# Finding checkerboard patterns via fractional 0–1 programming

Andrew Trapp*, Oleg A. Prokopyev*, Stanislav Busygin†

July 25, 2010

**Abstract**

Biclustering is a simultaneous partitioning of the set of samples and the set of their attributes (features) into subsets (clusters). Samples and features clustered together are supposed to have a high relevance to each other. In this paper we provide a new mathematical programming formulation for unsupervised biclustering. The proposed model involves the solution of a fractional 0-1 programming problem. A linear-mixed 0–1 reformulation as well as two heuristic-based approaches are developed. Encouraging computational results on clustering real DNA microarray data sets are presented. In addition, we also discuss theoretical computational complexity issues related to biclustering.

*Keywords: unsupervised biclustering, data mining, fractional 0–1 programming, heuristics*

## 1 Introduction

Let a data set of $n$ samples and $m$ features be given as a rectangular matrix $A = (a_{ij})_{m\times n}$, where the value $a_{ij}$ is the value (or, expression) of the $i$-th feature in the $j$-th sample. We consider the clustering of the samples into clusters

$$\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r,\ \mathcal{S}_k \subseteq \{1 \ldots n\},\ k = 1 \ldots r,$$

$$\mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_r = \{1 \ldots n\},\ \ \mathcal{S}_k \cap \mathcal{S}_\ell = \emptyset,\ k, \ell = 1 \ldots r,\ k \neq \ell.$$

*Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA 15261. E-mails: act25@pitt.edu, prokopyev@engr.pitt.edu

†Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611. E-mail: busygin@ufl.edu

This clustering should be done so that samples from the same cluster share certain common properties. Correspondingly, a feature $i$ may be assigned to one of the feature clusters

$$\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r,\ \mathcal{F}_k \subseteq \{1 \ldots m\},\ k = 1 \ldots r,$$

$$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_r = \{1 \ldots m\},\quad \mathcal{F}_k \cap \mathcal{F}_\ell = \emptyset,\ k, \ell = 1 \ldots r,\ k \neq \ell,$$

in such a way that features of the cluster $\mathcal{F}_k$ are "responsible" for creating the cluster of samples $\mathcal{S}_k$. Such a simultaneous clustering of samples and features is called *biclustering*, which can be formally defined as follows [8]:

**Definition 1** *A biclustering of a data set is a collection of pairs of sample and feature subsets* $\mathcal{B} = ((\mathcal{S}_1, \mathcal{F}_1), (\mathcal{S}_2, \mathcal{F}_2), \ldots, (\mathcal{S}_r, \mathcal{F}_r))$ *such that the collection* $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r)$ *forms a partition of the set of samples, and the collection* $(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r)$ *forms a partition of the set of features.*

The term biclustering was first used by Cheng and Church [9, 20]. Other names such as *co-clustering*, *bidimensional clustering*, and *subspace clustering* are also often used in the literature [20].

While standard clustering algorithms are applied to either features or samples of the data set separately, biclustering obtains simultaneous clustering in these two dimensions. In traditional clustering each feature (sample) cluster is characterized using all samples (features) of the data set. On the other hand, in biclustering each sample (feature) cluster is induced by a specific subset of samples (features). This allows for the discovery of local patterns of interest, which are relevant to a specific subset of features only for a specific subset of samples and vice versa. Intuitively, this implies that *"clustering derives a global model while biclustering produces a local model"* [20].

Biclustering has great significance in biomedical applications, especially for analysis of DNA microarray data sets, which are currently in wide use. DNA microarrays measure gene expression levels of thousands of genes simultaneously, allowing researchers to observe how the genes act in different types of cells under various conditions. A typical microarray dataset includes several classes of samples each of which represents a certain medical condition or type of cells. When biclustering is performed with high reliability, it is possible to not only diagnose conditions represented by sample clusters, but also identify genes (features) responsible for them or serving as their markers [30].

There are diverse criteria used to relate clusters of samples and clusters of features. In general, biclustering relies on some kind of pattern of interest

among elements of a bicluster (co-cluster) [8, 20]. In our previous work [7] we defined *consistent biclustering* criteria and developed an algorithm for supervised biclustering under the biclustering consistency conditions. The key advantage of the proposed criteria is that in contrast to other biclustering schemes, consistent biclustering is theoretically justified by the conic separation property [7]. In this paper we extend our previous work on consistent biclustering for the case of unsupervised learning and develop new optimization-based algorithms for handling the unsupervised biclustering problem under the biclustering consistency conditions. In addition, we also present some new theoretical computational complexity results.

The remainder of the paper is organized in the following manner. Section 2 considers related work in the area of biclustering. In Section 3, we discuss the notion of *consistent biclustering*. Section 4 is concerned with theoretical computational complexity issues of the biclustering problem. Section 5 covers a new mathematical programming formulation for unsupervised biclustering. In Section 6 two heuristic algorithms for unsupervised biclustering are presented, while in Section 7 we discuss our computational experiments. Section 8 rounds out the paper with some concluding remarks.

## 2 Related Work

Supervised and unsupervised biclustering has been considered in a number of works, among which we should mention the biclustering of expression data investigated by Y. Cheng and G.M. Church [9], Cho et al. [10], a paper of I.S. Dhillon on textual biclustering using bipartite spectral graph partitioning [11] and the double conjugated clustering algorithm by S. Busygin, G. Jacobsen and E. Krämer [5]. Other types of biclustering include, for example, the order-preserving submatrix problem discussed in [3].

The choice of the similarity measure is a key challenge in biclustering. Probably, the most established metric is so-called *mean squared residue score*, proposed by Y. Cheng and G.M. Church [9]. To formulate it, let us introduce the following notation. Let

$$\mu_{ik}^{(r)} = \frac{1}{|\mathcal{S}_k|} \sum_{j \in \mathcal{S}_k} a_{ij} \tag{1}$$

be the mean of the $i$-th row in the sample cluster $\mathcal{S}_k$,

$$\mu_{jk}^{(c)} = \frac{1}{|\mathcal{F}_k|} \sum_{i \in \mathcal{F}_k} a_{ij} \tag{2}$$

be the mean of the $j$-th column in the feature cluster $\mathcal{F}_k$, and

$$\mu_k = \frac{\sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} a_{ij}}{|\mathcal{F}_k||\mathcal{S}_k|}$$

be the mean value in the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$. The *residue* of element $a_{ij}$ is defined as

$$r_{ij} = a_{ij} - \mu_{ik}^{(r)} - \mu_{jk}^{(c)} + \mu_k, \tag{3}$$

$i \in \mathcal{F}_k$, $j \in \mathcal{S}_k$. Finally, the *mean squared residue* score of the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is defined as

$$H_k = \sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} (r_{ij})^2 .$$

This value is equal to zero if all columns of the bicluster are equal to each other (which also implies that all rows are equal). A bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is called a $\delta$-bicluster if $H_k \leq \delta$. After proving that finding the largest square $\delta$-bicluster is $NP$-hard, Cheng and Church used a simple greedy procedure starting from the entire data matrix and successively removing columns or rows contributing most to the mean squared residue score [9]. The residue of the form (3) is also utilized in FLOC algorithm by Yang et. al [29].

Recently, Cho et al. [10] also used the residue defined in (3), but computed all biclusters simultaneously, optimizing the total squared residue as:

$$\min \sum_{k=1}^{K} H_k, \tag{4}$$

where input parameter $K$ is the total number of biclusters. In addition to (3), Cho et al. [10] implemented an alternative, simpler residue measure, given by

$$r_{ij} = a_{ij} - \mu_k, \tag{5}$$

which is the same metric used in so-called "direct clustering" (also known as block clustering) by Hartigan [17]. An additional advantage of the framework by Cho et al. [10] is that they were able to prove that their algorithms monotonically decrease the objective function (4) and converge to a local minimum.

Another class of optimization-based methods poses biclustering as an optimization problem in information theory minimizing the resulting loss in mutual information [12, 13].

For further information on biclustering methodology we refer the reader to detailed surveys [8, 20].

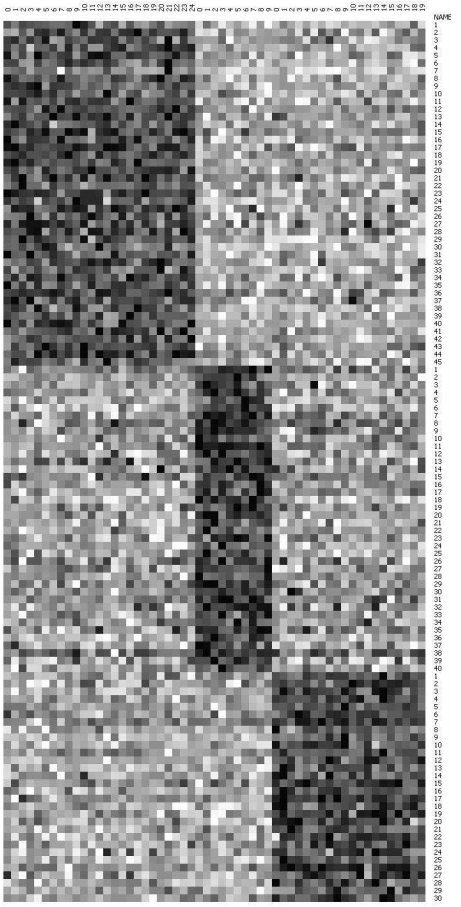


Figure 1: Example of consistent biclustering with 3 clusters of samples and features.

## 3 Consistent Biclustering

In this section we briefly review the notion of consistent biclustering, which has been introduced in [7]. Let each sample be already assigned to one of the clusters $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r$. Introduce a 0–1 matrix $S = (s_{jk})_{n \times r}$ such that $s_{jk} = 1$ if $j \in \mathcal{S}_k$, and $s_{jk} = 0$ otherwise. The sample class *centroids* can be computed as the matrix $C = (c_{ik})_{m \times r}$:

$$C = AS(S^T S)^{-1}, \tag{6}$$

whose $k$-th column represents the centroid of the cluster $\mathcal{S}_k$.

Consider a row $i$ of the matrix $C$. Each value in it gives us the average expression of the $i$-th feature in one of the sample clusters. As we want to identify the "checkerboard" pattern in the data, we have to assign the feature to the cluster where it is most expressed. So, let us assign the $i$-th feature to the cluster $\hat{k}$ with the maximal value $c_{i\hat{k}}$:

$$i \in \mathcal{F}_{\hat{k}} \;\Rightarrow\; \forall k = 1 \ldots r,\; k \neq \hat{k} :\; c_{i\hat{k}} > c_{ik} \tag{7}$$

Now, provided the clustering of all features into clusters $\mathcal{F}_1$, $\mathcal{F}_2$, …, $\mathcal{F}_r$, let us construct clustering of samples using the same principle of maximal average expression and see whether we will arrive at the same clustering as the initially given one. To do this, construct a 0–1 matrix $F = (f_{ik})_{m \times r}$ such that $f_{ik} = 1$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$ otherwise. Then, the feature cluster centroids can be computed in form of matrix $D = (d_{jk})_{n \times r}$:

$$D = A^T F(F^T F)^{-1}, \tag{8}$$

whose $k$-th column represents the centroid of the cluster $\mathcal{F}_k$. The condition on sample clustering we need to verify is

$$j \in \mathcal{S}_{\hat{k}} \;\Rightarrow\; \forall k = 1 \ldots r,\; k \neq \hat{k} :\; d_{j\hat{k}} > d_{jk} \tag{9}$$

Let us state now the definition of biclustering consistency as given in [7].

**Definition 2** *A biclustering $\mathcal{B}$ will be called consistent if both relations (7) and (9) hold, where the matrices $C$ and $D$ are defined as in (6) and (8).*

An example of the data set with three consistent biclusters of over-expressed values is visualized in Figure 1. It is easy to observe that the average expression of any feature from bicluster $\mathcal{B}_1$ on samples from bicluster $\mathcal{B}_1$ is greater than average expression of the same feature on samples

from biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. Correspondingly, for samples we see that the average expression of any sample from bicluster $\mathcal{B}_1$ on features from bicluster $\mathcal{B}_1$ is greater than the average expression of the same sample on features from biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. Similar observations can be made for biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. This "checkerboard" pattern may mean for microarray data, for example, strong up-regulation of certain genes under a cancer condition of a particular type (whose samples constitute one class of the data set).

The key advantage of consistent biclustering is the formal setup it provides for the desired separability of clusters. It was shown in [7] that consistent biclustering implies separability of the clusters by convex cones[1]:

**Theorem 1** *Let $\mathcal{B}$ be a consistent biclustering. Then there exist convex cones $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r \subseteq \mathbb{R}^m$ such that all samples from $\mathcal{S}_k$ belong to the cone $\mathcal{P}_k$ and no other sample belongs to it, $k = 1 \ldots r$.*

*Similarly, there exist convex cones $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_r \subseteq \mathbb{R}^n$ such that all features from $\mathcal{F}_k$ belong to the cone $\mathcal{Q}_k$ and no other feature belongs to it, $k = 1 \ldots r$.*

It also follows from the conic separability that the convex hulls of clusters are separated, i.e, they do not intersect.

The consistent biclustering criteria is rather strict and data sets may not necessarily contain biclusters satisfying constraints (7) and (9). Hence, we say that:

**Definition 3** *A data set is called biclustering-admitting if there exists some consistent biclustering for it.*

In supervised clustering the researcher is given a training set of samples whose classes are known, and this a priori information can be used to classify the test set of samples. Therefore, we also present an additional notion:

**Definition 4** *The data set is called conditionally biclustering-admitting with respect to a given (partial) classification of some samples and/or features if there exists a consistent biclustering preserving the given (partial) classification.*

[1] Recall that set $S$ is a convex cone if it is convex and a cone, which means that for any $x_1,\ x_2 \in S$ and $\lambda_1,\ \lambda_2 \geq 0$, we have $\lambda_1 x_1 + \lambda_2 x_2 \in S$, see, e.g., [6].

# 4 Computational Complexity

One of the open questions stated in [7] is to consider the computational complexity of consistent biclustering. Next we partially answer this question and show that finding the largest conditionally biclustering-admitting submatrix (CBASM) of a given data matrix is $NP$-hard even for two classes ($r = 2$). Consider the following decision version of this problem:

**Instance**: A real-valued data matrix $A = (a_{ij})_{m\times n}$, $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2 \subseteq \{1, \ldots, n\}$, $\tilde{\mathcal{F}}_1, \tilde{\mathcal{F}}_2 \subseteq \{1, \ldots, m\}$ such that $\tilde{\mathcal{S}}_1 \bigcap \tilde{\mathcal{S}}_2 = \tilde{\mathcal{F}}_1 \bigcap \tilde{\mathcal{F}}_2 = \emptyset$, and six integers $k$, $k_1$, $k_2$, and $\ell$, $\ell_1$ and $\ell_2$ such that $k \leq n$, $\ell \leq m$, $k_1 \leq |\tilde{\mathcal{S}}_1|$, $k_2 \leq |\tilde{\mathcal{S}}_2|$, $\ell_1 \leq |\tilde{\mathcal{F}}_1|$ and $\ell_2 \leq |\tilde{\mathcal{F}}_2|$.
**Question**: Are there sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \{1, \ldots, n\}$ and $\mathcal{F}_1, \mathcal{F}_2 \subseteq \{1, \ldots, m\}$ such that $\mathcal{S}_1 \bigcap \mathcal{S}_2 = \mathcal{F}_1 \bigcap \mathcal{F}_2 = \tilde{\mathcal{S}}_1 \bigcap \mathcal{S}_2 = \mathcal{S}_1 \bigcap \tilde{\mathcal{S}}_2 = \tilde{\mathcal{F}}_1 \bigcap \mathcal{F}_2 = \mathcal{F}_1 \bigcap \tilde{\mathcal{F}}_2 = \emptyset$, and $|S_1|+|S_2| \geq k$, $|F_1|+|F_2| \geq \ell$, $|\tilde{\mathcal{S}}_1 \bigcap \mathcal{S}_1| \geq k_1$, $|\tilde{\mathcal{S}}_2 \bigcap \mathcal{S}_2| \geq k_2$, $|\tilde{\mathcal{F}}_1 \bigcap \mathcal{F}_1| \geq \ell_1$, $|\tilde{\mathcal{F}}_2 \bigcap \mathcal{F}_2| \geq \ell_2$ and biclustering $\mathcal{B} = ((\mathcal{S}_1, \mathcal{F}_1), (\mathcal{S}_2, \mathcal{F}_2))$ is consistent?

The idea behind the above described decision problem is as follows. We want to check if there exists a submatrix of size $\ell \times k$, which is biclustering-admitting and partially preserves a given classification of samples (at least for $k_1$ and $k_2$ samples from $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2$, respectively) and features (at least for $\ell_1$ and $\ell_2$ features from $\tilde{\mathcal{F}}_1, \tilde{\mathcal{F}}_2$, respectively). All other samples and features from $\tilde{\mathcal{S}}_i$, $\tilde{\mathcal{F}}_i$, respectively ($i = 1, 2$), which do not belong to their respective $\mathcal{S}_i$ and $\mathcal{F}_i$ in the resulting consistent biclustering are considered to be outliers.

**Theorem 2** *CBASM problem is $NP$-complete.*

**Proof.** In the reduction described below we use the **Balanced Complete Bipartite Subgraph** problem, which is known to be $NP$-complete [14]:

**Instance**: Bipartite graph $G = (V, E)$, positive integer $K \leq |V|$.
**Question**: Are there two disjoint sets $V_1$, $V_2 \subseteq V$ such that $|V_1|=|V_2|=K$ and such that $u \in V_1$, $v \in V_2$ implies that $\{u, v\} \in E$?

Given a bipartite graph $G = (V, U, E)$ define that matrix $A = (a_{ij})_{m\times n}$ as follows. Let $m = |V| + 1$ and $n = |U| + 1$. Define the values of $a_{ij}$ as follows:

- $a_{ij} = 1$ if $i \leq |V|$, $j \leq |U|$ and $(i, j) \in E$
- $a_{i,(n+1)} = a_{(m+1),j} = 1 - \epsilon$ for $i = 1 \ldots m$, $j = 1 \ldots n$, where $\epsilon < \min\{\frac{1}{m}, \frac{1}{n}\}$
- $a_{(m+1),(n+1)} = 1$

- $a_{ij} = 0$, otherwise.

Let $\tilde{\mathcal{S}}_1 = \{1, 2, \ldots, n\}$, $\tilde{\mathcal{F}}_1 = \{1, 2, \ldots, m\}$, $\tilde{\mathcal{S}}_2 = \{n+1\}$, $\tilde{\mathcal{F}}_2 = \{m+1\}$, $k = K+1$, $\ell = K+1$, $k_1 = K$, $k_2 = 1$, $\ell_1 = K$, $\ell_2 = 1$.

Next we show that $G$ contains a balanced complete bipartite graph of size $K$ if and only if the matrix $A$ contains a submatrix of size $(K+1) \times (K+1)$ conditionally biclustering-admitting with respect to a given (partial) classification of samples $\tilde{\mathcal{S}}_1$, $\tilde{\mathcal{S}}_2$ and features $\tilde{\mathcal{F}}_1$, $\tilde{\mathcal{F}}_2$.

The first direction can be proven as follows. Suppose $G$ contains a balanced complete bipartite subgraph with nodes $V_1$ and $V_2$ of size $K$. Let $\mathcal{F}_1$ and $\mathcal{S}_1$ consists of all indices, which correspond to nodes from $V_1$ and $V_2$, respectively, while $\mathcal{S}_2 = \{n+1\}$ and $\mathcal{F}_2 = \{m+1\}$ by construction. Then for any $i \in \mathcal{F}_1$ and $j \in \mathcal{S}_1$

$$c_{i1} = \frac{\sum_{p \in V_2} a_{ip}}{K} = 1 > 1 - \epsilon = a_{i,(n+1)} = c_{i2} \tag{10}$$

$$d_{j1} = \frac{\sum_{q \in V_1} a_{qj}}{K} = 1 > 1 - \epsilon = a_{(m+1),j} = d_{j2} \tag{11}$$

For $(\mathcal{S}_2, \mathcal{F}_2)$ we have that

$$c_{(m+1),2} = a_{(m+1),(n+1)} = 1 > \frac{\sum_{p \in V_2} a_{(m+1),p}}{K} = 1 - \epsilon = c_{(m+1),1} \tag{12}$$

$$d_{(n+1),2} = a_{(m+1),(n+1)} = 1 > \frac{\sum_{q \in V_1} a_{q,(n+1)}}{K} = 1 - \epsilon = d_{(n+1),1} \tag{13}$$

Inequalities (10)-(13) imply that (7) and (9) are satisfied and the constructed submatrix is biclustering-admitting.

In order to show the opposite direction, assume that we have a submatrix of size $(K+1) \times (K+1)$, which is conditionally biclustering-admitting with respect to a given (partial) classification of samples and features described above. Let $V_1$ and $V_2$ correspond to indices from $\mathcal{F}_1$ and $\mathcal{S}_1$, respectively. Next we show that the subgraph induced by $V_1$ and $V_2$ is complete, i.e., if $i \in \mathcal{F}_1$ and $j \in \mathcal{S}_1$ then $a_{ij} = 1$. Suppose this is not true and there exist $i \in V_1$ and $j \in V_2$ such that $a_{ij} = 0$. Then

$$c_{i1} = \frac{\sum_{p \in V_2} a_{ip}}{K} \leq \frac{K-1}{K} = 1 - \frac{1}{K} \leq 1 - \epsilon = a_{i,(n+1)} = c_{i2}, \tag{14}$$

and (7) is not satisfied. This contradicts the initial assumption that the biclustering is consistent. Therefore, $a_{ij} = 1$.

■

# 5 Supervised and Unsupervised Biclustering

In this section we first review our previous work on supervised biclustering via fractional 0–1 programming [7]. We then demonstrate how this approach can be extended to handle the unsupervised case.

## 5.1 Supervised Biclustering

Supervised clustering consists of two routines. The first derives classification criteria while processing the training samples, while the second applies these criteria to the test samples. In genomic and proteomic data analysis, as well as in other data mining applications, where only a small subset of features is expected to be relevant to the classification of interest, the classification criteria should involve dimensionality reduction and feature selection. In [7], it was proposed to handle this task utilizing the notion of consistent biclustering. Namely, we select a subset of features of the original data set in such a way that the obtained subset of data becomes conditionally biclustering-admitting with respect to the given classification of training samples.

Formally, let us introduce a vector of 0–1 variables $x = (x_i)_{i=1\dots m}$, and consider the $i$-th feature selected if $x_i = 1$, and $x_i = 0$, otherwise. The condition of biclustering consistency, when only the selected features are used, becomes

$$\frac{\sum_{i=1}^m a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^m f_{i\hat{k}} x_i} > \frac{\sum_{i=1}^m a_{ij} f_{ik} x_i}{\sum_{i=1}^m f_{ik} x_i},\ \forall j \in \mathcal{S}_{\hat{k}},\ \hat{k}, k = 1 \dots r,\ \hat{k} \neq k. \tag{15}$$

We will use the fractional relations (15) as constraints of an optimization problem, which may incorporate various objective functions over $x$, depending on the desirable properties of the selected features. One general choice is to select the maximal possible number of features in order to lose the minimal amount of information provided by the training set. Thus one possible fractional 0-1 programming formulation based on the biclustering criterion is:

$$\max_{x \in \mathbb{B}^n} \sum_{i=1}^m x_i \tag{16}$$

subject to

$$\frac{\sum_{i=1}^m a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^m f_{i\hat{k}} x_i} \geq (1+t) \frac{\sum_{i=1}^m a_{ij} f_{ik} x_i}{\sum_{i=1}^m f_{ik} x_i},\ \forall j \in \mathcal{S}_{\hat{k}},\ \hat{k}, k = 1 \dots r,\ \hat{k} \neq k, \tag{17}$$

where (17) is a modified version of (15) and constant $t > 0$ is a parameter of the method. We will refer to $t$ as *the parameter of separation*. The introduction of $t$ may result in the improvement of quality of the feature selection since it strengthens the class separation. Notice also by doing so, we have replaced the strict inequalities by non-strict ones and made the feasible domain closed.

After the feature selection is done, we perform classification of test samples according to (9). That is, if $b = (b_i)_{i=1\ldots m}$ is a test sample, then we assign it to the class $\mathcal{F}_{\hat{k}}$ satisfying

$$\frac{\sum_{i=1}^{m} b_i f_{i\hat{k}} x_i^*}{\sum_{i=1}^{m} f_{i\hat{k}} x_i^*} > \frac{\sum_{i=1}^{m} b_i f_{ik} x_i^*}{\sum_{i=1}^{m} f_{ik} x_i^*},\ k = 1 \ldots r,\ \hat{k} \neq k, \tag{18}$$

where $x^*$ is the solution of the feature selection procedure given by (16)-(17).

The optimization problem (16)-(17) is a specific class of fractional 0–1 programming, which has been discussed extensively in [22, 23, 25, 27], among others. The optimization problem (16)-(17) can be reformulated as a linear mixed 0-1 programming problem by applying a similar approach to that used in [27] to linearize problems with fractional 0-1 objective functions. This technique as well as a simple heuristic algorithm for solving (16)-(17) were presented in [7]. The obtained features were used for classification of test data according to (18), providing excellent results for both the HuGE (59 samples $\times$ 7070 features) [32] and ALL vs. AML (72 samples $\times$ 6281 features) [15] data sets.

### 5.2 Unsupervised Biclustering

The goal of clustering is to partition the given data into subgroups of similar samples according to some similarity criteria relevant to the purpose of the performed analysis. In contrast to the case of supervised biclustering, where pre-classified training data is available, unsupervised biclustering does not use training data to develop appropriate classification criteria.

Consider assigning each sample to one of the clusters

$$\mathcal{S}_1,\ \mathcal{S}_2,\ \ldots,\ \mathcal{S}_r.$$

To facilitate this, let $S = (s_{jk})_{n\times r}$ now represent a set of 0–1 variables such that $s_{jk} = 1$ if $j \in \mathcal{S}_k$, and $s_{jk} = 0$, otherwise. Similarly, consider clustering all features into clusters

$$\mathcal{F}_1,\ \mathcal{F}_2,\ \ldots,\ \mathcal{F}_r.$$

Also, we now let $F = (f_{ik})_{m\times r}$ represent a set of 0–1 variables such that $f_{ik} = 1$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$, otherwise.

One can verify that the following sets of constraints:

$$s_{j\hat{k}} \left( \frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{k}}}{\sum_{i=1}^{m} f_{i\hat{k}}} - (1+t) \frac{\sum_{i=1}^{m} a_{ij} f_{ik}}{\sum_{i=1}^{m} f_{ik}} \right) \geq 0 \; \forall j, \; \hat{k}, \; k = 1 \ldots r, \; \hat{k} \neq k, \quad (19)$$

$$f_{i\hat{k}} \left( \frac{\sum_{j=1}^{n} a_{ij} s_{j\hat{k}}}{\sum_{j=1}^{n} s_{j\hat{k}}} - (1+t) \frac{\sum_{j=1}^{n} a_{ij} s_{jk}}{\sum_{j=1}^{n} s_{jk}} \right) \geq 0 \; \forall i, \; \hat{k}, \; k = 1 \ldots r, \; \hat{k} \neq k \quad (20)$$

enforce biclustering consistency given by (7) and (9), where the parameter of separation $t$ has the same meaning as in (17). Furthermore, the unsupervised biclustering formulation requires additional constraints related to the clustering of samples and features, namely

- each feature can be selected to at most one cluster, and each cluster must contain at least one feature:

$$\sum_{k=1}^{r} f_{ik} \leq 1 \; \forall i \quad \text{and} \quad \sum_{i=1}^{m} f_{ik} \geq 1 \; \forall k, \quad (21)$$

- each sample can be selected to at most one cluster, and each cluster must contain at least one sample:

$$\sum_{k=1}^{r} s_{jk} \leq 1 \; \forall j \quad \text{and} \quad \sum_{j=1}^{n} s_{jk} \geq 1 \; \forall k. \quad (22)$$

Depending on the clustering problem under consideration, conditions (21)-(22) can be modified one way or another. For example, we may allow certain features to belong to several biclusters, or require each sample to be clustered (i.e., there are no outliers) in which case constraints (21)-(22) should be modified correspondingly.

Combining constraints (19)-(20) and (21)-(22) with a suitable objective function, for instance

$$n \cdot \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik} + m \cdot \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk} \quad (23)$$

yields a fractional 0–1 programming problem. The goal of (23) is to select as many features and samples as possible while at the same time satisfying constraints on biclustering consistency. Samples which are not selected in any of the clusters in the solution of (23) subject to (19)-(20) and (21)-(22) are considered to be outliers.

### 5.3 Linear Mixed 0–1 Reformulation

Now we will demonstrate how to linearize fractional formulation (23), (19)-(20) and (21)-(22), making it amenable to solution via standard linear mixed-integer programming solvers such as CPLEX or XPRESS-MP.

In order to make conditions (19) and (20) more manageable, observe that they are equivalent to

$$\frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{k}}}{\sum_{i=1}^{m} f_{i\hat{k}}} - (1+t)\frac{\sum_{i=1}^{m} a_{ij} f_{ik}}{\sum_{i=1}^{m} f_{ik}} \geq -L_j^s(1 - s_{j\hat{k}}) \;\; \forall j,\; \hat{k},\; k,\; \hat{k} \neq k, \quad (24)$$

$$\frac{\sum_{j=1}^{n} a_{ij} s_{j\hat{k}}}{\sum_{j=1}^{n} s_{j\hat{k}}} - (1+t)\frac{\sum_{j=1}^{n} a_{ij} s_{jk}}{\sum_{j=1}^{n} s_{jk}} \geq -L_i^f(1 - f_{i\hat{k}}) \;\; \forall i,\; \hat{k},\; k,\; \hat{k} \neq k, \quad (25)$$

for large enough constants $L_i^f$ and $L_j^s$, which can be chosen, for example, as

$$L_j^s = \max_i a_{ij} - (1+t)\min_i a_{ij}; \quad L_i^f = \max_j a_{ij} - (1+t)\min_j a_{ij} \quad (26)$$

The following proposition, which can be found, for example, in [27], can be utilized to address our formulation:

**Proposition 1** *A polynomial mixed 0–1 term $z = xy$, where $x$ is a 0–1 variable, and $y$ is a continuous variable taking any positive value, can be represented by the following linear inequalities: (1) $y - z \leq M - Mx$; (2) $z \leq y$; (3) $z \leq Mx$; (4) $z \geq 0$, where $M$ is a large constant greater than $y$.*

Making use of this proposition, let us introduce new variables

$$u_k = \frac{1}{\sum_{i=1}^{m} f_{ik}},\; k = 1 \ldots r, \quad (27)$$

$$v_k = \frac{1}{\sum_{j=1}^{n} s_{jk}},\; k = 1 \ldots r, \quad (28)$$

$$z_{ik} = \frac{f_{ik}}{\sum_{\ell=1}^{m} f_{\ell k}},\; i = 1 \ldots m,\; k = 1 \ldots r, \quad (29)$$

$$y_{jk} = \frac{s_{jk}}{\sum_{\ell=1}^{n} s_{\ell k}},\; j = 1 \ldots n,\; k = 1 \ldots r. \quad (30)$$

By means of variable substitution, then, we replace the nonlinear 0–1 inequalities (24) and (25) with the following linear-mixed 0–1 constraints:

$$\sum_{i=1}^{m} a_{ij} z_{i\hat{k}} - (1+t)\sum_{i=1}^{m} a_{ij} z_{ik} \geq -L_j^s(1 - s_{j\hat{k}}) \;\; \forall j,\; \hat{k},\; k,\; \hat{k} \neq k, \quad (31)$$

$$\sum_{j=1}^{n} a_{ij} y_{j\hat{k}} - (1+t)\sum_{j=1}^{n} a_{ij} y_{jk} \geq -L_i^f (1 - f_{i\hat{k}}) \;\; \forall i,\; \hat{k},\; k,\; \hat{k} \neq k, \tag{32}$$

$$\sum_{i=1}^{m} z_{ik} = 1,\; k = 1 \ldots r, \;\; \sum_{j=1}^{n} y_{jk} = 1,\; k = 1 \ldots r, \tag{33}$$

$$u_k - z_{ik} \leq 1 - f_{ik},\; v_k - y_{jk} \leq 1 - s_{jk},\; z_{ik} \leq u_k,\; y_{jk} \leq v_k,\; \forall\, i, j, k, \tag{34}$$

$$z_{ik} \leq f_{ik},\; y_{jk} \leq s_{jk},\; z_{ik} \geq 0,\; y_{jk} \geq 0,\; \forall\, i, j, k. \tag{35}$$

Objective function (23) along with conditions (21)-(22) and (31)-(35) thus give a linear mixed 0–1 reformulation of the original nonlinear 0–1 programming problem. Using this reformulation, the number of new variables is $2r + (m+n)r$, yielding a total of $2r(m+n+1)$ variables.

# 6 Heuristic Approaches

Unfortunately, our linear mixed 0–1 reformulation can not be used for solving large-scale instances of the biclustering problem even with the best techniques implemented in modern integer programming solvers. As a consequence, we have developed two alternative heuristic approaches for solving the problem given by (23) and (19)-(22). The first algorithm iteratively solves a relaxation of the mixed integer programming problem given by (23), (21)-(22) and (31)-(35) (subsequently referred to as main MIP), while the second is a variation of a local search based heuristic.

## 6.1 Heuristic 1 (H1): MIP-based Heuristic

The first heuristic is an extension of the algorithm described in [7]. It solves relaxations of the main MIP in an iterative fashion until specified criteria are met.

To shed light on our motivations for this heuristic, consider the meaning of variables $z_{ik}$ and $y_{jk}$. We have introduced them so that

$$z_{ik} = \frac{f_{ik}}{\sum_{\ell=1}^{m} f_{\ell k}},\; i \in \mathcal{F}_k, \tag{36}$$

and

$$y_{jk} = \frac{s_{jk}}{\sum_{\ell=1}^{n} s_{\ell k}},\; j \in \mathcal{S}_k. \tag{37}$$

Thus, for $i \in \mathcal{F}_k$, $z_{ik}$ is the reciprocal of the cardinality of cluster $\mathcal{F}_k$ after feature selection, if the $i$-th feature is selected, and 0 otherwise; likewise, for

$j \in \mathcal{S}_k$, $y_{jk}$ is the reciprocal of the cardinality of cluster $\mathcal{S}_k$ if the $j$-th sample is selected, and 0 otherwise. This reveals that $z_{ik}$ and $y_{jk}$ are also binary variables just as $f_{ik}$ and $s_{jk}$ are, however, their nonzero values are just not set to 1. Though these nonzero values are not known until the optimal sizes of feature and sample clusters are obtained, knowing the values of $z_{ik}$ and $y_{jk}$ suffices to define the values of $f_{ik}$ and $s_{jk}$, and the system of constraints with respect only to the continuous variables $0 \leq z_{ik} \leq 1$ and $0 \leq y_{jk} \leq 1$ (that is, dropping constraints (34) involving $u_k$ and $v_k$) constitutes a linear relaxation of the main MIP. Furthermore, it can be strengthened by the system of inequalities connecting $z_{ik}$ to $f_{ik}$ and $y_{jk}$ to $s_{jk}$. Indeed, knowing that no more than $m_k$ features can be selected for cluster $\mathcal{F}_k$ and that no more than $n_k$ samples can be selected for cluster $\mathcal{S}_k$, it is valid to impose:

$$m_k z_{ik} \geq f_{ik} \ \forall i, k, \tag{38}$$

and

$$n_k y_{jk} \geq s_{jk} \ \forall j, k. \tag{39}$$

These inequalities strengthen the relaxation of the main MIP, and furthermore, we can prove:

**Theorem 3** *If $f^*$, $s^*$ is an optimal solution to (23), (21)-(22) and (31)-(35), and if $\forall k$ $m_k = \sum_{i=1}^{m} f_{ik}^*$, and if $\forall k$ $n_k = \sum_{j=1}^{n} s_{jk}^*$, then $f^*$, $s^*$ is also an optimal solution to (23), (21)-(22), (31)-(33), (35), and (38)-(39).*

**Proof.** Since the new program is a relaxation of the original (having 2 additional sets of constraints which are satisfied by any feasible solution to the original program), $f^*$ and $s^*$ must be feasible for the new program as well. Thus, we need only demonstrate that (23), (21)-(22), (31)-(33), (35), and (38)-(39) does not have a better solution. Let us assume the contrary, that is, there exists some $f^{**}$, $s^{**}$ such that

$$m \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk}^{**} + n \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik}^{**} > m \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk}^{*} + n \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik}^{*}.$$

On the other hand, we have that $f_{ik} \leq m_k z_{ik} \ \forall i, k$, and likewise that $s_{jk} \leq n_k y_{jk} \ \forall j, k$. Summing over $i$ and $k$, we can see

$$n \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik}^{**} \leq n \sum_{i=1}^{m} \sum_{k=1}^{r} m_k z_{ik},$$

and

$$m \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk}^{**} \leq m \sum_{j=1}^{n} \sum_{k=1}^{r} n_k y_{jk}.$$

Now summing these equations, and in conjunction with (33) along with our assumptions about $m_k$ and $n_k$, we have that

$$m \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk}^{**} + n \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik}^{**} \leq m \sum_{j=1}^{n} \sum_{k=1}^{r} n_k y_{jk} + n \sum_{i=1}^{m} \sum_{k=1}^{r} m_k z_{ik} =$$

$$m \sum_{k=1}^{r} \sum_{j=1}^{n} n_k y_{jk} + n \sum_{k=1}^{r} \sum_{i=1}^{m} m_k z_{ik} = m \sum_{k=1}^{r} n_k \sum_{j=1}^{n} y_{jk} + n \sum_{k=1}^{r} m_k \sum_{i=1}^{m} z_{ik} =$$

$$= m \sum_{k=1}^{r} n_k + n \sum_{k=1}^{r} m_k = m \sum_{j=1}^{n} \sum_{k=1}^{r} s_{jk}^{*} + n \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik}^{*},$$

which leads us to a contradiction. ■

The above proof is a generalization of Theorem 3 in [7] and provides the intuition behind our first heuristic algorithm.

**Algorithm 1**

1. *Assign* $m_k := m$ *and* $n_k := n$, $k = 1 \ldots r$.
2. *Solve the mixed 0–1 programming formulation (23) together with (21)-(22), (31)-(33), (35), and (38)-(39).*
3. *If* $m_k = \sum_{i=1}^{m} f_{ik}$ $\forall k$, *and* $n_k = \sum_{j=1}^{m} s_{jk}$ $\forall k$, *go to 6.*
4. *Assign* $m_k := \sum_{i=1}^{m} f_{ik}$ *and* $n_k := \sum_{j=1}^{n} s_{jk}$ $\forall k$.
5. *Go to 2.*
6. *STOP.*

Our computational experiments indicated that, though this heuristic performs much faster than solving the original linear mixed 0–1 reformulation, it is still not efficient enough for tackling reasonably large real-life microarray data sets.

### 6.2 Heuristic 2 (H2): Multi-start Iterative Heuristic

As an alternative to Algorithm 1, we also developed another iterative based heuristic. The heuristic is first primed by generating a random assignment of the $n$ samples to the $r$ clusters. Using this initial clustering of samples, the $m$ features are then clustered according to (7). Given the obtained partitioning of features, we perform new clustering of samples according to (9). The process continues in this manner, iteratively refining both row and column clusters, until two stopping conditions have been met:

1) $r$ clusters have been generated and,

2) no samples nor features have switched clusters for one full iteration.

It is possible that we may reach an iteration where one or more clusters may become empty. If fewer than $r$ clusters have been formed, the algorithm resets by generating another random assignment of samples to clusters and continues as before. The pseudocode of the above described routine (with some additional enhancements, which we will discuss later in this subsection) is outlined in Procedure 3.

The final result of Procedure 3 may substantially depend on the initial random assignment of samples to clusters. Therefore, we decided to run multiple trials of the procedure, hence the term *multi-start*, in order to determine to which cluster each sample predominantly belonged based on a simple majority vote, with ties broken arbitrarily.

Let us formalize our terminology by defining a *solution* as consisting of a particular assignment of samples to clusters, and let us define a *trial* as consisting of one pass of the heuristic Procedure 3 - typically consisting of multiple iterations, until stopping conditions 1) and 2) above have been met.

At the first step of the algorithm, we construct an initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$ in Procedure 1. We obtain our initial solution using multiple trials of Procedure 3. Among generated sample clusterings, we choose the one which minimizes a simple "diversity" metric (see Step 3 of Procedure 1).

Once the initial sample clustering $(\mathcal{S}_1^0, \ldots, \mathcal{S}_r^0)$ solution has been generated, the multi-start Procedure 2 begins. This procedure applies *MSLim* trials of Procedure 3 to generate a starting solution for use in the final biclustering of samples and features (Step 3 of Algorithm 2). For each trial the corresponding sample clusterings are recorded, and upon completion of the multi-start procedure, there will be a cluster to which each sample was

assigned a majority of times (ties broken arbitrarily). We assign each sample to its "majority" cluster and use this final sample clustering in the final trial of the algorithm.

The key issue with this multi-start Procedure 2 is that the actual clusters $k = 1 \ldots r$ are not unique across trials. For example, it is possible to have two identical sample clustering with differing cluster numbers. In a given trial samples 1 and 2 could be assigned to cluster 1, while samples 3 and 4 could be assigned to cluster 2. We should consider this clustering identical to the clustering where samples 1 and 2 are assigned to cluster 2 while samples 3 and 4 are assigned to cluster 1. In order to distinguish this and other similar situations we make use of the initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$ (see, Step 3 of Procedure 2).

**Algorithm 2**

1. *Call Procedure 1.*
2. *Call Procedure 2.*
3. *Using results from Procedure 2 as initial sample clustering, call Procedure 3 to obtain final biclustering.*

**Procedure 1** */* Construction of initial sample clustering */*

1. *DO generate random assignment of samples to clusters.*
2. *Call Procedure 3.*
3. *Evaluate "diversity" of the resulting solution by calculating*

$$d = \sum_{k=1}^{r} \left( \frac{n_k}{n} - \frac{1}{r} \right)^2,$$

   *where $n_k$ is the number of samples in cluster $k$.*
4. *If the obtained solution has a smaller value of $d$, store it as the current initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$.*
5. *WHILE initTrialCount < initLim;*
6. *STOP and return the obtained solution.*

**Procedure 2** */* Improve initial sample clustering via voting */*

1. *DO generate random assignment of samples to clusters.*
2. *Call Procedure 3 to obtain sample clustering $(\mathcal{S}_{1'}, \mathcal{S}_{2'}, \ldots, \mathcal{S}_{r'})$.*
3. *Permutate $i' \to j$ minimizing Euclidean distance of $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r)$ from $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$.*

**4.** *Record assignment of samples to clusters for current trial.*

**5.** *WHILE MSTrialCount < MSLim;*

**6.** *Categorize each sample according to the cluster to which it was assigned a majority of times. Break ties arbitrarily.*

**Procedure 3** */* Iteration based biclustering */*

**1.** *Assign features to clusters according to sample clustering. For a given feature, this is accomplished as follows:*

**2.** *For* $k = 1 \dots r$*, determine the number of entries* $n_k$ *and the sum of entries* $s_k$ *in this feature.*

**3.** $\forall k : n_k \geq 1$*, calculate the average expression* $\frac{s_k}{n_k}$*.*

**4.** *Across all* $k : n_k \geq 1$*, calculate the average expression* $\frac{s_k}{n_k}$*. Let cluster* $k^*$ *be the cluster with the largest average expression.*

**5.** $\forall \hat{k} : \hat{k} \neq k^*, n_{\hat{k}} \geq 1$ *in this feature, verify that* $\frac{s^*_k}{n^*_k} > (1+t)\,\frac{s_{\hat{k}}}{n_{\hat{k}}}$*, where* $t$ *is the parameter of separation.*

**6.** *If the inequality in Step (5) does not hold for at least one* $\hat{k}$*, this feature is labeled as unclustered; otherwise, the feature is assigned to cluster* $k^*$*.*

**7.** *Once all features have been assigned to a particular cluster according to sample clustering, cluster samples according to feature clustering by repeating similar steps.*

**8.** *If an entire pass occurs with no feature nor sample clustering changes, find the number of non-empty clusters* $\hat{r}$*.*

**9.** *If* $\hat{r} = r$*, STOP. Otherwise, GO TO Step (1) of the calling procedure and restart the current trial.*

Note that in step (6) of Procedure (3) features (and samples) can be labeled as *unclustered*, so that in steps (2–4), only *clustered* samples are considered in feature clustering, and likewise in sample clustering.

Algorithm (2) may continue indefinitely due to some features consistently alternating between a small number of clusters. In other words, some features have a tendency to make alternating moves each iteration. Two parameters are introduced to eliminate this possibility: *PercentCutOff* and *BeginFlopCheck*. For each feature, the actual ratio of the number of alternating moves per total number of iterations is calculated for a given trial. *PercentCutOff* specifies the permissible ratio of alternating feature moves per total number of iterations for a given trial. The *BeginFlopCheck* parameter works in conjunction with the *PercentCutOff* parameter by specifying

the number of iterations after which we begin checking for the *PercentCutOff* ratio; this allows for some early shuffling of features between clusters. After reaching *BeginFlopCheck* number of iterations, if the actual ratio is greater than *PercentCutOff*, the feature is discarded from future consideration for the remainder of this trial. These modifications cause rapid convergence for a given trial.

# 7 Computational Experiments

## 7.1 Test Data

Both artificial and real microarray data were used to test the efficacy of our algorithms. The following subsections detail the types of data used to conduct our computational experiments.

### 7.1.1 Trivial Data Sets

We generated a set of "toy" test instances in order to test the accuracy of the main MIP formulation. To create these instances, we varied the values of $m$ and $n$, using $r = 2, 3, 4$. The test instances were constructed with $m$ rows and $n$ columns, with entries formed by repeating the values $\{0, 1, \ldots, r\}$. An example with $m = 6$, $n = 6$, and $r = 3$ is displayed below. The table on the left presents the unsorted data, while the table on the right presents the data sorted according to sample and feature classification. We can easily identify 3 biclusters, each consisting of 2 samples and 2 features.

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 1 | 2 |
| 2 | 0 | 1 | 2 | 0 | 1 |
| 1 | 2 | 0 | 1 | 2 | 0 |
| 0 | 1 | 2 | 0 | 1 | 2 |
| 2 | 0 | 1 | 2 | 0 | 1 |
| 1 | 2 | 0 | 1 | 2 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| **2** | **2** | 0 | 0 | 1 | 1 |
| **2** | **2** | 0 | 0 | 1 | 1 |
| 1 | 1 | **2** | **2** | 0 | 0 |
| 1 | 1 | **2** | **2** | 0 | 0 |
| 0 | 0 | 1 | 1 | **2** | **2** |
| 0 | 0 | 1 | 1 | **2** | **2** |

### 7.1.2 HuGE Data Set

The biclustering algorithms were tested on the Human Gene Expression (HuGE) Index data set [32]. The purpose of the HuGE project is to provide a comprehensive database of gene expressions in normal tissues of different parts of human body and to highlight similarities and differences among the organ systems. We refer the reader to [18] for the detailed description of

these studies. The data set consists of 59 samples from 19 distinct tissue types. It was obtained using oligonucleotide microarrays capturing 7070 genes. The samples were obtained from 49 human individuals: 24 males with median age of 63 and 25 females with median age of 50. Each sample came from a different individual except for the first 7 BRA samples that were from different brain regions of the same individual and the 5th LI sample, which came from that individual as well. The HuGE data set is summarized in Table 1 below.

Table 1: Summary of HuGE Data Set

| **Tissue type** | **Abbreviation** | **# Samples** |
|---|---|---|
| blood | BD | 1 |
| brain | BRA | 11 |
| breast | BRE | 2 |
| colon | CO | 1 |
| cervix | CX | 1 |
| endometrium | END | 2 |
| esophagus | ES | 1 |
| kidney | KI | 6 |
| liver | LI | 6 |
| lung | LU | 6 |
| muscle | MU | 6 |
| myometrium | MYO | 2 |
| ovary | OV | 2 |
| placenta | PL | 2 |
| prostate | PR | 4 |
| spleen | SP | 1 |
| stomach | ST | 1 |
| testes | TE | 1 |
| vulva | VU | 3 |

We made use of the HuGE data set in testing both Algorithms 1 and 2. Test data for our first algorithm was generated by creating subsets of 2 of the 12 total tissue types having multiple samples. For these tissue types, all samples were included in the data set, and all genes were included except for those genes containing at least one negative expression. Subsets of the HuGE data set were also used to test the performance of Algorithm 2. We constructed 10 test data sets of tissue groupings; four sets having three

tissues each, three sets having four tissues each, and three sets having five tissues each. Only those tissues having multiple samples were considered in choosing these groupings. Furthermore, each of these ten tissue groupings were used to make two sets: one set containing all 7070 genes, and another set containing only those genes having no negative expressions across all tissues in that set.

## 7.2 Other Algorithms

We compared our unsupervised biclustering algorithms to three other publicly available biclustering algorithms [33]:

- euclidian co-clustering algorithm (further referred to as **CC-e**) [10];
- minimum squared residue co-clustering algorithm (**CC-r**) [10];
- information theoretic co-clustering algorithm (**CC-i**)[12];

These three algorithms represents recent algorithmic implementations of two classes of biclustering methodologies: (i) algorithms based on minimization of some residue measure (3), or (5), also utilized in [9, 10, 17, 29], and (ii) information theoretic based methods, see [12, 13]. Similar to the algorithms proposed in this paper, these methods are also optimization-based, differing only in the objective function metrics of their respective optimization problems. In Section 2 we provided a brief review of the methods. For a more detailed description we refer the reader to our recent survey on biclustering algorithms [8].

## 7.3 Environments and Parameter Values

Our testing environments are described below, as well as the choices for our parameters in tuning our algorithms.

### 7.3.1 Environments

The algorithmic testing was performed on multiple platforms. Both heuristic algorithms were compiled using Microsoft Visual Studio .NET 2003. They were run on Windows XP with a Pentium 4, 2.4GHz processor and 2GB of RAM. The source code for the three biclustering algorithms (**CC-e**, **CC-i**, and **CC-r**) was compiled with the GNU GCC compiler (version 4.1.2) and run on a Unix platform with an AMD Opteron 240, 1.394GHz speed processor and having 4GB of RAM.

### 7.3.2 Parameter Values

In testing heuristic **H1** we set the parameter of separation $t = 3$ in order to strongly differentiate the resulting clusters. The callable library of CPLEX 9.0 was used to formulate the problem instances as well as to perform the optimization. Additionally, in order to speed up the running time we adjusted some of the CPLEX MIP default settings. Our knowledge of what adjustments to make owes in large part to the *STOP* tool described in [1]; this open source tool determines which parameters work best for the problem formulation at hand. With this knowledge, we set the MIP Emphasis parameter to emphasis *feasibility over optimality*, which encourages CPLEX to focus on finding a feasible integer solution quickly. We also adjusted two cutting plane parameters. The first was the Mixed-Integer Rounding Cuts parameter, which we set to *aggressive*. The second was the Fractional Cuts, which we set to *off*, so that no fractional cuts would be generated by CPLEX.

Regarding branching rules, we chose to have CPLEX implement the *alternative best estimate* option for deciding subsequent nodes on which to branch, and we set the Dive parameter to *traditional dive*. The Variableselect parameter, which decides on which variable we branch, was set to *strong branching*. Additionally, we made use of two of CPLEX's built-in heuristics, namely the RINS heuristic and the node heuristic, to help quickly locate other feasible solutions. Both the RINS and node heuristic parameters were set to activate *every iteration*, and we restricted the number of nodes on which the RINS heuristic could operate to *5000*. Finally, because we believe that sample variables have a higher priority than feature variables in terms of branching strategies, we configured CPLEX to branch on sample variables before branching on feature variables.

In our testing of the **H2** heuristic, we also specified several parameters. In particular, *BeginFlopCheck* was set to 10 iterations, and *PercentCutOff* was set to 0.8. Thus, after 10 iterations, if a feature has alternated more than 80% of the time, that feature would be consider an outlier and no longer considered for this trial. These parameters were chosen using what the authors felt were reasonable values; we wanted to ensure that multiple iterations passed, and that the feature moved around frequently, before considering it an outlier. Moreover, limited testing revealed that the algorithm's convergence was not overly sensitive to these parameters. Ranges of *BeginFlopCheck* parameter from 5 to 15, and from approximately 60% to 95% for *PercentCutOff*, caused the algorithm to converge. Finally, in an effort to provide a more direct comparison with the three coclustering algorithms (**CC-e**, **CC-i**, and **CC-r**), we set the parameter of separation $t$ to 0

for the **H2** heuristic, so that, in effect, our algorithm was not influenced by this parameter.

Table 2: Summary of MIP Performance on Trivial Data Sets

| **m** | **n** | **r** | **seconds** |
|---|---|---|---|
| 4 | 4 | 2 | 0.016 |
| 8 | 8 | 2 | 0.016 |
| 16 | 16 | 2 | 0.047 |
| 32 | 32 | 2 | 0.172 |
| 64 | 64 | 2 | 0.75 |
| 128 | 64 | 2 | 6.735 |
| 128 | 128 | 2 | 2.2 |
| 256 | 256 | 2 | 20.208 |
| 512 | 256 | 2 | 97.545 |
| 1024 | 256 | 2 | 555.395 |
| 2048 | 256 | 2 | 842.359 |
| 4096 | 256 | 2 | 3534.412 |
| 6 | 6 | 3 | 0.093 |
| 12 | 12 | 3 | 1.437 |
| 24 | 24 | 3 | 9.578 |
| 288 | 24 | 3 | 513.794 |
| 576 | 24 | 3 | 751.184 |
| 1152 | 24 | 3 | 1242.25 |
| 2304 | 24 | 3 | 4791.955 |
| 8 | 8 | 4 | 1.485 |
| 16 | 16 | 4 | 37.391 |
| 24 | 24 | 4 | 104.313 |
| 48 | 24 | 4 | 139.891 |
| 96 | 24 | 4 | 1133.384 |

### 7.4 Results

This subsection presents the computational results we found using the aforementioned algorithms and parameters.

### 7.4.1 MIP Results on Trivial Data Sets

In this section we describe our application of the main MIP (23), (21)-(22) and (31)-(35) to some "toy" instances we generated according to section 7.1.1. Every generated "toy" instance was successfully solved to optimality by CPLEX 9.0. The size of the "toy" instances, as well as the time required to solve them to optimality, are reported in Table 2. Unfortunately we were not able to experience the same level of success on subsets of the HuGE dataset using the main MIP in conjunction with CPLEX 9.0, and thus we employed Algorithms 1 and 2 to perform unsupervised biclustering.

However, the results on "toy" instances indicate the potential for utilizing the main MIP for biclustering rather large instances of data with 0–1 values. This situation may appear, for example, in the constraint matrix of linear (integer) programming problems. Then biclustering will correspond to finding an arrow-head structure within these matrices, which could be used for application of decomposition techniques (e.g., Benders', Dantzig-Wolfe) [21].

### 7.4.2 MIP-based Heuristic Results on Subsets of HuGE Data Set

The MIP-based Heuristic **H1** recovered significant biclusters of samples to features with the subsets of HuGE data chosen as in 7.1.2. In order to perform this biclustering, it was helpful to implement a 3 hour time limit for the first iteration (in conjunction with an optimality gap setting of 1%), followed by 2 hour time limits for each subsequent iteration (and no optimality gap). As mentioned previously, the parameter of separation was chosen as $t = 3$; this high setting of $t$ yielded tight clusters.

The developed MIP-based heuristic performs fairly well for $r = 2$ and $n = 20 \sim 30$ samples. Two heatmaps identifying exemplary clusterings with no errors and $r = 2$ are included in Figures 2(a) and 2(b). Figure 2(a) depicts the results of **H1** on the HuGE data subset including all of the liver and brain samples, using only those features with nonnegative values, which gives $m = 1534$ features and $n = 17$ samples. The resulting biclusters are characterized by liver tissue in the first 6 samples and 12 features, while the remaining 11 samples and 46 features are characterized by brain tissue. **H1** and **CC-i** have no errors on this data set, while **CC-e** and **CC-r** have 7 and 3 errors, respectively.

Figure 2(b) displays the results of **H1** on the HuGE data subset including all of the liver and muscle samples, again using only those features with nonnegative values, giving $m = 2097$ features and $n = 12$ samples. **H1**

Table 3: Iterative Heuristics: Test Results on HuGE Data Set. The rightmost four columns contain the number of *misclassifications* per algorithm, followed by percentage of *correct classifications*. A "+" superscript on the data set name indicates that only positive genes from the HuGE data set were used, while no "+" indicates that all genes were included. The "% correct $r$ =" heading contain two percentages for the specified level of $r$: the first is the percent correct for only nonnegative data, while the second is for all data.

| **Organs** | **\| S \|** | **\| F \|** | **Cce** | **Cci** | **CCr** | **H2** |
|---|---|---|---|---|---|---|
| BRA-LI-MU$^+$ | 23 | 1309 | 13 (43.5) | **0 (100)** | 8 (65.2) | **0 (100)** |
| BRA-LI-MU | 23 | 7070 | 12 (47.8) | — | 3 (87.0) | **0 (100)** |
| KI-MYO-VU$^+$ | 11 | 3245 | 4 (63.6) | 3 (72.7) | **0 (100)** | 6 (45.5) |
| KI-MYO-VU | 11 | 7070 | 5 (55.6) | — | **3 (72.7)** | 4 (63.6) |
| BRA-BRE-LU$^+$ | 19 | 1602 | 8 (57.9) | 3 (84.2) | 4 (79.0) | **2 (89.5)** |
| BRA-BRE-LU | 19 | 7070 | 7 (63.2) | — | 8 (57.9) | **4 (79.0)** |
| KI-MU-OV$^+$ | 14 | 2440 | 7 (50.0) | **3 (78.6)** | 7 (50.0) | **3 (78.6)** |
| KI-MU-OV | 14 | 7070 | 6 (57.1) | — | 5 (64.3) | **0 (100)** |
| **% Correct** $r = 3$ | | | 52.2/53.7 | **86.6**/— | 71.6/71.6 | 83.6/**85.8** |
| BRA-BRE-KI-MU$^+$ | 25 | 1389 | 10 (60.0) | 2 (92.0) | 8 (68.0) | **0 (100)** |
| BRA-BRE-KI-MU | 25 | 7070 | 9 (64.0) | — | 11 (56.0) | **4 (84.0)** |
| LI-LU-PR-VU$^+$ | 19 | 2301 | 11 (42.1) | **4 (79.0)** | 7 (63.2) | 6 (68.4) |
| LI-LU-PR-VU | 19 | 7070 | 11 (42.1) | — | 8 (57.9) | **6 (68.4)** |
| BRE-KI-PL-PR$^+$ | 14 | 2564 | 8 (42.9) | 6 (57.1) | 6 (57.1) | **2 (85.7)** |
| BRE-KI-PL-PR | 14 | 7070 | 7 (50.0) | — | 6 (57.1) | **4 (71.4)** |
| **% Correct** $r = 4$ | | | 50.0/51.7 | 79.3/— | 63.8/60.3 | **86.2/81.0** |
| BRA-BRE-END-LI-LU$^+$ | 27 | 1377 | 16 (40.7) | 13 (51.9) | 12 (55.6) | **9 (66.7)** |
| BRA-BRE-END-LI-LU | 27 | 7070 | 12 (55.6) | — | 9 (66.7) | **8 (70.4)** |
| BRA-KI-MU-MYO-VU$^+$ | 28 | 1429 | 16 (42.9) | **4 (85.7)** | 9 (67.9) | 12 (57.1) |
| BRA-KI-MU-MYO-VU | 28 | 7070 | 13 (53.6) | — | **7 (75.0)** | 8 (71.4) |
| KI-LU-MU-PL-PR$^+$ | 24 | 2048 | 13 (45.8) | 12 (50.0) | 9 (62.5) | **7 (70.8)** |
| KI-LU-MU-PL-PR | 24 | 7070 | 12 (50.0) | — | 9 (62.5) | **2 (91.7)** |
| **% Correct** $r = 5$ | | | 43.0/48.1 | 63.3/— | 62.0/65.2 | **64.6/70.9** |
| **Overall % Correct** | | | 48.0/51.0 | 75.5/— | 65.7/65.9 | **77.0/78.7** |

identified 6 samples and 24 features representing liver tissue, and 6 samples and 35 features representing muscle tissue. As in the previous data set, **H1** performed perfectly, while **CC-e**, **CC-i** **CC-r** have 6, 3 and 3 errors, respectively.

Unfortunately, the performance of the algorithm **H1** substantially declined when $r > 2$.

### 7.4.3 Iterative Heuristic Results on Subsets of HuGE Data Set

As described in section 7.1.2, we created subsets of tissue groupings from the HuGE data set to test the performance of **H2**. Thus, after obtaining

these results, we compared its performance with those of publicly available biclustering algorithms (see section 7.2). The computational results of these algorithms have been summarized in Table 3. The rightmost four columns detail the performance of each of the four algorithms.

From Table 3 we can see that in 13 of the 20 data sets, **H2** had the highest percentage of correct classifications, while in another 2 data sets, **H2** tied for the highest percentage. Only in 5 of the 20 data sets was **H2** outperformed, and of these, only once (KI-MYO-VU without negatives) was it outperformed by all of the three other algorithms. As noted previously, the **CC-i** algorithm only operates on nonnegative data, and so its performance could not be measured on data sets containing negative entries.

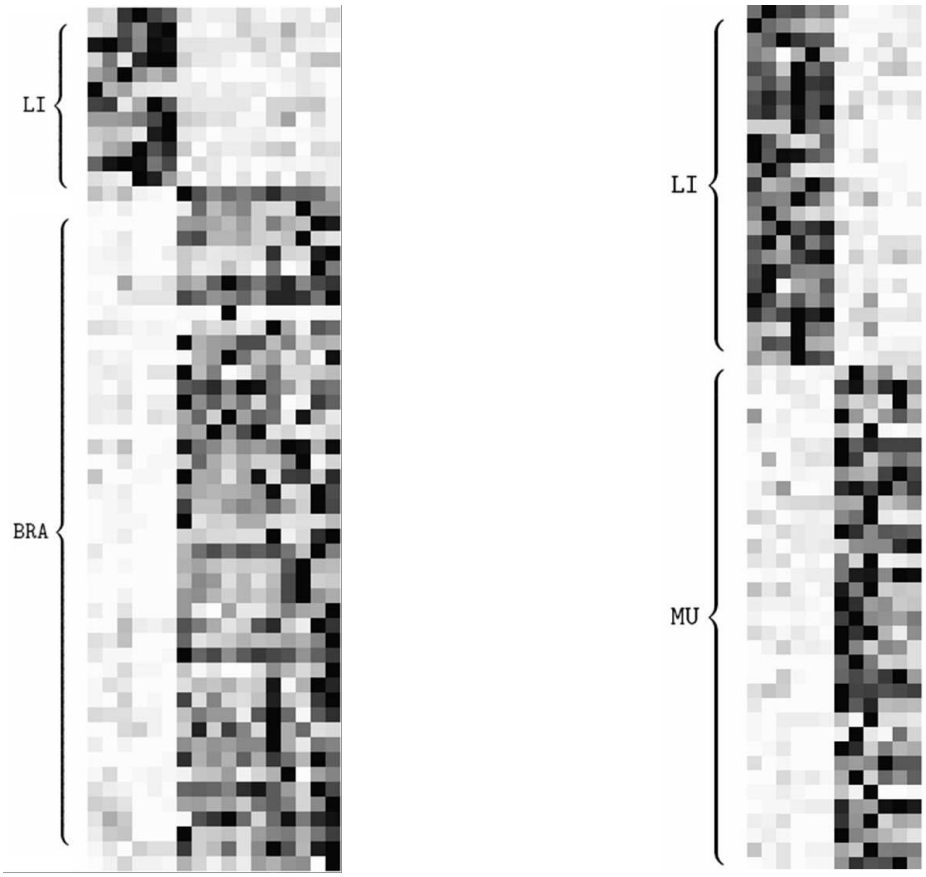
Further performance measures are evidenced in the rows with the "% correct $r$ =" heading. For each value of $r$ ($r = 3$, $r = 4$, and $r = 5$, as well as for the overall case), the percentage of correct classifications for each heuristic was calculated. The first percentage reported is the performance on non-negative data, while the second percentage is the performance on all data. Among the four algorithms, **H2** had the highest percentage of correct classifications on 7 of the 8 measures, including both of the overall measures.

Figures 3(a) and 3(b) are heatmaps of exemplary biclusterings we found.

# 8 Concluding Remarks

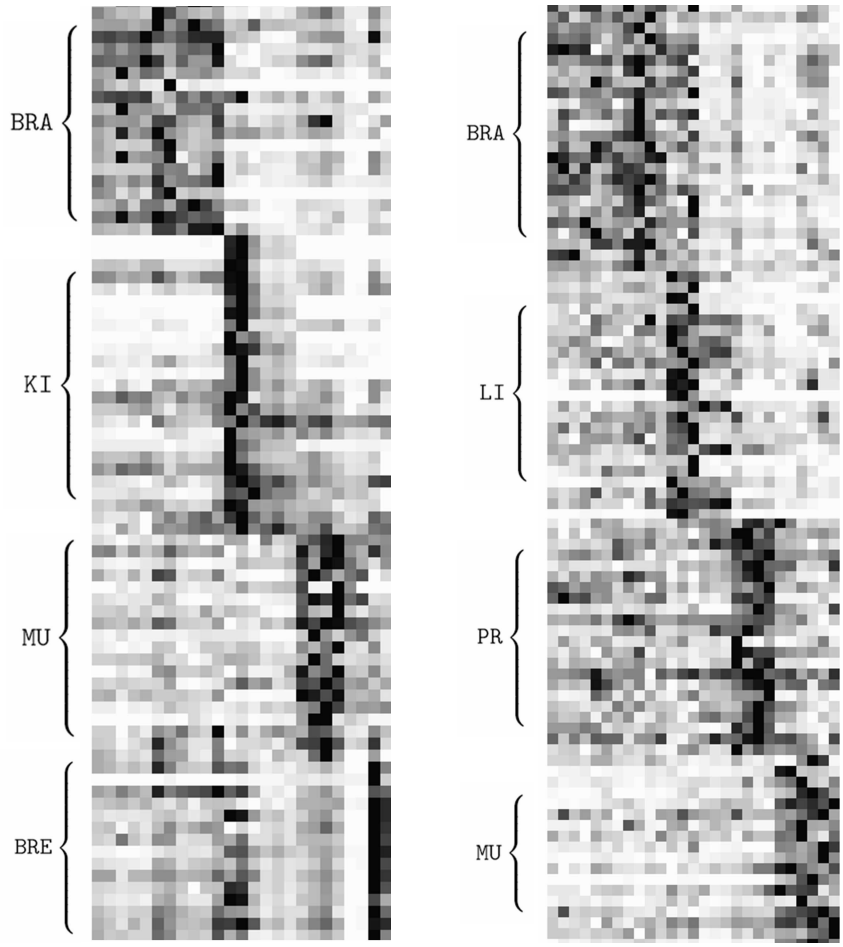
We have extended our previous work on supervised biclustering for the case of unsupervised learning. The key advantage of the proposed techniques is that in contrast to other biclustering schemes, consistent biclustering is justified by the conic separation property [7]. We discussed the computational complexity of consistent biclustering proving that the problem of finding the largest conditionally biclustering admitting submatrix is $NP$-hard. We formulated and presented a fractional 0–1 program for unsupervised biclustering under the biclustering consistency conditions defined in [7]. Furthermore, as the fractional 0–1 program contains non-linear constraints, we presented an equivalent linear mixed 0–1 programming reformulation. We then discussed two heuristic algorithms to conduct unsupervised biclustering, one involving iterative resolving of a specific relaxation of the original linear mixed 0–1 programming problem, while the other is an efficient local search based heuristic which compared favorably when tested against three other biclustering algorithms from the literature.

It is important to note that in some of the biclustering methodologies a direct one to one correspondence between clusters of samples and clusters




(a) (b)

Figure 2: Heatmaps with (a) results of **H1** on 11 BRA and 6 LI samples; (b) results of **H1** on 6 LI and 6 MU.




(a) (b)

Figure 3: Heatmaps with (a) results of **H2** on 11 BRA, 2 BRE, 6 KI, 6 MU samples; (b) results of **H2** on 11 BRA, 6 LI, 6 MU, 4 PR samples.

of features is not required. Moreover, the numbers of sample and feature clusters are allowed to be different. This way we may consider not only pairs $(\mathcal{S}_k, \mathcal{F}_k)$, but also other pairs $(\mathcal{S}_k, \mathcal{F}_\ell)$, $k \neq \ell$. Another possible generalization is to allow *overlapping* of co-clusters, i.e., to consider so-called *soft* biclustering. In the future, we plan to extend our work to generate overlapping biclusters and soft biclustering that allows weighted memberships in multiple biclusters.

## 9 Acknowledgments

The authors would like to thank both referees for their comments and suggestions that have greatly improved the quality of the paper.

## References

[1] M. Baz, B. Hunsaker, J.P. Brooks, A. Gosavi, Automated Tuning of Optimization Parameters, Technical Report, University of Pittsburgh, 2007.

[2] A. Ben-Dor, L. Bruhn, I. Nachman, M. Schummer, and Z. Yakhini, Tissue classification with gene expression profiles, *Journal of Computational Biology*, 7 (2000), pp. 559–584.

[3] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, Discovering Local Structure in Gene Expression Data: the Order-Preserving Submatrix Problem, *Journal of Computational Biology*, 10 (2000), pp. 373–384.

[4] A. Ben-Dor, N. Friedman, and Z. Yakhini, Class discovery in gene expression data, *Procceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB)* (2001).

[5] S. Busygin, G. Jacobsen, E. Krämer, Double Conjugated Clustering Applied to Leukemia Microarray Data, *SDM 2002 Workshop on Clustering High Dimensional Data and its Applications*, 2002.

[6] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[7] S. Busygin, O.A. Prokopyev, P.M. Pardalos, Feature Selection for Consistent Biclustering via Fractional 0–1 Programming, *Journal of Combinatorial Optimization*, Vol. 10/1 (2005), pp. 7–21.

[8] S. Busygin, O.A. Prokopyev, P.M. Pardalos, Biclustering in Data Mining, *Computers & Operations Research*, 35 (2008), pp. 2964–2987.

[9] Y. Cheng, G.M. Church, Biclustering of Expression Data, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (2000), pp. 93–103.

[10] H. Cho, I.S. Dhillon, Y. Guan, S. Sra, Minimum Sum-Squared Residue Co-clustering of Gene Expression Data, *Proceedings of the 4th SIAM International Conference on Data Mining(SDM)*, April 22-24, 2004, Lake Buena Vista, Florida.

[11] I.S. Dhillon, Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, August 26-29, 2001, San Francisco, CA.

[12] I.S. Dhillon, Y. Guan, Information Theoretic Clustering of Sparse Co-Occurrence Data, *Proceedings of the Third IEEE International Conference on Data Mining (ICDM03)*, November 19-22, 2003, Melbourne, FL.

[13] I.S. Dhillon, S. Mallela, D. Modha, Information-Theoretic Co-clustering, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 24-27, 2003, Washington, DC.

[14] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of $NP$-Completeness, W.H. Freeman, San Francisco, 1979.

[15] T.R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, *Science*, 286 (1999), pp. 531–537.

[16] P. Hansen, M. Poggi de Aragão, C.C. Ribeiro, Hyperbolic 0–1 programming and query optimization in information retrieval, *Mathematical Programming*, 52 (1991), pp. 256–263.

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

[18] L-L. Hsiao, Dangond, F., Yoshida, T., Hong, R., Jensen, RV., Misra, J., Dillon, W., Lee, KF., Clark, KE., Haverty, P., Weng, Z., Mutter, G., Frosch, MP., MacDonald, ME., Milford, EL., Crum, CP., Bueno, R., Pratt, RE., Mahadevappa, M., Warrington, JA., Stephanopoulos, G., Stephanopoulos, G., and Gullans, SR, A Compendium of Gene Expression in Normal Human Tissues, *Physiol. Genomics*, 7 (2001), pp. 97–104.

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

[20] S. Madeira, A. L. Oliviera, Biclustering Algorithms for Biological Data Analysis: A Survey, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1, (2004) 24–45.

[21] R. Martin, Large Scale Linear and Integer Optimization: A Unified Approach, Kluwer Academic Publishers, Boston/Dordrecht/London, 1999.

[22] O. A. Prokopyev, H.-X. Huang, P.M. Pardalos, On complexity of unconstrained hyperbolic 0–1 programming problems, *Operations Research Letters*, 33 (2005), pp. 312–318.

[23] O.A. Prokopyev, C. Meneses, C.A.S. Oliveira, P.M. Pardalos, On Multiple-Ratio Hyperbolic 0–1 Programming Problems, *Pacific Journal of Optimization*, Vol. 1/2 (2005), pp. 327–345.

[24] S. Saipe, Solving a (0,1) hyperbolic program by branch and bound, *Naval Research Logistics Quarterly*, 22 (1975), pp. 497–515.

[25] M. Tawarmalani, S. Ahmed, N. Sahinidis, Global Optimization of 0–1 Hyperbolic Programs, *Journal of Global Optimization*, 24 (2002), pp. 385–416.

[26] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, NIPS (2001).

[27] T.-H. Wu, A note on a global approach for general 0–1 fractional programming, *European Journal Operational Research*, 101 (1997), pp. 220–223.

[28] E.P. Xing, R.M. Karp, CLIFF: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts, *Bioinformatics Discovery Note*, 1 (2001), pp. 1–9.

[29] J. Yang, W. Wang, H. Wang, P. Yu, $\delta$-clusters: Capturing Subspace Correlation in a Large Data Set, *Proceedings of the 18th IEEE International Conference on Data Engineering* (2002), pp. 517–528.

[30] National Human Genome Research Institute, DNA Microarray Fact Sheet, `http://www.genome.gov/page.cfm?pageID=10000533#2`, last accessed January 2008.

[31] CAMDA 2001 Conference.
`http://bioinformatics.duke.edu/camda/camda01/`.

[32] HuGE Index.org Website.
`http://www.hugeindex.org`.

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

[34] ILOG Inc. CPLEX 9.0 User's Manual, 2004.