# Comparative Analysis of Biclustering Algorithms

Doruk Bozdağ
Biomedical Informatics
The Ohio State University
bozdagd@bmi.osu.edu

Ashwin S. Kumar
Computer Science and Engineering
Biomedical Informatics
The Ohio State University
ashwin@bmi.osu.edu

Umit V. Catalyurek
Biomedical Informatics
Electrical and Computer Engineering
The Ohio State University
umit@bmi.osu.edu

## ABSTRACT

Biclustering is a very popular method to identify hidden co-regulation patterns among genes. There are numerous biclustering algorithms designed to undertake this challenging task, however, a thorough comparison between these algorithms is even harder to accomplish due to lack of a ground truth and large variety in the search strategies and objectives of the algorithms. In this paper, we address this less studied, yet important problem and formally analyze several biclustering algorithms in terms of the bicluster patterns they attempt to discover. We systematically formulate the requirements for well-known patterns and show the constraints imposed by biclustering algorithms that determine their capacity to identify such patterns. We also give experimental results from a carefully designed testbed to evaluate the power of the employed search strategies. Furthermore, on a set of real datasets, we report the biological relevance of clusters identified by each algorithm.

## Categories and Subject Descriptors

J.3 [**Biology and genetics**]

## Keywords

Biclustering, microarray, gene expression

## 1. INTRODUCTION

Analyzing variations in expression levels of genes under different conditions (samples) is important to understand the underlying complex biological processes that the genes participate in. In gene expression data analysis, expression levels of genes in each sample are represented by a real-valued data matrix with rows and columns representing the genes and the samples, respectively. The goal is to identify genes that have correlated expression values in various samples.

Clustering is the most widely used technique to discover interesting patterns in gene expression data. Genes assigned to the same cluster are likely to have related biological functions, hence they are good candidates for further wet laboratory analysis. Most of the traditional clustering algorithms operate in full space, i.e., take all columns (samples) into account while assigning a row (gene) into a cluster [8, 18]. In gene expression data analysis, however, due to the diversity of sample sources, functionally related genes may not exhibit a similar pattern in all samples but in a subset of them. To address this observation, biclustering (subspace clustering) approaches have emerged, where only a subset of columns are considered to determine the assignment of rows into clusters [4]. Since such a subset of columns are also initially unknown, biclustering can be viewed as simultaneous clustering of rows and columns.

Following the work of Cheng and Church [4], biclustering has quickly become popular in analyzing gene expression data and numerous biclustering algorithms have been proposed [7, 9, 10, 11, 12, 14, 17, 19, 20, 21]. Since each algorithm focuses on identification of different bicluster patterns, it is a very challenging task to thoroughly evaluate these algorithms. Comparative analysis of biclustering techniques has only been considered in a few studies in the literature [13, 16]. In this work, we attempt to address this problem and systematically compare several biclustering algorithms to answer the following three crucial questions regarding these algorithms:

- What kind of bicluster patterns can it discover?
- How powerful is the search technique?
- Are the resulting clusters biologically relevant?

We classify biclustering patterns into two, based on whether the pattern is defined on a single cluster or multiple clusters. If a bicluster pattern is defined on a single bicluster, we call the pattern a *local pattern*. Otherwise, we call the pattern a *global pattern*. Example of a local pattern is a scaling pattern, where every row in a bicluster is a multiple of another row (when considered across the columns in the bicluster). In a local pattern, no information is required about the elements outside the bicluster. This is in contrast to global patterns, where the membership of a row (column) to a bicluster depends on the elements of the row (column) external to the bicluster, and/or on the membership of the row (column) to other biclusters. Such external dependencies bring an additional layer of complexity in analysis of

| $\alpha_i$ | $\pi_j$ | $\beta_i$ | Simplified $\hat{a}_{ij}$ ($i \in I$, $j \in J$) | Bicluster pattern |
|---|---|---|---|---|
| 0 | any | constant | a constant | Constant bicluster |
| any | 0 | constant | | |
| constant | constant | 0 | | |
| constant | constant | constant | | |
| 0 | any | varying | $\beta_i$ | Constant rows |
| any | 0 | varying | | |
| varying | constant | any | | |
| any | constant | varying | | |
| constant | varying | 0 | $\pi_j$ | Constant columns |
| constant | varying | constant | | |
| constant | varying | varying | $\pi_j + \beta_i$ | Shifting |
| varying | varying | 0 | $\alpha_i \times \pi_j$ | Scaling |
| varying | varying | varying | $\alpha_i \times \pi_j + \beta_i$ | Shift-scale |

**Table 1: Well known bicluster patterns, corresponding expected value expressions, and constraints on $\alpha_i$, $\pi_j$ and $\beta_i$ factors to obtain each of these patterns. *Constant* refers to non-zero constant values, *varying* refers to non-constant values, and *any* refers to any value without any constraints.**

global patterns. Therefore, as a first step to understand bicluster patterns, in this work, we focus our attention to local patterns. In particular, we consider identification of local shifting (additive) and scaling (multiplicative) patterns which are extremely useful to characterize co-regulation relationships among genes. We analyze quality criteria and search strategies of several algorithms, and discuss the types of bicluster patterns each algorithm can identify. For this purpose we identified and analyzed four algorithms that can discover local patterns: Cheng and Church [4], OPSM [2], HARP [22] and CPB [3]. In order to decouple the objective and the search technique of an algorithm as much as possible, we designed an extensive testbed consisting of synthetic datasets that correspond to various input scenarios. Furthermore, we included two additional well-known algorithms that seek global patterns (SAMBA [19] and MSSRCC [5]) in our experimental studies, and we applied the algorithms on real datasets to provide insight about the biological relevance of resulting clusters.

The paper is organized as follows. In the following section, we describe the shifting and scaling patterns and bicluster quality metrics. Analytical discussion about the considered algorithms are given in Section 3. We report results from our experiments in Section 4, and conclude in Section 5.

## 2. BICLUSTER PATTERNS AND QUALITY METRICS

In this section, we introduce the notation and show how various local patterns can be modeled as a variation of shifting and scaling patterns.

A gene expression data matrix $A = (R, C)$ can be represented by a set of rows $R$ and a set of columns $C$ denoting genes and samples, respectively. Each entry in this matrix is denoted by $(r, c)$ and the expression level of gene $r$ in sample $c$ is represented by $a_{rc}$, $r \in R$, $c \in C$. A bicluster $B = (I, J)$ is composed of a subset of rows $I \subset R$ and a subset of columns in $J \subset C$, where all $a_{ij}$, for $i \in I$ and $j \in J$, are expected to fit to a predetermined target pattern, possibly with small deviations. Therefore, each $a_{ij}$ value in a bicluster can be decomposed into two components:

$$a_{ij} = \hat{a}_{ij} + \epsilon_{ij} \tag{1}$$

where $\hat{a}_{ij}$ is the expected value of $a_{ij}$ that would maximize its match to the target pattern and $\epsilon_{ij}$ is the deviation from the expected value. $\epsilon_{ij}$ is also referred to as a residue and it is commonly used to quantify the quality of a bicluster. The most commonly used bicluster quality metric is the Mean Squared Residue (MSR) [4], which is defined as

$$MSR = \frac{1}{|I||J|} \sum_{j \in J, i \in I} \epsilon_{ij}^2 \tag{2}$$

A bicluster that matches perfectly to a target pattern has $\epsilon_{ij} = 0$, hence its MSR score is also 0.

Types of patterns searched by most biclustering algorithms can be modeled using an appropriate expression for $\hat{a}_{ij}$. For example, shifting patterns and scaling patterns can be represented as $\hat{a}_{ij} = \pi_j + \beta_i$ and $\hat{a}_{ij} = \alpha_i \times \pi_j$, respectively. Here, $\alpha_i$ and $\beta_i$ represent scaling and shifting factors associated with row $i$, and $\pi_j$ represents a base value associated with column $j$. A more general expression that encompasses both shifting and scaling patterns can be stated as $\hat{a}_{ij} = \alpha_i \times \pi_j + \beta_i$ [1, 15]. We refer to this pattern as a *shift-scale* pattern, since it represents a simultaneously shifting and scaling pattern. The shift-scale pattern expression can be reduced to simpler pattern expressions by applying some constraints on $\alpha_i$, $\pi_j$ and $\beta_i$ factors. For instance, if $\alpha_i$ is constrained to be equal to a constant $\alpha$, the shift-scale pattern expression reduces to $\hat{a}_{ij} = \alpha \times \pi_j + \beta_i = \pi'_j + \beta_i$, which is a shifting pattern expression. A set of well known bicluster patterns and corresponding expected value expressions are given in Table 1. Furthermore, constraints on $\alpha_i$, $\pi_j$ and $\beta_i$ factors to obtain each of these patterns are also shown in this table.

## 3. ANALYSIS OF BICLUSTERING ALGORITHMS

In this section, we formally analyze four biclustering algorithms that seek local patterns in terms of the types of patterns they can discover and discuss their search strategies.

| $\alpha_i$ | $\pi_j$ | $\beta_i$ | Bicluster pattern |
|---|---|---|---|
| 0 | any | any | constant bicluster, constant rows |
| constant | any | any | constant bicluster, constant columns, shifting |
| any | 0 | any | constant bicluster, constant rows |
| any | constant | any | constant bicluster, constant rows |

**Table 2: All possible non-trivial cases to satisfy condition in (8) for Cheng and Church algorithm.**

## 3.1 Cheng and Church Algorithm

The first and the most popular expression for residue $\epsilon_{ij}$ is defined by Cheng and Church [4]:

$$\epsilon_{ij} = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \tag{3}$$

where $a_{iJ}$, $a_{Ij}$ and $a_{IJ}$ are the mean of the entries in of row $i$, column $j$, and the entire bicluster, respectively, for $i \in I$ and $j \in J$. In their biclustering algorithm [4], Cheng and Church considered minimization of MSR as their objective.

Aguilar-Ruiz [1] showed that defining $\epsilon_{ij}$ as in (3) allows capturing shifting pattens, but not shift-scale patterns. In their analysis they considered a bicluster that has a perfect shift-scale pattern, i.e., they set $a_{ij} = \alpha_i \times \pi_j + \beta_i$. Let $\overline{\alpha}$ and $\overline{\beta}$ denote the mean values of $\alpha_i$ and $\beta_i$ over the rows in $I$, and $\overline{\pi}$ denote the mean value of $\pi_j$ over the columns in $J$. Then,

$$\begin{aligned} a_{iJ} &= \frac{1}{|J|}\sum_{j\in J} a_{ij} \\ &= \frac{1}{|J|}\sum_{j\in J}(\alpha_i \times \pi_j + \beta_i) \\ &= \alpha_i \times \overline{\pi} + \beta_i \end{aligned} \tag{4}$$

$$\begin{aligned} a_{Ij} &= \frac{1}{|I|}\sum_{i\in I} a_{ij} \\ &= \frac{1}{|I|}\sum_{i\in I}(\alpha_i \times \pi_j + \beta_i) \\ &= \overline{\alpha} \times \pi_j + \overline{\beta} \end{aligned} \tag{5}$$

$$\begin{aligned} a_{IJ} &= \frac{1}{|I||J|}\sum_{i\in I, j\in J} a_{ij} \\ &= \frac{1}{|I||J|}\sum_{i\in I, j\in J}(\alpha_i \times \pi_j + \beta_i) \\ &= \overline{\alpha} \times \overline{\pi} + \overline{\beta} \end{aligned} \tag{6}$$

Plugging these values in (3) gives

$$\begin{aligned} \epsilon_{ij} &= (\alpha_i \times \pi_j + \beta_i) - (\alpha_i \times \overline{\pi} + \beta_i) \\ &\quad -(\overline{\alpha} \times \pi_j + \overline{\beta}) + (\overline{\alpha} \times \overline{\pi} + \overline{\beta}) \\ &= (\alpha_i - \overline{\alpha})(\pi_j - \overline{\pi}) \end{aligned} \tag{7}$$

In order to have a zero MSR value, each residue $\epsilon_{ij}$ should also be zero. This condition can be expressed as

$$(\alpha_i - \overline{\alpha})(\pi_j - \overline{\pi}) = 0 \tag{8}$$

for all $i \in I$ and $j \in J$ [1]. Possible ways of non-trivially satisfying this condition and corresponding bicluster patterns from Table 1 are shown in Table 2. Biclusters with scaling or shift-scale patterns cannot have a zero MSR value when Cheng and Church (CC) algorithm is used, as can be seen from this table. Therefore, those patterns may not be identified with CC and other algorithms that use the residue definition in (3).

CC is an iterative algorithm that starts with $I = R$ and $J = C$. At each iteration, single or a group of rows/columns are removed from the bicluster to improve the MSR value. This is followed by a row/column addition phase where rows or columns are iteratively added to the cluster until a user-specified threshold MSR value is reached. To decide which row or column to move, a score similar to MSR is defined for each row and each column. The score for row $i$ and column $j$ are respectively the sum of the squared residues across the columns in $J$ and rows in $I$. CC algorithm discovers biclusters one at a time and each discovered bicluster is masked by random numbers in the original matrix before the search for the next one begins. Returned biclusters depend on the value of the threshold MSR which in turn depends on the range of data values. Therefore, choosing an appropriate MSR threshold is not trivial and may have a significant impact on the resulting biclusters.

## 3.2 HARP Algorithm

In [22], Yip *et al.* introduced a quality metric slightly different from MSR. In the HARP algorithm they proposed, quality of a bicluster is measured as the sum of the relevance indices of the columns. Relevance index $R_{Ij}$ for column $j \in J$ is defined as

$$R_{Ij} = 1 - \frac{\sigma^2_{Ij}}{\sigma^2_{.j}} \tag{9}$$

where $\sigma^2_{Ij}$ (local variance) and $\sigma^2_{.j}$ (global variance) are the variance of the values $a_{ij}$ in column $j$ for $i \in I$ and $i \in R$, respectively. Note that the relevance index for a column is maximized if its local variance is zero, provided that the global variance is not. Local variance is computed as

$$\sigma_{Ij} = \frac{1}{|I|}\sum_{i\in I}(a_{ij} - a_{Ij})^2 \tag{10}$$

where $a_{Ij}$ denotes the mean of the values in column $j$ over the rows in the bicluster. In order to understand the types of biclusters that can be identified using relevance index, again consider a bicluster with potentially shifting and scaling patterns, thus set $a_{ij} = \alpha_i \times \pi_j + \beta_i$. Then, $a_{Ij}$ can be

| $\alpha_i$ | $\pi_j$ | $\beta_i$ | Bicluster pattern |
|---|---|---|---|
| 0 | any | 0 | none |
| 0 | any | constant | constant bicluster |
| constant | any | 0 | constant bicluster, constant columns |
| constant | any | constant | constant bicluster, constant columns |
| any | 0 | 0 | none |
| any | 0 | constant | constant bicluster |

**Table 3: All possible non-trivial cases to satisfy condition in (12) for the HARP algorithm.**

simplified as in (5), and the local variance becomes

$$\begin{aligned} \sigma_{Ij}^2 &= \frac{1}{|I|}\sum_{i\in I}\big((\alpha_i\times\pi_j+\beta_i)-(\overline{\alpha}\times\pi_j+\overline{\beta})\big)^2 \\ &= \frac{1}{|I|}\sum_{i\in I}\big((\alpha_i-\overline{\alpha})\times\pi_j-(\beta_i-\overline{\beta})\big)^2 \end{aligned} \quad (11)$$

Therefore, local variance of column $j \in J$ becomes zero only if

$$(\alpha_i-\overline{\alpha})\times\pi_j-(\beta_i-\overline{\beta})=0 \quad (12)$$

for all $i \in I$. Note that the bicluster quality is maximized only if the local variance of all columns in $J$ are zero. Therefore, the constraint in (12) should be satisfied for all $j \in J$ to maximize quality. In Table 3, all possible ways of non-trivially satisfying this condition and corresponding bicluster patterns are given.

As shown in Table 3, the only types of bicluster patterns that maximize the quality in HARP algorithm are constant bicluster and constant column patterns. Thus, one can conclude that the HARP algorithm is not suitable to discover constant row, shifting, scaling or shift-scale patterns.

## 3.3 Correlated Pattern Biclusters Algorithm (CPB)

The goal in CPB [3] algorithm is to identify biclusters, where the Pearson's Correlation Coefficient (PCC) between every pair of rows $i \in I$ are above a threshold with respect to the columns $j \in J$. PCC between two rows $i, \ell \in I$ with respect to the columns $j \in J$ is defined as:

$$PCC=\frac{\sum_{j\in J}(a_{ij}-a_{iJ})(a_{\ell j}-a_{\ell J})}{\sqrt{\sum_{j\in J}(a_{ij}-a_{iJ})^2\sum_{j\in J}(a_{\ell j}-a_{\ell J})^2}} \quad (13)$$

where $a_{iJ}$ and $a_{\ell J}$ respectively denote the mean of the entries in rows $i$ and $\ell$ over the columns in the bicluster. To investigate the relationship between PCC and shifting and scaling patterns, let $a_{ij}=\alpha_i\times\pi_j+\beta_i$ as before. Then, $a_{iJ}$ is simplified as given in (4). Thus,

$$\begin{aligned} a_{ij}-a_{iJ} &= (\alpha_i\times\pi_j+\beta_i)-(\alpha_i\times\overline{\pi}+\beta_i) \\ &= \alpha_i(\pi_j-\overline{\pi}) \end{aligned} \quad (14)$$

Similarly,

$$a_{\ell j}-a_{\ell J}=\alpha_\ell(\pi_j-\overline{\pi}) \quad (15)$$

Plugging (14) and (15) into (13) gives

$$\begin{aligned} PCC &= \frac{\sum_{j\in J}\alpha_i(\pi_j-\overline{\pi})\alpha_\ell(\pi_j-\overline{\pi})}{\sqrt{\sum_{j\in J}(\alpha_i(\pi_j-\overline{\pi}))^2\sum_{j\in J}(\alpha_\ell(\pi_j-\overline{\pi}))^2}} \\ &= \frac{\alpha_i\alpha_\ell\sum_{j\in J}(\pi_j-\overline{\pi})^2}{\sqrt{\alpha_i^2\alpha_\ell^2\big(\sum_{j\in J}(\pi_j-\overline{\pi})^2\big)^2}} \\ &= 1 \end{aligned} \quad (16)$$

provided that the denominator is not zero. This result indicates that if a bicluster has a shift-scale pattern, there is perfect correlation between any pair of rows in $I$ with respect to the columns in $J$. In Table 4, all possible cases that ensure a non-zero denominator value in the PCC expression is given. The table shows that besides shift-scale patterns, PCC can also be used to discover shifting, scaling and constant column patterns. However it may not be effective identifying constant bicluster or constant row patterns.

CPB algorithm starts by randomly initializing $I$ and $J$ sets and iteratively improves the bicluster by moving rows and columns in and out of the cluster. To represent the general tendency of rows in the columns of the bicluster, a tendency vector is $T$ is utilized. A row is included in the cluster only if its PCC with this tendency vector is above a certain threshold. Each row is associated with a scaling and a shifting factor, $\alpha_i$ and $\beta_i$, and each column is associated with a tendency value $t_j$, which is the $j^{th}$ element of the tendency vector $T$. For each $a_{ij}$ value in the bicluster, a corresponding normalized $a'_{ij}$ value is computed by first subtracting $\beta_i$ from $a_{ij}$, and then dividing by $\alpha_i$. $t_j$ is then updated by setting it equal to the mean of $a'_{ij}$ values across the rows in $I$. Then, least squares fitting is applied to the values $a'_{ij}$ in each row $i$ to find parameters that maximize their fit to the updated $t_j$ values. The slope and intercept of the fit are assigned to $\alpha_i$ and $\beta_i$ respectively, and the entire process is repeated until all $\alpha_i$, $\beta_i$ and $t_j$ values converge. Then, the bicluster is updated to ensure that only rows having large PCC with the new $T$ vector, and columns having entries with small $a'_{ij}-t_j$ difference over all rows are included. In general, since the search strategy of the CPB algorithm accounts for both shifting and scaling factors, it is well-suited for identifying shift-scale patterns.

## 3.4 Order Preserving Submatrix (OPSM)

An order preserving submatrix is defined as a submatrix (bicluster) where there exists a permutation of columns in $J$ such that the sequence of values in every row $i \in I$ is strictly increasing [2]. This condition be stated as

$$a_{ij}<a_{ik} \textbf{ iff } a_{\ell j}<a_{\ell k} \quad (17)$$

| $\alpha_i$ | $\pi_j$ | $\beta_i$ | Bicluster pattern |
|---|---|---|---|
| constant | varying | any | constant columns, shifting |
| varying | varying | any | scaling, shift-scale |

**Table 4: All possible non-trivial cases that i) ensure the denominator is non-zero in the PCC expression in (16) for CPB algorithm, ii) satisfy condition in (18) for OPSM algorithm.**

for all $i, \ell \in I$ and for all $j, k \in J$, such that $i \neq \ell$ and $j \neq k$. In other words,

$$a_{ij} - a_{ik} < 0 \textbf{ iff } a_{\ell j} - a_{\ell k} < 0 \tag{18}$$

Assuming that the bicluster has a shift-scale pattern, condition in (18) becomes

$$\begin{aligned}&(\alpha_i \times \pi_j + \beta_i) - (\alpha_i \times \pi_k + \beta_i) < 0\\ \textbf{iff } &(\alpha_\ell \times \pi_j + \beta_\ell) - (\alpha_\ell \times \pi_k + \beta_\ell) < 0\end{aligned}$$

which can be rearranged as

$$\alpha_i(\pi_j - \pi_k) < 0 \textbf{ iff } \alpha_\ell(\pi_j - \pi_k) < 0 \tag{19}$$

In order to satisfy the condition in (18), $\alpha_i$ and $\alpha_\ell$ should be non-zero, and, $\pi_j$ and $\pi_k$ should be varying. The possible scenarios satisfying this condition are the same as those given in Table 4. In other words, OPSM may potentially identify the same type of biclusters as CPB.

The expected value expression $\hat{a}_{ij}$ that OPSM algorithm seeks can be stated as

$$\hat{a}_{ij} = \alpha_i \times \pi^i_j + \beta_i \tag{20}$$

where $\pi^i$ is a vector of length $|J|$ associated with row $i$. Although this expression is less restrictive compared to the shift-scale pattern expression (since $\pi^i_j$ do not have to be the same for rows $i \in I$), the condition in (17) still needs to be satisfied. Let $\overline{\pi^i}$ denote the mean of $\pi^i_j$ values across rows $j \in J$. Setting $a_{ij}$ to the $\hat{a}_{ij}$ expression in (20) and plugging it in the PCC equation in (13) gives

$$\begin{aligned}PCC &= \frac{\sum_{j\in J} \alpha_i(\pi^i_j - \overline{\pi^i})\alpha_\ell(\pi^\ell_j - \overline{\pi^\ell})}{\sqrt{\sum_{j\in J}(\alpha_i(\pi^i_j - \overline{\pi^i}))^2 \sum_{j\in J}(\alpha_\ell(\pi^\ell_j - \overline{\pi^\ell}))^2}}\\ &= \frac{\sum_{j\in J}(\pi^i_j - \overline{\pi^i})(\pi^\ell_j - \overline{\pi^\ell})}{\sqrt{\sum_{j\in J}(\pi^i_j - \overline{\pi^i})^2 \sum_{j\in J}(\pi^\ell_j - \overline{\pi^\ell})^2}}\end{aligned} \tag{21}$$

provided that $\alpha_i$ and $\alpha_\ell$ are not zero. One can observe that the expression in (21) is equal to the PCC between vectors $\pi^i$ and $\pi^\ell$. Therefore, the distribution of PCC values between rows in an order preserving submatrix is the same as the distribution of PCC values between *any* two vectors having the same column ordering.

Using this observation, we designed a small experiment to see the practical values of PCC in an order preserving submatrix. We generated 20 row vectors of length $t$ and sorted the values in each vector to ensure that they have the same column ordering. The values for the vectors were chosen from a uniform distribution between 0 and 1. The *smallest* PCC value between any pair of these vectors was found to be 0.83, 0.94 and 0.96 for $t = 20$, $t = 40$ and $t = 60$, respectively. These results show that even when random values are used, requiring to have the same column ordering results in extremely high PCC values between the rows of a bicluster. This implies that the type of the bicluster pattern sought by the OPSM algorithm resembles a shift-scale pattern especially as the number columns gets larger.

# 4. EXPERIMENTAL RESULTS

In this section we report results from our extensive experiments on synthetically generated as well as real datasets to demonstrate the performances of the algorithms discussed.

## 4.1 Synthetic datasets

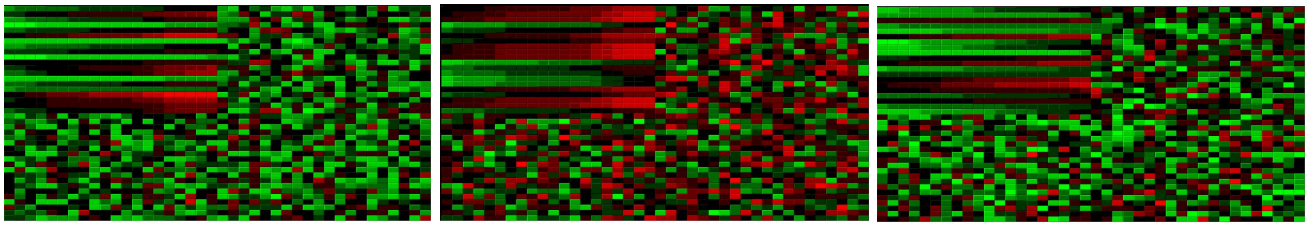
We generated synthetic datasets by implanting biclusters in fixed-size matrices; one in each. All matrices comprised 1000 rows and 120 columns. There were 3 types of implanted biclusters, namely shift-scale biclusters, shift biclusters and order-preserving biclusters. The datasets containing the respective biclusters were named correspondingly. The shift-scale biclusters were generated according to the shift-scale equation discussed earlier ($\hat{a}_{ij} = \alpha_i \times \pi_j + \beta_i$). The $\alpha_i$, $\beta_i$ and $\pi_j$ values were randomly chosen from a uniform distribution between 0 and 1. The matrices were generated in exactly the same manner, thereby ensuring that the distribution of values remained the same throughout the matrix. After generation, the elements of the matrix were shuffled, followed by implanting of the bicluster in the matrix, as it was. The matrix with implanted bicluster was then subject to random permutation of its rows and columns. The shift datasets and order-preserving datasets were constructed in exactly the same manner, except that the value generation was done according to their respective patterns discussed earlier. In Figure 1, examples of 20x20 biclusters embedded into 40x40 synthetic datasets, before random row and column permutation, are displayed for each type of dataset considered.

In general, when it comes to evaluating the quality of a biclustering result, there are two main factors which are to be taken into consideration, namely, the portion of the implanted bicluster the algorithm was is able to return and the portion external or irrelevant to the implanted bicluster which the algorithm returns. We define two metrics to evaluate cluster quality, namely U, which is the uncovered portion of the implanted bicluster and E, which is the portion of the output cluster external to the implanted cluster.

## 4.2 Algorithms used

We have used publicly available implementation of the CC algorithm[1], and for the OPSM algorithm we used the implementation in BiCAT (Biclustering Analysis Toolbox)[2]. Un-

[1] `http://cheng.ececs.uc.edu/biclustering`
[2] `http://www.tik.ethz.ch/~sop/bicat/`



(a) shift-scale (b) shift (c) order preserving

**Figure 1: Examples of synthetic datasets.**

| Algorithm | Target bicluster patterns | Metric/Constraint |
|---|---|---|
| CC [4] | Local: constant bicluster, constant columns, constant rows, shifting | MSR |
| HARP [22] | Local: constant bicluster, constant columns | Relevance index |
| CPB [3] | Local: constant columns, shifting, scaling, shift-scale | PCC |
| OPSM [2] | Local: constant columns, shifting, scaling, shift-scale | Ordered columns |
| SAMBA [19] | Global: biclusters with large variance | Heaviest bi-cliques |
| MSSRCC [5] | Global: biclusters with small combined mean-squared residue | MSR |

**Table 5: Properties of the biclustering algorithms.**

fortunately, there is no publicly available implementation of the HARP algorithm. Since in Section 3.2 we analytically showed that the HARP algorithm is not suitable to discover constant row, shifting, scaling or shift-scale patterns, we decided to exclude HARP from experimental evaluation. We have implemented the CPB algorithm.

To further enhance our experimental setup, we included two well-known biclustering algorithms, namely the Minimum Sum-Squared Residue-based CoClustering algorithm (MSSRCC) [5][3] and the Statistical Algorithmic Method for Bicluster Analysis (SAMBA) [19][4], These two algorithms seek global patterns, hence they are significantly different from the other algorithms previously discussed in terms of their search techniques and evaluation criteria.

The SAMBA algorithm [19] transforms the matrix into a bipartite graph where the genes and conditions collectively form the set of vertices. There is an edge between a gene and a condition only when the gene deviates sharply from its normal value, under that condition. It essentially finds the $K$ heaviest bi-cliques in the graph. It can find only those biclusters where the variance of elements is large and clustering stops as soon as all columns are covered. This algorithm is deterministic.

The MSSRCC algorithm [5] performs simultaneous clustering of the data matrix, by making use of the mean-squared residue as the objective function to be minimized. In a k-means fashion, it first clusters columns in such a way that that the overall residue is minimized, then performs a similar clustering on the rows. This algorithm forces every row/column into a cluster. Hence, it cannot detect overlapping clusters and it necessarily returns the number of clusters as decided by the input cluster split. This algorithm is not deterministic, i.e., results are likely to change on each run.

The properties of the biclustering algorithms considered in this paper are summarized in Table 5.

For CPB, CC and MSSRCC, the number of runs on each dataset was set to 100. Since OPSM and SAMBA algorithms are deterministic we ran them once. For each of the experiments on synthetic datasets, unless mentioned otherwise, the values for the various parameters of algorithms were as follows. For the CPB algorithm, the value of threshold PCC was set to 0.9. The value of the threshold MSR in CC was set to 0.01, which was found to be the best value after various trials. In SAMBA, the default values were used, i.e., the degree limit was set to 50, number of bi-cliques, $K$, to be returned to 20 and the overlap ratio to 30. In case of OPSM, the number of best partial models to be selected was set to 100 throughout. For MSSRCC, the cluster split was given according to the size of the implanted bicluster, so that the expected sizes of the output clusters were the same as the implanted biclusters.

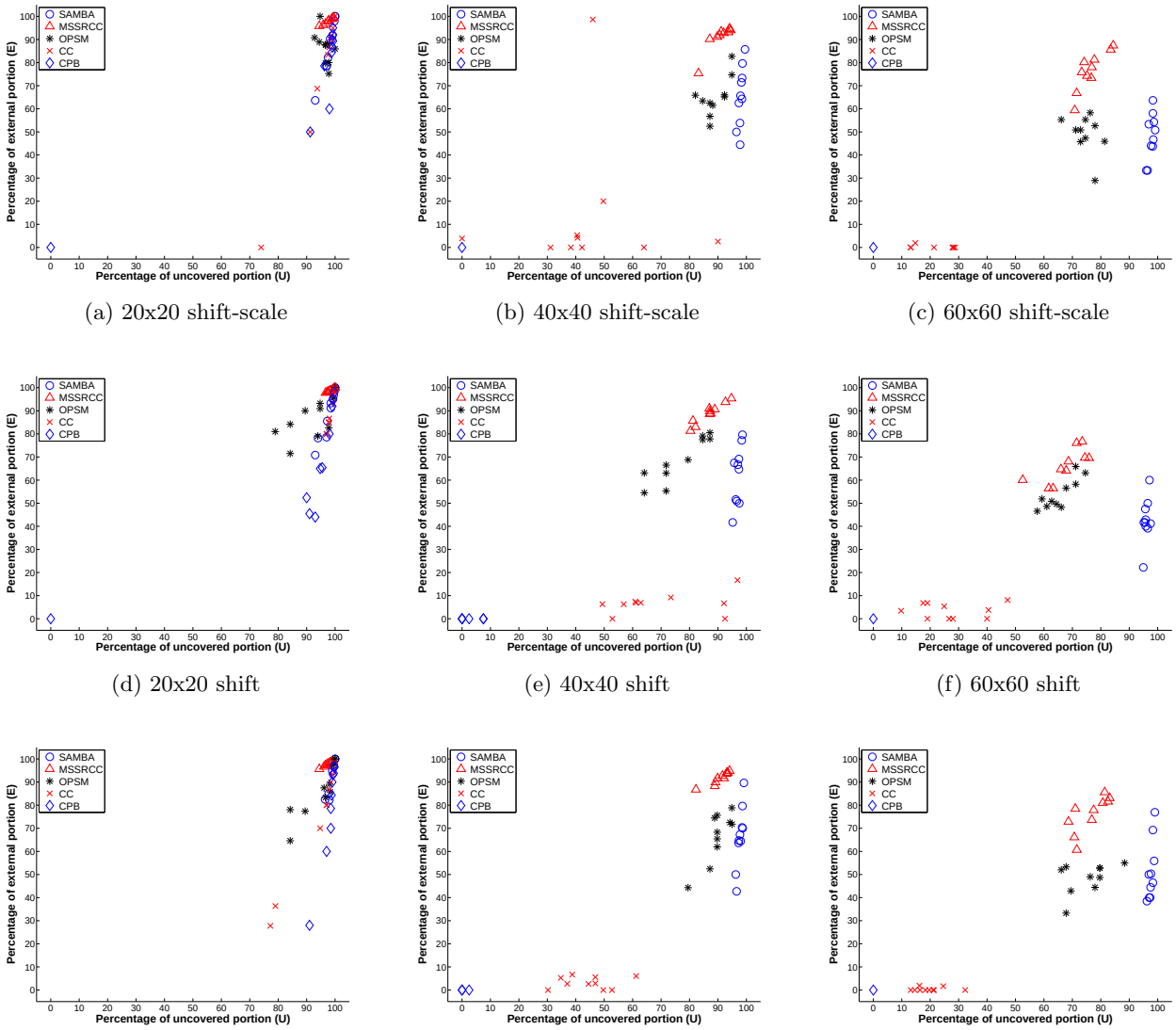
The implementations that ran fastest were CPB, MSSRCC and SAMBA, followed by CC, which was also reasonably fast. OPSM was rather time-consuming.

#### 4.2.1 *Effect of implanted bicluster type and size*

For the first set of experiments, we implanted 3 different types of biclusters, namely shift-scale, shift and order-preserving biclusters, with 3 different bicluster sizes: small (20x20), medium (40x40) and large (60x60). Figure 2 displays the results of these 9 experiments. The results indicate improved performances of all algorithms as we go from 20x20 datasets to 60x60 datasets. Thus, the ease of finding an implanted bicluster increases with bicluster size. A good performance is expected from CPB and OPSM, since both shift and shift-scale datasets have a perfect PCC and the

[3] `http://www.cs.utexas.edu/users/dml/Software/cocluster.html`

[4] Implementation is available in EXPANDER (EXpression Analyzer and DisplayER) at `http://acgt.cs.tau.ac.il/expander/`.

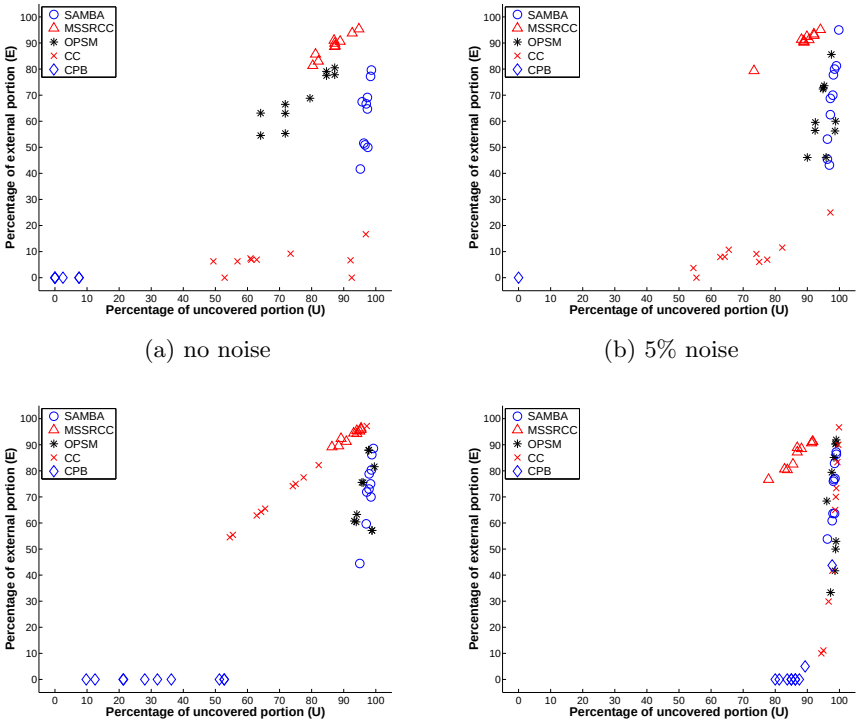



(a) 20x20 shift-scale (b) 40x40 shift-scale (c) 60x60 shift-scale

(d) 20x20 shift (e) 40x40 shift (f) 60x60 shift

(g) 20x20 order preserving (h) 40x40 order preserving (i) 60x60 order preserving

**Figure 2: Scatter plot of U and E values of the algorithms on the base synthetic datasets with implanted biclusters.**




(a) no noise (b) 5% noise

(c) 10% noise (d) 20% noise

**Figure 3: Results on datasets with varying noise levels. 40x40 shift datasets were used.**

order-preserving dataset has a very large PCC between the rows of the implanted bicluster as discussed in Section 3.4. The CPB Algorithm provided the best performance overall, finding a perfect match in most cases. A perfect match is a cluster whose U and E values are both zero. On the 40x40 and 60x60 datasets, it hardly missed any rows and columns and returned absolutely no external rows or columns (E=0). CPB also found one perfect match on the 20x20 datasets. CC performed as the second best algorithm overall, and did not return too many external rows and columns, i.e., the E value was consistently low throughout. OPSM performed better than MSSRCC, especially while finding medium and large biclusters in order preserving and shift-scale datasets. The MSSRCC algorithm works well only in exhaustive cases, i.e., when the entire matrix is made up of implanted biclusters, which is not the case in our datasets. We verified this, by running MSSRCC on a special 120x120 dataset comprising 9 implanted clusters. The results indicated that up to 30% of the clusters found had U and E values between 0 and 5 while up to 60% of the clusters had U and E values between 80 and 100. The SAMBA algorithm does not find any of the patterns because the algorithm cannot find shift and scale patterns but only constant patterns with large variance.

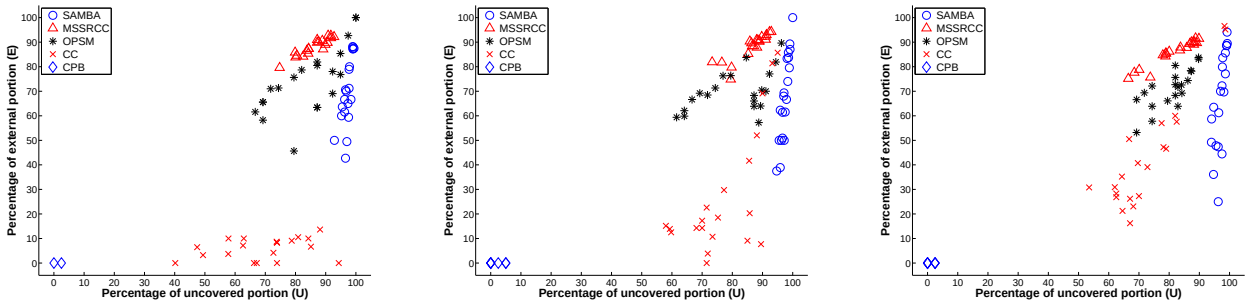
We have also tested implanting larger biclusters (up to 500x60), and found out that, except CPB, the performance of the algorithms improved marginally as the size of the implanted bicluster increased. CPB on the other hand, consistently returned a perfect match in all cases.

### 4.2.2 *Effect of noise*

Our second set of experiments evaluates the effect of noise in the input dataset. We generated three noise datasets, corresponding to three selected noise levels: 5%, 10% and 20%. The noise datasets were prepared from the 40x40 shift datasets by incrementing each value by a random percentage, up to 5%, 10% or 20% for the corresponding datasets. The results are displayed in Figure 3. In general, we see that with the increase in noise level, the performance of all algorithms degrades as expected. CPB gave the best results and was least affected by noise compared to other algorithms. With increasing noise level, CPB demonstrated an increase in U value, but the E value remained at 0. This indicates that, even though CPB fails to identify the implanted bicluster entirely under high noise, it is still successful at filtering out external rows and columns. CC, on the other hand, showed an increase in both U and E values with increasing noise. OPSM was most affected by noise, because the addition of noise to the dataset no longer preserved the the same ordering in columns.

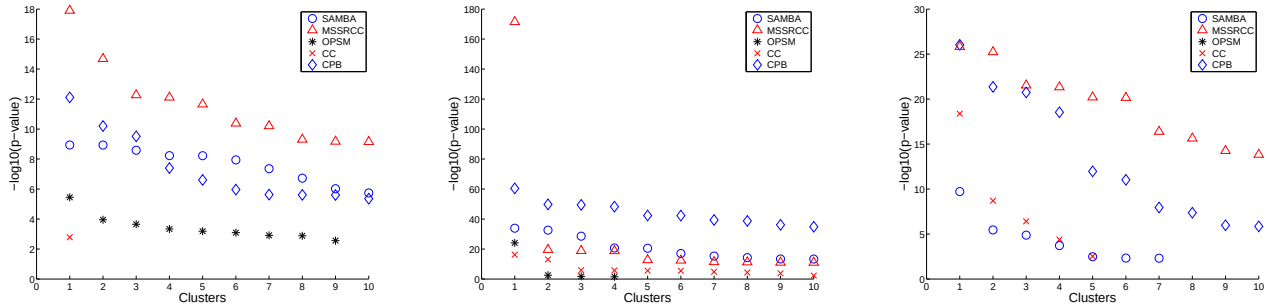
### 4.2.3 *Effect of overlap*

In the third set of experiments (Figure 4), we have evaluated effect of existence of overlapping clusters. We have generated 40x40 shift datasets with two implanted biclusters instead of one. There were three different types of datasets corresponding to the overlap ratio. The overlap ratio is defined as the ratio of the overlapping rows and columns in the implanted biclusters. For example, an overlap ratio of 0.5 indicates that 50% of the rows and 50% of the columns in




(a) no overlap (b) 25% overlap (c) 50% overlap

**Figure 4: Results on datasets with overlapping implanted biclusters. The overlap ratio shows the fraction of rows and columns overlapping in two biclusters. 40x40 shift datasets were used.**




(a) GDS1406, mouse (b) GDS1611, yeast (c) GDS1739, drosophila

**Figure 5: -log10 transformed p-values of the top 10 clusters for each algorithm on the three real datasets.**

two implanted biclusters are overlapping. The overlap ratios we used were 0, 0.25 and 0.5. CPB, OPSM and SAMBA are capable of detecting overlaps. Owing to their nature, CC and MSSRCC algorithms cannot detect overlapping patterns. As the overlap increased, CC showed an increase in U and E values and CPB remained unaffected.

## 4.3 Results on real datasets

We tested the algorithms on three large real datasets to determine the biological relevance of their results. The datasets we used were those of yeast (GDS1611), mouse (GDS1406) and drosophila (GDS1739) from the Gene Expression Omnibus (GEO) database [6]. The yeast dataset comprises 9275 genes and 96 conditions, the mouse dataset, 12422 genes and 87 conditions, and the drosophila dataset, 13966 genes and 54 conditions. The resulting biclusters were evaluated based on the enrichment of Gene Ontology (GO) terms. For each bicluster, the GO term with the smallest p-value is reported. In Figure 5, -log10 transformed p-values of the top 10 clusters with the smallest associated p-values are presented for each algorithm for each of the four datasets.

As can be seen in Figure 5(a), the clusters returned by the MSSRCC algorithm were statistically the most significant ones for the GDS1406 dataset, followed by those of CPB and SAMBA algorithms. On the GDS1611 dataset on the other hand, the clusters with the lowest associated p-values were returned by the CPB algorithm, followed by those of SAMBA and one of the OPSM clusters. However, the most significant cluster was identified by the MSSRCC algorithm. MSSRCC and CPB also performed well on the GDS1739 dataset, and CC algorithm also identified one of the most significant clusters for this dataset. However, in general CC and OPSM performed relatively poorly in our experiments on real datasets.

# 5. CONCLUSIONS

In this paper we analyzed and compared several biclustering algorithms including Cheng and Church, Correlated Pattern Biclusters and OPSM, in terms of the types of bicluster patterns they seek and the strength of their search strategies. First, we analytically modeled six well known biclustering patterns, then we examined algorithms with respect to the types of patterns that each of them can identify based on the constraints imposed by their quality criteria. Our mathematical analysis as well as experiments showed that Correlated Pattern Biclusters algorithm performs significantly better than the other considered algorithms that can identify local patterns, and can be a good candidate for searching simultaneously shifting and scaling patterns. The clusters identified by the CPB as well as the MSSRCC algorithm (which considers global patters) from real datasets

were found to be more significantly enriched than the clusters identified by other algorithms. This indicates that the patterns sought by these algorithms may have higher biological relevance. The analysis methods introduced in this paper can be extended to formally analyze various other biclustering algorithms both in terms of the target bicluster patterns and search strategies.

## 6. ACKNOWLEDGMENTS

This work was supported in parts by National Institutes of Health / National Cancer Institute Grant R01CA141090; by the U.S. DOE SciDAC Institute Grant DE-FC02-06ER2775; by the U.S. National Science Foundation under Grants CNS-0643969, OCI-0904809, OCI-0904802 and CNS-0403342; and an allocation of computing time from the Ohio Supercomputer Center.

## 7. REFERENCES

[1] J. S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20):3840–3845, 2005.

[2] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proc. International Conference on Computational Biology*, pages 49–57, 2002.

[3] D. Bozdag, J. D. Parvin, and U. Catalyurek. A biclustering method to discover co-regulated genes using diverse gene expression datasets. In *Proc. of International Conference on Bioinformatics and Computational Biology*, volume 5462, pages 151–163, April 2009.

[4] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. of the International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[5] H. Cho, I. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue based co-clustering of gene expression data. In *SIAM International Conference on Data Mining*, pages 114–125, 2004.

[6] R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, 2002.

[7] J. Gu and J. Liu. Bayesian biclustering of gene expression data. *BMC Genomics*, 9(Suppl 1):S4, 2008.

[8] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[9] D. Jiang, J. Pei, and A. Zhang. DHC: A density-based hierarchical clustering method for time series gene expression data. In *Proc. of the IEEE International Symposium on Bioinformatics and Bioengineering*, pages 393–400, 2003.

[10] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

[11] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research*, 37(15):e101–, 2009.

[12] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. In *Proc. of the IEEE International Conference on Data Mining*, page 187, 2003.

[13] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology Bioinformatics*, 1(1):24–45, 2004.

[14] T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. *In Pac. Symp. Biocomp.*, 8, 2003.

[15] J. A. Nepomuceno, A. T. Lora, J. S. Aguilar-Ruiz, and J. G. García-Gutiérrez. Biclusters evaluation based on shifting and scaling patterns. *Intelligent Data Engineering and Automated Learning*, pages 840–849, 2007.

[16] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006.

[17] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl_1):S243–252, 2001.

[18] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[19] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Supplement 1):136–144, 2002.

[20] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD*, 2002.

[21] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proc. of the IEEE International Symposium on Bioinformatics and Bioengineering*, page 321, 2003.

[22] K. Y. Yip, D. W. Cheung, and M. K. Ng. Harp: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1387–1397, 2004.