# MicroCluster: Efficient Deterministic Biclustering of Microarray Data

**Lizhuang Zhao and Mohammed J. Zaki,** *Rensselaer Polytechnic Institute*

Biclustering[1] has proved of great value for finding interesting patterns in microarray expression data, which record the expression levels of many genes, for different biological samples. (For more on gene expression, see the related sidebar.) Biclustering can identify the coexpression patterns of a subset of genes that might be relevant to a subset of the samples of interest.

*MicroCluster can mine different types of arbitrarily positioned and overlapping clusters of genetic data to find interesting patterns.*

Mining microarray data for biclusters presents four main challenges. First, biclustering is NP-hard,[1] so many proposed bicluster-mining algorithms use heuristic methods or probabilistic approximations, which decrease the final clustering results' accuracy. Second, microarray data is susceptible to noise, owing to varying experimental conditions, so methods should handle noise well. Third, given that we don't understand the cell's complex gene regulation circuitry, clustering methods should allow overlapping clusters that share subsets of genes or samples. Finally, the methods should be flexible enough to mine several (interesting) types of clusters.

MicroCluster is an efficient, deterministic, and complete biclustering method that addresses these challenges. Our approach has four key features. First, we mine only the maximal biclusters satisfying certain homogeneity criteria. Second, the clusters can be arbitrarily positioned anywhere in the input data matrix, and they can have arbitrary overlapping regions. Third, MicroCluster uses a flexible definition of a cluster that lets it mine several types of biclusters (which previously were studied independently). Finally, MicroCluster can delete or merge biclusters that have large overlaps. So, it can tolerate some noise in the data set and let users focus on the most important clusters. We've developed a set of metrics to evaluate the clustering quality and have tested MicroCluster's effectiveness on several synthetic and real data sets.

## Preliminary concepts

Let $G = \{g_0, g_1, \ldots, g_{n-1}\}$ be a set of $n$ genes, and let $S = \{s_0, s_1, \ldots, s_{m-1}\}$ be a set of $m$ biological samples (or conditions). A microarray data set is a real-valued $n \times m$ matrix $D = G \times S = \{d_{ij}\}$ (with $i \in [0, n-1]$, $j \in [0, m-1]$), whose rows correspond to genes and whose columns correspond to samples. Each entry $d_{ij}$ records the (absolute or relative) expression level of gene $g_i$ in sample $s_j$. For example, figure 1 shows a data set with 10 genes and seven samples. For clarity, certain cells are blank; we assume that random expression values fill them.

A bicluster $C$ is a submatrix of $D$, where $C = X \times Y = \{c_{ij}\}$, with $X \subseteq G$ and $Y \subseteq S$, provided certain conditions of homogeneity are satisfied. For example, a simple condition might be that all values $c_{ij}$ are identical or approximately equal (a constant pattern). You can also define other homogeneity conditions, such as a similar column or row pattern or a scaling or shifting pattern.[2] Given $\mathcal{B}$, the set of all biclusters that satisfy the given homogeneity conditions, then $C = X \times Y \in \mathcal{B}$ is a *maximal bicluster* if and only if there doesn't exist $C' = X' \times Y' \in \mathcal{B}$ such that $C \subset C'$ (that is, $X \subset X'$ and $Y \subset Y'$).

Let $C = X \times Y$ be a bicluster, and let

$$\begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$$

be an arbitrary $2 \times 2$ submatrix of $C$. We call $C$ a cluster if and only if it's a maximal bicluster satisfying these properties:



1541-1672/05/$20.00 © 2005 IEEE
*Published by the IEEE Computer Society*

- If $r_i = |c_{ib}/c_{ia}|$ and $r_j = |c_{jb}/c_{ja}|$, then $(\max(r_i, r_j)/\min(r_i, r_j)) - 1 \le \varepsilon$, where $\varepsilon$ is a maximum ratio threshold. This threshold ensures that the ratios of column values across any two rows in the cluster are similar.
- If $c_{ia} \times c_{ib} < 0$, then $\text{sign}(c_{ia}) = \text{sign}(c_{ja})$ and $\text{sign}(c_{ib}) = \text{sign}(c_{jb})$, where $\text{sign}(x)$ returns $-1$ if $x$ is negative or 1 if $x$ is nonnegative (the preprocessing step replaces expression values of zero with a small random positive correction value). This lets us easily mine data sets having negative expression values. (It also prevents us from reporting that, for example, expression ratio –5/5 is equal to 5/–5.)
- If $r_x = \max(c_{xa}, c_{xb})/\min(c_{xa}, c_{xb})$, then $r_x - 1 \le \delta^r$, where $x \in \{i, j\}$ and $\delta^r$ is a maximum row range threshold. So, the cluster can allow at most $\delta^r$ variation in the rows.
- If $r_y = \max(c_{iy}, c_{jy})/\min(c_{iy}, c_{jy})$, then $r_y - 1 \le \delta^c$, where $y \in \{a, b\}$ and $\delta^c$ is a maximum column range threshold. So, the cluster can allow at most $\delta^c$ variation in the columns.
- $|X| \ge mr$ and $|Y| \ge mc$, where $mr$ denotes the minimum row cardinality threshold and $mc$ denotes the minimum column cardinality threshold. This lets us find only meaningfully large clusters.

## Gene Expressions

In every organism, different genes are expressed in different types of cells and tissues at different times. Analysis of these gene expressions can help us understand disease states, drug functions, and so on. Genes with similar or correlated expression levels (that is, genes that display similar behaviors, at least in terms of the amount of their presence) are said to be *coexpressed*. From gene coexpression, we can infer the gene coregulation mechanism or the involved genes' biological function (coexpressed genes might share the same regulation mechanism or be functionally related).

Sample

| Gene | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
|---|---|---|---|---|---|---|---|
| $g_0$ | | 1.0 | 1.0 | | 1.0 | 1.0 | 1.0 |
| $g_1$ | 3.0 | 2.5 | | | 2.0 | | 1.0 |
| $g_2$ | | 5.0 | | | 5.0 | | 5.0 |
| $g_3$ | 6.6 | 5.5 | | | | | 2.0 |
| $g_4$ | 9.0 | 7.5 | | | 6.0 | | 3.0 |
| $g_5$ | 6.6 | | | | 4.4 | | 2.0 |
| $g_6$ | | 3.0 | | | 3.0 | | 3.0 |
| $g_7$ | | 8.0 | 8.0 | | 8.0 | 8.0 | |
| $g_8$ | 6.0 | 5.0 | | | 4.0 | | 2.0 |
| $g_9$ | | 4.0 | 4.0 | | 4.0 | 4.0 | 4.0 |

(a)

Sample

| Gene | $s_3$ | $s_0$ | $s_6$ | $s_4$ | $s_1$ | $s_5$ | $s_2$ |
|---|---|---|---|---|---|---|---|
| $g_5$ | | 6.6 | 2.0 | 4.4 | | | |
| $g_2$ | | | 5.0 | 5.0 | 5.0 | | |
| $g_6$ | | | 3.0 | 3.0 | 3.0 | | |
| $g_0$ | | | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $g_9$ | | | 4.0 | 4.0 | 4.0 | 4.0 | 4.0 |
| $g_7$ | | | | 8.0 | 8.0 | 8.0 | 8.0 |
| $g_3$ | | 6.6 | 2.0 | | 5.5 | | |
| $g_4$ | | 9.0 | 3.0 | 6.0 | 7.5 | | |
| $g_8$ | | 6.0 | 2.0 | 4.0 | 5.0 | | |
| $g_1$ | | 3.0 | 1.0 | 2.0 | 2.5 | | |

(b)

**Figure 1. An (a) example microarray data set and (b) some clusters.**

### Lemma 1 (symmetry property)

Given the bicluster $C$ and the $2 \times 2$ submatrix mentioned previously, if $r_i = |c_{ib}/c_{ia}|$, $r_j = |c_{jb}/c_{ja}|$, $r_a = |c_{ja}/c_{ia}|$, and $r_b = |c_{jb}/c_{ib}|$, then $(\max(r_i, r_j)/\min(r_i, r_j)) - 1 \le \varepsilon \Leftrightarrow (\max(r_a, r_b)/\min(r_a, r_b)) - 1 \le \varepsilon$.

Here's the proof: Without loss of generality, assume that $r_i \ge r_j$, then $|c_{ib}/c_{ia}| \ge |c_{jb}/c_{ja}| \Leftrightarrow |c_{ja}/c_{ia}| \ge |c_{jb}/c_{ib}| \Leftrightarrow r_a \ge r_b$. Also, $\max(r_i, r_j)/\min(r_i, r_j) = r_i/r_j = |c_{ib}/c_{ia}|/|c_{jb}/c_{ja}| = |c_{ja}/c_{ia}|/|c_{jb}/c_{ib}| = r_a/r_b = \max(r_a, r_b)/\min(r_a, r_b)$.

### Lemma 2 (shifting cluster)

Let $C = X \times Y = \{c_{xy}\}$ be a submatrix of data set $D = \{d_{xy}\}$, and let

$$C_{2,2} = \begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$$

be an arbitrary $2 \times 2$ submatrix of $C$. Let

$$e^D = \{e^{d_{xy}}\}$$

be the new data set obtained by applying the exponential function (base $e$) to each value $d_{xy}$ in $D$. If $e^C$ is a scaling cluster in $e^D$, then $C$ is a shifting cluster in $D$.

Here's the proof: Let

$$e^{C_{2,2}} = \begin{bmatrix} e^{c_{ia}} & e^{c_{ib}} \\ e^{c_{ja}} & e^{c_{jb}} \end{bmatrix}$$

be any $2 \times 2$ submatrix of $e^C$. Without loss of generality, assume that

$$0 \le \frac{|e^{c_{ib}} / e^{c_{ia}}|}{|e^{c_{jb}} / e^{c_{ja}}|} - 1 \le \varepsilon$$

then

$$0 \le |c_{ib} - c_{ia}| - |c_{jb} - c_{ja}| \le \ln(1+\varepsilon) = \varepsilon'$$

that is, $C$ is a shifting cluster of $D$ with window size $\varepsilon'$.

This lemma shows that we can mine shifting biclusters by first transforming each value in the data set by taking its exponent. Any (scaling) cluster discovered in the transformed data set is, by this lemma, a shifting cluster in the original data set.

### Obtaining clusters

Figure 1b shows examples of different clusters that we can obtain from our example data set by some row or column permutations. Let $mr = mc = 3$, and let $\varepsilon = 0.01$. If we let $\delta^r = \delta^c = \infty$ (that is, unconstrained), then $C_1 = \{g_1, g_4, g_8\} \times \{s_0, s_1, s_4, s_6\}$ is a scaling cluster; that is, each row or column is some scalar multiple of another row or column. We also discover two other maximal overlapping clusters: $C_2 = \{g_0, g_2, g_6, g_9\} \times \{s_1, s_4, s_6\}$ and $C_3 = \{g_0, g_7, g_9\} \times \{s_1, s_2, s_4, s_5\}$. If we set $mc = 2$, we find another maximal cluster $C_4 = \{g_0, g_2, g_6, g_7, g_9\} \times \{s_1, s_4\}$, which is subsumed by $C_2$ and $C_3$. As we'll show later, MicroCluster can delete such a cluster in the final steps.

Different choices of row and column range thresholds ($\delta^r$ and $\delta^c$) give MicroCluster the flexibility to produce different kinds of clusters. For example, if $\delta^r = \delta^c \approx 0$, then we obtain clusters with approximately identical values. We obtain row constant clusters by constraining only $\delta^r \approx 0$ and column constant clusters by constraining only $\delta^c \approx 0$. When $\delta^r \ne 0$ and $\delta^c \ne 0$, we obtain a scaling cluster, and by Lemma 2 we can mine shifting clusters as well.

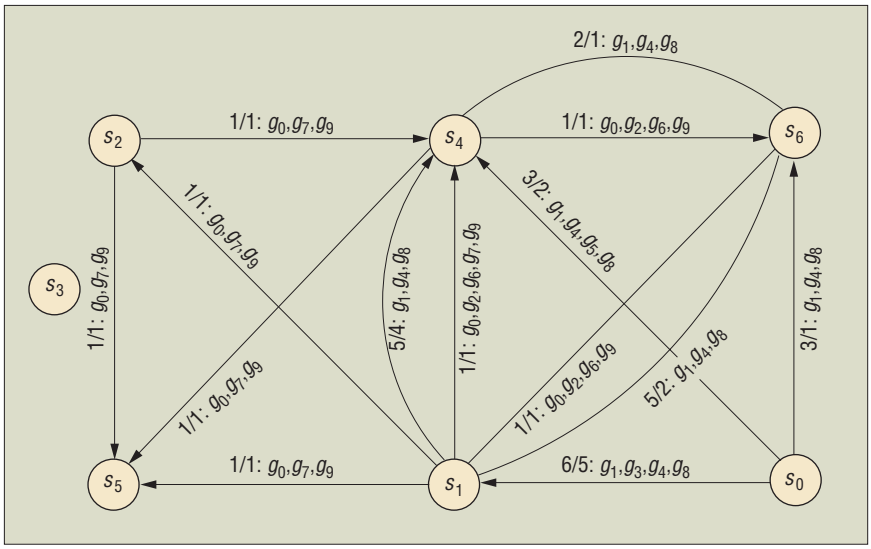



**Figure 2. A weighted, directed range multigraph compactly represents the possible valid ratio ranges and the genes that meet those ranges.**

```
Input          : parameters: ε, mr, mc, δ^r, δ^c, range graph M, set of genes G and samples S
Output         : cluster set C
Initialization : C = ∅, call MicroCluster(C = G × ∅, S)
MicroCluster (C = X × Y, P):;
1  if C satisfies δ^r, δ^c then
2      if |C.X| ≥ mr and |C.Y| ≥ mc then
3          if ∄ C′ ∈ C, such that C ⊂ C′ then
4              Delete any C″ ∈ C, if C″ ⊂ C
5              C ← C + C
6  foreach s_b ∈ P do
7      C^new ← C
8      C^new.Y ← C^new.Y + s_b
9      P ← P − s_b
10     if C.Y = ∅ then
11         MicroCluster(C^new, P)
       else
12         forall s_a ∈ C.Y and each R_i^ab ∈ R^ab satisfying |G(R_i^ab) ∩ C.X| ≥ mr do
13             C^new.X ← (∩_all s_a ∈ C.Y G(R_i^ab)) ∩ C.X
14             if |C^new.X| ≥ mr then
15                 MicroCluster(C^new, P)
```

**Figure 3. The MicroCluster algorithm's clique-mining step.**

## The MicroCluster algorithm

Because of the symmetry property, MicroCluster always mines a data set that has more rows than columns, transposing the input data set (matrix) if necessary. MicroCluster has three main steps:

1. Find the valid *ratio ranges* for all pairs of columns, and construct a range multigraph.
2. Mine the maximal clusters from the range multigraph.
3. Optionally delete or merge clusters.

### Constructing a range multigraph

Given $D$, $mr$, $mc$, and $\varepsilon$, let $s_a$ and $s_b$ be any two columns in $D$ and let

$$r_x^{ab} = \frac{d_{xa}}{d_{xb}}$$

be the ratio of the expression values of gene $g_x$ in columns $s_a$ and $s_b$, where $x \in [0, n-1]$. A ratio range is an interval of ratio values $[r_l, r_u]$, with $r_l \le r_u$. Let

$$\mathcal{G}_{ab}\left(\left[r_l, r_u\right]\right) = \left\{g_x : r_x^{ab} \in \left[r_l, r_u\right]\right\}$$

be the *gene set*, the set of genes whose ratios with regard to columns $s_a$ and $s_b$ lie in the given ratio range.

First, MicroCluster quickly tries to summarize the valid ratio ranges that can contribute to some cluster. More formally, we call a ratio range valid if and only if

- $(\max(|r_u|, |r_l|)/\min(|r_u|, |r_l|)) - 1 \le \varepsilon$,
- $|\mathcal{G}_{ab}([r_l, r_u])| \ge mr$,
- $r_x^{ab} < 0$ exists for some gene $g_x$, then all the values $\{d_{xa}\}/\{d_{xb}\}$ in the same column have the same sign (negative or nonnegative), and
- $[r_l, r_u]$ is maximal with regard to $\varepsilon$; that is, we can't add another gene to $\mathcal{G}_{ab}([r_l, r_u])$ and yet preserve the $\varepsilon$ bound.

Given the set of all valid ranges across any pairs of columns $s_a$, $s_b$ with $a < b$, given as

$$\mathcal{R}^{ab} = \left\{R_i^{ab} = \left[r_{l_i}^{ab}, r_{u_i}^{ab}\right] : s_a, s_b \in S\right\}$$

we construct a weighted, directed, range multigraph $M = (V, E)$, where $V = S$ (all columns), and for each

$$R_i^{ab} \in \mathcal{R}^{ab}$$

there exists a weighted, directed edge $(s_a, s_b) \in E$ with weight

$$w = \frac{r_{u_i}^{ab}}{r_{l_i}^{ab}}$$

In addition, each edge in the range multigraph has an associated gene set corresponding to the range on that edge. For example, suppose $mc = 3$, $mr = 3$, and $\varepsilon = 0.01$. Figure 2 shows the range multigraph constructed from figure 1.

### Mining clusters

Construction of the range multigraph filters out most unrelated data. Next, MicroCluster uses a depth-first search on the range multigraph to mine all the clusters (see figure 3). It takes as input the set of parameter values $\varepsilon$, $mr$, $mc$, $\delta^r$, $\delta^c$, $M$, $G$, and $S$. It will output the

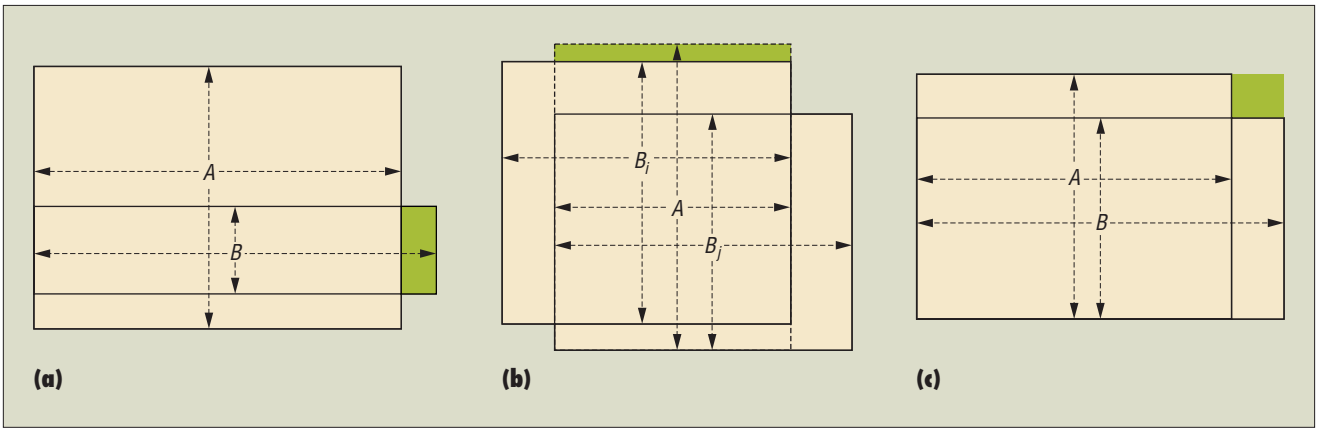



**Figure 4. Three deletion or merging cases: (a) deleting cluster *B*, (b), deleting cluster *A*, and (c) merging *A* and *B*.**

final set of all clusters $C$. MicroCluster is a recursive algorithm that at each call accepts a current *candidate* cluster $C = X \times Y$ and a set of not-yet-expanded samples $P$. The initial call is made with a cluster $C = G \times \varnothing$ with all genes $G$, but no samples, and with $P = S$.

Lines 1 through 5 of figure 3 check whether $C$ meets the thresholds $mr$ and $mc$ (line 1) and $\delta^r$ and $\delta^c$ (line 2). If so, we next check whether some maximal cluster $C' \in \mathcal{C}$ already contains $C$ (line 3). If not, we first remove any cluster $C'' \in \mathcal{C}$ that $C$ has already subsumed (line 4) and then add $C$ to $\mathcal{C}$ (line 5).

Lines 6 through 15 generate a new candidate cluster by expanding the current candidate by one more sample and constructing the appropriate gene set for the new candidate, before making a recursive call. MicroCluster begins by adding to $C$ each new sample $s_b \in P$ (line 6), to obtain a new candidate $C^{new}$ (lines 7 and 8). Microcluster removes already-processed samples from $P$ (line 9). Let $s_a$ be all samples added to $C$ until the previous recursive call. If no previous vertex $s_a$ exists (which happens initially) (line 10), we simply call MicroCluster with the new candidate (line 11). Otherwise, MicroCluster tries all combinations of each qualified range edge $R_i^{ab}$ between $s_a$ and $s_b$ for all $s_a \in C.Y$ (line 12), obtains their gene set intersection

$$\cap_{all\ s_a \in C.Y} \mathcal{G}\left(R_i^{ab}\right)$$

and intersects with $C.X$ to obtain the valid genes in $C^{new}$ (line 13). If $C^{new}$ has at least $mr$ genes, then another recursive call to MicroCluster is made (lines 14 and 15).

## Deleting or merging clusters

This step is important because real data can be noisy, and having many clusters with large overlaps only makes it harder for users to select the important ones.

Let $A = X_A \times Y_A$ and $B = X_B \times Y_B$ be any two mined clusters. We define the *span* of a cluster $C = X \times Y$ as the set of gene sample pairs belonging to the cluster, given as $L_C = \{(g_i, s_j) \mid g_i \in X, s_j \in Y\}$. Then we can define these derived spans:

- $L_{A \cup B} = L_A \cup L_B$,
- $L_{A-B} = L_A - L_B$, and
- $L_{A+B} = L_{(X_A \cup X_B) \times (Y_A \cup Y_B)}$.

MicroCluster deletes or merges the involved clusters if any of these overlap conditions exist:

- *Delete B*. If $L_A > L_B$ and if $|L_{B-A}|/|L_B| < \eta$, then delete $B$. As figure 4a illustrates, this means that if the cluster with the smaller span ($B$) has only a few extra elements, then delete that cluster.
- *Delete A*. This is a generalization of the previous case. For a cluster $A$, if a set of clusters $\{B_i\}$ exists such that

$$\frac{\left|L_A - L_{\cup_i B_i}\right|}{|L_A|} < \eta$$

  then delete cluster $A$. As figure 4b shows, $A$ is mostly covered by the $\{B_i\}$'s and therefore can be deleted.
- *Merge A and B*. If $|L_{A+B-A-B}|/|L_{A+B}| < \gamma$, merge $A$ and $B$ into one cluster $(X_A \cup X_B) \times (Y_A \cup Y_B)$ (see figure 4c).

Here, $\eta$ and $\gamma$ are user-defined thresholds.

## Complexity analysis

Constructing the range multigraph takes $O(|G||S|^2)$ time. Cluster mining, which corresponds to constrained maximal clique enumeration, is the most expensive of the three steps. The precise number of clusters mined depends on the data set and the input parameters.

Nevertheless, for microarray data sets MicroCluster will likely be very efficient for two reasons. First, the range multigraph prunes away much of the noise and irrelevant information. Second, MicroCluster keeps intermediate gene sets for all candidate clusters, so it can prune the search the moment input criteria aren't met. Deleting and merging apply to only those pairs of clusters that actually overlap, which can be determined in $O(|\mathcal{C}| \log (|\mathcal{C}|))$ time.

## Parameter selection

The seven input parameters ($\varepsilon$, $mr$, $mc$, $\delta^r$, $\delta^c$, $\eta$, and $\gamma$) are indispensable for generating different kinds of biclusters satisfying user requirements. Moreover, setting these parameters is relatively easy (we show an experimental justification later). In the beginning, you might not set $\eta$ and $\gamma$. You can set these later for removing highly overlapped clusters or for tolerating noise. $\delta^r$ and $\delta^c$ restrict scaling or shifting patterns to be constant column or row patterns. You can also leave these unconstrained initially, to obtain more interesting clusters. However, if you're interested in constant patterns only, you can set $\delta^r$ and $\delta^c$ to gain further pruning and speedup.

For a specific data set, $\varepsilon$ decides the cluster pattern's accuracy, and $mr$ and $mc$ decide a qualified cluster's minimum size; together, they affect the running time for generating clusters. For MicroCluster, $\varepsilon$ and $mr$ affect the number of edges between vertices in the multigraph, which affects the running speed and results. In contrast, $mc$ affects the minimum search depth and cluster deletion. Because at first you might not know the data

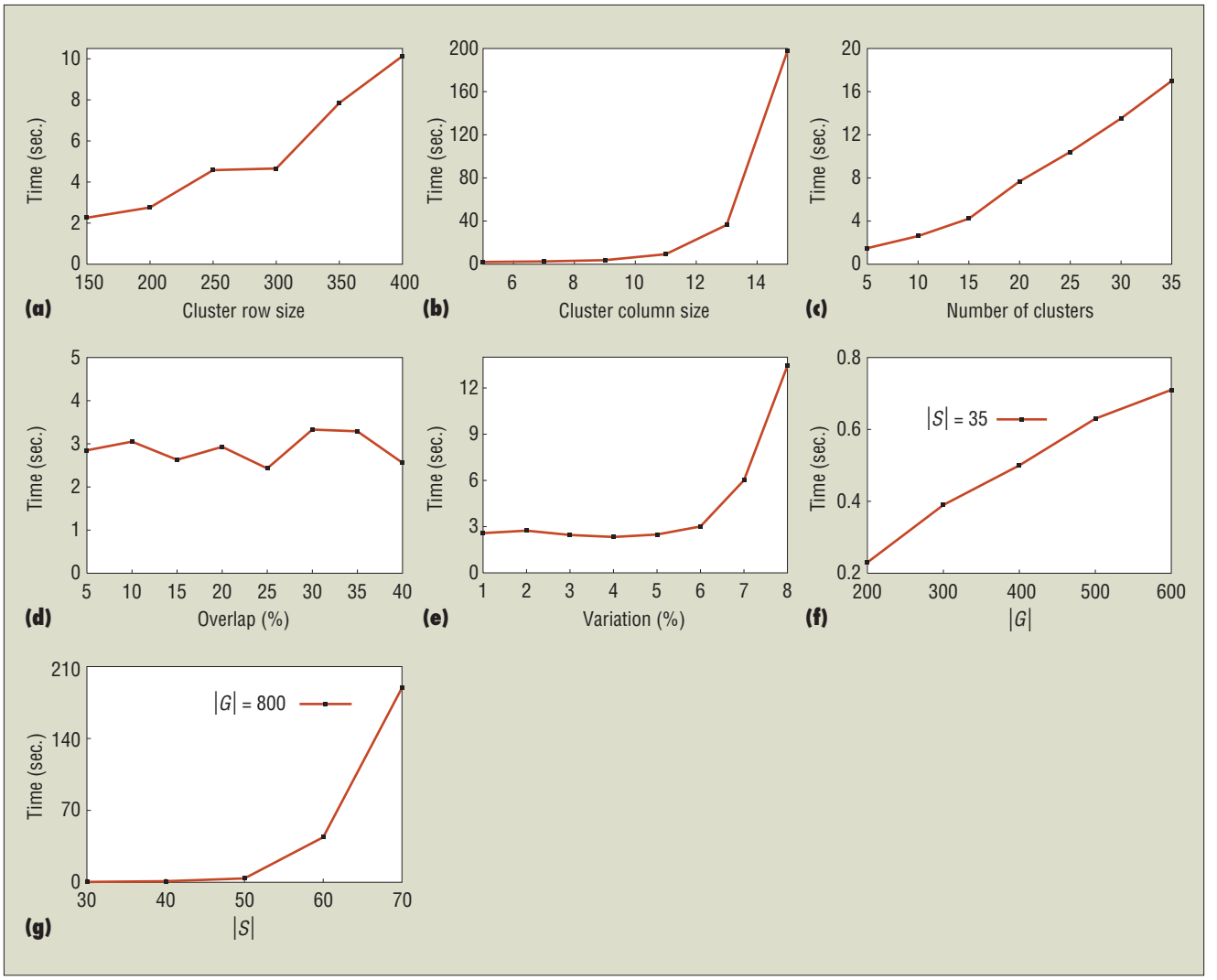



**Figure 5. MicroCluster's performance on synthetic data: (a) cluster row size, (b) cluster column size, (c) number of clusters, (d) overlap percentage, (e) variation (noise), (f) total number of rows $|G|$, and (g) total number of columns $|S|$.**

distribution, you can set these three parameters to be highly constrained—that is, small $\varepsilon$ (for example, 1 percent of the value range) and large *mr* and *mc* (for example, 30 percent of the original data set size). Then according to concrete requirements and the real running time, you can relax or strengthen these constraints. After obtaining the initial clusters, you can apply $\eta$ and $\gamma$ according to your noise tolerance and overlapping allowance.

## Evaluating MicroCluster's effectiveness

As we mentioned before, we evaluated MicroCluster on both synthetic and real data sets. Unless otherwise noted, we performed all experiments on a Fedora virtual machine (448 Mbytes of memory, a 1.4 GHz Pentium-M) over Windows XP through VMware middleware.

### Synthetic data sets

The synthetic data sets let us embed clusters and then test how MicroCluster performs for varying input parameters in isolation. The input parameters to the synthetic generator were the total number of genes and samples, number of clusters to embed, percentage of overlapping clusters, row and column ranges for the cluster sizes, and amount of noise for the expression values. Once all the clusters are generated, the generator assigned random values to the noncluster regions. We generated synthetic data with these default parameters: a $5{,}000 \times 40$ data matrix, 10 clusters, a cluster column size of 8 and a row size of 200, a 20 percent overlap, and a 3 percent noise level. For each experiment, we kept all default parameters except the varying parameter. We also chose appropriate parameter values for MicroCluster so that it found all embedded clusters.

Figures 5a through 5e show MicroCluster's sensitivity to the different parameters. The time increases linearly with cluster row size (figure 5a) but exponentially with cluster column size (figure 5b). The time is also linear with regard to the number of clusters (figure 5c), whereas the overlap percentage doesn't seem to affect the time much (figure 5d). Finally, as noise increases (figure 5e), the time to mine the clusters increases because the chance is greater that a random gene or sample can belong to some cluster.

We next varied the total number of rows

## Related Work on Subspace and Biclustering Algorithms

Researchers have proposed many subspace and biclustering algorithms.[1] We briefly review the most relevant.

CLIQUE (*Cl*ustering *in Que*st) introduced the problem of subspace clustering; it's a complete algorithm that finds axis-aligned dense subspaces.[2] (Given a data set with $d$ dimensions, a subspace is restricted to some smaller subset of dimensions $k << d$. Furthermore, even within each of the $k$ dimensions that are retained, the subspace is confined to some subrange of the possible values for those dimensions.) PROCLUS (*Pro*jected *Clus*tering) uses projective clustering to find axis-aligned subspaces by partitioning the set of points and then uses hill-climbing to refine the partitions.[3] Such subspace methods aren't designed to mine coherent patterns from microarray data sets.

For microarray analysis, $\delta$-biclustering introduced the concept of biclusters.[4] It uses mean-squared residue of a submatrix ($X \times Y$) to find biclusters. If a submatrix with enough size has a mean-squared residue less than a threshold $\delta$, it's a $\delta$-bicluster. Initially, $\delta$-biclustering starts with the whole data matrix, then repeatedly adds or deletes a row or column from the current matrix greedily until convergence occurs. After finding a cluster, it replaces the submatrix with random values and continues to find the next best cluster. This process iterates until it can find no further clusters. One limitation of $\delta$-biclustering is that it might converge to a local optimum. Also, it can easily miss overlapping clusters owing to the random-value substitutions it performs.

CLIFF (*Cl*ustering via *I*terative *F*eature *F*iltering) iterates between feature filtering and sample partitioning.[5] It first calculates $k$ best features (genes) according to their intrinsic discriminability, using current partitions. Then it partitions the samples with these features by keeping the minimum normalized weights. This process iterates until convergence occurs.

SAMBA (Statistical-Algorithmic Method for Bicluster Analysis) uses a bipartite graph to model and implement clustering.[6] It repeatedly finds the maximal highly connected subgraph in the bipartite graph. Then it performs local improvement by adding or deleting a single vertex until no further improvement is possible.

HCS (Highly Connected Subgraph) is a full-space clustering algorithm.[7] It cuts a graph into subgraphs by removing some edges, and repeats until all the vertices in each subgraph are similar enough.

xMotif uses a Monte Carlo method to find biclusters; it requires all gene expressions to be similar across all the samples in a bicluster.[8] It randomly picks a seed sample $s$ and a sample subset $d$ (called a *discriminating set*), and then finds all such genes that are conserved across all the samples in $d$.

These methods share several drawbacks. First, some of them are randomized methods based on shrinking and expansion, which sometimes results in incomplete clusters. Second, none of them can deal properly with overlapping clusters. Third, the greedy methods will lead to a local optimum and might miss important clusters. In general, none of them are deterministic, so they can't guarantee that they'll find all valid (overlapping) clusters.

Among the recent methods most similar to MicroCluster is pCluster,[9] which defines a cluster $C$ as a submatrix of the original data set, such that for any $2 \times 2$ submatrix

$$\begin{bmatrix} c_{xa} & c_{xb} \\ c_{ya} & c_{yb} \end{bmatrix}$$

of $C$, $|(c_{xa} - c_{ya}) - (c_{xb} - c_{yb})| < \delta$, where $\delta$ is a threshold. The algorithm first scans the data set to find the set of all column-pair and row-pair maximal clusters, called an *MDS* (maximal dimension set). Then it prunes the data set, in turn using the row-pair MDS and the column-pair MDS. It then mines the final clusters on the basis of a prefix tree data structure. MaPle (*Ma*ximal *P*attern-Based *Cl*ustering) is an extension of pCluster.[10] By skipping trivial subclusters and pruning nonpromising cluster candidates earlier, it performs more quickly than pCluster (approximately twice as fast, on the basis of the experiments of Jian Pie and his colleagues[10]).

Both pCluster and MaPle run more slowly than MicroCluster, which is 1.4 to 20 times faster than pCluster, as we show in the main article. These methods don't delete or merge similar clusters, so they can't properly handle highly overlapping clusters with noise, which commonly occur in real data sets. In contrast, MicroCluster uses deletion and merging to handle these problems. Also, pCluster and MaPle both have an exponential runtime dependence on the number of genes, which is much larger than the number of samples. In contrast, MicroCluster can transpose the data sets and thus has an exponential runtime dependence only on the number of samples, not genes.

### References


1. S.C. Madeira and A.L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 1, no. 1, 2004, pp. 24–45.
2. R. Agrawal et al., "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. 1998 ACM SIGMOD Conf. Management of Data* (SIGMOD 98), ACM Press, 1998, pp. 94–105.
3. C.C. Aggarwal et al., "Fast Algorithms for Projected Clustering," *Proc. 1999 ACM SIGMOD Conf. Management of Data* (SIGMOD 99), ACM Press, 1999, pp. 61–72.
4. Y. Cheng and G.M. Church, "Biclustering of Expression Data," *Proc. 8th Int'l Conf. Intelligent Systems for Molecular Biology* (ISMB 00), ACM Press, 2000, pp. 93–103.
5. E.P. Xing and R.M. Karp, "CLIFF: Clustering High-Dim Microarray Data via Iterative Feature Filtering Using Normalized Cuts," *Bioinformatics*, vol. 17, suppl. 1, 2001, pp. S306–S315.
6. A. Tanay, R. Sharan, and R. Shamir, "Discovering Statistically Significant Biclusters in Gene Expression Data," *Bioinformatics*, vol. 18, suppl. 1, 2002, pp. S136–S144.
7. E. Hartuv et al., "An Algorithm for Clustering cDNAs for Gene Expression Analysis," *Proc. 3rd Ann. Int'l Conf. Computational Molecular Biology*, ACM Press, 1999, pp. 188–197.
8. T.M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," *Proc. Pacific Symp. Biocomputing* (Biocomputing 03), World Scientific Press, 2003, pp. 77–88; http://helix-web.stanford.edu/psb03/murali.pdf.
9. H. Wang et al., "Clustering by Pattern Similarity in Large Data Sets," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 02), ACM Press, 2002, pp. 394–405.
10. J. Pei et al., "MaPle: A Fast Algorithm for Maximal Pattern-Based Clustering," *Proc. 3rd IEEE Int'l Conf. Data Mining* (ICDM 03), IEEE CS Press, 2003, pp. 19–22.

($|G|$) or columns ($|S|$) in a data set, keeping these default parameters: 5 clusters, a cluster column size between 20 and 27 percent (that is, for $|S|$ total columns, a cluster will have a column size between $0.2|S|$ and $0.27|S|$), a cluster row size between 10 and 14 percent, 20 percent overlap, and 2 percent noise.

Figures 5f and 5g show the running time. If we fix $|S| = 35$ and vary $|G|$, the time increases linearly (figure 5f), but if we fix $|G| = 800$ and vary $|S|$, the time increases exponentially (figure 5g).

## Real data sets

We used four real data sets:

- *Yeast Elutriation* (1,536 × 14) (genome-www.stanford.edu/cellcycle/data/rawdata/individual.html),
- *Yeast* (2,884 × 17),[1]
- *CAMDA'04* (7,091 × 46) (www.camda.duke.edu/camda04), and
- *Cancer* (12,625 × 13) (http://pepr.cnmcresearch.org).

To analyze MicroCluster's performance on these sets, we defined four metrics:

- *Cluster#* is just $|C|$.
- *Element_Sum* is the sum of the spans of all clusters—that is, $Element\_Sum = \sum_{C \in C} |L_C|$.
- *Coverage* is the span of the union of all clusters—that is, $Coverage = \left|L_{\cup_{C \in C} C}\right|$.
- *Overlap* is $(Element\_Sum - Coverage)/Coverage$.

Element_Sum will count an overlapping element as many times as the number of clusters containing that element. Coverage counts each element only once, as long as it is part of some cluster. So, Overlap measures the degree of average overlap among the clusters. With these metrics, we can analyze the clustering results systematically.

***How parameter changes affect the output.*** We applied MicroCluster to a real subset (50 × 14) of the Yeast Elutriation data set with different input parameters to analyze how parameter changes affect the final output. The default parameters are $mc = 4$, $mr = 7$, $\varepsilon = 8$ percent, $\eta = 7$ percent, and $\gamma = 7$ percent. For each experiment, we kept all default parameter settings, except for the varying parameter as marked by the $x$-axis.

Figures 6a through 6c show that the original (before deletion and merging) Cluster# and Overlap vary more than linearly as $mr$, $mc$, or $\varepsilon$ changes. However, after deletion and merging, Cluster# and Overlap change gradually as $mr$, $mc$, or $\varepsilon$ changes. Coverage reflects the real information conveyed by the clusters, which changes little during deletion or merging, as figures 6a through 6c show. In other words, deletion and merging remove much redundant information while retaining good coverage. Figures 6d and 6e tell us that changing $\eta$ and $\gamma$ affects the final Cluster#, Overlap, and Coverage (after deletion and merging) approximately linearly.

Our experiments also confirmed MicroCluster's robustness. Between two sequential experiments (changing one parameter only), the second experiment's original clusters were always a subset of the first experiment's clusters. That is, MicroCluster generated a consistent cluster change. So, you can always modify the input parameters to get the most satisfying clustering (for example, compromising between the number of clusters found and the desired coverage).

> The clusters capture different aspects of process, function, and location, although there are some similarities due to the overlap among the clusters.

***Finding relevant clusters.*** Figure 7 illustrates four types of clusters that MicroCluster mined.

Figure 7a represents a bicluster mined on CAMDA'04.

From the Cancer data set, we tested MicroCluster on gene expression data of human cancer from 13 (six high-grade and seven low-grade) pediatric astrocytomas with 12,625 genes. (An astrocytoma is a nervous-system tumor beginning in the brain or spinal cord in small, star-shaped cells called *astrocytes*.) Our experiments show MicroCluster can separate these two kinds of samples very well. It outputs one cluster with 254 genes and six samples ($\varepsilon = 0.02$), all belonging to the low-grade class. Figure 7b shows the gene expression pattern.

Figures 7c and 7d are constant column and constant row clusters, respectively, mined on a subset of the Yeast data set. The color change (green-black-red) represents a value ranging from 60 (light green) to 140 (light red).

We compared MicroCluster with pCluster[2] and MaPle,[3] which are also deterministic algorithms. However, they capture only a subset of the bicluster patterns that MicroCluster does (that is, they can obtain the same clusters if we let $\delta^r = \delta^c = \infty$ and $\eta = \gamma = 0$). We used a Cygwin/WindowsXP PC with 768 MBytes of memory and a 1.4 GHz Pentium M processor. We obtained the pCluster and MaPle executable codes from their authors. We used small thresholds to get the same outputs from all methods: a cluster tolerance of $\delta = 1$ for pCluster and $\delta = 0$ for MaPle, and $\varepsilon = 0.1$ percent for MicroCluster. (For unknown reasons, when $\delta \geq 1$, MaPle runs very slowly—45 seconds for 8 × 50, 615 sec. for 8 × 45, and more than 2,000 sec. for other combinations in table 1.)

Table 1 shows that whereas all the algorithms found exactly the same clusters, MicroCluster is 1.4 to 20 times faster than pCluster and faster than MaPle in most cases for the unbalanced microarray data set. (For more on pCluster and MaPle, see the "Related Work on Subspace and Biclustering Algorithms" sidebar on the previous page.)

We also checked if any clusters that MicroCluster discovered share a common gene process, function, or cellular location, using the *Gene Ontology* (www.geneontology.org) project data. This project aims to develop three structured, controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components, and molecular functions, in a species-independent manner. We used the yeast genome gene ontology term finder (www.yeastgenome.org) to verify the biological significance of MicroCluster's result (four clusters obtained from the Yeast data set, with $\varepsilon = 0.01$, $mr = 60$, and $mc = 5$).

Table 2 shows the shared GO terms used to describe each cluster's set of genes. We display only the statistically significant shared terms with a $p$-value less than 0.01. Also, when multiple hierarchical terms involve the same group of genes, we report only the most significant terms. For example, for cluster $C_1$, we find significant genes involved in meiotic gene conversion and cell cycle checkpoint. We can see that the clusters capture different aspects of process,

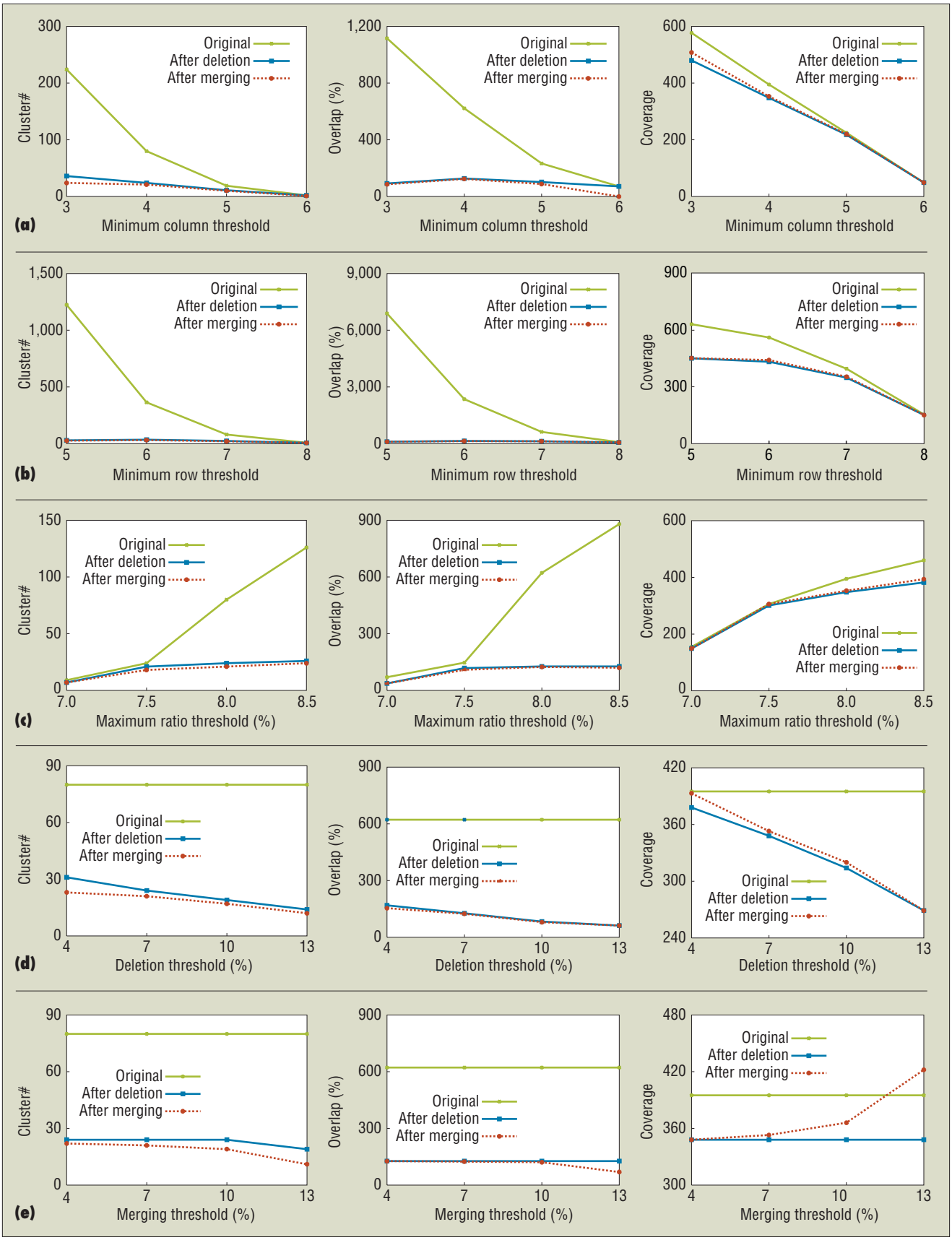



**Figure 6. How parameter changes affect MicroCluster's final output, for Cluster# (the number of clusters), Overlap, and Coverage: the (a) minimum column threshold, *mc*; (b) minimum row threshold, *mr*; (c) maximum ratio threshold, $\varepsilon$; (d) deletion threshold, $\eta$; and (e) merging threshold, $\gamma$.**

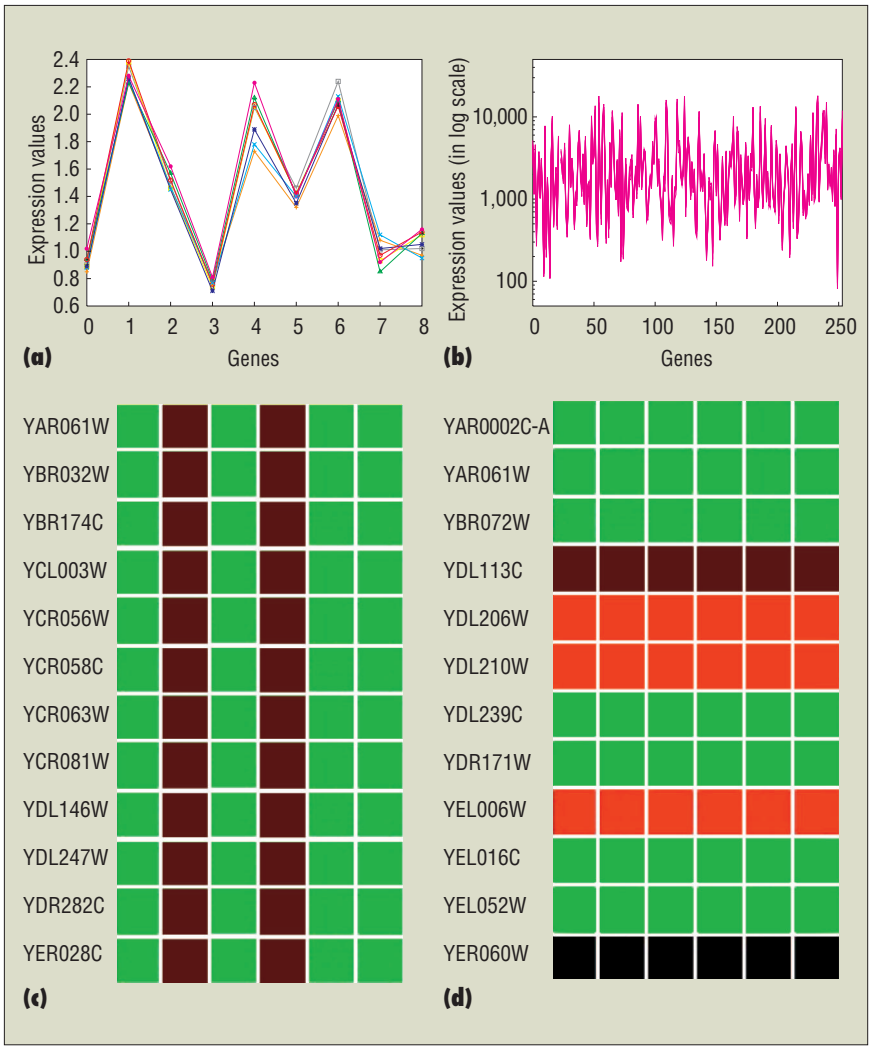



**Figure 7. Example clusters in the (a) CAMDA'04 and (b) Cancer data sets, and a (c) constant column and (d) constant row cluster. In figure 7d, the color change (green-black-red) represents a gene expression value ranging from 60 (light green) to 140 (light red).**

**Table 1. A comparison of MicroCluster (MC), pCluster (PC), and MaPle on the Yeast data set.**

| | | *mc** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 6 | | | 7 | | | 8 | | |
| *mr** | | MC | PC | MaPle | MC | PC | MaPle | MC | PC | MaPle |
| 40 | Runtime (sec.) | 639 | 883 | 228 | 18 | 150 | 80 | 13 | 27 | 34 |
| | Cluster# | | 1,195 | | | 106 | | | 5 | |
| | Element_Sum | | 333,502 | | | 32,782 | | | 1,688 | |
| | Coverage | | 6,456 | | | 2,234 | | | 682 | |
| 45 | Runtime (sec.) | 162 | 876 | 165 | 12 | 148 | 67 | 11 | 23 | 27 |
| | Cluster# | | 554 | | | 35 | | | 1 | |
| | Element_Sum | | 169,534 | | | 12,082 | | | 392 | |
| | Coverage | | 5,349 | | | 1,483 | | | 392 | |
| 50 | Runtime (sec.) | 43 | 871 | 138 | 11 | 146 | 58 | 10 | 22 | 20 |
| | Cluster# | | 243 | | | 11 | | | 0 | |
| | Element_Sum | | 81,456 | | | 4,200 | | | 0 | |
| | Coverage | | 4,178 | | | 961 | | | 0 | |

* *mr* is the minimum row cardinality threshold; *mc* is the minimum column cardinality threshold.

function, and location, although there are some similarities due to the overlap among the clusters. These clusters are related mainly to meosis, transport, and localization. Some genes are localized in membranes.

For comparison, we also ran SAMBA[4] and xMotif[5] on the same data set, also with *mr* = 60 and *mc* = 5. SAMBA found four highly similar clusters (each containing approximately 140 genes), any two of which overlap more than 97 percent. So, SAMBA actually obtained only one cluster containing 140 genes. xMotif found three clusters (containing 36, 86, and 120 genes, respectively) that had some similarities to our clusters. However, many genes reported in a cluster had "unknown biological process/function" as a significant GO term. Furthermore, xMotif gave different results each time, even with the same input parameters, because of its randomized nature. These results indicate that MicroCluster can find potentially biologically significant clusters where other approaches might not be so effective. (For more on SAMBA and xMotif, see the "Related Work on Subspace and Biclustering Algorithms" sidebar.)

Biclustering shows obvious advantages over traditional full-space clustering algorithms for microarray data. A good microarray-biclustering algorithm should have the basic properties of accuracy, noise handling, and feasible speed. To be accurate, MicroCluster mines multiple patterns and uses an enumeration method that guarantees not missing any qualified clusters, which stochastic or probabilistic algorithms can't do. To deal with microarray data's inherent noise, MicroCluster's merging and deletion stages control the noise tolerance in a cluster appropriately; their effectiveness is clearly illustrated by the performance evaluation metrics we defined in this article. Furthermore, MicroCluster gains a reasonable running time in two ways, which are verified on large data sets. First, it exploits microarray data's symmetry and enumerates the not-so-overwhelming sample combinations. Second, it applies multiple pruning techniques to speed up enumeration. To decrease the cost of cluster mining further, we plan to develop additional new techniques for pruning the search space. ■

Table 2. Significant shared Gene Ontology project terms (process, function, and component) for genes in different clusters.

| Cluster | Gene ontology term | Gene names | p-value |
|---|---|---|---|
| $C_1$ (62 × 5) | Process: meiotic gene conversion | SAE3, REC114 | 0.00740 |
| | Process: cell cycle checkpoint | KCC4, RFX1, CSM3 | 0.00768 |
| $C_2$ (62 × 5) | Process: meiotic DNA recombinase assembly | RAD57, SAE3 | 0.00086 |
| | Process: carbohydrate transport | MAL31, MPH2, YEA4 | 0.00265 |
| | Function: solute:cation symporter activity | MAL31, UGA4, FCY21 | 0.00012 |
| | Function: transporter activity | MAL31, UGA4, MPH2, YEA4, YEL006W, FCY21, YFL054C, PMC1, VHT1, QDR1, YMR034C | 0.00085 |
| | Function: carbohydrate transporter activity | MAL31, MPH2, YEA4 | 0.00201 |
| $C_3$ (60 × 5) | Process: transport, establishment of localization | ERP1, MAL31, ATG20, UGA4, MPH2, YEA4, YEL006W, YFL054C, PMC1, VHT1, MIP6, QDR1, PEP8, SAL1, LAS17, SYT1 | 0.0018 |
| | Process: response to drug | SNG1, SLI1, QDR1 | 0.00220 |
| | Function: solute:cation symporter activity | MAL31, UGA4, FCY21 | 0.00011 |
| | Function: transporter activity | MAL31, UGA4, MPH2, YEA4, YEL006W, FCY21, YFL054C, PMC1, VHT1, QDR1, YMR034C, SAL1 | 0.00015 |
| | Function: metal ion binding | SAL1, IZH4 | 0.00695 |
| | Location: membrane | ATG20, UGA4, ADY3, MPH2, COQ4, YEL006W, FCY21, YFL054C, PMC1, VHT1, SNG1, SLI1, MIP6, QDR1, PEP8, SAL1, IZH4 | 0.00203 |
| $C_4$ (61 × 5) | Process: carbohydrate transport | MAL31, MPH2, YEA4 | 0.00253 |
| | Process: polyamine transport | UGA4, TPO2 | 0.00327 |
| | Process: transport, establishment of localization | ERP1, SSA3, MAL31, ATG20, UGA4, MPH2, YEA4, YEL006W, YFL054C, PMC1, VHT1, TPO2, VMR1, QDR1, PEP5 | 0.00570 |
| | Function: transporter activity | MAL31, UGA4, MPH2, YEA4, YEL006W, FCY21, YFL054C, PMC1, VHT1, TPO2, VMR1, QDR1, YMR034C | 4.26e–05 |
| | Function: solute:cation symporter activity | MAL31, UGA4, FCY21 | 0.00011 |
| | Function: carbohydrate transporter activity | MAL31, MPH2, YEA4 | 0.00191 |
| | Function: polyamine transporter activity | UGA4, TPO2 | 0.00912 |
| | Location: vacuolar membrane | UGA4, PMC1, TPO2, PEP5 | 0.00871 |


## Acknowledgments

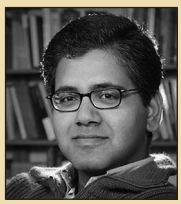
This work was supported in part by US National Science Foundation Career Award IIS-0092978, US Department of Energy Career Award DE-FG02-02ER25538, and NSF grants EIA-0103708 and EMT-0432098.


## References


1. Y. Cheng and G.M. Church, "Biclustering of Expression Data," *Proc. 8th Int'l Conf. Intelligent Systems for Molecular Biology* (ISMB 00), ACM Press, 2000, pp. 93–103.
2. H. Wang et al., "Clustering by Pattern Similarity in Large Data Sets," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 02), ACM Press, 2002, pp. 394–405.
3. J. Pei et al., "Maple: A Fast Algorithm for Maximal Pattern-Based Clustering," *Proc. 3rd IEEE Int'l Conf. Data Mining* (ICDM 03), IEEE CS Press, 2003, pp. 19–22.
4. A. Tanay, R. Sharan, and R. Shamir, "Discovering Statistically Significant Biclusters in Gene Expression Data," *Bioinformatics*, vol. 18, suppl. 1, 2002, pp. S136–S144.
5. T.M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," *Proc. Pacific Symp. Biocomputing* (Biocomputing 03), World Scientific Press, 2003, pp. 77–88; http://helix-web.stanford.edu/psb03/murali.pdf.




## The Authors




**Lizhuang Zhao** is a PhD student of computer science at Rensselaer Polytechnic Institute. His research interests are data mining and bioinformatics—for example, subspace clustering of microarray gene expression data and logical-relationship analysis of binary-valued Boolean data sets. He has received master's degrees in computer science from both the Harbin Institute of Technology and the Rensselaer Polytechnic Institute. He's a student member of ACM SIGMOD. Contact him at the Dept. of Computer Science, Rensselaer Polytechnic Inst., Troy, NY 12180; zhaol2@cs.rpi.edu.

**Mohammed J. Zaki** is an associate professor of computer science at Rensselaer Polytechnic Institute. His research interests involve developing novel data mining techniques, with applications in bioinformatics, Web mining, and so on. He is an action editor for *Data Mining and Knowledge Discovery: An International Journal*, is an associate editor for *IEEE Transactions on Knowledge and Data Engineering*, and is on the editorial board of the *International Journal of Data Warehousing and Mining*, the *International Journal of Data Mining and Bioinformatics*, and *Scientific Programming*. He received his PhD in computer science from the University of Rochester. Contact him at the Dept. of Computer Science, Rensselaer Polytechnic Inst., Troy, NY 12180; zaki@cs.rpi.edu.