A Generalized Framework for Mining Arbitrarily Positioned Overlapping Co-clusters

Omar Odibat* Chandan K. Reddy†

Abstract

The goal of co-clustering is to simultaneously cluster both rows and columns in a given matrix. Motivated by several applications in text mining, recommendation systems and bioinformatics, different methods have been developed to discover local patterns that cannot be identified by traditional clustering algorithms. In spite of much research in this domain, existing co-clustering algorithms have some critical limitations in terms of identifying co-clusters with different types of correlations in the data and the ability to capture overlapping co-clusters in the data matrix. In this paper, we present a new deterministic co-clustering algorithm, POsitive and NEgative correlation based Overlapping Co-Clustering (PONEOCC), which can be used to find significant co-clusters efficiently. Our algorithm uses a novel ranking-based objective function that is optimized to simultaneously find large co-clusters with minimum residual errors. It allows positively and negatively correlated objects to be members of the same co-clusters and can extract overlapping co-clusters. In addition, the co-clusters can be arbitrarily positioned in the data matrix. We evaluated our algorithm on several synthetic and real-world gene expression datasets, and the experimental results showed that PONEOCC is able to find biologically significant co-clusters and also outperformed some of the well-known existing co-clustering algorithms in terms of the quality, size and biological significance of the co-clusters.

Keywords: Co-clustering; positive and negative correlation; biclustering; gene expression data.

1 Introduction

Traditional clustering algorithms, such as $k-$means and hierarchical clustering, assign every data point to a cluster based on a similarity measure computed across all the features. In some applications, traditional clustering algorithms cannot capture the structural patterns in the data [1]. Since these algorithms assume that correlated rows (columns) share similar patterns across all the columns (rows), such algorithms fail to discover local patterns that exist in subsets of rows (columns) [8] .

Given a data matrix with two entities (objects, features), such as $(words, documents)$ in text mining, $(users, movies)$ in recommendation systems and $(genes, samples)$ in bioinformatics, a subset of rows may be inter-related under a subset of columns forming blocks of substructures (co-clusters). For example, a set of genes may be co-expressed under a subset of samples and applying traditional clustering techniques cannot capture such blocks [1]. However, co-clustering has emerged as a powerful tool to simultaneously cluster both dimensions of a data matrix by utilizing the relationship between the two entities [22]. Co-clustering helps in discovering local patterns that cannot be identified by the traditional one-way clustering algorithms.

Compared to traditional one-dimensional clustering, co-clustering is considered more informative and more scalable [2] because it simultaneously measures the degree of coherence in the samples samples and in the attributes of a given matrix. [11]. Moreover, considering co-clusters rather than the whole data matrix reduces the noise induced on the whole data set [10]. Co-clustering has been used in several applications such as clustering microarray data [18], identifying protein interactions [14], collaborative filtering [9], text mining [5], matrix approximation [22]. In this paper, we focus on applying co-clustering in biological applications such as gene expression data analysis for local patterns.

In the (κ, ℓ) co-clustering model [1, 8], the goal is to find a grid structure comprised of κ row clusters and ℓ column clusters such that a certain objective function is maximized or minimized. In such a model, not all the rows and columns of the original data matrix participate in this partitioning. However, the assumption here is that the rows in each row cluster should be correlated under each of the ℓ column clusters, and the columns in each column cluster should be correlated under each of the κ clusters. Such an assumption may not hold when a subset of rows is correlated in a limited subset of columns or vice versa. To overcome this limitation, we develop a novel co-clustering algorithm that is able to capture a subset of rows that are correlated in any subset of the ℓ column clusters as well as to capture a subset of columns that are correlated in any subset of the κ row clusters.

The remainder of this paper is organized as follows: Section 2 presents a brief overview of the related research. Section 3 summarizes the motivation and the contributions of our work. Section 4 explains the importance of co-clustering in gene expression analysis and highlights the key challenges addressed by our framework. Section 5 presents the different steps of the proposed *PONEOCC* algorithm. Section 6

*Dept. of Computer Science, Wayne State University, Detroit, MI 48202, USA. Email: odibat@wayne.edu

†Dept. of Computer Science, Wayne State University, Detroit, MI 48202, USA. Email: reddy@cs.wayne.edu.

presents the experimental results of the proposed algorithms on synthetic and real datasets along with the biological evaluation and comparisons with other algorithms available in the literature. Finally, we conclude our discussion in Section 7.

2 Relevant Literature

Discovering the set of co-cluster in a given data matrix is an important challenge. It has been proved that the problem of finding all the significant co-clusters is an NP-hard problem [6]. In this section, we present some of the existing algorithms and describe more about some of the popularly used ones.

2.1 Co-clustering Algorithms Cheng and Church (CC) [6] proposed a Mean Squared Residue (MSR) function to measure the homogeneity of the co-clusters. This MSR measure gives lower scores to co-clusters with lower variance. The algorithm starts with the original data matrix; then a set of row/column deletions and additions are applied to produce one co-cluster, which will be replaced with random numbers. This procedure is repeated until a certain number of co-clusters is obtained. The algorithm has two main limitations: (i) It finds only one co-cluster at a time, and (ii) random interference (masking the discovered co-clustered with random numbers) reduces the quality of the co-clusters and obstructs the discovery of other co-clusters.

The Order-Preserving Submatrices (OPSMs) [4] algorithm finds local patterns in which the expression levels of all genes induce the same linear ordering of the experiments. A co-cluster is considered order-preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. However, the OPSM algorithm finds only one co-cluster at a time and captures only positively correlated genes. Our algorithm, on the other hand, finds all the co-clusters simultaneously so that both types of correlation are allowed to be in the same co-cluster.

Iterative Signature Algorithm (ISA) [12] is a statistical co-clustering algorithm which defines a transcription module (co-cluster) as a co-regulated set of genes under a set of experimental conditions. ISA starts from a set of randomly selected genes (or conditions) that are iteratively refined until they are mutually consistent. At each iteration, a threshold is used to remove noise and to maintain co-regulated genes and the associated co-regulating conditions.

Robust Overlapping Co-clustering (ROCC) [8] is a co-clustering algorithm that works with several Bregman divergences. This model allows overlapping co-clusters and finds k row clusters and l column clusters simultaneously. ROCC performs co-clustering through two steps. In the first step the Bregman co-clustering algorithm [2] is used to find co-clusters arranged in a grid structure. In the second step, co-clusters with large errors are pruned; then similar co-clusters are merged. This algorithm does not handle the negative correlation among the rows. In [18], a survey of co-clustering algorithms is presented. In our work, we primarily compare our results with some of these popular algorithms used for finding co-clusters in gene-expression data.

2.2 Our Contributions In this paper, we present a novel deterministic co-clustering algorithm, POsitive and NEgative correlation based Overlapping Co-Clustering (PONEOCC), which can be used to extract significant co-clusters from gene expression data efficiently. Some of the important contributions of PONEOCC are as follows:

- A novel ranking-based objective function is proposed to find the co-clusters with minimum errors.
- Both of positively and negatively correlated genes are allowed to be members of the same co-cluster.
- Ability to extract arbitrarily positioned overlapping co-clusters simultaneously.
- Ability to identify large co-clusters that are biologically significant.

3 Motivation

In this section, we present our contributions with some examples to illustrate the capabilities of the proposed algorithm, and then we present an overview of the proposed algorithm.

Given a $m \times n$ data matrix, the goal of co-clustering is to find a row mapping (ρ) that maps the rows to the κ row clusters and a column mapping (γ) that maps the columns to the ℓ column clusters:

$$\rho : \{1,2,...,m\} \longrightarrow \{1,2,...,\kappa\}$$

$$\gamma : \{1,2,...,n\} \longrightarrow \{1,2,...,\ell\}$$

During the search process, an objective function is optimized and the co-clusters are scored so that the algorithm outputs the co-clusters with high scores (or low errors). This objective function is based on the scores of all the $\kappa * \ell$ co-clusters. In other words, the rows (columns) are assigned to the co-clusters based on the overall objective function.

When the algorithm is completed, the co-clusters with the highest scores are reported, and those with low scores (high errors) are pruned [8]. However, this optimizing procedure can miss some of the co-clusters that have strongly correlated genes just because these genes are not correlated in other co-clusters. To illustrate this point we present the following two examples.

3.1 Motivating Example 1: In Figure 1(a), an example of 9 co-clusters arranged in a 3×3 grid structure is shown.

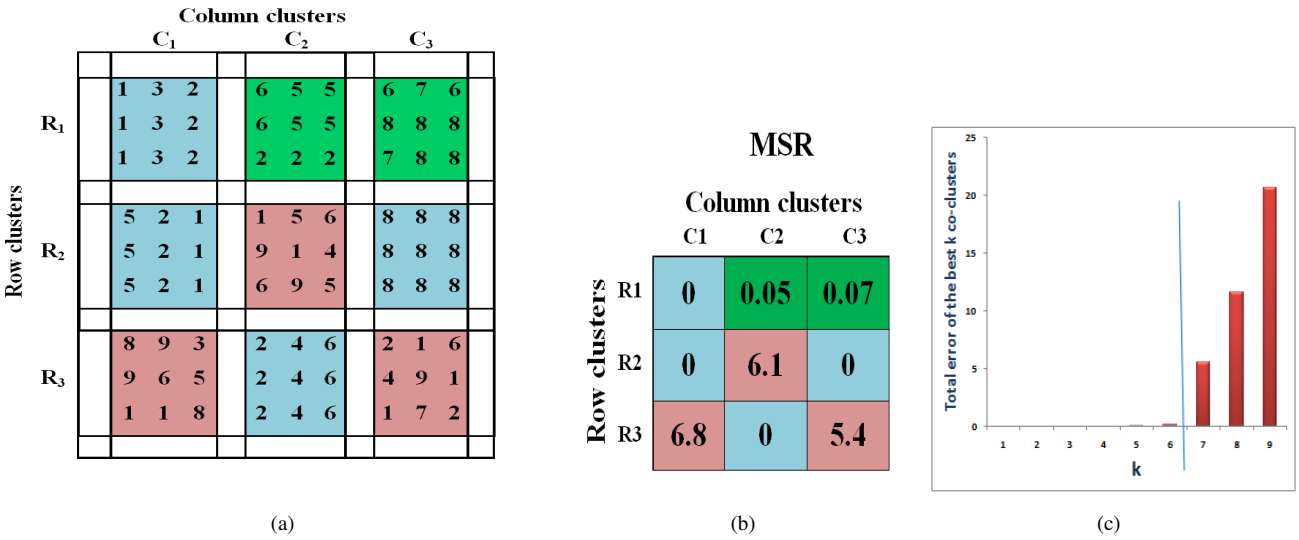


Figure 1: Motivating example 1: (a) Nine co-clusters arranged in a 3×3 grid structure. (b) The error of each co-cluster measured by MSR [6]. (c) The accumulated sum of the error of the best K co-clusters is shown in the Y-axis. The value of K is shown on the X-axis (the cut-off is based on elbow point criterion).

The corresponding error for each co-cluster is shown in Figure 1(b). The error is measured by the mean-squared residue (MSR) score given by Equation (5.1). Let us consider the co-cluster present in the intersection of the third row cluster and the second column cluster. This co-cluster has an error of 0, which means that this is a perfect co-cluster. However, since the other co-clusters in the same row cluster have high error values, this co-cluster will be missed by the existing algorithms. Our objective function depends only on the score of the top-ranked co-clusters. So, in this example if 70% of the co-clusters are included in the objective function (as represented by the vertical line shown in Figure 1(c)), then the proposed algorithm will be able to identify the six best co-clusters regardless of the score of the three remaining co-clusters. The co-clusters found by our algorithm are the highly ranked ones which are unknown in advance, arbitrarily positioned, and can be changed during the iterative re-assignment step.

3.2 Motivating Example 2: Figure 2 shows two co-clusters of size 4×4. The MSR of the first co-cluster is 0.098, and the MSR of the second co-cluster is 2.723, which means that the first co-cluster is more homogenous than the second one. Given a new row, as shown in Figure 2, the question is: can we add it to the current co-clusters or not? If this row is to be added; then only the error of the first co-cluster will be reduced. Specifically, the MSR of the first co-cluster will be reduced to 0.085, but the error of the second co-cluster will be increased to 4.47. That is, the average MSR of the two co-clusters before adding the new row is 1.41 while the average MSR of the two co-clusters after adding the new row is 2.273. Therefore, the row will not be added to the current co-clusters because of the high error of the second co-cluster, which will be pruned eventually. In this paper, we propose a new objective function that considers the score of the top-ranked co-clusters when rows (or columns) are to be added/removed. Therefore, when our algorithm is applied to this example, this row will be added because it improves the score of the co-cluster that has the maximum score already. We will show that by using the new objective function, it is possible to obtain improved results by focusing on the discovery of high quality co-clusters.

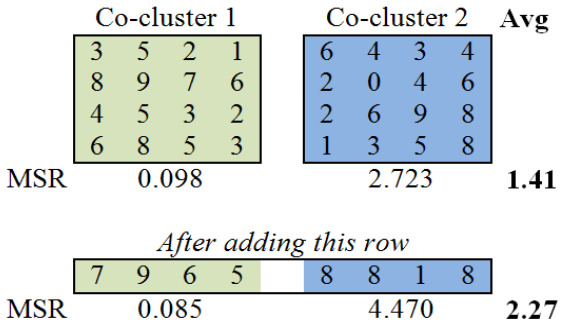


Figure 2: Motivating example 2: Two co-clusters are shown with their corresponding MSR. The problem here is to decide whether to add the new row to the current solution or not.

3.3 Overview of the PONEOCC Algorithm In this subsection, we briefly present the main steps of the proposed algorithm. More details of the proposed algorithm will be given in detail in Section 5. The goal of this algorithm is to find a set of arbitrarily positioned overlapping co-clusters. As shown in Figure 3(a), the algorithm starts with a two dimensional matrix (objects $\times$ features) as an input. In the first step, Figure 3(b), a divisive approach is used for initialization. Basically, it starts with all the rows and columns in one co-cluster; then the algorithm divides the co-cluster with the largest error. This iterative procedure continues until the κ row clusters and the ℓ column clusters are obtained. The core co-clustering step, Figure 3(c), finds the optimal row and column clusterings (ρ, γ). This is the most important step in our algorithm in which the rows and columns of the data matrix are re-ordered.

In the third step, Figure 3(d), similar co-clusters are merged using a hierarchical agglomerative approach. The two most similar co-clusters are merged. Finally, in the fourth step, Figure 3(e), more rows and columns are added to each co-cluster individually. The goal of this refinement step is to obtain larger overlapping co-clusters.

4 Co-clustering Gene Expression Data

The recent advances in DNA microarray technology allow genomewide expression profiling. In analyzing such high-dimensional biological data, one can observe that the activities of genes are not independent of each other. Therefore, it is important to study groups of genes rather than to perform a single gene analysis. However, traditional clustering techniques, such as $k-$means, assume that related genes should have similar expression profiles in all the samples [16]. This assumption does not hold in all of the experiments. From the biological perspective, not all the genes are involved in each biological pathway and some of these pathways may be active under a subset of the conditions [18]. Co-clustering was proposed to overcome the limitations of traditional clustering algorithms [24].

There are several requirements that should be considered while searching for co-clusters in gene expression data. A subset of genes can be correlated only in a small subset of conditions due to the heterogeneity of the samples which could be taken from different patients. Moreover, a gene can be involved in more than one biological pathway; therefore, there is a need for a co-clustering algorithm that allows overlapping between the co-clusters [8], i.e., the same gene can be a member of more than one co-cluster. In addition, since genes can be positively or negatively correlated [13], it is important to allow both types of correlation in the same co-cluster. Furthermore, the co-clusters can be arbitrarily positioned in the gene expression data. Finally, there are several types of co-clusters that can be biologically relevant [18]. The existing algorithms do not include all of these important requirements. The proposed algorithm supports the discovery of large and possibly overlapping co-clusters that contain positively and negatively correlated genes.

4.1 Positive and Negative Correlation There are different types of correlations between genes in any cell. Examples of such relationships are positive and negative correlations [19]. Figure 4 shows an example these correlations. In a positive correlation, genes show similar patterns while in a negative correlation, genes show opposite patterns. Since it is possible that genes with both types of correlations exist in the same biological pathway [13], there is a need for a computational model that captures both types of correlations simultaneously. However existing co-clustering algorithms capture positive correlations only. In this paper, we introduce a novel algorithm that can be used in a systematic way to capture positive and negative correlations simultaneously.

4.2 Overlapping Co-clusters Discovering overlapping patterns is a challenging task in data mining [17]. For example, a gene can be involved in more than one biological process. Therefore, that gene can belong to more than one co-cluster. One of the main advantages of our algorithm is that it allows overlapping between coclusters, as shown in Figure 3(c), which helps in understanding the different roles played by a particular gene in a living cell.

4.3 Finding Large Co-clusters Extracting large co-clusters is helpful in gaining more knowledge about new genes or new patients. In our algorithm, the last two steps help in computing large and overlapping co-clusters. The third step, Figure 3(c), merges similar co-clusters into larger co-clusters. The fourth step, Figure 3(d), adds more genes and conditions to each co-cluster individually.

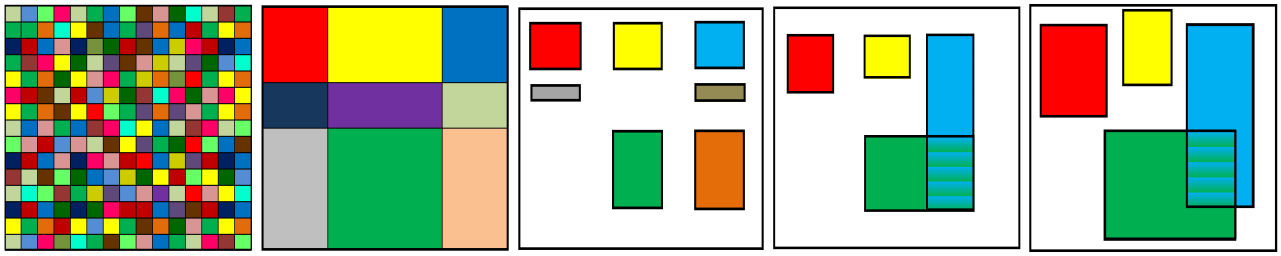
5 The Proposed Framework

In this section, we explain the proposed co-clustering algorithm. The main steps of the proposed algorithm are illustrated in Figure 3. The notations used in this paper are described in Table 1.

5.1 Measuring the Coherence Expression: Cheng and Church proposed using the mean-squared residue (MSR) score as a measurement of the coherence between genes [6]. Given a gene expression submatrix X that has I genes and J conditions, the residue is computed as follows:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (x_{ij} - x_{Ij} - x_{iJ} + x_{IJ})^2 \tag{5.1}$$

where $x_{iJ} = \frac{\sum_{j \in J} x_{ij}}{|J|}$ is the row mean, $x_{Ij} = \frac{\sum_{i \in I} x_{ij}}{|I|}$ is the column mean and $x_{IJ} = \frac{\sum_{i \in I, j \in J} x_{ij}}{|I| * |J|}$ is the overall mean of the matrix X. Given two genes, a and b, and J conditions the



(a) Input data matrix (b) **Step 1**: Initialization (c) **Step 2**: Core Co-clustering (d) **Step 3**: Merging (e) **Step 4**: Refinement

Figure 3: Main steps of the proposed algorithm which finds arbitrarily positioned overlapping co-clusters. (a) The input is a two dimensional matrix. (b) **Step1**: the initialization step partitions the data matrix into κ by ℓ grid structure using a divisive approach. (c) **Step 2**: the core-clustering step finds the optimal row and column clusterings (ρ,γ). (d) **Step 3**: similar co-clusters are merged using agglomerative approach. (d) **Step 4**: In the refinement step, more rows and columns are added to each co-cluster individually.

Table 1: Notations used in this paper

Notation	Description
D	data matrix $D \in \mathbb{R}^{m\times n}$
m	total number of rows (genes) in D
n	total number of columns (conditions) in D
κ	number of row clusters
ℓ	number of column clusters
ρ	mapping of row clusters
γ	mapping of column clusters
η	number of optimized co-clusters
α,k	indices for row clusters
β,l	indices for column clusters
θ	vector containing the signs of genes

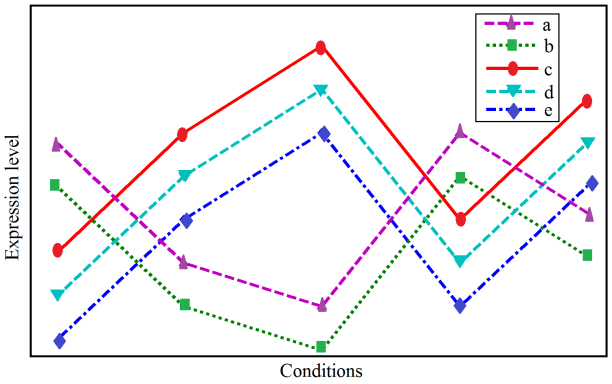


Figure 4: Different types of relationships between genes in one co-cluster. The genes $\{a,b\}$ are positively correlated with each other, and the genes $\{c,d,e\}$ are positively correlated with each other. However, the genes $\{a,b\}$ are negatively correlated with the genes $\{c,d,e\}$.

MSR, defined in Eq. (5.1), can be re-written as follows:

$$\begin{aligned} h(a,b,J) &= \frac{1}{2|J|}\sum_{j\in J}\left(a_j-\bar{a}-\frac{a_j+b_j}{2}+\frac{\bar{a}+\bar{b}}{2}\right)^2 \\ &\quad +\frac{1}{2|J|}\sum_{j\in J}\left(b_j-\bar{b}-\frac{a_j+b_j}{2}+\frac{\bar{a}+\bar{b}}{2}\right)^2 \\ &= \frac{1}{|J|}\sum_{j\in J}\left(\frac{(a_j-\bar{a})-(b_j-\bar{b})}{2}\right)^2 \end{aligned} \tag{5.2}$$

where $\bar{a}$ and $\bar{b}$ represent the mean of the expression values for genes a and b respectively. The goal here is to find co-clusters with low MSR values. A perfect co-cluster will have MSR $= 0$. We modified this function to ensure that the results are normalized to $[0,1]$ with 1 indicating a perfect co-cluster. First, the data is normalized, so that the values of the genes are between 0 and 1. Second, we used the following modified function:

$$H'(I,J) = 1-\frac{1}{|I||J|}\sum_{i\in I,j\in J}(x_{ij}-x_{Ij}-x_{iJ}+x_{IJ})^2 \tag{5.3}$$

Note that based on (5.3), an optimal co-cluster has a value of 1, which results from the case where $(a_j-\bar{a}) = (b_j-\bar{b})\ \forall j\in J$. In a negative correlation, genes have opposite patterns. i.e. the two negatively correlated genes will get a perfect score when $(a_j-\bar{a}) = -(b_j-\bar{b})\ \forall j\in J$. A positive correlation between any two genes is measured as

$$h'_{+}(a,b,J) = 1-\frac{1}{|J|}\sum_{j\in J}\left(\frac{(a_j-\bar{a})-(b_j-\bar{b})}{2}\right)^2 \tag{5.4}$$

and a negative correlation between any two genes is measured as

$$h'_{-}(a,b,J) = 1-\frac{1}{|J|}\sum_{j\in J}\left(\frac{(a_j-\bar{a})+(b_j-\bar{b})}{2}\right)^2 \tag{5.5}$$

Algorithm 1 Update_F(κ, ℓ, ρ ,γ, η)

1: **Input:** Number of row clusters (κ)
Number of column clusters (ℓ)
Rows clustering (ρ)
Columns clustering (γ)
Number of optimized co-clusters (η)
2: **Output:** The value of the objective function (F:)
3: **Procedure:**
/* Get the score (Π) of each co-cluster */
4: **for** each $\alpha \in [1,\kappa]$ **do**
5: **for** each $\beta \in [1,\ell]$ **do**
6: $\Pi(\alpha,\beta) \leftarrow H^{'}(u,v,s) : \rho(u) = \alpha, \gamma(v) = \beta$
7: **end for**
8: **end for**
/* Rank the co-clusters based on their scores */
9: **for** each $\alpha \in [1,\kappa]$ **do**
10: **for** each $\beta \in [1,\ell]$ **do**
11: $\Re(\alpha,\beta) \leftarrow I(\sum_{k=1}^{\kappa}\sum_{l=1}^{\ell} I(\Pi(k,l) > \Pi(\alpha,\beta)) < \eta)$
12: **end for**
13: **end for**
/* Compute the value of the objective function */
14: $F \leftarrow \sum_{\alpha=1}^{\kappa}\sum_{\beta=1}^{\ell} \Pi(\alpha,\beta) * \Re(\alpha,\beta)$

5.2 Ranking-based Objective Function Let $\Re$ be a ranked list of the scores computed using the $H^{'}$ measure, the objective function is computed as the sum of the top η scores, where η is the number of co-clusters to be included in the function ($1 \leq \eta \leq \kappa * \ell$). The value of the objective function is the average of the top η co-clusters. This objective function optimizes for the best co-clusters rather than considering all the co-clusters. The η co-clusters are not known in advance and can be changed in the algorithm due to the change in the row/colmn clustering.

In this paper, η is computed as $\eta = r * \kappa * \ell$ where r is the ratio of co-clusters that will contribute in the computation of the objective function. In this paper, we set r to 70%. The scores are computed using the objective function F as shown in Algorithm 1. First, the scores of each co-cluster are computed *(lines 4-8)*. Second, the scores are ranked *(lines 9-13)*. Finally, the objective function is computed as the sum of the scores of the top ranked co-clusters *(line 14)*. $I(x)$ is the indicator function that is 1 if the value of x is true, and it is 0 otherwise. The objective function will be computed for each possible change in the row/column mapping to maintain non-decreasing values for the objective function. Figure 3 shows the main steps of the proposed approach. Next, we present the steps of the proposed algorithm in detail.

To handle negative and positive correlations, the algorithm maintains a vector that contains the signs of the genes (θ). The signs indicate the type of correlation. For instance, the signs for the genes (a,b,c,d,e) in Figure 4 are $\theta = (+1,-1,+1,+1,-1)$. The values of genes with negative sign values can be transformed so that all the genes are positively correlated. If $b^{'} = 1-b$ and $e^{'} = 1-e$; then the new data matrix contains the genes $(a,b^{'},c,d,e^{'})$ and $\theta = (+1,+1,+1,+1,+1)$.

Algorithm 2 *Initialize*(D,m,n,κ, ℓ)

1: **Input:** Data matrix (D)
No. of rows (m)
No. of columns (n)
No. of row clusters (κ)
No. of column clusters (ℓ)
2: **Output:** Row clustering (ρ)
Columns clustering (γ)
3: **Procedure:**
/* Assign all the rows and columns to one co-cluster */
4: $k \leftarrow 1,\ l \leftarrow 1$
5: $\rho(g) \leftarrow k, \forall [g]_1^m$
6: $\gamma(c) \leftarrow l, \forall [c]_1^n$
/* Iterative partitioning of the co-clusters */
7: **while** $k < \kappa$ **or** $l < \ell$ **do**
8: **if** $k < \kappa$ **then**
9: $k \leftarrow k+1$
10: $\alpha \leftarrow \arg\min_{\alpha} \sum_{l=1}^{\ell} H^{'}(u,v) \ : \rho(u) = \alpha, \gamma(v) = l$
11: $(g_i,g_j) \leftarrow \arg\min_{(i,j)} \sum_{\substack{1 \leq l \leq \ell \\ (g_i,g_j) \in \alpha}} h_{+}^{'}(g_i,g_j,v) \ : \gamma(v) = l$
12: **for** $g_a \in \alpha, g_a \neq g_i, g_a \neq g_j$ **do**
13: **if** $\sum_{l=1}^{\ell} h_{+}^{'}(g_a,g_i,v) > \sum_{l=1}^{\ell} h_{+}^{'}(g_a,g_j,v) \ : \gamma(v) = l$ **then**
14: $\rho(g_a) \leftarrow k$
15: **end if**
16: **end for**
17: **end if**
/* Partition the column clusters */
18: **if** $l < \ell$ **then**
19: $l \leftarrow l+1$
20: $\beta \leftarrow \arg\min_{\beta} \sum_{k=1}^{\kappa} H^{'}(u,v) \ : \rho(u) = k, \gamma(v) = \beta$
21: $(c_i,c_j) \leftarrow \arg\min_{(i,j)} \sum_{\substack{1 \leq k \leq \kappa \\ (c_i,c_j) \in \beta}} h_{+}^{'}(c_i,c_j,u) \ : \rho(u) = k$
22: **for** $c_b \in \beta, c_b \neq c_i, c_b \neq c_j$ **do**
23: **if** $\sum_{k=1}^{\kappa} h_{+}^{'}(c_b,c_i,u) > \sum_{k=1}^{\kappa} h_{+}^{'}(c_b,c_j,u) \ : \rho(u) = k$ **then**
24: $\gamma(c_b) \leftarrow k$
25: **end if**
26: **end for**
27: **end if**
28: **end while**

5.3 Input The inputs to this algorithm include the original matrix $D \in \mathbb{R}^{m \times n}$, the number of genes m and the number of conditions n, number of row clusters κ and the number of column clusters ℓ. The κ and ℓ parameters are common parameters in the co-clustering methods that use the $\kappa \times \ell$ model such as [8], and they can be set based on the size of the data matrix. The ratio r is also specified so that the objective function seeks the best mapping for the top η co-clusters, where $\eta = r * \kappa * \ell$. We set r to 70% in all of the

experiments. The data is normalized so that the values of each genes are in $[0,1]$. Therefore, the objective function has a lower bound of 0 and upper bound of 1.

5.4 PONEOCC Algorithm The PONEOCC algorithm has four major steps which are shown in Figure 3 and described in detail below.

5.4.1 STEP 1: Initialization Inspired by the bisecting K-means cluster technique [23], we used a deterministic algorithm for the initialization as shown in Algorithm 2. In this approach, each row in the dataset is mapped to one of the κ clusters, and each column is mapped to one of the ℓ clusters resulting in a checkerboard structure as shown in Figure 3(b). The initialization algorithm is a divisive algorithm that starts with the complete data assigned to one co-cluster (lines 4-6). In each iteration, the row cluster with the highest error (lowest $H^{'}$) is selected *(line 10)*. Then, the two genes with the lowest correlation score are identified *(line 11)*. These two genes will be the seeds for the next partition to be computed *(lines 12-17)*. Each of the remaining genes will be added to the first (second) partition if it is more correlated to the first (second) gene. This is done using $h^{'}_{+}$ function. Therefore, the gene sign vector will contain only positive signs after this step *(lines 5-6)* (Algorithm 3). The conditions are clustered in the same manner *(lines 18-27)*. Eventually, we will have the complete dataset organized as a $\kappa \times \ell$ grid structure.

5.4.2 STEP 2: Core Co-clustering (ρ,γ) This is the most important step that finds the optimal row and column clusterings (ρ,γ) as shown in Figure 3(c). In this iterative algorithm, ρ and γ are updated in each iteration. To update ρ, each row is considered for one of the following three possible actions as described in Algorithm 3:

- Exclude the current row from any row cluster by setting ρ to 0 *(lines 10-11)*.
- Find the best row cluster to include the current row as a positively correlated row *(lines 12-14)*.
- Find the best row cluster to include the current row as a negatively correlated row *(lines 15-17)*.

The objective function, F, is computed for each possible action, and the action to be carried out is the one corresponding to the maximum value of the three objective function values*(lines 18-26)*. Following this strategy, the value of the objective function is maintained to be non-decreasing. One of the advantages of the proposed algorithm is that it allows different types of correlation to be in the same co-cluster. The θ sign vector is important because it keeps track of the type of correlation for each row (positive and negative: $+1$ and -1).

Algorithm 3 *PONEOCC*(D,m,n,κ, ℓ,η)

1: **Input:** Data matrix (D)
No. of rows (m)
No. of columns (n)
No. of row clusters (κ)
No. of column clusters (ℓ)
No. of optimized co-clusters (η)
2: **Output:**: A set of co-clusters (ζ)
3: **Procedure:**
4: **Step1** : initialization
5: $(\rho,\gamma) \leftarrow$ Initialize(D,m,n,κ,ℓ)
/* Initialize the sign of each row (gene) to be positive */
6: $\theta(g) \leftarrow 1 \forall g \in \{1,2,...,m\}$
7: **Step**2 : core co_clustering
8: **repeat**
9: **for** $a = 1 : m$ **do**
10: $\rho_0 \leftarrow \rho$, $\rho_0(a) \leftarrow 0$
/* Compute F when the current row is excluded */
11: $F_0 \leftarrow$ Update_$F(\kappa,\ell,\eta,\rho_0,\gamma)$
12: $\rho_{gP} \leftarrow \rho$, $\theta(a) \leftarrow 1$
13: $g_P \leftarrow \arg\max_{g_P}$ Update_$F(\kappa,\ell,\eta,\rho_{g_P},\gamma,\theta)$
/* Current row is considered positively correlated */
14: $F_{g_P} \leftarrow$ Update_$F(\kappa,\ell,\eta,\rho_{g_P},\gamma,\theta)$
15: $\rho_{g_N} \leftarrow \rho$, $\theta(a) \leftarrow -1$
16: $g_N \leftarrow \arg\max_{g_N}$ Update_$F(\kappa,\ell,\eta,\rho_{g_N},\gamma,\theta)$
/* Current row is considered negatively correlated */
17: $F_{g_N} \leftarrow$ Update_$F(\kappa,\ell,\eta,\rho_{g_N},\gamma,\theta)$
/* Select the best row clustering */
18: **if** $F_{g_P} > F_{g_N}$ **and** $F_{g_P} > F_{g_0}$ **then**
19: $\rho \leftarrow \rho_{g_P}$
20: $\theta(a) \leftarrow 1$
21: **else if** $F_{g_N} > F_{g_P}$ **and** $F_{g_N} > F_{g_0}$ **then**
22: $\rho \leftarrow \rho_{g_N}$
23: $\theta(a) \leftarrow -1$
24: **else**
25: $\rho \leftarrow \rho_0$
26: **end if**
27: **end for**
/* Column clustering */
28: **for** $b = 1 : n$ **do**
29: $\gamma_0 \leftarrow \gamma$, $\gamma_0(b) \leftarrow 0$
/* Compute F when the current column is excluded */
30: $F_0 \leftarrow$ Update_$F(\kappa,\ell,\eta,\rho,\gamma_0)$
31: $\gamma_c \leftarrow \gamma$
32: $c \leftarrow \arg\max_c$ Update_$F(\kappa,\ell,\eta,\rho,\gamma_c(b)=c,\theta)$
33: $F_c \leftarrow$ Update_$F(\kappa,\ell,\eta,\rho,\gamma_c,\theta)$
/* Select the best column clustering */
34: **if** $F_c > F_0$ **then**
35: $\gamma \leftarrow \gamma_c$
36: **else**
37: $\gamma \leftarrow \gamma_0$
38: **end if**
39: **end for**
40: **until** convergence
41: **Step3** : Merging similar co_clusters
42: **Step4** : Refinement

The column mapping, γ, is calculated in a similar manner *(lines 28-39)*, but there is no consideration for negatively correlated columns. The algorithm stops when the difference between two consecutive values of the objective function is lower than a certain threshold value. After convergence, the result will be a non-overlapping set of co-clusters. Each of these co-clusters includes a subset of the rows/columns of the original dataset.

5.4.3 STEP 3: Merging the Co-clusters In this step, similar co-clusters are merged. Before merging, the co-clusters with the lowest scores are pruned. If there is no pre-determined threshold for pruning the co-clusters, the top η co-clusters will be retained, and the remaining co-clusters will be pruned. The similarity between any two co-clusters is defined using the $H^{'}$ function of the union of the rows and columns of the co-clusters, and the merging is performed following an agglomerative clustering approach. The two most similar co-clusters are merged in each iteration. The goal of this step is two-fold: (i) It allows the discovery of large co-clusters, and (ii) it allows overlapping co-clusters as shown in Figure 3(d).

5.4.4 STEP 4: Refinement To maximize the size of the obtained co-clusters after the merging step, the algorithm adds more rows and columns to each co-cluster individually. Therefore, the same row/column can be added to several co-clusters. Given a certain co-cluster with I rows and J columns, adding a row i or a column j is subject to the following constraints:

$$H(I \cup i, J, \theta) \geq H(I, J, \theta)$$

$$H(I, J \cup j, \theta) \geq H(I, J, \theta)$$

This will be added first for the positively correlated genes; then it will be added for the negatively correlated genes. This step helps in identifying larger co-clusters and also allows for more overlapping co-clusters as shown in Figure 3(e).

If there are missing values in the original data matrix, the H function can be modified as follows

(5.6)

$$H^{'}(I,J) = 1 - \frac{1}{|I||J|} \sum_{i \in I, j \in J} w_{ij} * ((x_{ij} - x_{Ij} - x_{iJ} + x_{IJ})^2)$$

Where w_{ij} is a binary weight matrix that contains 0 if the corresponding data point is missing in D, otherwise it contains 1.

6 Experimental Results

To demonstrate the performance of the proposed PONEOCC algorithm, several experiments were conducted using both synthetic and real-world gene expression datasets. We compared the results of the proposed algorithm against four other popular co-clustering algorithms, namely, CC, ISA, OPSM and ROCC, which were briefly described in Section 2. We used BiCAT software [3] to run CC, ISA and OPSM algorithms. The default parameter settings for each algorithm were used. The code for the ROCC was obtained from the authors of [8], and the parameters were set using a similar approach to the ones described in their paper.

6.1 Results on Synthetic Data Our algorithm was evaluated on a number of synthetic datasets. Two types of datasets were used, one of them without noise and the other one with 10% noise. The size of the randomly generated dataset is 200×150. Two co-clusters were implanted in each dataset, and the size of each co-cluster is 50×50.

The evaluation of the synthetic datasets was done using two metrics as defined in [20]. The first metric is the average co-cluster relevance which indicates the extent to which the generated co-clusters represent the implanted co-clusters in the gene dimension. The second metric is the average module recovery which determines how well each of the implanted co-clusters is discovered by the co-clustering algorithm.

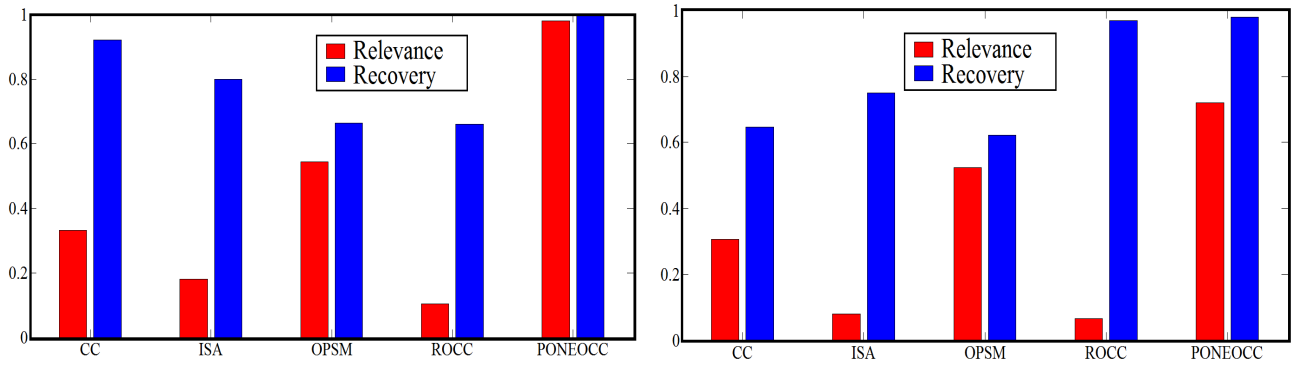
Given a set of implanted co-clusters denoted by M_{opt}, and a set of discovered co-clusters denoted by M, each co-cluster M_i contains G_i genes and C_i conditions, the gene match score is defined as follows:

$$S_G(M_1, M_2) = \frac{1}{M_1} \sum_{(G1,C1) \in M_1} \arg\max_{(G_2,C_2)} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}$$

the average co-cluster relevance is computed as $S_G(M, M_{opt})$, and the average module recovery is computed as $S_G(M_{opt}, M)$. Both metrics take values in the range of $[0, 1]$.

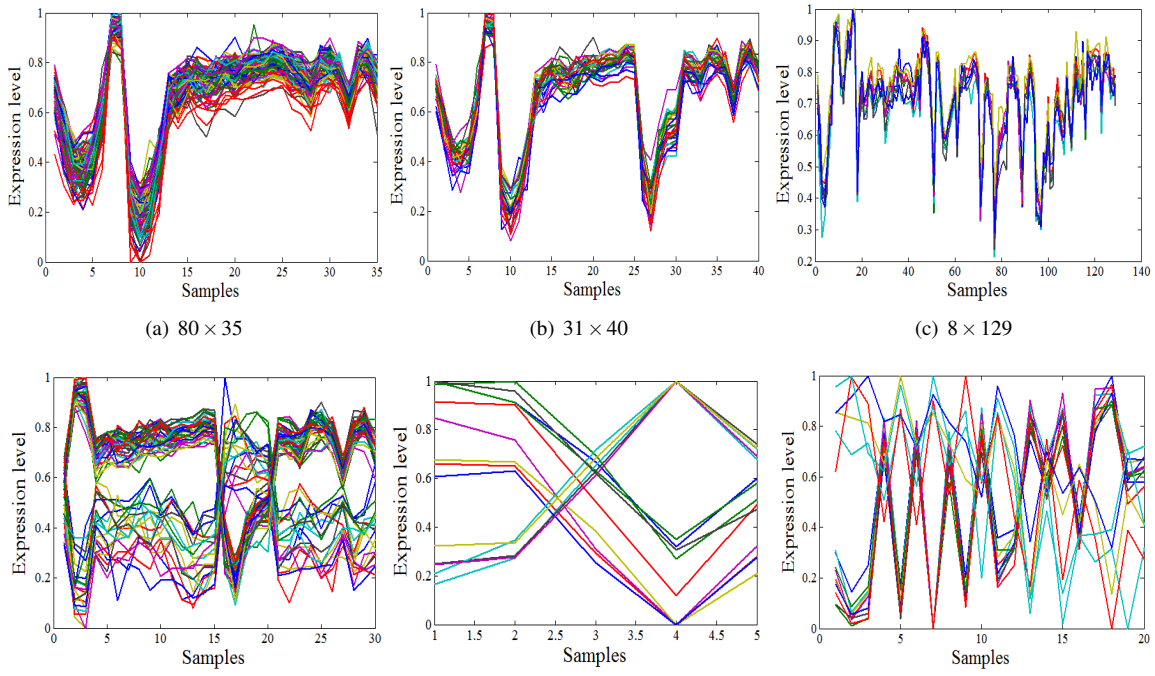
Figure 5 shows the results of different co-clustering algorithms when applied to the synthetic datasets. The proposed algorithm outperformed the other algorithms. PONEOCC produced the highest values for the average co-cluster relevance and the average module recovery. A major improvement is obtained in the average co-cluster relevance metric. This improvement results from the fact that the proposed algorithm prunes co-clusters with high errors during the search process, and it optimizes for high-quality co-clusters. As a result, fewer random data points are added to the co-clusters obtained by our algorithm. On the other hand, other algorithms, such as ROCC, prune the co-clusters with high error values after all the co-clusters are obtained.

6.2 Results on Real Gene Expression Data Table 2 shows a description of the seven gene expression datasets used in our experiments. For each dataset, we ran the five algorithms, and we measured the quality, the size and the biological significance of the obtained co-clusters. Figure 6 shows some samples of the obtained co-clusters. The co-



(a) without noise (b) with 10% noise

Figure 5: The results on the synthetic datasets. 10% noise was added to the data to test the performance of different co-clustering algorithms.



(a) 80×35 (b) 31×40 (c) 8×129

(d) 60×30 (e) 14×5 (f) 18×20

Figure 6: Examples of the co-clusters identified by the proposed algorithm. The three co-clusters in the first row contain only the positively correlated genes which show similar patterns. These co-clusters were obtained from the Gasch yeast dataset. The three co-clusters in the second row contain positively and negatively correlated genes which show opposite patterns. These co-clusters were obtained from (d) Gash yeast, (e) Scleroderma and (f) Causton yeast datasets.

clusters in the first row of Figure 6 contain positively correlated genes, while the co-clusters in the second row of the same figure contain both types of correlations. In these co-clusters, the negatively correlated genes are shown as the mirror image of the other genes.

Table 2: Description of the real-world datasets.

Dataset Name	No. of genes	No. of samples
Arabidopsis thaliana [1]	734	69
Gasch yeast [1]	2993	197
Cho yeast [2]	6240	17
Causton yeast [2]	4960	11
Lung cancer [3]	1000	197
Leukemia [3]	5000	38
Scleroderma [4]	2533	27

In evaluating the results of different co-clustering algorithms, we are interested in evaluating two metrics: the quality of the co-clusters, and the volume of the co-clusters. The quality of the co-clusters was measured using the H' function (see Equation 5.3). Therefore, the score of each co-cluster is between 0 and 1. A perfect co-cluster is assigned a score of 1. Table 3, shows the results of all of the five co-clustering algorithms on the seven datasets. No results were obtained using the ISA algorithm when running on the Cho, Causton and Leukemia datasets using the BiCat software; hence, the corresponding cells are given as $-$. As shown in this table, the proposed algorithm produced the highest co-clusters score for all the datasets. Regarding the average volume of the co-clusters, our algorithm outperformed the other algorithms in three datasets, and in the other datasets, our algorithm produced relatively large co-clusters. These results confirmed one of our initial claims that the proposed algorithm was designed to identify high-quality co-clusters.

When working with biological data, we are interested in identifying the biological significance of the results. The biological significance was estimated using the p-values with different significance levels = 5%, 1% and 0.1%. The hypergeometric distribution is used to calculate the probability of having at least k genes from a co-cluster of size n genes by chance in a biological process containing f genes from a total size of N genes as follows:

$$P = 1 - \sum_{i=0}^{k} \frac{\binom{f}{i}\binom{N-f}{n-i}}{\binom{N}{n}}$$

This test measures if a co-cluster is enriched with genes from a particular category to a greater extent than that would be expected by chance [15]. The range of the p-values is from 0 to 1. Lower p-values indicate biological significance [7]. As shown in Figure 7, PONEOCC algorithm outperformed the existing algorithms in the majority of the cases. A high quality co-cluster contains strongly correlated genes, which suggests that the genes are involved in the same biological pathway. Therefore, it is likely to have a significant p-value.

[1] *http://www.tik.ee.ethz.ch/sop/download/*

[2] *http://yfgdb.princeton.edu/download/yeast_datasets/*

[3] *http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi/*

[4] *http://genome-www.stanford.edu/scleroderma/data.shtml/*

7 Conclusion

In this paper, we presented a novel co-clustering framework (PONEOCC) to cluster large-scale gene expression data. It uses a novel objective function that is optimized to simultaneously find large co-clusters with minimum errors, and it allows positively and negatively correlated genes to be in the same co-cluster. The co-clusters can be arbitrarily positioned in the gene expression matrix and can overlap. Furthermore, the algorithm performs well on noisy data, and it can handle missing values. The experimental results on synthetic and real-word datasets showed that the proposed algorithm can extract biologically and statistically significant co-clusters from gene expression data. The proposed algorithm was compared to some of the existing algorithms, and the comparisons showed that PONEOCC outperformed the other methods that are available in the literature. We plan to extend our framework to handle non-linear subspace correlations [21].

References

[1] Aris Anagnostopoulos, Anirban Dasgupta, and Ravi Kumar. Approximation algorithms for co-clustering. In *PODS '08: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 201–210, New York, NY, USA, 2008.

[2] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *J. Mach. Learn. Res.*, 8:1919–1986, 2007.

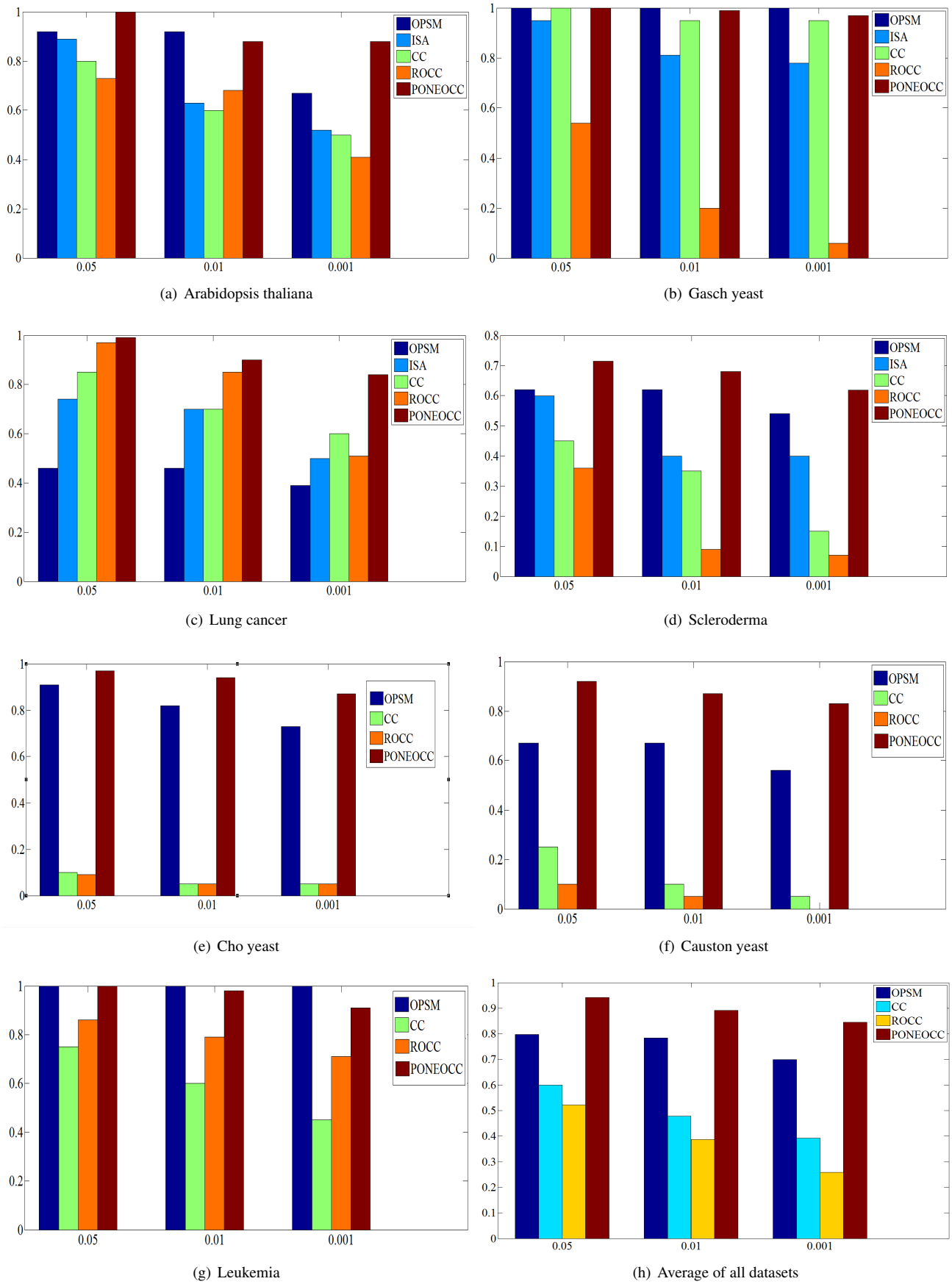
[3] Simon Barkow, Stefan Bleuler, Amela Prelic, Philip Zimmermann, and Eckart Zitzler. BicAT: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[4] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of computational biology*, 10(3-4):373–384, 2003.

[5] Stanislav Busygin, Oleg Prokopyev, and Panos M. Pardalos. Biclustering in data mining. *Comput. Oper. Res.*, 35(9):2964–2987, 2008.

[6] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[7] Hyuk Cho and Inderjit S. Dhillon. Coclustering of human cancer microarrays using minimum sum-squared residue co-



(a) Arabidopsis thaliana

(b) Gasch yeast

(c) Lung cancer

(d) Scleroderma

(e) Cho yeast

(f) Causton yeast

(g) Leukemia

(h) Average of all datasets

Figure 7: Proportion of co-clusters that are significantly enriched in each dataset (a) to (g), and the average of all the datasets is shown in (h). The biological significance is measured using the P-values: 5%, 1% and 0.1%, respectively.

Table 3: The average score of co-clusters and the average volume of the co-clusters obtained from CC, ISA, OPSM, ROCC and PONEOCC methods for each dataset. (average row number, average column number) are provided for each volume value. These two numbers were rounded. The maximum values for the score and the volume are highlighted.

Dataset	Average score of co-clusters					Average volume of co-clusters				
	CC	ISA	OPSM	ROCC	PONEOCC	CC	ISA	OPSM	ROCC	PONEOCC
Arabidopsis	0.9996	0.9569	0.9969	0.9952	**0.9998**	146.2 (19,8)	40.6 (20,2)	330.7 (98,8)	534.1 (41,28)	**2282.1** (191,12)
Gasch	0.9844	0.9907	0.9966	0.9945	**0.9987**	2424 (304,43)	572.1 (67,9)	2019.6 (522,9)	2320.7 (115,25)	**2582.5** (272,29)
Cho	0.9322	-	0.9923	0.9854	**0.996**	950.5 (80,12)	-	**2015** (682,7)	757.9 (152,6)	1958 (392,5)
Causton	0.922	-	0.9907	0.9831	**0.9965**	2202.9 (219,10)	-	2656.3 (941,6)	800 (200,4)	**3897.5** (780,5)
Lung	0.9857	0.9665	0.9951	0.9947	**0.999**	**7772.1** (106,50)	538.1 (47,11)	574.9 (169,9)	1212.3 (55,24)	1631.9 (171,17)
Leukemia	0.9715	-	0.9963	0.9775	**0.9984**	7611.2 (310,15)	-	**8475** (708,20)	2544 (190,10)	3543.7 (219,13)
Scleroderma	0.9838	0.9813	0.9862	0.9895	**0.995**	**2273.8** (110,16)	15 (8,2)	1303.4 (403,8)	426 (63,10)	1949 (380,7)

clustering. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(3):385–400, 2008.

[8] Meghana Deodhar, Gunjan Gupta, Joydeep Ghosh, Hyuk Cho, and Inderjit Dhillon. A scalable framework for discovering coherent co-clusters in noisy data. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 241–248, 2009.

[9] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 625–628, Washington, DC, USA, 2005.

[10] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, 97:12079–12084, 2000.

[11] Kuo-Wei Hsu, Arindam Banerjee, and Jaideep Srivastava. I/o scalable bregman co-clustering. In *PAKDD'08: Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 896–903, 2008.

[12] Jan Ihmels, Sven Bergmann, and Naama Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[13] Liping Ji and Kian-Lee Tan. Mining gene expression data for positive and negative co-regulated gene clusters. *Bioinformatics*, 20(16):2711–2718, 2004.

[14] Jinyan Li, Kelvin Sim, Guimei Liu, and Limsoon Wong. Maximal quasi-bicliques with balanced noise tolerance: Concepts and co-clustering applications. In *Proc. SIAM Int. Conf. on Data Mining SDM'08*, pages 72–83, April 2008.

[15] Jinze Liu, Jiong Yang, and Wei Wang. Biclustering in gene expression data by tendency. In *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference*, CSB '04, pages 182–193, Washington, DC, USA, 2004.

[16] Junwan Liu, Zhoujun Li, Xiaohua Hu, and Yiming Chen. Biclustering of microarray data with mospo based on crowding distance. *BMC Bioinformatics*, 10(Suppl 4):S9, 2009.

[17] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. A generative pattern model for mining binary datasets. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1109–1110, New York, NY, USA, 2010.

[18] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.

[19] Omar Odibat, Chandan K. Reddy, and Craig N. Giroux. Differential biclustering for gene expression analysis. In *Proceedings of the ACM Conference on Bioinformatics and Computational Biology (BCB)*, pages 275–284, 2010.

[20] Amela Prelic, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Buhlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006.

[21] Chandan K. Reddy and Mohammad S. Aziz. Modeling local nonlinear correlations using subspace principal curves. *Stat. Anal. Data Min.*, 3:332–349, October 2010.

[22] Hanhuai Shan and Arindam Banerjee. Residual bayesian co-clustering for matrix approximation. In *Proc. SIAM International Conference on Data Mining*, pages 223–234, 2010.

[23] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In Marko Grobelnik, Dunja Mladenic, and Natasa Milic-Frayling, editors, *KDD-2000 Workshop on Text Mining, August 20*, pages 109–111, 2000.

[24] Wen-Hui Yang, Dao-Qing Dai, and Hong Yan. Finding correlated biclusters from gene expression data. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2010.