde{J}_c}\bar{\boldsymbol{A}}_{\cdot j'}\Big).\end{aligned}$$

Therefore, if column j belongs to cluster c,

$$\begin{aligned}&(n_c-1)\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2 - n_c\|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c}\|^2\\ &= \Big(\sum_{j'\in \tilde{J}_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\Big)\Big(\sum_{j'\in \tilde{J}_c}\bar{\boldsymbol{A}}_{\cdot j'}\Big)\\ &\quad - \Big(\sum_{j'\in J_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\Big)\Big(\sum_{j'\in J_c}\bar{\boldsymbol{A}}_{\cdot j'}\Big)\\ &= -\bar{\boldsymbol{A}}^T_{\cdot j}\sum_{j'\in J_c}\bar{\boldsymbol{A}}_{\cdot j'} - \sum_{j'\in J_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\bar{\boldsymbol{A}}_{\cdot j} + \bar{\boldsymbol{A}}^T_{\cdot j}\bar{\boldsymbol{A}}_{\cdot j}\\ &= -2\sum_{j'\in J_c}\bar{\boldsymbol{A}}^T_{\cdot j'}\bar{\boldsymbol{A}}_{\cdot j} + \bar{\boldsymbol{A}}^T_{\cdot j}\bar{\boldsymbol{A}}_{\cdot j}.\end{aligned} \tag{4.15}$$

Similarly, we get

$$\begin{aligned}&(n_{c'}+1)\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c'}\|^2 - n_{c'}\|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c'}\|^2\\ &= 2\sum_{j'\in J_{c'}}\bar{\boldsymbol{A}}^T_{\cdot j'}\bar{\boldsymbol{A}}_{\cdot j} - \bar{\boldsymbol{A}}^T_{\cdot j}\bar{\boldsymbol{A}}_{\cdot j}.\end{aligned} \tag{4.16}$$

Thus, if we store $\bar{\boldsymbol{A}}^T\bar{\boldsymbol{A}}, \|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c}\|^2$ and $\|\bar{\boldsymbol{A}}\boldsymbol{C}_{\cdot c'}\|^2$ in main memory, then we can compute $\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c}\|^2$ and $\|\bar{\boldsymbol{A}}\widetilde{\boldsymbol{C}}_{\cdot c'}\|^2$ in constant time according to (4.15) and (4.16) respectively. Therefore, $\delta(c')$ in step $(\star)$ of Algorithm 4.3 can be updated in constant time. Following the same idea, if we additionally store $\boldsymbol{A}^T\boldsymbol{A}, \|\boldsymbol{A}\boldsymbol{C}_{\cdot c}\|^2$ and $\|\boldsymbol{A}\boldsymbol{C}_{\cdot c'}\|^2$ in main memory, we can also update $\delta(c')$ based on (4.14) efficiently.

In our implementation we go one step further and employ a "ping-pong" approach wherein we alternate between the invocations of the batch and incremental algorithms (see Figures 1 and 2 to assess usefulness).

4.3 Spectral approximation for initialization Till now we have tacitly assumed the presence of some initialization (as induced by $\boldsymbol{R}$ and $\boldsymbol{C}$) for our algorithms. In this section we look more carefully at a method for a principled initialization scheme.

In minimizing the original objective functions the difficulty is introduced by the strong structural constraints on

$\boldsymbol{R}$ and $\boldsymbol{C}$—viz., constraining $\boldsymbol{R}$ and $\boldsymbol{C}$ to be cluster indicator matrices. If we relax those constraints to just seek column orthogonal matrices $\boldsymbol{R}$ and $\boldsymbol{C}$, i.e., $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_k$ and $\boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}_l$, then the minimization is dramatically eased.

Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be the singular value decomposition (SVD) of $\boldsymbol{A}$ and $\boldsymbol{A}_s = \boldsymbol{U}_s\Sigma_s\boldsymbol{V}_s^T$ be the rank-s SVD approximation to $\boldsymbol{A}$ (note that if $s > \mathrm{rank}(\boldsymbol{A})$ then $\boldsymbol{A}_s = \boldsymbol{A}$). We use the fact that the best rank-s approximation to a matrix $\boldsymbol{A}$, where the approximation error is measured by the Frobenius norm, is given by $\boldsymbol{A}_s$ [10]. This fact allows us to show that both the residues (3.4) and (3.5) are minimized by selecting $\boldsymbol{R} = \boldsymbol{U}_k$ and $\boldsymbol{C} = \boldsymbol{V}_l$. We find that the minimum residue is achieved when $\hat{\boldsymbol{A}} = \boldsymbol{A}_s = \boldsymbol{U}_s\boldsymbol{\Sigma}_s\boldsymbol{V}_s^T$ where $s = \min(k,l)$ for the first problem, and $s = \max(k,l)$ for the second. We verify the second claim as follows. Let $\boldsymbol{R} = \boldsymbol{U}_k$ and $\boldsymbol{C} = \boldsymbol{V}_l$, then

$$\begin{aligned}\hat{\boldsymbol{A}} &= \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A} + \boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T - \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T \\ &= \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{A} + \boldsymbol{A}\boldsymbol{V}_l\boldsymbol{V}_l^T - \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{A}\boldsymbol{V}_l\boldsymbol{V}_l^T \\ &= \boldsymbol{A}_k + \boldsymbol{A}_l - \boldsymbol{A}_k\boldsymbol{V}_l\boldsymbol{V}_l^T \\ &= \boldsymbol{A}_s, \quad s = \max(k,l).\end{aligned}$$

Note that the relaxation above allows us to obtain lower bounds on both the objective functions. The squared residues are lower bounded by $\sigma_{s+1}^2 + \cdots + \sigma_{\mathrm{rank}(\boldsymbol{A})}^2$. Due to their global nature, spectral techniques seem to offer an ability for superior initializations. After obtaining a relaxed solution we have to somehow obtain a co-clustering. One approach is to cluster the rows of $\boldsymbol{R}$ and $\boldsymbol{C}$ using k-means and obtain row and column clusters from the clustered $\boldsymbol{R}$ and $\boldsymbol{C}$. We could also follow an approach based on QR factorization as proposed by Zha et al. [19].

4.4 Computational complexity We briefly remark on the computational complexity of our algorithms. Consider Algorithm 4.1. We need not carry out an explicit computation of $\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^T$. Instead, we just need to compute $\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{C}$ and update the cluster assignment vectors ρ and γ appropriately. The former takes $O(N)$ time, whereas the computations for updating the clusterings can be performed in $O(N(k+l))$ time per iteration. Thus the overall complexity of Algorithm 4.1 is $O(t(k+l)N)$ where t is the number of iterations. It is easy to observe that the computational complexity of Algorithm 4.2 is the same. Algorithm 4.3 can be implemented to require $O(nl)$ operations if we make use of the speedup suggestions in Section 4.2.

5 Experimental results

We now provide experimental results to illustrate the behavior of our algorithms. We witness that co-clustering allows us to capture the "trends" of genes over a subset of the total number of conditions. We select two commonly used gene expression datasets, viz., a yeast *Saccharomyces cerevisiae* cell cycle expression dataset from Cho et al. [4] and human B-cell lymphoma expression dataset from [1]. The preprocessed gene expression matrices are obtained from http://arep.med.harvard.edu/biclustering/ [3].

5.1 Description of the datasets The yeast cell cycle dataset contains 2884 genes and 17 conditions. To avoid distortion or biases arising from the presence of missing values in the data matrix we remove all the genes that had any missing value. This step results in a matrix of size 2882×17. The preprocessed matrix contains integers in the range 0 to 595. More details about this particular dataset and its preprocessing can be found in Cheng and Church [3]. The human lymphoma dataset has 4026 genes and 96 conditions. The preprocessed data matrix has integer entries in the range -749 to 642. After removing all the genes having any missing value as before, the matrix is reduced to a smaller matrix of size 854×96. We note that though the size of the matrix is substantially reduced, without recourse to some principled missing value replacement it is improper to use the entire data matrix.

5.2 Implementation details Our algorithms are implemented in C++, all experiments are performed on a PC(Linux, Intel Pentium 2.53GHz), and all figures are generated with MATLAB. We tested a variety of co-clusterings with differing number of row and column clusters. However, we illustrate results with 50 gene clusters and 2 condition clusters for the yeast dataset and 20 gene clusters and 2 condition clusters for the human lymphoma dataset. These cluster numbers are chosen with consideration to the dimension of the data matrix and to make it easy to illustrate co-clusters. We do not want to put too many or too few genes in each co-cluster, but we want to demonstrate coherence of a subset of genes over a subset of conditions in each co-cluster. In all our experiments, we set $\tau = 10^{-2}\|\boldsymbol{A}\|^2$ for both Algorithm 4.1 and Algorithm 4.2. Also, we fix $\tau = 10^{-5}\|\boldsymbol{A}\|^2$ and 20 as the chain length for local search. With random initialization, Algorithm 4.1 generates the co-clusters in 16 seconds for the yeast dataset and in 5 seconds for the lymphoma dataset whereas Algorithm 4.2 takes 14 and 20 seconds respectively for these datasets.

5.3 Analysis of co-clustering results To demonstrate the advantage of spectral initialization we conduct the following set of experiments. Each algorithm is run 20 times on the yeast and the lymphoma datasets, with random and spectral initialization, respectively. We averaged the initial and final objective function values over these 20 trials. Table 1 shows the averaged values for the yeast dataset. Observe that spectral initialization yields lower initial objective function values than random initialization and better final objective function values for both algorithms. Similarly Table 2 shows

Table 1: Average initial and final objective function values over 20 runs for the yeast data set. Here $\|\boldsymbol{A}\|^2 = 2.89236 \times 10^9$.

	Random	Spectral
Alg. 4.1 (initial)	6.6081×10^8	$\mathbf{3.9277 \times 10^8}$
(final)	5.4192×10^7	$\mathbf{5.4115 \times 10^7}$
Alg. 4.2 (initial)	5.0466×10^8	$\mathbf{3.6359 \times 10^8}$
(final)	1.9337×10^7	$\mathbf{1.9278 \times 10^7}$

Table 2: Average initial and final objective function values over 20 runs for the human lymphoma dataset. Here $\|\boldsymbol{A}\|^2 = 5.63995 \times 10^8$.

	Random	Spectral
Alg. 4.1 (initial)	5.6328×10^8	$\mathbf{5.3239 \times 10^8}$
(final)	4.8209×10^8	$\mathbf{4.8129 \times 10^8}$
Alg. 4.2 (initial)	5.0754×10^8	$\mathbf{4.1596 \times 10^8}$
(final)	2.7036×10^8	$\mathbf{2.6854 \times 10^8}$

that spectral initialization performs better than random initialization for the human lymphoma dataset.

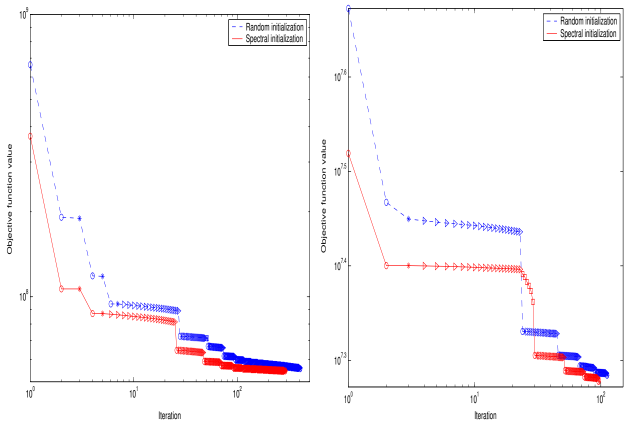

Figure 1: Objective function value vs. iteration by Algorithm 4.1 (left plot) and Algorithm 4.2 (right plot) for the yeast dataset. (50 gene clusters and 2 condition clusters)

Figures 1 and 2, in logarithmic scales, show the monotonic decrease in objective function values with the progress of iterations for both algorithms. In the figures, iteration refers to one of the followings: row batch update (denoted as a circle), column batch update (an asterisk), row local search step (a triangle), and column local search step (a square). As shown in the figures, though initial objective function values

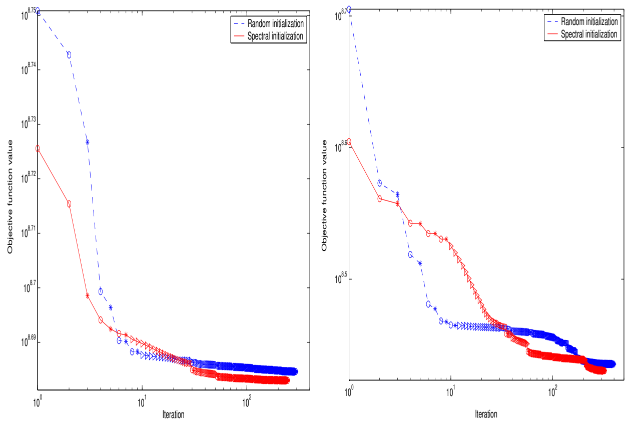

Figure 2: Objective function value vs. iteration by Algorithm 4.1 (left plot) and Algorithm 4.2 (right plot) for the lymphoma dataset. (20 gene clusters and 2 condition clusters)

with random and spectral initialization are quite different, the final objective function values are similar. We ascribe this to the employment of both batch update and incremental local search in "ping-pong" manner, where the incremental local search algorithm refines the clustering produced by the batch algorithms and triggers further runs of the batch algorithms. Thus this ping-pong strategy produces stair-shaped objective function curves as shown in Figures 1 and 2. For example, in the left plot of Figure 1, the algorithms will terminate after 6 batch steps (with random initialization) and 4 batch steps (with spectral initialization) without the incremental algorithm. However, after taking the chain of incremental local searches, the objective function values decrease several times until it converges. We also want to mention that our incremental local search algorithm can remove empty clusters generated by the batch algorithms because moving a vector into an empty cluster always decreases objective function values.

Due to space limitations, we present only some exemplary co-clusters obtained by our co-clustering algorithms with spectral initialization. In Figures 3-7, x-axis lists the number of the conditions and y-axis gives the gene expression level.

Figure 3 shows four co-clusters of yeast data generated by Algorithm 4.1, while Figure 4 shows eight co-clusters generated by Algorithm 4.2. From the figures we see that both Algorithm 4.1 and Algorithm 4.2 can identify groups of genes and groups of conditions that exhibit similar expression patterns. In other words, they discover a set of genes that display homogeneous patterns of expression levels over a subset of conditions.

Each cluster in Figure 5, from top to bottom and from

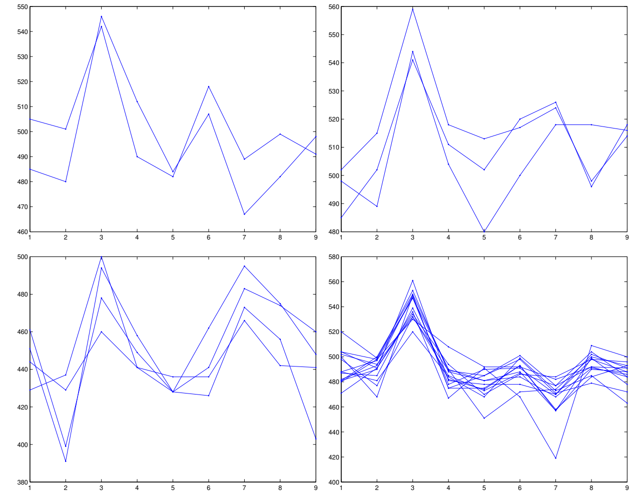

Figure 3: Co-clusters discovered from the yeast dataset by Algorithm 4.1 using spectral initialization. Each co-cluster consists of the following number of genes and conditions in the format of (number of genes; number of conditions) from top-left to bottom-right plot: $(2; 9)$, $(3; 9)$, $(4; 9)$, and $(14; 9)$.

left to right, is generated by combining the two co-clusters in each row of Figure 4. The average expression level (a thick red line) for each cluster is shown for interpretation purpose. These four concatenated clusters are closely related with the clusters of Tavazoie et al. [16], where one-way Euclidean k-means clustering algorithm was applied to cluster genes into different regulation classes, as follows: The top-left cluster is related to their cluster 1, the top-right cluster is related to their cluster 7, the bottom-left cluster is related to their cluster 2, and the bottom-right cluster is related to their cluster 12. Thus co-clustering does not prevent us from discovering relations that can be discovered by one-way clustering.

We observe that Algorithm 4.2 appears to generate more meaningful co-clusters than Algorithm 4.1 because the residue measure used by Algorithm 4.2 captures the coherence trends of genes and conditions of the form $\boldsymbol{e}\boldsymbol{x}^T + \boldsymbol{y}\boldsymbol{e}^T$ while the residue used in Algorithm 4.1 captures the uniformity of a co-cluster.

We conducted similar experiments on the human lymphoma dataset and some exemplary co-clusters are shown in Figures 6 and 7. As before, Algorithm 4.2 appears to capture more meaningful co-clusters than Algorithm 4.1. Both algorithms discover several co-clusters that consist of only several genes, but large number of conditions. For example, the top-left co-cluster in Figure 7 has 4 genes and the top-right co-cluster in Figure 7 contains 5 genes, behaving similarly across 83 out of total 96 conditions. Also, we observe that several co-clusters that contain a large subset of genes behaving similarly across a small number of conditions are discovered. For example, the bottom-left co-cluster consists of 72 genes and the bottom-right co-cluster has 106 genes, containing 13 conditions for both co-clusters. These co-clusters display the broad trends over genes or conditions, either of which may contain a smaller subset of related co-clusters. All results shown herein were obtained by running our algorithms with spectral initialization without any further post-processing, however, our algorithms could be recursively applied on each co-cluster to obtain a finer partition.

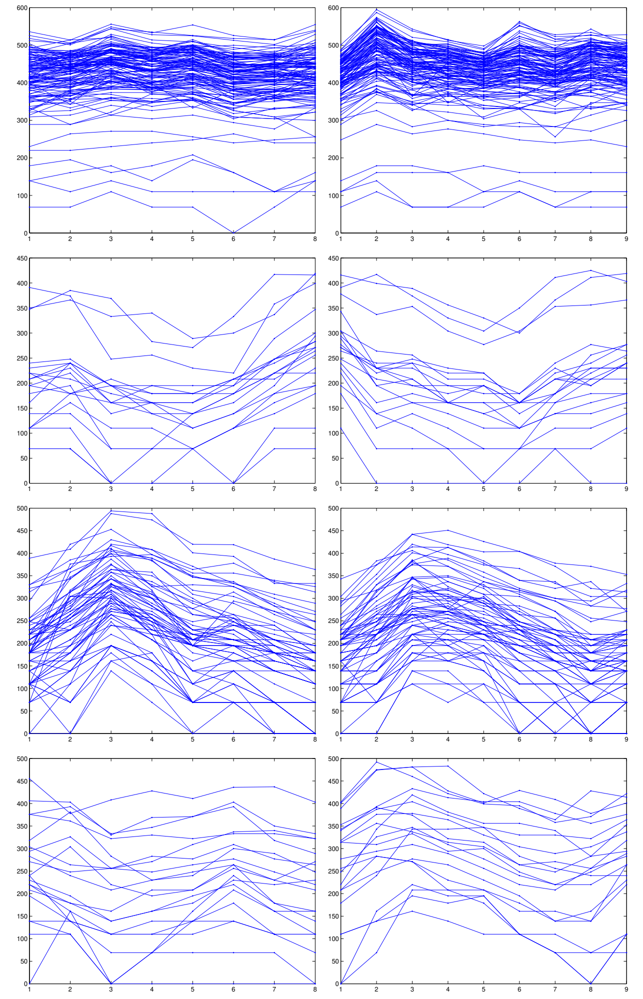

Figure 4: Co-clusters discovered from the yeast dataset by Algorithm 4.2 using spectral initialization. Note that two co-clusters in the same row have same genes. Each co-cluster consists of the following number of genes and conditions in the format of (number of genes; number of conditions) from top-left to bottom-right plot: $(124; 8)$, $(124; 9)$, $(19; 8)$, $(19; 9)$, $(63; 8)$, $(63; 9)$, $(20; 8)$, and $(20; 9)$.

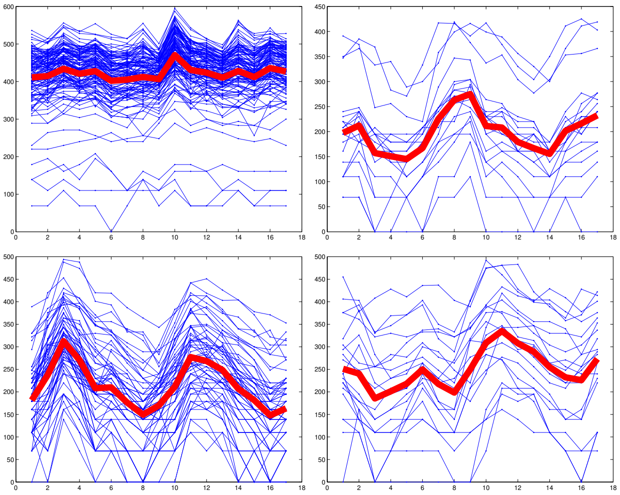

Figure 5: Gene clusters obtained by combining adjacent co-clusters in Figure 4 for comparison with Tavazoie et al. [16]. Each cluster consists of the following number of genes and conditions in the format of (number of genes; number of conditions) from top-left to bottom-right plot: $(124; 17)$, $(19; 17)$, $(63; 17)$, and $(20; 17)$. Each of these plots is closely related to clusters discovered in Tavazoie et al. [16], where one-way clustering was used to cluster genes.

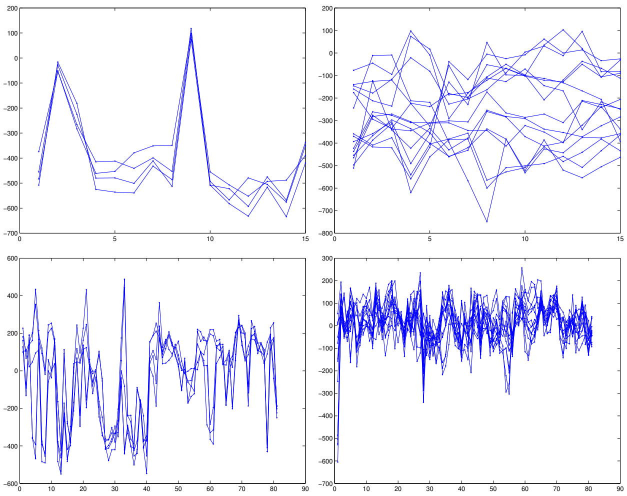

Figure 6: Co-clusters discovered from the human lymphoma dataset by Algorithm 4.1 using spectral initialization. Each co-cluster consists of the following number of genes and conditions in the format of (number of genes; number of conditions) from top-left to bottom-right plot: $(4; 15)$, $(5; 15)$, $(5; 81)$, and $(15; 81)$.

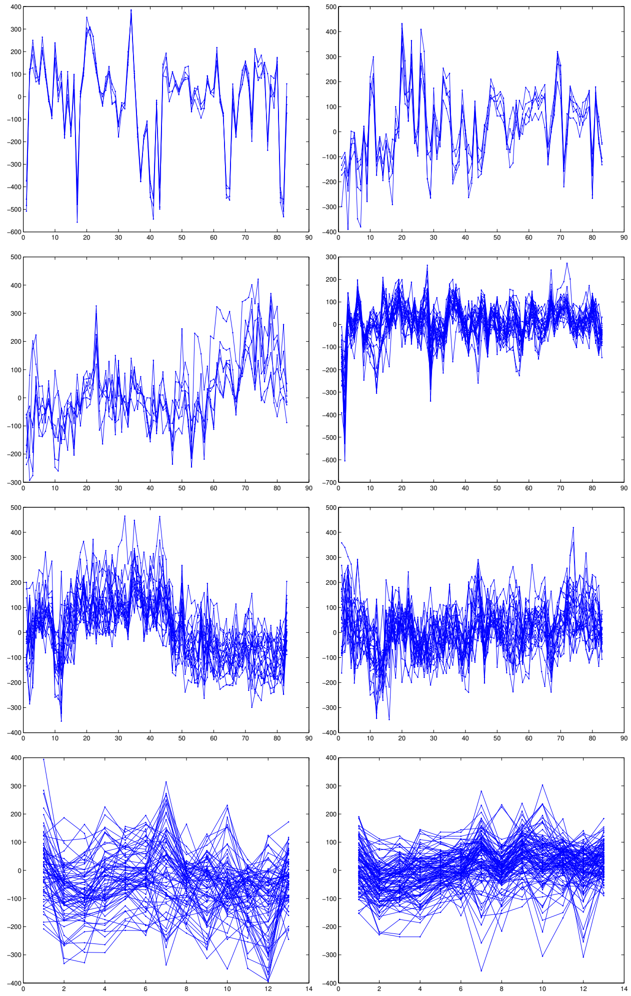

Figure 7: Co-clusters discovered from the human lymphoma dataset by Algorithm 4.2 using spectral initialization. Each co-cluster consists of the following number of genes and conditions in the format of (number of genes; number of conditions) from top-left to bottom-right plot: $(4; 83)$, $(5; 83)$, $(7; 83)$, $(26; 83)$, $(21; 83)$, $(25; 83)$, $(72; 13)$, and $(106; 13)$.

6 Conclusions & future work

Our main contributions in this paper are: 1) we propose two efficient k-means like co-clustering algorithms to simultaneously find k row and l column clusters, 2) an initialization method using spectral relaxation of a trace optimization problem, 3) a local search strategy that prevents poor local optima and empty clusters. We expect this framework to have as broad applicability to co-clustering as the popular k-means algorithm has to one-way clustering. An issue that is the subject of future exploration is a "good" way of evaluating co-clusterings, especially in the absence of truelabels for one or both dimensions of the data. We also plan to investigate the biological significance of co-clustering results on various gene expression datasets.

In contrast to [3], our co-clustering algorithms produce non-overlapping clusters. In the future, we plan to extend our work to generate overlapping clusters like in [3] and soft clustering that allows weighted memberships in multiple clusters. We envisage a generalization of our methods to multi-way k-means type procedures applied to multi-dimensional data matrices on contingency tables.

An intriguing future application of co-clustering would be to fill in missing values that frequently occur in gene expression matrices. Anti-correlation has been observed to imply functional similarity of genes [9]; we plan to extend our co-clustering algorithms to detect such anti-correlations.

References

[1] A. A. Alizadeh and M. B. Eisen et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–510, 2000.

[2] D. Baier, W. Gaul, and M. Schader. Two-mode overlapping clustering with applications to simultaneous benefit segmentation and market structuring. In *Classification and Knowledge Organization*, pages 557–566. Springer, 1997.

[3] Y. Cheng and G. Church. Biclustering of expression data. In *Proceedings ISMB*, pages 93–103. AAAI Press, 2000.

[4] R. Cho, M. Campbell, E Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[5] J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–686, 1997.

[6] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of The 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2001)*, pages 269–274, 2001.

[7] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, 2002.

[8] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98, 2003.

[9] I. S. Dhillon, E. M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19(13):1612–1619, September 2003.

[10] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1: 211–218, 1936.

[11] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67 (337):123–129, March 1972.

[12] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

[13] Y. Kluger, R. Basri, J.T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.*, 13:703–716, 2003.

[14] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[16] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature genetics*, 22(3), 1999.

[17] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of dna microarray data. Technical report, Department of Health Research and Policy, Statistics, Genetics and Biochemistry, Stanford University, October 1999.

[18] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. *Proc. of 3rd IEEE Symposium on BioInformatics and BioEngineering (BIBE'03)*, pages 321–327, 2003.

[19] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Neural Info. Processing Systems*, 2001.

[20] H. Zha, X. He, C. Ding, M. Gu, and H.D. Simon. Bipartite graph partitioning and data clustering. In *Proc. 10th Int'l Conf. Information and Knowledge Management (ACM CIKM)*, 2001.

[21] B. Zhang, G. Kleyner, and M. Hsu. A local search approach to k-clustering. Technical Report HPL-1999-119, HP Lab Palo Alto, Oct. 1999.