Biclustering of Expression Data

Yizong Cheng[‡§*] and George M. Church[‡†]
[‡]Department of Genetics, Harvard Medical School, Boston, MA 02115
[§]Department of ECECS, University of Cincinnati, Cincinnati, OH 45221
yizong.cheng@uc.edu, church@salt2.med.harvard.edu

Abstract

An efficient node-deletion algorithm is introduced to find submatrices in expression data that have low mean squared residue scores and it is shown to perform well in finding co-regulation patterns in yeast and human. This introduces "biclustering", or simultaneous clustering of both genes and conditions, to knowledge discovery from expression data. This approach overcomes some problems associated with traditional clustering methods, by allowing automatic discovery of similarity based on a subset of attributes, simultaneous clustering of genes and conditions, and overlapped grouping that provides a better representation for genes with multiple functions or regulated by many factors.

Keywords: microarray, gene expression pattern, clustering

Introduction

Gene expression data are being generated by DNA chips and other microarray techniques and they are often presented as matrices of expression levels of genes under different conditions (including environments, individuals, and tissues). One of the usual goals in expression data analysis is to group genes according to their expression under multiple conditions, or to group conditions based on the expression of a number of genes. This may lead to discovery of regulatory patterns or condition similarities.

The current practice is often the application of some agglomerative or divisive clustering algorithm that partitions the genes or conditions into mutually exclusive groups or hierarchies. The basis for clustering is often the similarity between genes or conditions as a function of the rows or columns in the expression matrix. The similarity between rows is often a function of the row vectors involved and that between columns a function of the column vectors. Functions that have been used include Euclidean distance (or related coefficient of correlation and Gaussian similarity) and the dot product (or a nonlinear generalization of it, as used in kernel methods) between the vectors. All conditions are given equal weights in the computation of gene similarity and vice versa. One must doubt not only the rationale of equally weighing all conditions or all genes, but that of giving the same weight to the same condition for the similarity computation between all the genes and vice versa as well. Any such formula leads to the discovery of some similarity groups at the expense of obscuring some other similarity groups.

In expression data analysis, beyond grouping genes and conditions based on overall similarity, it is sometimes needed to salvage information lost during oversimplied similarity and grouping computation. One of the goals of doing so is to disclose the involvement of a gene or a condition in multiple pathways, some of which can only be discovered under the dominance of more consistent ones.

In this article, we introduce the concept of *bicluster*, corresponding to a subset of genes and a subset of conditions with a high similarity score. Similarity is not treated as an function of pairs of genes or pairs of conditions. Instead, it is a measure of the coherence of the genes and conditions in the bicluster. This measure can be a symmetric function of the genes and conditions involved and thus the finding of biclusters is a process that groups genes and conditions simultaneously. If we project these biclusters onto the dimension of genes or that of conditions, then we can see the result as clustering of either genes or conditions, into possibly overlapping groups.

A particular score that applies to expression data transformed by a logarithm and augmented by the additive inverse is the *mean squared residue*. The *residue* of element a_{ij} in the bicluster indicated by the subsets I and J is

$$a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}, \tag{1}$$

where a_{iJ} is the mean of the ith row in the bicluster, a_{Ij} the mean of the jth column in the bicluster, and a_{IJ} that of all elements in the bicluster. The mean squared residue is the variance of the set of all elements in the bicluster, plus the mean row variance and the mean column variance. We want to find biclusters with low mean squared residue, in particular, large and maximal

*Tel: (513) 556-1809, Fax: (513) 556-7326.
†Tel: (617) 432-7266, Fax: (617) 432-7663.

ones with scores below a certain threshold.

A special case for a perfect score (a zero mean squared residue) is a constant bicluster of elements of a single value. When a bicluster has a non-zero score, it is always possible to remove a row or a column to lower the score, until the remaining bicluster becomes constant.

The problem of finding a maximum bicluster with a score lower than a threshold includes the problem of finding a maximum biclique (complete bipartite subgraph) in a bipartite graph as a special case. If a maximum biclique is one that maximizes the number of vertices involved (maximizing $|I| + |J|$), then the problem is equivalent to finding a maximum matching in the bipartite complement and can be solved using polynomial time max-flow algorithms. However, this approach often results in a submatrix with maximum perimeter and zero area, particularly in the case of expression data, where the number of genes may be hundreds times more than the number of conditions.

If the goal is to find the largest balanced biclique, for example, the largest constant square submatrix, then the problem is proven to be NP-hard (Johnson, 1987). On the other hand, the hardness of finding one with the maximum area is still unknown.

Divisive algorithms for partitioning data into sets with approximately constant values have been proposed by Morgan and Sonquist (1963) and Hartigan (1972). The result is an hierarchy of clusters, and the algorithms foretold the more recent decision tree procedures. Hartigan (1972) also mentioned that the criterion for partitioning may be other than a constant value, for example, a two-way analysis of variance model, which is quite similar to the mean squared residue scoring proposed in this article. Rather than a divisive algorithm, our approach is more of the type of node deletion (Yannakakis, 1981).

The term *biclustering* has been used by Mirkin (1996) to describe "simultaneous clustering of both row and column sets in a data matrix". Other terms that have been associated to the same idea include "direct clustering" (Hartigan, 1972) and "box clustering" (Mirkin, 1996). Mirkin (1996) presents a node addition algorithm, starting with a single cell in the matrix, to find a maximal constant bicluster.

The algorithms mentioned above either find one constant bicluster, or find a set of mutually exclusive near-constant biclusters that cover the data matrix. There are ample reasons to allow biclusters to overlap in expression data analysis. One of the reasons is that a single gene may participate in multiple pathways that may or may not be co-active under all conditions. The problem of finding a minimum set of biclusters, either mutually exclusive or overlapping, to cover all the elements in a data matrix is a generalization of the problem of covering a bipartite graph by a minimum set of bicliques, either mutually exclusive or overlapping, which has been shown to be NP-hard (Orlin, 1977). Nau, Markowsky, Woodbury, and Amos (1978) had an interesting application of biclique covering on the interpretation of leukocyte-serum immunological reaction matrices, which are not unlike the gene-condition expression matrices.

In expression data analysis, the uttermost important goal may not be finding the maximum bicluster or even finding a bicluster cover for the data matrix. More interesting is the finding of a set of genes showing strikingly similar up-regulation and down-regulation under a set of conditions. A low mean squared residue score plus a large variation from the constant may be a good criterion for identifying these genes and conditions.

In the following sections, we present a set of efficient algorithms that find these interesting gene and condition sets. The basic iterate in the method consists of the steps of masking null values and biclusters that have been discovered, coarse and fine node deletion, node addition, and the inclusion of inverted data.

Methods

A gene-condition expression matrix is a matrix of real numbers, with possible null values as some of the elements. Each element represents the logarithm of the relative abundance of the mRNA of a gene under a specific condition. The logarithm transformation is used to convert doubling or other multiplicative changes of the relative abundance into additive increments.

Definition 1. Let X be the set of genes and Y the set of conditions. Let a_{ij} be the element of the expression matrix A representing the logarithm of the relative abundance of the mRNA of the ith gene under the jth condition. Let $I \subset X$ and $J \subset Y$ be subsets of genes and conditions. The pair (I, J) specifies a submatrix A_{IJ} with the following *mean squared residue* score.

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

where

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \quad a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}, \quad (3)$$

and

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} = \frac{1}{|I|} \sum_{i \in I} a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{Ij} \quad (4)$$

are the row and column means and the mean in the submatrix (I, J). A submatrix A_{IJ} is called a *δ-bicluster* if $H(I, J) \leq \delta$ for some $\delta \geq 0$. □

The lowest score $H(I, J) = 0$ indicates that the gene expression levels fluctuate in unison. This includes the trivial or constant biclusters where there is no fluctuation. These trivial biclusters may not be very interesting but need to be discovered and masked so more interesting ones can be found. The *row variance* may be an accompanying score to reject trivial biclusters.

$$V(I,J) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{Ij})^2. \quad (5)$$

The matrix $a_{ij} = ij, i, j > 0$ has the property that no submatrix of a size larger than a single cell has a score lower than 0.5. A $K \times K$ matrix of all 0s except one 1 has the score

$$h_K = \frac{1}{K^6}(K-1)\left[(K-1)^3 + 2(K-1)^2 - 2\right]. \quad (6)$$

A matrix with elements randomly and uniformly generated in the range of $[a, b]$ has an expected score of $(b-a)^2/12$. This result is independent of the size of the matrix. For example, when the range is $[0, 800]$, the expected score is 53,333.

A translation (addition by a constant) to the matrix will not affect the $H(I, J)$ score. A scaling (multiplication by a constant) will affect the score (by the square of the constant), but will have no affect if the score is zero. Neither translation nor scaling affects the ranking of the biclusters in a matrix.

Theorem 1. The problem of finding the largest square δ-bicluster ($|I| = |J|$) is NP-hard.

Proof. We construct a reduction from the BALANCED COMPLETE BIPARTITE SUBGRAPH problem (GT24 in Garey and Johnson, 1979) to this problem. Given a bipartite graph (V_1, V_2, E) and a positive integer K, form a real-valued matrix A with $a_{ij} = 0$ if and only if $(i, j) \in E$ and $a_{ij} = 2ij$ otherwise, for $i, j > 0$. If the largest square $1/h_K$-bicluster in A has a size larger than or equal to K, then there is a $K \times K$ biclique (complete bipartite subgraph). Since the BALANCED COMPLETE BIPARTITE SUBGRAPH problem is NP-complete, the problem of this theorem is NP-hard. □

Node Deletion

Every expression matrix contains a submatrix with the perfect score ($H(I, J) = 0$), and any single element is such a submatrix. Certainly the kind of biclusters we look for should have a maximum size, both in terms of the number of genes involved ($|I|$) and in terms of the number of conditions ($|J|$).

If we start with a large matrix, say, the one with all the data, then the question is how to select a submatrix with a low H score. A greedy method is to remove the row or column to achieve the largest decrease of the score. This requires the computation of the scores of all the submatrices that may be the consequences of any row or column removal, before each choice of removal can be made. This method (Algorithm 0) requires time in $O((n+m)nm)$, where n and m are the row and column sizes of the expression matrix, to find one bicluster.

Algorithm 0 (Brute-Force Deletion and Addition).

Input: A, a matrix of real numbers, and $\delta \geq 0$, the maximum acceptable mean squared residue score.

Output: A_{IJ}, a δ-bicluster that is a submatrix of A with row set I and column set J, with a score no larger than δ.

Initialization: I and J are initialized to the gene and condition sets in the data and $A_{IJ} = A$.

Iteration:

1. Compute the score H for each possible row/column addition/deletion and choose the action that decreases H the most. If no action will decrease H, or if $H <= \delta$, return A_{IJ}.

Algorithm 0, although a polynomial-time one, will not be efficient enough for a quick analysis of most expression data matrices. We propose in the following Algorithm 1 with time complexity in $O(nm)$ and Algorithm 2 in $O(m \log n)$. The combination of the two will provide a very efficient node-deletion algorithm for finding a bicluster of a low score. The correctness and efficiency of these algorithms are based on a number of lemmas, in which rows (or columns) are treated as points in a space where a distance is defined.

Lemma 1. Let S be a finite set of points in a space in which a non-negative real-valued function of two arguments, d is defined. Let $m(S)$ be a point that minimizes the function

$$f(s) = \sum_{x \in S} d(x, s). \quad (7)$$

Define the measure

$$E(S) = \frac{1}{|S|}\sum_{x \in S} d(x, m(S)). \quad (8)$$

Then, the removal of any non-empty subset

$$R \subset \{x \in S : d(x, m(S)) > E(S)\} \quad (9)$$

will only make

$$E(S - R) < E(S). \quad (10)$$

Proof. Condition (10) can be rewritten as

$$\frac{A'}{|S-R|} < \frac{A+B}{|S|}, \quad (11)$$

where

$$A = \sum_{x \in S-R} d(x, m(S)),\ A' = \sum_{x \in S-R} d(x, m(S-R)), \quad (12)$$

$$B = \sum_{x \in R} d(x, m(S)). \quad (13)$$

The definition of the function m requires that $A' \leq A$. Thus, a sufficient condition for the inequality (11) is

$$\frac{A}{|S-R|} < \frac{A+B}{|S|}, \tag{14}$$

which is equivalent to

$$E(S) = \frac{A+B}{|S|} < \frac{B}{|R|} = \frac{1}{|R|}\sum_{x\in R} d(x, m(S)). \tag{15}$$

Clearly, (9) is a sufficient condition for this inequality and therefore also for (10). □

Lemma 2. Suppose the set removed from S is

$$R \subset \{x \in S : d(x, m(S)) > \alpha E(S)\} \tag{16}$$

with $\alpha \geq 1$. Then the reduction rate of the score $E(S)$ can be characterized as

$$\frac{E(S) - E(S-R)}{E(S)} > \frac{\alpha - 1}{|S|/|R| - 1}. \tag{17}$$

When a single point x is removed, the reduction rate has the bound

$$E(S) - E(S-R) > \frac{d(x, m(S)) - E(S)}{|S| - 1}. \tag{18}$$

Proof. Using notation in Lemma 1, we have now

$$\alpha E(S) = \alpha\frac{A+B}{|S|} < \frac{B}{|R|} = \frac{1}{|R|}\sum_{x\in R} d(x, m(S)). \tag{19}$$

This leads to

$$\alpha|R|A < (|S| - \alpha|R|)B, \tag{20}$$

or, equivalently,

$$|S|A < (|S| - \alpha|R|)(A+B). \tag{21}$$

This is the same as

$$\frac{A}{|S-R|} < \frac{|S| - \alpha|R|}{|S-R|}\frac{A+B}{|S|}. \tag{22}$$

Using the inequality $A' \leq A$ and the facts that $E(S-R) = A'/|S-R|$ and $E(S) = (A+B)/|S|$, this leads to the inequality

$$E(S-R) < \frac{|S| - \alpha|R|}{|S-R|}E(S), \tag{23}$$

which is (17). Inequality (18) can be derived from (17). □

Theorem 2. The set of rows that can be completely or partially removed with the net effect of decreasing the score of a bicluster A_{IJ} is

$$R = \left\{ i \in I; \frac{1}{|J|}\sum_{j\in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 > H(I,J) \right\} \tag{24}$$

Proof. Let the points in Lemma 1 be $|J|$-dimensional real-valued vectors and S be the set of vectors b_i with components $b_{ij} = a_{ij} - a_{iJ}$ for $i \in I$ and $j \in J$. The function d is defined as

$$d(b_i, b_k) = \sum_{j\in J}(b_{ij} - b_{kj})^2. \tag{25}$$

In this case,

$$m(S) = \frac{1}{|I|}\sum_{i\in I} b_i \tag{26}$$

and has the components $a_{Ij} - a_{IJ}$. □

There is also a similar result for the columns.

Lemma 2 acts as a guide on the trade-off between two types of node deletion, that of deleting one node a time, and that of deleting a set of node a time, before the score is recalculated. These two algorithms are listed below.

Algorithm 1 (Single Node Deletion).

Input: A, a matrix of real numbers, and $\delta \geq 0$, the maximum acceptable mean squared residue score.

Output: A_{IJ}, a δ-bicluster that is a submatrix of A with row set I and column set J, with a score no larger than δ.

Initialization: I and J are initialized to the gene and condition sets in the data and $A_{IJ} = A$.

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{Ij} for all $j \in J$, a_{IJ}, and $H(I,J)$. If $H(I,J) <= \delta$, return A_{IJ}.
2. Find the row $i \in I$ with the largest

$$d(i) = \frac{1}{|J|}\sum_{j\in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

and the column $j \in J$ with the largest

$$d(j) = \frac{1}{|I|}\sum_{i\in I}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

remove the row or column whichever with the larger d value by updating either I or J.

The correctness of Algorithm 1 is shown by Theorem 2, in the sense that every removal decreases the score. Because there are only finite number of rows and columns to remove, the algorithm terminates in no more than $n+m$ iterates, where n and m are the number of genes and the number of conditions in the initial data matrix. However, it may happen that all $d(i)$ and $d(j)$ are equal to $H(I,J)$ for $i \in I$ and $j \in J$ and hence Theorem 2 does not apply. In this case, the removal

of one of them may still decrease the score, unless the score is already zero.

Step 1 in each iterate requires time in $O(nm)$ and a complete recalculation of all d values in Step 2 is also an $O(nm)$ effort. The selection of the best row and column candidates takes $O(\log n + \log m)$ time. When the matrix is bi-level, specifying "on" and "off" of the genes, the update of various variables after the removal of a row takes only $O(m)$ time and that after the removal of a column only $O(n)$ time. In this case, the algorithm can be made very efficient even for whole genome expression data, with overall running time in $O(nm)$. But, for non-bi-level matrices, updates are more expensive and it is advisable to use the following Multiple Node Deletion before the matrix is reduced to a manageable size, when Single Node Deletion is appropriate.

Algorithm 2 (Multiple Node Deletion).

Input: A, a matrix of real numbers, $\delta \geq 0$, the maximum acceptable mean squared residue score, and $\alpha > 1$, a threshold for multiple node deletion.

Output: A_{IJ}, a δ-bicluster that is a submatrix of A with row set I and column set J, with a score no larger than δ.

Initialization: I and J are initialized to the gene and condition sets in the data and $A_{IJ} = A$.

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{Ij} for all $j \in J$, a_{IJ}, and $H(I, J)$. If $H(I, J) <= \delta$, return A_{IJ}.
2. Remove the rows $i \in I$ with
$$\frac{1}{|J|}\sum_{j\in J}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2 > \alpha H(I,J)$$
3. Recompute a_{Ij}, a_{IJ}, and $H(I, J)$.
4. Remove the columns $j \in J$ with
$$\frac{1}{|I|}\sum_{i\in I}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2 > \alpha H(I,J)$$
5. If nothing has been removed in the iterate, switch to Algorithm 1.

The correctness of Algorithm 2 is guaranteed by Lemma 2. When α is properly selected, the Multiple Node Deletion phase of Algorithm 2 (before the call of Algorithm 1) requires a number of iterates in $O(\log n + \log m)$, which is usually extremely fast. Without updating the score after the removal of each node, the matrix may shrink too much and one may miss some large δ-biclusters (although later runs of the same algorithm may find them). One may also choose an adaptive α based on the score and size during the iteration.

Node Addition

After node deletion, the resulting δ-bicluster may not be maximal, in the sense that some rows and columns may be added without increasing the score. Lemma 3 and Theorem 3 below mirror Lemma 1 and Theorem 2 and provide a guideline for node addition.

Lemma 3. Let S, d, $m(S)$, and $E(S)$ be defined as same as those in Lemma 1. Then, the addition to S of any non-empty subset

$$R \subset \{x \notin S : d(x, m(S)) \leq E(S)\} \tag{27}$$

will not increase the score E:

$$E(S + R) \leq E(S). \tag{28}$$

Proof. The condition (28) can be rewritten as

$$\frac{A'}{|S+R|} \leq \frac{A-B}{|S|}, \tag{29}$$

where

$$A = \sum_{x\in S+R} d(x, m(S)),\ A' = \sum_{x\in S+R} d(x, m(S+R)), \tag{30}$$

$$B = \sum_{x\in R} d(x, m(S)). \tag{31}$$

The definition of the function m requires that $A' \leq A$. Thus, a sufficient condition for the inequality (29) is

$$\frac{A}{|S+R|} \leq \frac{A-B}{|S|}, \tag{32}$$

which is equivalent to

$$E(S) = \frac{A-B}{|S|} \geq \frac{B}{|R|} = \frac{1}{|R|}\sum_{x\in R} d(x, m(S)). \tag{33}$$

Clearly, (27) is a sufficient condition for this inequality and therefore also for (28). □

Theorem 3. The set of rows that can be completely or partially added with the net effect of decreasing the score of a bicluster A_{IJ} is

$$R = \left\{ i \notin I; \frac{1}{|J|}\sum_{j\in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I,J) \right\} \tag{34}$$

Proof. Similar to the proof of Theorem 2. □

There is also a similar result for the columns.

Algorithm 3 (Node Addition).

Input: A, a matrix of real numbers, I and J signifying a δ-bicluster.

Output: I' and J' such that $I \subset I'$ and $J \subset J'$ with the property that $H(I', J') \leq H(I, J)$.

Iteration:

1. Compute a_{iJ} for all i, a_{Ij} for all j, a_{IJ}, and $H(I, J)$.
2. Add the columns $j \notin J$ with
$$\frac{1}{|I|}\sum_{i \in I}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$
3. Recompute a_{iJ}, a_{IJ}, and $H(I, J)$.
4. Add the rows $i \notin I$ with
$$\frac{1}{|J|}\sum_{j \in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$
5. For each row i still not in I, add its inverse if
$$\frac{1}{|J|}\sum_{j \in J}(-a_{ij} + a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I, J)$$
6. If nothing is added in the iterate, return the final I and J as I' and J'.

Lemma 3 and Theorem 3 guarantee the addition of rows and columns in Algorithm 3 will not increase the score. However, the resulting δ-bicluster may still not be maximal because of two reasons. The first is that Lemma 3 only gives a sufficient condition for adding rows and columns and it is not necessarily a necessary condition. The second reason is that by adding rows and columns, the score may decrease to the point it is much smaller than δ. Each iterate in Algorithm 3 only adds rows and columns according to the current score, not δ.

Step 5 in the iteration adds inverted rows into the bicluster. These rows form "mirror images" of the rest of the rows in the bicluster and can be interpreted as co-regulated but receiving the opposite regulation. These inverted rows cannot be added to the data matrix at the beginning, because that would make all $a_{Ij} = 0$ and also $a_{IJ} = 0$.

Algorithm 3 is very efficient. Its time efficiency is comparable with the Multiple Node Deletion phase of Algorithm 2 and in the order of $O(mn)$.

Clearly, addition of nodes does not have to take place after all deletion is done. Sometimes an addition may decrease the score more than any deletion. A single node deletion and addition algorithm based on the Lemmas and thus more efficient than Algorithm 0 is possible to set up.

Experimental Methods

The biclustering algorithms were tested on two sets of expression data, both having been clustered using conventional clustering algorithms. The yeast *Saccharomyces cerevisiae* cell cycle expression data from Cho *et al.* (1998) and the human B-cells expression data from Alizadeh *et al.* (2000)were used.

Data Preparation

The yeast data contain 2,884 genes and 17 conditions. These genes were selected according to Tavazoie *et al.* (1999). The genes were identified by their SGD ORF names (Ball *et al.*, 2000) from http://arep.med.harvard.edu/network_discovery. The relative abundance values (percentage of the mRNA for the gene in all mRNAs) were taken from a table prepared by Aach, Rindone, and Church (2000). (Two of the ORF names did not have corresponding entries in the table and thus there were 34 null elements.) These numbers were transformed by scaling and logarithm $x \rightarrow 100\log(10^5 x)$ and the result was a matrix of integers in the range between 0 and 600. (The transformation does not affect the values 0 and -1 (null element).)

The human data was downloaded from the Web site for supplementary information for the article by Alizadeh *et al.* (2000). There were 4,026 genes and 96 conditions. The expression levels were reported as log ratios and after a scaling by a factor of 100, we ended up with a matrix of integers in the range between -750 and 650, with 47,639 missing values (12.3% of the matrix elements).

The matrices after the above preparation, along with the biclustering results can be found at http://arep.med.harvard.edu/biclustering.

Missing Data Replacement

Missing data in the matrices were replaced with random numbers. The expectation was that these random values would not form recognizable patterns and thus would be the leading candidates to get removed in node deletion.

The random numbers used to replace missing values in the yeast data were generated so that they form a uniform distribution between 0 and 800. For the human data, the uniform distribution was between -800 and 800.

Determining Algorithm Parameters

The 30 clusters reported in Tavazoie *et al.* (1999) were used to determine the δ value in Algorithms 1 and 2. From the discussion before, we know that a completely random submatrix of any size for the value range (0 to 800) has a score about 53,000. The clusters reported in Tavazoie *et al.* (1999) have scores in the range between 261 (Cluster 3) and 996 (Cluster 7), with a median of 630 (Clusters 8 and 14). A δ value (300) close to the lower end of this range was used in the experiment, to detect more refined patterns.

rows	columns	low	high	peak	tail
3	6	10	6870	390	15.5%
3	17	30	6600	480	6.83%
10	6	110	4060	800	0.064%
10	17	240	3470	870	0.002%
30	6	410	2460	960	$< 10^{-6}$
30	17	480	2310	1040	$< 10^{-6}$
100	6	630	1720	1020	$< 10^{-6}$
100	17	700	1630	1080	$< 10^{-6}$

Table 1: Score distributions estimated by randomly selecting one million submatrices for each size combination. The columns correspond to the number of rows, the number of columns, the lowest score, the highest score, the peak score, and the percentage of submatrices with scores below 300.

Submatrices of different sizes were randomly generated (one million times for each size) from the yeast matrix and the distributions of scores along with the probability that a submatrix of the size has a score lower than 300 were estimated and listed in Table 1.

The δ value used in the experiment with human data was 1,200, because of the doubling in the range and the quadrupling of the variance in the data, compared to the yeast data.

Algorithm 1 (Single Node Deletion) becomes quite slow when the number of rows in the matrix is in the thousands, which is common in expression data. A proper α must be determined to run the accelerated Algorithm 2 (Multiple Node Deletion). Lemma 2 gives some guidance to the determination of α. Our aim was to find an α as large as possible and still allow the program to find 100 biclusters in less than 10 minutes. When the number of conditions is less than 100, which was the case for both data sets, Steps 3 and 4 were not used in Algorithm 2, so deletion of conditions started only when Algorithm 1 was called. The α used in both experiments was 1.2.

Node Addition

Algorithm 3 was used after Algorithm 2 and Algorithm 1 (called by Algorithm 2), to add conditions and genes to further reduce the score. Only one iterate of Algorithm 3 was executed for each bicluster, based on the assumption that further iterates would not add much.

Step 5 of Algorithm 3 was performed, so many biclusters contain a "mirror image" of the expression pattern.

These additions were performed using the original data set (without the masking described below).

Masking Discovered Biclusters

Because the algorithms are all deterministic, repeated run of them will not discover different biclusters, unless discovered ones are masked.

Each time a bicluster was discovered, the elements in the submatrix representing it were replaced by random numbers, exactly like those generated for the missing values (see Missing Data Replacement above). This made it very unlikely that elements covered by existing biclusters would contribute to any future pattern discovery. The masks were not used during node addition.

The steps described above are summarized in Algorithm 4 below.

Algorithm 4 (Finding a Given Number of Biclusters).

Input: A, a matrix of real numbers with possible missing elements, $\alpha \geq 1$, a parameter for multiple node deletion, $\delta \geq 0$, the maximum acceptable mean squared residue score, and n, the number of δ-biclusters to be found.

Output: n δ-biclusters in A.

Initialization: Missing elements in A are replaced with random numbers from a range covering the range of non-null values. A' is a copy of A.

Iteration for n times:

1. Apply Algorithm 2 on A', δ, and α. If the row (column) size is small (less than 100), do not perform multiple node deletion on rows (columns). The matrix after multiple node deletion is B.
2. (Step 5 of Algorithm 2) Apply Algorithm 1 on B and δ and the matrix after single node deletion is C.
3. Apply Algorithm 3 on A and C and the result is the bicluster D.
4. Report D, and replace the elements in A' that are also in D with random numbers.

Implementation and Display

These algorithms were implemented using C and run on a Sun Ultra10 workstation. With the parameters specified above, 100 biclusters were discovered from each data set in less than 10 minutes. Plots were generated for each bicluster, showing the expression levels of the genes in it under the conditions in the bicluster. 30 biclusters for the yeast data were ploted in Figures 1, 2, 3, and 4, and 24 for the human data in Figures 5 and 6. In the captions, "Bicluster" denotes a bicluster discovered using our algorithm and "Cluster" denotes a cluster discovered in Tavazoie *et al.* (1999). Detailed descriptions for these biclusters can be found in http://arep.med.harvard.edu/biclustering.

Results

From visual inspection of the plots one can see that this biclustering approach works as well as conventional clustering methods, when there were clear patterns over *all* attributes (conditions when the genes are clustered, or genes when conditions are clustered). The δ param-

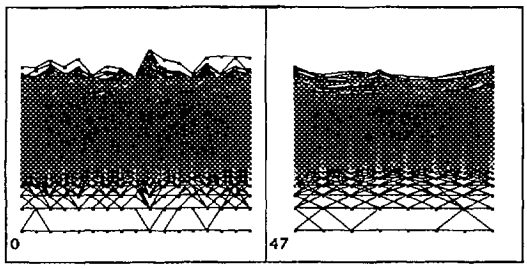

Figure 1: The first bicluster (Bicluster 0) discovered by Algorithm 4 from the yeast data and the flattest one (Bicluster 47, with a row variance 39.33) are examples of the "flat" biclusters the algorithm has to find and mask before more "interesting" ones may emerge. The bicluster with the highest row variance (4,162) is Bicluster 93 in Figure 3. The quantization effect visible at lower ends of the expression levels was due to a lack of the number of significant digits both before and after logarithm.

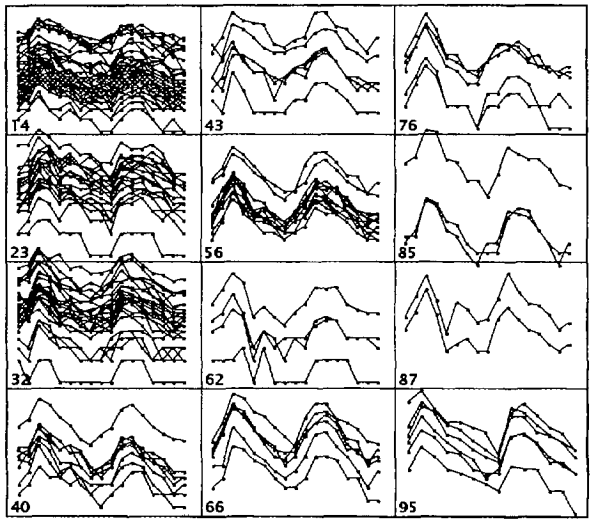

Figure 2: 12 biclusters with numbers indicating the order they were discovered using the algorithms. These biclusters are clearly related to Cluster 2 in Tavazoie *et al.* (1999), which has a score of 757. They subdivides Cluster 2's profile into similar but different ones. These biclusters have scores less than 300. They also include 10 genes from Cluster 14 of Tavazoie *et al.* (1999) and 6 from other clusters. All biclusters plotted here except Bicluster 95 contain all 17 conditions, indicating that these conditions form a cluster well, with respect to the genes included here.

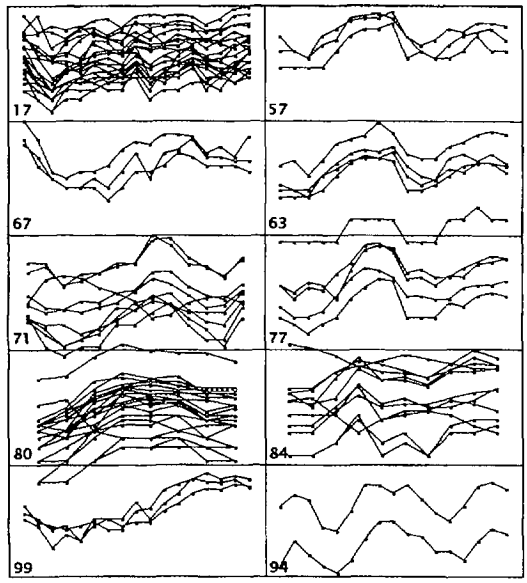

Figure 3: 10 biclusters discovered in the order labeled. Biclusters 17, 67, 71, 80, and 99 (on the left column) contain genes in Clusters 4, 8, and 12 of Tavazoie *et al.* (1999), while biclusters 57, 63, 77, 84, and 94 represent Cluster 7.

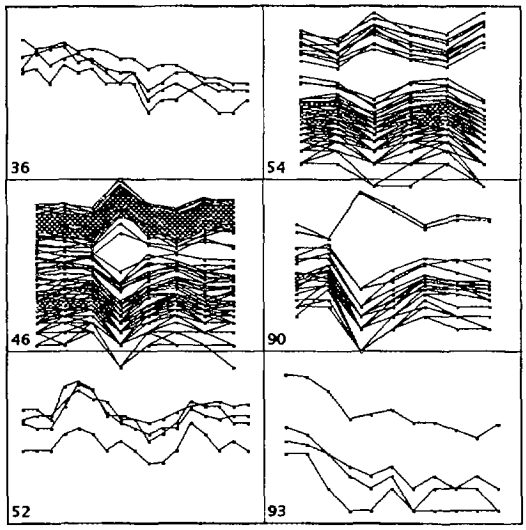

Figure 4: 6 biclusters discovered in the order labeled. Bicluster 36 corresponds to Cluster 20 of Tavazoie *et al.* (1999). Bicluster 93 corresponds to Cluster 9. Bicluster 52 and 90 correspond to Clusters 14 and 30. Bicluster 46 corresponds to Clusters 1, 3, 11, 13, 19, 21, 25, and 29, and Bicluster 54 corresponds to Clusters 5, 15, 24, and 28. Notice that some biclusters have less than half of the 17 conditions and thus represent shared patterns in many clusters discovered using similarity based on all the conditions.

eter gives a powerful tool to fine-tune the similarity requirements. This explains the correspondence between one or more biclusters to each of the better clusters discovered in Tavazoie *et al.* (1999). These includes the clusters associated to the highest scoring motifs (Clusters 2, 4, 7, 8, 14, and 30 of Tavazoie *et al.*). For other clusters from Tavazoie *et al.*, there were not clear correspondence to our biclusters. Instead, Biclusters 46 and 54 represent the common features under some of the conditions of these lesser clusters.

Coverage of the Biclusters

In the yeast data experiment, the 100 biclusters covered 2,801, or 97.12% of the genes, 100% of the conditions, and 81.47% of the cells in the matrix.

The first 100 biclusters from the human data covered 3,687, or 91.58% of the genes, 100% of the conditions, and 36.81% of the cells in the data matrix.

Sub-Categorization of Tavazoie's Cluster 2

Figure 2 shows 12 biclusters containing mostly genes classified to Cluster 2 in Tavazoie *et al.*. Each of these biclusters clearly represents a variation to the common theme for Cluster 2. For example, Bicluster 87 contains genes with three sharp peaks in expression (CLB6 and SPT21). Genes in Bicluster 62 (ERP3, LPP1, and PLM2) showed also three peaks, but the third of these is rather flat. Bicluster 56 shows a clear-cut double-peak pattern with DNA replication genes CDC9, CDC21, POL12, POL30, RFA1, RFA2, among others. Bicluster 66 contains two-peak genes with an even sharper image (CDC45, MSH6, RAD27, SWE1, and PDS5). On the other hand, Bicluster 14 contains those genes with barely recognizable double peaks.

Broad Strokes and Fine Drawings

Figures 5 and 6 show various biclusters discovered in the human lymphoma expression data. Some of them represent a few genes closely following each other, through almost all the conditions. Others show large numbers genes displaying a broad trend and its mirror image. These "broad strokes" often involve smaller subsets of the conditions and genes depicted in some of the "fine drawings" may be added to these broad trends during the node addition phase of the algorithm. These biclusters clearly say a lot about the regulatory mechanism and also the classification of conditions.

Comparison with Alizadeh's Clusters

We did a comparison of the first 100 biclusters discovered in the human lymphoma data with the clusters discovered in Alizadeh *et al.* (2000). Hierarchical clustering was used in Alizadeh *et al.* on both the genes and the conditions. We used the root division in each hierarchy in our comparison. We call the two clusters generated by the root division in a hierarchy the *primary clusters*.

Out of the 100 biclusters, only 10 have conditions exclusively from one or the other primary cluster on

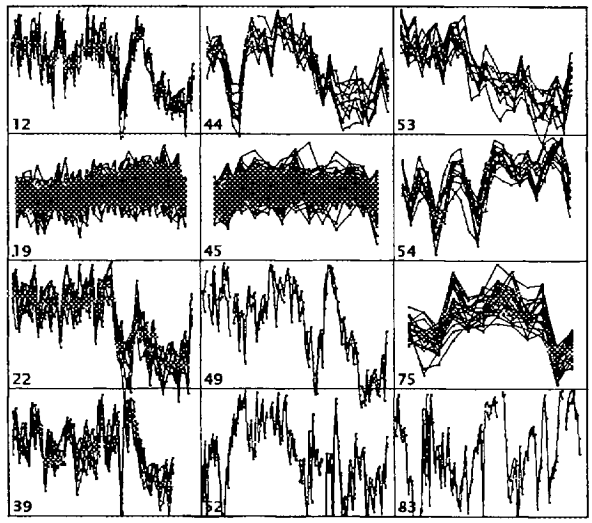

Figure 5: 12 biclusters discovered in the order labeled from the human expression data. Scores were 1,200 or lower. The numbers of genes and conditions in each are reported in the format of (bicluster label, number of genes, number of conditions) as follows. (12, 4, 96), (19, 103, 25), (22, 10, 57), (39, 9, 51), (44, 10, 29), (45, 127, 13), (49, 2, 96), (52, 3, 96), (53, 11, 25), (54, 13, 21), (75, 25, 12), (83, 2, 96)

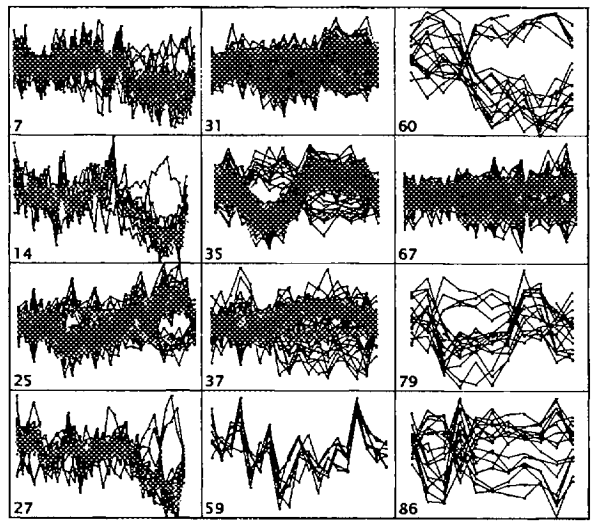

Figure 6: Another 12 biclusters discovered in the order labeled from the human expression data. The numbers of genes and conditions in each bicluster (whose label is the first figure in the triple) are as follows. (7, 34, 48), (14, 15, 46), (25, 57, 24), (27, 23, 30), (31, 158, 17), (35, 102, 13), (37, 59, 18), (59, 8, 19), (60, 18, 11), (67, 102, 20), (79, 18, 11), (86, 18, 11) Mirror images can be seen in most of these biclusters.

conditions. Bicluster 19 in Figure 5 is heavily biased towards one primary condition cluster, while Bicluster 67 in Figure 6 is heavily biased towards the other.

A similar proportion of the biclusters have genes exclusively from one or the other primary cluster on genes. The tendency is that the higher the row variance (5) and the number of columns (conditions) involved are, the more likely the genes in the bicluster will come from one primary cluster. We define *mix* as the percentage of genes from the minority primary cluster in the bicluster, and plot mix versus row variance in Figure 7. Each digit in the Figure 7 represents a bicluster, with the digit indicating the number of conditions in the bicluster.

All the biclusters shown in Figures 5 and 6 have row variance (squared-rooted) greater than 100. Many of them have zero percent mix (Biclusters 7, 12, 14, 22, 39, 44, 49, 52, and 53 from one primary cluster and Biclusters 25, 54, 59, and 83 from the other). This demonstrates the correctness of the use of row variance as a means for separating interesting biclusters from the trivial ones. Some biclusters have high row variances but also high mix scores (those in the upper right quadrant of the plot). These invariably involve a narrower view of the conditions, with the digit "1" indicating the number of conditions involved is below 20%. Genes in these biclusters are distributed on both sides of the root division of hierarchical clustering. These biclusters show the complementary role that biclustering plays to one-dimensional clustering. Biclustering allows us to focus on the right subsets of conditions to see the apparent co-regulatory patterns not seen with the global scope.

For example, Bicluster 60 suggests that under the 11 conditions, mostly for chronic lymphocytic leukemia (CLL), FMR2 and the interferon-γ receptor α chain behave against their stereotype and get down-regulated. As another example, Bicluster 35 suggests that under the 13 conditions, CD49B and SIT appear similar to genes classified by hierarchical clustering into the other primary cluster.

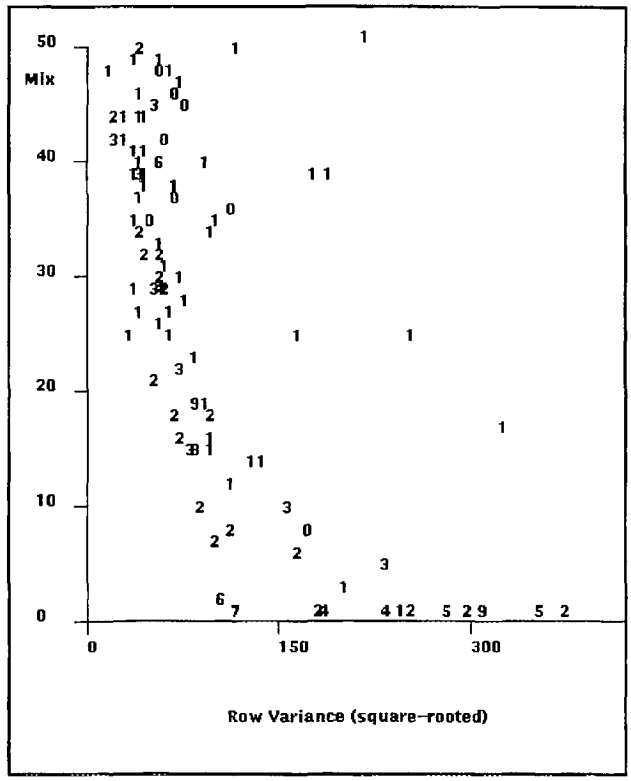

Figure 7: A plot of the first 100 biclusters in the human lymphoma data. Three measurements are involved. The horizontal axis is the square root of the row variance, defined in (5). The vertical axis is the *mix*, or the percentage of genes misclassified across the root division generated by hierarchical clustering in Alizadeh *et al.* (2000). We assumed that genes forming mirror image expression patterns are naturally from the other side of the root division. The digits used to represent the biclusters in the plot also indicate the sizes of the condition sets in the biclusters. The digit n indicates that the number of conditions is between $10n$ and $10(n+1)$ percent of the total number of conditions.

Discussion

We have introduced a new paradigm, biclustering, to gene expression data analysis. The concept itself can be traced back to 1960's and 1970's, although it has been rarely used or even studied. To gene expression data analysis, this paradigm is relevant, because of the complexity of gene regulation and expression, and the sometimes low quality of the gathered raw data.

Biclustering is performed on the expression matrix, which can be viewed as a weighted bipartite graph. The concept of bicluster is a natural generalization of the concept of biclique in graph theory. There are one-dimensional clustering methods based on graphs constructed from similarity scores between genes, for example, the hierarchical clustering method in Alizadeh *et al.* (2000), and the finding of highly connected subgraphs in Hartuv *et al.* (1999). A major difference here

is that biclustering does not start from or require the computation of overall similarity between genes.

The relation between biclustering and clustering is similar to that between the instance-based paradigm and the model-based paradigm in supervised learning. Instance-based learning (for example, nearest-neighbor classification) views data locally, while model-based learning (for example, feed-forward neural networks) finds the globally optimally fitting models.

Biclustering has several obvious advantages over clustering. First, biclustering automatically selects genes and conditions with more coherent measurement and drops those representing random noise. This provides a method for dealing with missing data and corrupted measurements.

Secondly, biclustering groups items based on a similarity measure that depends on a context, which is best defined as a subset of the attributes. It discovers not only the grouping, but the context as well. And to some extent, these two become inseparable and exchangeable, which is a major difference between biclustering and clustering rows after clustering columns.

Most expression data result from more or less complete sets of genes but very small portions of all the possible conditions. Any similarity measure between genes based on the available conditions becomes anyway context-dependent. Clustering genes based on a measure like this is no more representative than biclustering.

Thirdly, biclustering allows rows and columns to be included in multiple biclusters, and thus allows one gene or one condition to be identified by more than one function categories. This added flexibility correctly reflects the reality in the functionality of genes and overlapping factors in tissue samples and experiment conditions.

By showing the NP-hardness of the problem, we tried to justify our efficient but greedy algorithms. But the nature of NP-hardness implies that there may be sizable biclusters with good scores evading the search by any efficient algorithm. Just like most efficient conventional clustering algorithms, one can say that the best biclusters can be found in most cases, but one cannot say that it will be found in all the cases.

Acknowledgments

This research was conducted at the Lipper Center for Computational Genetics at the Harvard Medical School, while the first author was on academic leave from the University of Cincinnati.

References

Aach, J., Rindone, W., and Church, G.M. 2000. Systematic management and analysis of yeast gene expression data. *Genome Research* in press.

Alizadeh, A.A. et al. 2000. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 403:503-510.

Ball, C.A. et al. 2000. Integrating functional genomic information into the *Saccharomyces* Genome Database. *Nucleic Acids Res.* 28:77-80.

Cho, R.J. et al. A genome wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell* 2:65-73.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: Freeman.

Hartigan, J.A. 1972. Direct clustering of a data matrix. *JASA* 67:123-129.

Hartuv, E. et al. 1999. An algorithm for clustering cDNAs for gene expression analysis. *RECOMB '99*, 188-197.

Johnson, D.S. 1987. The NP-completeness column: an ongoing guide. *J. Algorithms* 8:438-448.

Mirkin, B. 1996. *Mathematical Classification and Clustering.* Dordrecht: Kluwer.

Morgan, J.N. and Sonquist, J.A. 1963. Problems in the analysis of survey data, and a proposal. *JASA* 58:415-434.

Nau, D.S., Markowsky, G., Woodbury, M.A., and Amos, D.B. 1978. A mathmatical analysis of human leukocyte antigen serology. *Math. Biosci.* 40:243-270.

Orlin, J. 1977. Containment in graph theory: covering graphs with cliques, *Nederl. Akad. Wetensch. Indag. Math.* 39:211-218.

Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., and Church, G.M. 1999. Systematic determination of genetic network architecture. *Nature Genetics* 22:281-285.

Yannakakis, M. 1981. Node deletion problems on bipartite graphs. *SIAM J. Comput.* 10:310-327.