JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 15, Number 10, 2008
© Mary Ann Liebert, Inc.
Pp. 1275–1293
DOI: 10.1089/cmb.2007.0219

An Efficient Voting Algorithm for Finding Additive Biclusters with Random Background

JING XIAO,[1] LUSHENG WANG,[2] XIAOWEN LIU,[2] and TAO JIANG[1,3]

ABSTRACT

The biclustering problem has been extensively studied in many areas, including e-commerce, data mining, machine learning, pattern recognition, statistics, and, more recently, computational biology. Given an $n \times m$ matrix A $(n \geq m)$, the main goal of biclustering is to identify a subset of rows (called objects) and a subset of columns (called properties) such that some objective function that specifies the quality of the found bicluster (formed by the subsets of rows and of columns of A) is optimized. The problem has been proved or conjectured to be NP-hard for various objective functions. In this article, we study a probabilistic model for the implanted additive bicluster problem, where each element in the $n \times m$ background matrix is a random integer from $[0, L-1]$ for some integer L, and a $k \times k$ implanted additive bicluster is obtained from an error-free additive bicluster by randomly changing each element to a number in $[0, L-1]$ with probability θ. We propose an $O(n^2m)$ time algorithm based on voting to solve the problem. We show that when $k \geq \Omega(\sqrt{n \log n})$, the voting algorithm can correctly find the implanted bicluster with probability at least $1 - \frac{9}{n^2}$. We also implement our algorithm as a C++ program named VOTE. The implementation incorporates several ideas for estimating the size of an implanted bicluster, adjusting the threshold in voting, dealing with small biclusters, and dealing with overlapping implanted biclusters. Our experimental results on both simulated and real datasets show that VOTE can find biclusters with a high accuracy and speed.

Key words: additive bicluster, computational biology, gene expression data analysis, polynomial-time algorithm, probability model.

1. INTRODUCTION

Biclustering has proved extremely useful for exploratory data analysis. It has important applications in many fields, for example, e-commerce, data mining, machine learning, pattern recognition, statistics, and computational biology (Yang et al., 2002). Data arising from, for example, text analysis, market-basket data analysis, web logs, and microarray experiments are usually arranged in a co-occurrence

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China.
[2] Department of Computer Science, City University of Hong Kong, Hong Kong.
[3] Department of Computer Science and Engineering, University of California, Riverside, California.

table or matrix, such as a word-document table, product-user table, cpu-job table, or webpage-user table. Discovering a large bicluster in a product-user matrix indicates, for example, which users share the same preferences. Biclustering has therefore applications in recommender systems and collaborative filtering, identifying web communities, load balancing, discovering association rules, etc.

Recently, biclustering has become an important approach to microarray gene expression data analysis (Cheng and Church, 2000). The underlying bases for using biclustering in the analysis of gene expression data are as follows: (i) similar genes may exhibit similar behaviors only under a subset of conditions instead of all conditions, and (ii) genes may participate in more than one function, resulting in a regulation pattern in one context and a different pattern in another. Using biclustering algorithms, one may obtain subsets of genes that are co-regulated under certain subsets of conditions.

Given an $n \times m$ matrix A, the main goal of biclustering is to identify a subset of rows (called *objects*) and a subset of columns (called *properties*) such that a pre-determined objective function which specifies the quality of the bicluster (consisting of the found subsets of rows and columns) is optimized. Biclustering is also known under several different names, for example, "co-clustering," "two-way clustering," and "direct clustering." The problem was first introduced by Hartigan in the 1970s (Hartigan, 1972). Since then, it has been extensively studied in many areas and many approaches have been introduced. Several objective functions have also been proposed for measuring the quality of a bicluster; almost all of them have been proved or conjectured to be NP-hard (Lonardi et al., 2004; Peeters, 2003).

Let $A(I, J)$ be an $n \times m$ $(n \geq m)$ matrix, where $I = \{1, 2, \ldots, n\}$ is the set of rows representing the genes and $J = \{1, 2, \ldots, m\}$ is the set of columns representing conditions. In practice, the number of genes is much bigger than the number of conditions. Each element $a_{i,j}$ of $A(I, J)$ is an integer in $[0, L-1]$ indicating the weight of the relationship between object i and property j. A *bicluster* of $A(I, J)$ is a submatrix of $A(I, J)$. For any given subset $I' \subseteq I$ and subset $J' \subseteq J$, $A(I', J')$ denotes the bicluster of $A(I, J)$ that contains only the elements $a_{i,j}$ satisfying $i \in I'$ and $j \in J'$. When a bicluster contains only a single row i and a column set J', we simply use $A(i, J')$ to represent it. Similarly, we use $A(I', j)$ to represent the bicluster consisting of a row set I' and a single column j.

The following are two popular models of biclusters that assume different relationships between objectives (or genes) (Yang et al., 2002).

Constant model: A bicluster $A(I', J')$ is an *error-free constant* bicluster if for each column $j \in J'$, $a_{i,j} = c_j$ for all $i \in I'$, where c_j is a constant for any column j.

Additive model: A bicluster $A(I', J')$ is an *error-free additive* bicluster if for any pair of rows i_1 and i_2 in $A(I', J')$, $a_{i_1,j} - a_{i_2,j} = c_{i_1,i_2}$ for every column j, where c_{i_1,i_2} is a constant for any pair of rows i_1 and i_2.

Clearly, the additive model is more general than the constant model. In microarray gene expression analysis, the additive model can be used to capture groups of genes whose expression levels change in the same/simlar way under the same set of conditions (Madeira and Oliveira, 2004). The additive model also covers several other popular models in the literature as its special cases. For example, a multiplicative bicluster B is a submatrix that looks like:

$$\begin{pmatrix} b_{1,1} & b_{1,2}, \ldots, & b_{1,m'} \\ c_1 b_{1,1} & c_1 b_{1,2}, \ldots, & c_1 b_{1,m'} \\ c_2 b_{1,1} & c_2 b_{1,2}, \ldots, & c_2 b_{1,m'} \\ & \cdots & \\ c_{n'-1} b_{1,1} & c_{n'-1} b_{1,2}, \ldots, & c_{n'-1} b_{1,m'} \end{pmatrix},$$

where each row can be obtained from another by multiplying a number (Madeira and Oliveira, 2004). If we replace each element $b_{1,j}$ (or $c_i b_{1,j}$) with $\log b_{1,j}$ (or $\log(c_i b_{1,j})$, respectively), we get an additive bicluster. For a detailed discussion on various models of biclusters, see Madeira and Oliveira (2004). The additive model has many applications and has been extensively studied in the literature (Barkow et al., 2006; Kluger et al., 2003; Li et al., 2006; Liu and Wang, 2007; Lonardi et al., 2004; Madeira and Oliveira, 2004; Peeters, 2003; Prelić et al., 2006). It was first implicitly applied to microarray gene expression analysis by Cheng and Church (2000), who proposed the mean squared residue score H to measure the coherence of the rows and columns in a bicluster. It is easy to show that for an error-free additive bicluster, its score H is 0. Several efficient heuristic algorithms for solving the additive model were in Prelić et al. (2006) and Yang et al. (2002). Recently, Liu and Wang (2007) proposed the maximum similarity score to measure the quality of an additive bicluster. They designed an algorithm that runs in $O(nm(n+m)^2)$ time to find an optimal solution under such a score. To our best knowledge, this is the first score admitting a polynomial time algorithm for additive biclusters.

In this article, we will focus on the additive model for biclusters. In particular, we study a probabilistic model, in which the background matrix and a size $k \times k$ additive bicluster are generated based on certain probability methods, and then we implant the additive bicluster by replacing a size $k \times k$ submatrix of the background matrix with the $k \times k$ additive bicluster. This probabilistic model has recently been used in the literature for evaluating biclustering algorithms (Liu and Wang, 2007; Prelić et al., 2006).

The probabilistic additive model: More precisely, our probabilistic model for generating the implanted bicluster and background matrix is as follows. Let $A(I, J)$ be an $n \times m$ matrix, where each element $a_{i,j}$ is a random number in $[0, L-1]$ generated independently. Let B be an error-free $k \times k$ additive bicluster. The additive bicluster B' with noise is generated from B by changing each element $b_{i,j}$, with probability θ, into a random number in $[0, L-1]$. We then implant B' into the background matrix $A(I, J)$ and randomly shuffle its rows and columns to obtain a new matrix $A'(I, J)$. For convenience, we will still denote the elements of $A'(I, J)$ as $a_{i,j}$'s.

From now on, we will consider the matrix $A'(I, J)$ as the input matrix. Let $I_B \subseteq I$ and $J_B \subseteq J$ be the row and column sets of the implanted bicluster in A'. The implanted bicluster is denoted as $A'(I_B, J_B)$.

The implanted additive bicluster problem: Given the $n \times m$ matrix $A'(I, J)$ with an additive bicluster implanted as described above, find the implanted additive bicluster B'.

Our results: We propose an $O(n^2m)$ time algorithm for finding an implanted bicluster based on a simple voting technique. We show that when $k \geq \Omega(\sqrt{n \log n})$, the voting algorithm can correctly find the implanted bicluster with probability at least $1 - 9n^{-2}$. We also implement our algorithm as a C++ program named VOTE. In order to make the program applicable in a real setting, the implementation has to incorporate several ideas for estimating the size of an implanted bicluster, adjusting the threshold in voting, dealing with small biclusters, and dealing with overlapping biclusters. Our experiments on both simulated and real datasets show that VOTE can find implanted additive biclusters with high accuracy and efficiency. More specifically, VOTE has a performance/accuracy comparable to the best programs that were recently compared in the literature (Prelić et al., 2006; Liu and Wang, 2007), but with a much faster speed.

To our knowledge, the work in bioinformatics that is the most related to our above result is the work of Ben-Dor et al. (1999) concerning the clustering of gene expression patterns. In the article, they studied a probabilistic graph model, where each gene is a vertex in a *clique graph* H and each group of related genes form a clique in H. The (error-free) clique graph consists of d disjoint cliques, and the input graph is obtained from the clique graph H by (1) removing each edge in H with probability $\alpha < 0.5$ and (2) adding each edge not in H with probability $\alpha < 0.5$. They designed an algorithm that can successfully recover the original clique graph H with a high probability. Due to the difference in the models, our voting algorithm is totally different from their algorithm.

We note in passing that the problem of finding an implanted clique/distribution in a random graph has also been studied in the graph theory community (Alon et al., 1998; Feige and Krauthgamer, 2000; Kucera, 1995). Kucera Kucera (1995) claimed that when the size of the implanted clique is at least $\Omega(\sqrt{n \log n})$, where n is the number of vertices in the input random graph, a simple approach based on counting the degrees of vertices can find the clique with a high probability. Alon et al. (1998) gave an improved algorithm that can find an implanted clique of size at least $\Omega(\sqrt{n})$ with a high probability. Feige and Krauthgamer (2000) gave an algorithm that can find implanted cliques of similar sizes in semi-random graphs. It is easy to see that this problem of finding implanted cliques is a special case of our implanted bicluster problem, where the input matrix is binary and all the elements in the bicluster matrix are 1's. We observe that while it may be easy to modify Kucera's simple degree-based method to work for implanted constant biclusters under our probabilistic model, it is not obvious that the above results would directly imply our results on implanted additive biclusters. Moreover, these methods cannot easily be extended to discover multiple cliques/biclusters as often required in practice.

In the rest of the article, we first present the voting algorithm and analyze its theoretical performance on the above probabilistic model. We then describe the implementation of the C++ program VOTE, and present the experimental results. Some concluding remarks are given at the end. For the convenience of the reader, the proofs of some of the technical lemmas in the theoretical analysis will be deferred to the Appendix.

2. THE THREE-PHASE VOTING ALGORITHM

We start the construction of the algorithm with some interesting observations. Recall that B is an error-free $k \times k$ additive bicluster and A' is the random input matrix with a noisy additive bicluster B' implanted.

Observation 1. *Consider the k rows in B. There are at least $\frac{k}{L}$ rows that are identical. This implies that there exists a row set $I_C \subseteq I_B$ with $|I_C| \geq \frac{k}{L}$ such that $A'(I_C, J_B)$ is a* constant *bicluster with noise.*

Consider a row $i_1 \in I_B$ and a column $j_1 \in J_B$. For each row $i_2 \in I_B$, $c_{i_1,i_2} = a_{i_1,j_1} - a_{i_2,j_1}$ is an integer in $[a_{i_1,j_1} - L + 1, a_{i_1,j_1}]$. Based on the value c_{i_1,i_2}, we can partition I_B into L different row sets $I_B^d = \{i_2 | i_2 \in I_B \;\&\; c_{i_1,i_2} = d\}$, $d = a_{i_1,j_1} - L + 1, \ldots, a_{i_1,j_1}$. Let I_C be one of the row sets with the maximum cardinality, $|I_C| = \max_d |I_B^d|$. Then, $A'(I_C, J_B)$ is a constant bicluster (since $a_{i,j} = a_{i',j}$ for any $\{i, i'\} \subseteq I_C$ and $j \in J_B$) and $|I_C| \geq \frac{k}{L}$. Let $|I_C| = l$.

Our algorithm has three phases. In the first phase of the algorithm, we want to find the row set I_C in $A'(I, J)$. In order to vote, we first convert the matrix $A'(I, J)$ into a *distance* matrix $D(I, J)$ containing the same sets of rows and columns, and then focus on $D(I, J)$.

Distance matrix. Given an $n \times m$ matrix $A'(I, J)$, we can convert it into a distance matrix based on a row in the matrix. Let $i^* \in I$ be any row in the matrix A. We refer to row i^* as the *reference* row. Define $d_{i,j} = a_{i,j} - a_{i^*,j}$. In the transformation, we subtract the reference row i^* from every row in $A'(I, J)$. We use $D(I, J)$ to denote the $n \times m$ distance matrix containing the set of rows I and the set of columns J with every element $d_{i,j}$. For a row $i \in I$ and a column set $J' \subseteq J$, the number of occurrences of u, $u \in [-L + 1, L - 1]$, in $D(i, J')$ is the number of elements with value u in $D(i, J')$, denoted by $f(i, J', u) = |\{d_{i,j} | d_{i,j} = u \;\&\; j \in J'\}|$. The number of occurrences of the element that appears the most in $D(i, J')$ is $f^*(i, J') = \max_u f(i, J', u)$. Similarly, for a row set $I' \subseteq I$ and a column $j \in J$, the number of occurrences of u in $D(I', j)$ is the number of elements with value u in $D(I', j)$, denoted by $f(I', j, u)$. The number of occurrences of the element that appears the most in $D(I', j)$ is $f^*(I', j) = \max_u f(I', j, u)$.

Observation 2. *Suppose that we use a row $i^* \in I_C$ as the reference row. For each row i in I_C, the expected number of* 0*'s in row i of $D(I, J)$ is at least $\frac{m-k}{L} + (1 - \theta)^2 k$. For each row i in $I_B - I_C$, the expected number of* 0*'s in row i of $D(I, J)$ is at most $\frac{m-k}{L} + \frac{2\theta k}{L}$. For each row i in $I - I_B$, the expected number of* 0*'s in row i of $D(I, J)$ is at most $\frac{m-k}{L} + \frac{k}{L}$.*

Proof. For each row i in I_C, there are k elements from B' and $m - k$ elements from A. Based on the model, among the k elements in B', the expected number of 0's in row i of $D(I, J)$ is at least $(1 - \theta)^2 k$ (since in both row i and row i^*, there are at least $(1 - \theta)k$ of the k elements remaining unchanged) and the expected number of 0's among the $m - k$ elements in A is $\frac{m-k}{L}$. Similarly, for each row $i \in I_B - I_C$, there are k elements from B' and $m - k$ elements from A. Among the k elements in B', if the element $d_{i,j}$ is 0 in $D(I, J)$, then it must be one of the three cases: (1) $a_{i,j}$ is changed to $a_{i^*,j}$ and $a_{i^*,j}$ remains the same, (2) $a_{i^*,j}$ is changed to $a_{i,j}$ and a_{ij} remains the same, and (3) both $a_{i,j}$ and $a_{i^*,j}$ are changed to an identical number in $[0, L - 1]$. The expected numbers for the three cases are $\frac{\theta(1-\theta)k}{L}$, $\frac{\theta(1-\theta)k}{L}$ and $\frac{\theta^2 k}{L}$, respectively. The total expected number of 0's for the three cases is at most $\frac{2\theta k}{L}$. For the $m - k$ elements in A, the expected number of 0's is $\frac{m-k}{L}$. Therefore, for each row i in $I_B - I_C$, the expected number of 0's in row i of $D(I, J)$ is at most $\frac{m-k}{L} + \frac{2\theta k}{L}$. For each row i in $I - I_B$, the expected number of 0's in row i of $D(I, J)$ is at most $\frac{m}{L}$ since the probability that each element $a_{i,j}$ in row i is identical to $a_{i^*,j}$ is $\frac{1}{L}$. ■

Based on the observation, the expected number of 0's in each row of I_C, which is at least $\frac{m-k}{L}+(1-\theta)^2k$, is much more than that in the other rows. Therefore, with a high probability, the rows with the most 0's are in I_C as long as the reference row i^* is in I_C. Note that, $\frac{m-k}{L}+(1-\theta)^2k = \frac{m}{L}+[(1-\theta)^2-\frac{1}{L}]k$ and the voting algorithm works when $k \geq \Omega(\sqrt{m\log m})$. Thus, we will show that $\frac{m}{L}+4\sqrt{m\log m}$ is a good threshold on the number of 0's to differentiate the rows in I_C from the rows that are not in I_C. More specifically, we can use this threshold to find a row set I_0 by applying the following voting method.

The first phase voting:

```
1. for $i = 1$ to $n$ do
2.       compute $f(i, J, 0)$.
3. select rows $i$ such that $f(i, J, 0) > \frac{m}{L} + 4\sqrt{m\log m}$ to form $I_0$.
```

When m and k are sufficiently large and θ is sufficiently small, we can prove that, with high probability, the row set I_0 is equal to I_C. The proof will be given in the next section (in Lemma 4 and the discussion following its proof). If we cannot find any row i such that $f(i, J, 0) > \frac{m}{L}+4\sqrt{m\log m}$, then the whole algorithm will not output any bicluster. However, this has never happened in our experiments.

In the second phase voting of the algorithm, we attempt to find the column set J_B of the implanted bicluster. It is based on the following observation.

Observation 3. *For a column j in J_B, the expected number of occurrences of the element that appears the most in $D(I_C, j)$ is at least $(1-\theta)|I_C|$. For a column j in $J - J_B$, the expected number of occurrences of an element u in $D(I_C, j)$ is at most $\frac{1}{L}|I_C|$.*

Proof. In the error free matrix B, all rows in I_C are identical. For a column $j \in J_B$, the corresponding column in B has the same element, say u, in all rows I_C. After adding noise with probability θ, the expected number of unchanged u's is $(1-\theta)|I_C|$. Therefore, in the column $D(I_C, j)$, the expected number of occurrences of $u - a_{i^*,j}$ is at least $(1-\theta)|I_C|$.

For a column $j \in J - J_B$ and a row $i \in I_C$, if $a_{i^*,j_2} + u \in 1, 2, \ldots, L-1$, the probability that $a_{i,j_2} = a_{i^*,j_2} + u$ is $1/L$, otherwise, the probability is 0. Therefore, the expected number of occurrences of u in $D(I_C, j_2)$ is at most $\frac{1}{L}|I_C|$. ■

With high probability (and again assuming that θ is sufficiently small), the number of occurrences of the element that appears the most in the columns of J_B is greater than the number of occurrences of the element that appears the most in the columns of $J - J_B$. That is, for two columns $j_1 \in J_B$ and $j_2 \notin J_B$, with high probability, $f^*(I_0, j_1) > \frac{|I_0|}{2} > f^*(I_0, j_2)$. Based on the property, we can use voting to find a column set J_1.

The second phase voting:

```
1. for $j = 1$ to $m$ do
2.       compute $f^*(I_0, j)$.
3. select columns $j$ such that $f^*(I_0, j) > \frac{|I_0|}{2}$ to form $J_1$.
```

We can prove (in the next section) that, with high probability, J_1 is equal to the implanted column set J_B.

Similarly, the third phase voting of the algorithm is designed to locate the row set I_B of the implanted bicluster. But, before the voting, we need to correct corrupted columns of the distance matrix $D(I, J)$ caused by the elements of the reference row i^* that were changed during the generation of B'. Recall that $f^*(I_0, j) = \max_u f(I_0, j, u)$. Let $f(I_0, j, u_j) = f^*(I_0, j)$. For every $j \in J_1$, if $u_j \neq 0$, then the element $a_{i*,j}$ was changed when B' was generated (assuming $J_1 = J_B$), and we can thus correct each element $d_{i,j}$ in the jth column of the matrix $D(I, J)$ by subtracting u_j from it.

In the following, let us assume that the entries in the submatrix $D(I, J_B)$ have been adjusted according to the correct reference row i^* as described above. The following observation holds.

```
The Three Phase Voting Algorithm
Input:  An $n \times m$ matrix $A'(I, J)$, an integer $k$, noise level $\theta$, and $L$.
Output: A bicluster $A'(I_1, J_1)$.
1. for each row $i^* \in I$, do
2.     construct the $n \times m$ distance matrix $D(I, J)$ from $A'(I, J)$ with reference row $i^*$.
3.     find a row set $I_0$ by using the first phase voting.
4.     if $|I_0| \geq \frac{k}{L}$, then
5.         find a column set $J_1$ by using the second phase voting.
6.         correct the corrupted columns in submatrix $D(I, J_1)$.
7.         find a row set $I_1$ by using the third phase voting.
8.         if $|I_1| \geq k$ and $|J_1| \geq k$, output $A'(I_1, J_1)$ and return.
```

FIG. 1. The three-phase voting algorithm.

Observation 4. *For a row i in I_B, the expected number of occurrences of the element that appears the most in $D(i, J_B)$ is at least $(1 - \theta)k$. For a row i in $I - I_B$, the expected number of occurrences of the element that appears the most in $D(i, j_B)$ is $\frac{k}{L}$.*

We can thus find a row set I_1 in $A'(I, J_1)$ as follows.

The third phase voting:

```
1. for $i = 1$ to $n$ do
2.     compute $f^*(i, J_1)$.
3. select rows $i$ such that $f^*(i, J_1) > \frac{|J_1|}{2}$ to form $I_1$.
```

We can prove (in the next section) that, if $|I_1| \geq k$, with high probability, I_1 is equal to the implanted row set I_B. Therefore, a voting algorithm based on the above procedures, as given in Figure 1, can be used to find the implanted bicluster with high probability. Since the time complexity of the steps 2–7 of the algorithm is $O(nm)$ and these steps are repeated n times, the time complexity of the algorithm is $O(n^2m)$. Note that, if for any phase, we cannot find any row or column that satisfies the bounds, then the algorithm will not output any bicluster. However, this has never happened in our experiments.

3. ANALYSIS OF THE ALGORITHM

In this section, we will prove that, with high probability, the above voting algorithm correctly outputs the implanted bicluster.

Recall that in the submatrix $A'(I_B, J_B)$, each element was changed with probability θ to generate B' from B. We will show that, with high probability, there exists a row $i \in I_C$ such that row i has at least $(1 - \delta)(1 - \theta)k$ unchanged elements in $A'(i, J_B)$ for any $0 < \delta < 1$.

In the analysis, we need the following two lemmas from Li et al. (2002) and Motwani and Raghavan (1995).

Lemma 1 (Motwani and Raghavan, 1995). *Let $X_1, X_2, \ldots, X_n$ be n independent random binary (0 or 1) variables, where X_i takes on the value of 1 with probability p_i, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then for any $0 < \delta < 1$,*

(1) $\mathbf{Pr}(X > (1 + \delta)\mu) < [\frac{\mathrm{e}^{\delta}}{(1+\delta)^{(1+\delta)}}]^{\mu}$,

(2) $\mathbf{Pr}(X < (1 - \delta)\mu) \leq e^{-\frac{1}{2}\mu\delta^2}$.

Lemma 2 (Li et al., 2002). *Let X_i, $1 \le i \le n$, X and μ be defined as in Lemma 1. Then for any* $0 < \epsilon < 1$,

(1) $\mathbf{Pr}(X > \mu + \epsilon n) \le e^{-\frac{1}{3}n\epsilon^2}$,
(2) $\mathbf{Pr}(X < \mu - \epsilon n) \le e^{-\frac{1}{2}n\epsilon^2}$.

Lemma 3. *For any* $0 < \delta < 1$, *with probability at least* $1 - e^{-\frac{1}{2L}(1-\theta)k^2\delta^2}$, *there exists a row* $i \in I_C$ *that has at least* $(1-\delta)(1-\theta)k$ *unchanged elements in* $A'(i, J_B)$.

Proof. Let $x_{i,j}$ be a $0/1$ random variable, where $x_{i,j} = 0$ if $a_{i,j}$ is changed in generating B', and $x_{i,j} = 1$ otherwise. Based on the probabilistic model, the expected value of $\sum_{i \in I_C} \sum_{j \in J_B} x_{i,j}$ is $(1-\theta)kl$. By Lemma 1 and $l \ge \frac{k}{L}$,

$$Pr\left(\sum_{i \in I_C} \sum_{j \in J_B} x_{i,j} < (1-\delta)(1-\theta)kl\right) \le e^{-\frac{1}{2}(1-\theta)kl\delta^2} \le e^{-\frac{1}{2L}(1-\theta)k^2\delta^2}.$$

Note that, if $\sum_{j \in J_B} x_{i,j} < (1-\delta)(1-\theta)k$ for all rows $i \in I_C$, then we have $\sum_{i \in I_C} \sum_{j \in J_B} x_{i,j} < (1-\delta)(1-\theta)kl$. Thus,

$$Pr\left(\forall i \in I_C, \sum_{j \in J_B} x_{i,j} < (1-\delta)(1-\theta)k\right) \le Pr\left(\sum_{i \in I_C} \sum_{j \in J_B} x_{i,j} < (1-\delta)(1-\theta)kl\right)$$
$$\le e^{-\frac{1}{2L}(1-\theta)k^2\delta^2}. \quad (1)$$

The inequality (1) implies the lemma. ■

Suppose that there is a row $i^* \in I_C$ with $(1-\delta)(1-\theta)k$ unchanged elements in $A'(i^*, J_B)$. Now, let us consider the distance matrix $D(I, J)$ with the reference row i^*. We now show that, with high probability, the rows in I_C have more 0's than those in $I - I_C$ in matrix $D(I, J)$. That is, with high probability, our algorithm will find the row set I_C in the first phase voting.

Lemma 4. *Let* $i^* \in I_C$ *be the reference row with* $(1-\delta)(1-\theta)k$ *unchanged elements in* $A'(i^*, J_B)$, *and* $D(I, J)$ *the distance matrix as described in Section 2. When* $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ *and* $k \ge \frac{8}{\alpha}\sqrt{m \log m}$, *with probability at least* $1 - m^{-7} - nm^{-5}$, $f(i, J, 0) > \frac{m}{L} + \frac{\alpha}{2}k$ *for all* $i \in I_C$, *and* $f(i, J, 0) < \frac{m}{L} + \frac{\alpha}{2}k$ *for all* $i \in I - I_C$,

Proof. Let J_C be a subset of J_B such that $|J_C| = (1-\delta)(1-\theta)k$, and for all $j \in J_C$, $a_{i^*,j}$ is unchanged. Consider a row $i \in I_C$, $i \ne i^*$. Let $X_1, X_2, \ldots, X_m$ be m random variables. For each random variable X_j, $X_j = 1$ if $d_{i,j} = 0$, otherwise, $X_j = 0$. Then, $f(i, J, 0) = \sum_{j=0}^{m} X_j$. We consider two different column sets: J_C and $J - J_B$. (1) For $j \in J_C$, we have $Pr(X_j = 1) = 1 - \theta$ and $Pr(X_j = 0) = \theta$. The expectation of X_j is $\mu_j = 1 - \theta$. (2) For $j \in J - J_B$, $Pr(X_j = 1) = \frac{1}{L}$ and $Pr(X_j = 0) = 1 - \frac{1}{L}$. The expectation of X_j is $\mu_j = \frac{1}{L}$. Let $J_D = J_C \cup (J - J_B)$. From the above analysis,

$$\sum_{j \in J} \mu_j \ge \sum_{j \in J_D} \mu_j = [(1-\delta)(1-\theta)k](1-\theta) + \frac{m-k}{L} = \frac{m}{L} + \alpha k. \quad (2)$$

By the definition, the random variables $X_1, X_2, \ldots, X_m$ in row i are independent. By Lemma 2,

$$\begin{aligned} Pr\left(f(i,J,0) < \frac{m}{L} + \frac{\alpha}{2}k\right) &= Pr\left(\sum_{j\in J} X_j < \left(\frac{m}{L} + \alpha k\right) - \frac{\alpha}{2}k\right) \\ &\le Pr\left(\sum_{j\in J} X_j < \sum_{j\in J}\mu_j - 4\sqrt{\frac{\log m}{m}}m\right) \\ &\le e^{-8\log m} \\ &\le m^{-8}. \end{aligned} \tag{3}$$

Since $|I_C| \le k$, the probability that there exists a row in I_C with no more than $(\frac{m}{L} + \frac{\alpha}{2}k)$ 0's in $D(I,J)$ is at most $km^{-8} \le m^{-7}$.

Now, we consider a row $i \in I - I_C$. Let $Y_1, Y_2, \ldots, Y_m$ be m random variables. For each random variable Y_j, $Y_j = 1$ if $d_{i,j} = 0$, otherwise, $Y_j = 0$. We have $Pr(Y_j = 1) = \frac{1}{L}$. The expectation of Y_j is $\mu_j = \frac{1}{L}$. From the analysis, we have $\sum_{j\in J}\mu_j = \frac{m}{L}$. The random variable $Y_1, Y_2, \ldots, Y_m$ in row i are also independent. By Lemma 2,

$$Pr\left(\sum_{j\in J} Y_j > \frac{m}{L} + m\epsilon\right) \le \exp\left(-\frac{1}{3}m\epsilon^2\right).$$

Now let $\epsilon = 4\sqrt{\frac{\log m}{m}}$, we get

$$Pr\left(f(i,J,0) > \frac{m}{L} + \frac{\alpha}{2}k\right) \le Pr\left(\sum_{j\in J} Y_j \ge \frac{m}{L} + m\epsilon\right) \le m^{-16/3} \le m^{-5}.$$

Since $|I - I_B| \le n$, the probability that there exists a row in $I - I_B$ with at least $(\frac{m}{L} + \frac{\alpha}{2}k)$ 0's in $D(I,J)$ is at most nm^{-5}. Therefore, the lemma holds. ■

The above lemma shows that, when a row i^* with $(1-\delta)(1-\theta)k$ unchanged elements in $A'(i, J_B)$ is selected as the reference row, and m and k are large enough, $I_0 = I_C$ with high probability. Next, we prove that, with high probability, our algorithm will find the implanted column set J_B.

Lemma 5. *Suppose that the row set I_0 found in the first phase voting of Algorithm 1 is indeed equal to I_C. With probability at least $1 - ke^{-\frac{(1-2\theta)^2}{8L}k} - L(m-k)e^{-\frac{(L-2)^2}{12L^3}k}$, the column set J_1 found in the second phase voting of Algorithm 1 is equal to J_B.*

Proof. The ideas of the proof is the same as those in the above lemma. For the benefit of readability, we defer the proof to the Appendix. ■

Similarly, we can prove that, with high probability, our algorithm will find the implanted row set I_B.

Lemma 6. *Suppose that the column set J_1 found in the second phase voting of Algorithm 1 is indeed equal to J_B. With probability at least $1 - ke^{-\frac{(1-2\theta)^2}{8}k} - 2L(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$, the row set I_1 found in the third phase voting of Algorithm 1 is equal to I_B.*

Proof. Again, for the sake of readability, we defer the proof to the Appendix. ■

Finally, we can prove that, with high probability, no column or row other than those in the implanted bicluster will be output by the voting algorithm.

Lemma 7. *With probability at least* $1 - Ln(m-k)e^{-\frac{(L-2)^2}{12L^3}k} - 2Ln(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$*, no column or row of* $A'(I, J)$ *other than those in* $A'(I_B, J_B)$ *will be output by the Algorithm 1.*

Proof. See the Appendix. ■

Based on Lemmas 3, 4, 5, 6, and 7, we can show that, when m and k are large enough, the three-phase voting algorithm can find the implanted bicluster with high probability. Let c be a constant such that $c < \min\{\frac{(1-\theta)\delta^2 k}{2L}, \frac{(1-2\theta)^2}{8L}, \frac{(L-2)^2}{12L^3}\}$. In most applications, we may assume that $n < m^3$. Then, we have the following theorem.

Theorem 8. *When* $n < m^3$, $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ *and* $k \geq \max\{\frac{8}{\alpha}\sqrt{m \log m}, \frac{8 \log m}{c} + \log(2L)\}$*, the voting algorithm correctly outputs the implanted bicluster with probability at least* $1 - 9m^{-2}$.

Proof. It follows from Lemma 3 that, when $k \geq \frac{2 \log m}{c}$, we can find a row with at least $(1-\delta)(1-\theta)k$ unchanged elements in $A'(i, J_B)$ with probability $1 - m^{-2}$. Suppose that such a row is selected as the reference row. Lemma 4 shows that, when $n < m^3$, $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ and $k \geq \frac{8}{\alpha}\sqrt{m \log m}$, the row set I_C will be correctly found in the first phase voting with probability $1 - 2m^{-2}$. If the row set I_C is found, Lemma 5 shows that, when $k \geq \frac{3 \log m}{c} + \log L$, the implanted column set will be correctly found in the second phase voting with probability $1 - 2m^{-2}$. Similarly, if the implanted column set is found, in the third phase voting, when $k \geq \frac{5 \log m}{c} + \log(2L)$, the implanted row set will be found with probability $1 - 2m^{-2}$. Therefore, when all the required conditions hold, all the rows and columns in the implanted bicluster will be found by our algorithm with probability $1 - 7m^{-2}$.

It also follows from Lemma 7 that, when $n < m^3$ and $k \geq \frac{8 \log m}{c} + \log(2L)$, with probability $1 - 2m^{-2}$, no other row or column will be output by our algorithm. Therefore, our algorithm will correctly output the implanted bicluster with probability $1 - 9m^{-2}$. ■

If we replace m by n in the above analysis, the same proof shows that

Corollary 9. *When* $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ *and* $k \geq \max\{\frac{8}{\alpha}\sqrt{n \log n}, \frac{8 \log n}{c} + \log(2L)\}$*, the voting algorithm correctly outputs the implanted bicluster with probability at least* $1 - 9n^{-2}$.

In the practice of microarray data analysis, the number of conditions m is much smaller than the number of genes n. Thus, Theorem 8 allows the parameter k to be smaller (i.e., it works for smaller implanted biclusters) than Corollary 9, although it assumes an upper bound on n ($n < m^3$) and has a slightly worse success probability.

4. THE IMPLEMENTATION OF THE VOTING ALGORITHM

The voting algorithm described in Section 2 is originally based on the probabilistic model for generating the implanted additive bicluster. Many assumptions have been used to prove its correctness. To deal with real data, we have to carefully resolve the following issues.

Estimation of the bicluster size. In the voting algorithm, we assume that the size k of the implanted bicluster is part of the input. However, in practice, the size of the implanted bicluster is unknown. Here we develop a method to estimate the size of the bicluster. We first set k to be a large number such that $k \geq |J_B|$. Let q be the maximum number of rows such that $f(i, J, u) > (m-k)Pr(d_{i,j} = u) + k$ among all $u \in [-L+1, L-1]$. Our key observation here is that if k is greater than $|J_B|$, then q will be smaller than $|I_B|$. If k is smaller than $|J_B|$, then q will be greater than $|I_B|$. Thus, we can gradually decrease the value of k while observing that the value of q increases accordingly. The process stops when $q \geq 2k$.

To set the initial value of k such that $k \geq |J_B|$, we set $k = 3 \cdot \max_u(Pr(d_{i,j} = u)) \cdot m$. This worked very well in our experiments.

Dealing with rectangular biclusters. Many interesting biclusters in the practice of microarray gene expression data are non-square. Without loss of generality, assuming $|I_B| \geq |J_B|$. To obtain rectangular biclusters, we estimate the size of the bicluster as a $k \times k$ square, where $k = |J_B|$ in this case. We then use the first phase voting and the second phase voting normally. The third phase voting may automatically generate a rectangular bicluster by selecting all the rows i such that $f^*(i, J_1) > \frac{|J_1|}{2}$.

Discretization of real data. We need to do discretization before we can apply our algorithm. After we obtain matrix $D(I, J)$, we will transform $D(i, J)$ into a discrete matrix, where each element is an integer in $[0, L-1]$. We will do that row by row. Let min and max be the smallest number and the biggest number in the row, respectively. We divide the range [min, max] into L disjoint ranges of the same size. All the numbers in the i-th range will be mapped to integer $i-1$. Note that, in our probability model, we did not consider small deviations. Based on the discretization method, if small deviations happen in the middle of the L ranges, we can still get the correct discrete value. However, if the small deviations happen at the ends of the L ranges, then they may lead to wrong discrete value. This is a disadvantage of our method. However, the voting algorithm can still find the rows and columns as long as most of the values in the rows and columns are correctly discretized.

Adjusting the threshold used in the first phase voting for a real input matrix. In Step 3 of the first phase voting, we use the threshold $f(i, J, 0) > \frac{m}{L} + 4\sqrt{m \log m}$ to select rows to form I_0. This is based on the assumption that in the random background matrix, $d_{i,j} = 0$ with probability $\frac{1}{L}$. In order for the algorithm to work for any input data, we consider the distribution of numbers in the whole input matrix. We calculate the probability $Pr(d_{i,j} = l)$ for each $l \in [-L+1, L+1]$ in the discrete matrix. Here $Pr(d_{i,j} = l) = \frac{p}{n \times m}$, where p is the number of l's in the input discrete matrix. In Step 3 of the first phase voting, we choose all the rows such that $f(i, J, u) > (m-k)Pr(d_{i,j} = u) + k$. In this way, we were able to make our algorithm to work well for real microarray data where the background did not seem to follow some simple uniform/normal distribution.

When $|I_c|$ is too small for voting. Recall that I_c is the set of the rows identical to the reference row I^* in the implanted bicluster. In other words, the set I_c contains all the rows i with $d_{i,j} = 0$ for $j \in J_B$. The expectation of $|I_c|$ is $\frac{k}{L}$. When k is small and L is large, $|I_c|$ (and thus I_0) could be too small for the voting in the second phase to be effective. To enhance the performance of the algorithm, we consider the set I_B^u for each $u \in [-L+1, L-1]$ as defined in the beginning of Section 2, and approximate it using a set I_0^u in the algorithm just like how we approximated the set $I_C = I_B^0$ by the set I_0 in the first phase voting. Thus, the second phase voting becomes:

The second phase voting:

```
1. for $j = 1$ to $m$ do
2.     compute $f(I_0^u, j, u)$ for each $u \in [-L+1, L-1]$.
3. select columns $j$ such that $\sum_{u=-L+1}^{L-1} f(I_0^u, j, u) > (\sum_{u=-L+1}^{L-1} |I_0^u|)/2$ to form $J_1$.
```

Dealing with multiple and overlapping biclusters. In microarray gene expression analysis, a real input matrix may contain multiple biclusters of interest, some of which could overlap. We could modify the voting algorithm to find multiple implanted biclusters by forcing it to go through all the n rounds (i.e., considering each of the n rows as the reference row) and recording all the biclusters found. If the two biclusters found in two different rounds overlap (in terms of the area) by more than 25% of the area of the smaller biclcuster, then we consider them as the same bicluster and eliminate the smaller one. Eventually, the biclusters found in all n rounds (that were not eliminated) would be output, in the decreasing order of sizes.

5. EXPERIMENTAL RESULTS

We have implemented the above voting algorithm in C++ and produced a software, named VOTE. In this section, we will compare VOTE with some well-known biclustering algorithms in the literature on both simulated and real microarray datasets. The tests were performed on a desktop PC with P4 3.0-G CPU and 512-M memory running Windows operating system.

To evaluate the performance of different methods, we use a measure (called *match score*) similar to the score introduced by Prelić et al. (2006). Let M_1, M_2 be two sets of biclusters. The match score of M_1 with respect to M_2 is given by

$$S(M_1, M_2) = \frac{1}{|M_1|} \sum_{A(I_1,J_1)\in M_1} \max_{A(I_2,J_2)\in M_2} \frac{|I_1 \cap I_2| + |J_1 \cap J_2|}{|I_1 \cup I_2| + |J_1 \cup J_2|}.$$

Let M_{opt} denote the set of implanted biclusters and M the set of the output biclusters of a biclustering algorithm. $S(M_{opt}, M)$ represents how well each of the true biclusters is discovered by a biclustering algorithm.

5.1. Simulated datasets

Following the method used by Prelić et al. (2006) and Liu and Wang (2007), we consider an $n \times m$ background matrix A. Let $L = 30$. We generate the elements in the background matrix A such that the data fits the standard normal distribution with the mean of 0 and the standard deviation of 1. To generate an additive $b \times c$ bicluster, we first randomly generate the expression values in a reference row $(a_1, a_2, \ldots, a_c)$ according to the standard normal distribution. To obtain a row $(a_{i1}, a_{i2}, \ldots, a_{ic})$ in the additive bicluster, we randomly generate a distance d_i (based on the standard normal distribution) and set $a_{i,j} = a_j + d_i$ for $j = 1, 2, \ldots, c$. After we obtain the $b \times c$ additive bicluster, we add some noise by randomly selecting $\theta \cdot b \cdot c$ elements in the bicluster and changing their values to a random number (according to the standard normal distribution). Finally, we insert the obtained bicluster into the background matrix A and shuffle the rows and columns. We compare our program, VOTE, with several well-known programs for biclustering from the literature, including ISA (Ihmels et al., 2004), CC (Cheng and Church, 2000), OPSM (Ben-Dor et al., 2002), and RMSBE (Liu and Wang, 2007). The program OPSM is originally designed for *order preserving* biclusters. (A bicluster is order preserving if its columns can be permuted so that every row is monotonically increasing.) Obviously, an error-free additive bicluster is also an order preserving bicluster. However, when errors are added into a additive bicluster, only part of the bicluster is still order preserving biclusters. Here we also include OPSM in our comparison in various cases though it is not fair to OPSM in some cases. The parameter settings of different methods are listed in Table 1.

Testing the performance on small biclusters. First, we test the ability of finding small implanted additive biclusters. Let $n = m = 100$ and $b = c = 15 \times 15$, and consider implanted biclusters generated with different noise levels θ in the range of $[0, 0.25]$. For each case, we run 100 instances and calculate the average match score. As illustrated in Table 2, the variances of the match scores of the biclusters found by the programs RMSBE and VOTE are very small when the noise level is small, but they increase quickly as the noise gets larger. Figure 2 shows that VOTE and RMSBE perform very well at all noise levels.

Testing the performance on biclusters of different sizes. Since RMSBE has the best performance among the existing programs considered here, we compare VOTE with RMSBE on different bicluster sizes. In this test, the noise level is set as $\theta = 0.15$. The sizes of the implanted (square) biclusters vary from 25×25 to 100×100 and the background matrix is of size 500×500. For each case, we run 100 instances and calculate the average match score. Table 3 shows the variances of the match scores of the

TABLE 1. PARAMETER SETTINGS FOR DIFFERENT BICLUSTERING METHODS

Method	*Type of bicluster*	*Parameter setting*
BiMax (Prelić et al., 2006)	Constant	Minimum number of genes and chips: 4
ISA (Ihmels et al., 2004)	Constant/additive	$t_g = 2.0$, $t_c = 2.0$, $seeds = 500$
CC (Cheng and Church, 2000)	Constant	$\delta = 0.5$, $\alpha = 1.2$
CC (Cheng and Church, 2000)	Additive	$\delta = 0.002$, $\alpha = 1.2$
RMSBE (Liu and Wang, 2007)	Constant/additive	$\alpha = 0.4$, $\beta = 0.5$, $\gamma = \gamma_e = 1.2$
OPSM (Ben-Dor et al., 2002)	Order preserving	$l = 100$
SAMBA (Tanay et al., 2002)	Constant	$D = 40$, $N_1 = 4$, $N_2 = 4$, $k = 20$, $L = 10$

TABLE 2. MATCH SCORE VARIANCES OF RMSBE AND VOTE IN THE TEST OF PERFORMANCE ON SMALL BICLUSTERS AT DIFFERENT NOISE LEVELS

	Noise level					
	0	*0.05*	*0.10*	*0.15*	*0.20*	*0.25*
Variance (RMSBE)	0.0	0.0002	0.0004	0.0005	0.0009	0.0008
Variance (VOTE)	0.0	0.0	0.0001	0.0005	0.008	0.03

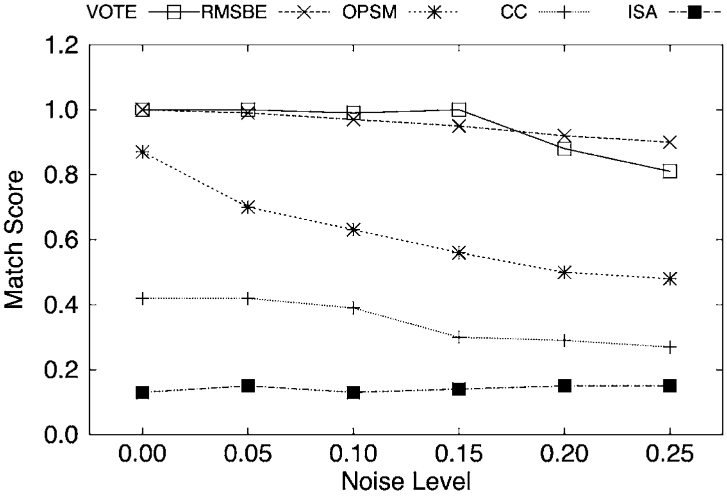

FIG. 2. Performance on small additive biclusters.

TABLE 3. MATCH SCORE VARIANCES OF RMSBE AND VOTE IN THE TEST OF PERFORMANCE ON SMALL BICLUSTERS OF DIFFERENT SIZES

	Size											
	25	*27*	*29*	*30*	*32*	*34*	*36*	*38*	*40*	*50*	*75*	*100*
Variance (RMSBE)	0.012	0.012	0.018	0.021	0.028	0.019	0.006	0.002	0.0005	0.0003	0.00008	0.00006
Variance (VOTE)	0	0.019	0.15	0.13	0.0001	0.0	0.0	0.0	0.0	0.0	0.0	0.0

biclusters found by the two programs, which are small except when the size of the implanted bicluster reaches below 32. As illustrated in Figure 3, VOTE outperforms RMSBE when the size of the implanted bicluster is greater than 32, while RMSBE is better at finding small biclusters.

Testing the performance on matrices of different k/n ratios. In Figure 3, the accuracy of VOTE drops quickly when the ratio k/n decreases below 30/500. In real applications, the ratio may be quite small. To use VOTE in practice, it is important to find the minimum ratio k/n that can guarantee a good performance. Here, we test VOTE and RMSBE on matrices of various sizes to find such a minimum k/n ratio. The noise level is set as $\theta = 0.15$ and the size of the background matrix is in the range $n = m = 100, 200, \ldots, 1000$. For each fixed n and k, we run 50 instances and calculate the average match score. For each fixed n, we try to find the smallest k such that the average match score of the obtained bicluster with respect to the implanted $k \times k$ bicluster is at least 80%. The values of k attained by VOTE and RMSBE for each n with match score $\geq$ 80% are listed in the upper half of Table 4. As shown in the table, the variances of the match scores of the biclusters found by the two programs are generally pretty high here since k is at its minimum value. Note that the match score of RMSBE depends on the quality of the reference row and column. When the reference row and column contain

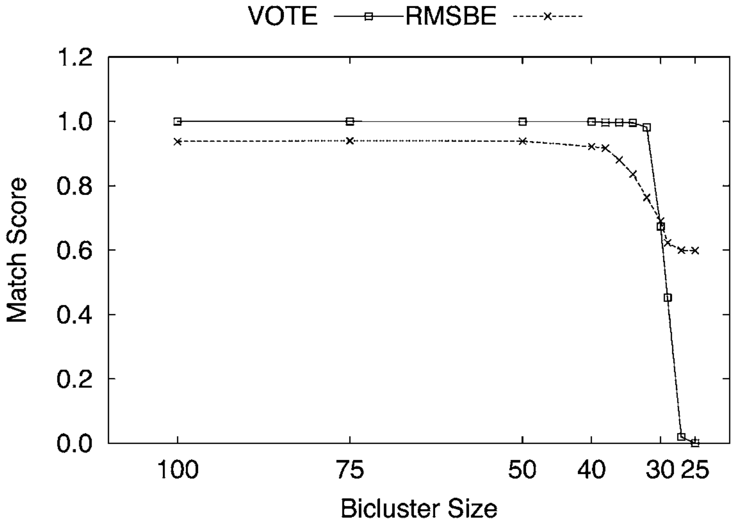

FIG. 3. Performance on biclusters of different sizes.

noise, RMSBE may miss some rows and columns in the resulting bicluster. However, VOTE is able to correct some noise in the reference row (Step 6 in Figure 1). Moreover, since VOTE does not require a reference column, it does not suffer from the impact of a noisy reference column. In general, VOTE outperforms RMSBE when n is large (i.e., it allows for a smaller k value). It should be pointed out that when the implanted bicluster size is smaller than the k value listed in Table 4, the match score of VOTE drops more quickly than that of RMSBE as illustrated in Figure 3. We also observe that (the results are not shown in this article), when the average match score is bigger than 80%, say, for example, 99%, VOTE is much better than RMSBE, but when the average match score is smaller than 69%, RMSBE performs better. To illustrate the latter point, we compare the two programs for match score 60%. The values of k attained by VOTE and RMSBE for each n with match score $\leq$ 60% are listed in the lower half of Table 4.

We observe that the above k values attained by VOTE are much smaller than the theoretical bound given in Corollary 9. For example, using $\alpha = 0.8$, $\theta = 0.09$ and $L = 30$, when $n = 1000$, k would have to be at least 831 in order for the success probability of VOTE to be close to 1. This shows that the theoretical bound could be very conservative in practice.

Identification of overlapping biclusters. To test the ability of finding overlapping biclusters, we first generate two $b \times b$ additive biclusters with o overlapped rows and columns. The parameter o is called

TABLE 4. MINIMUM k OF DIFFERENT MATRIX SIZES

n	*100*	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1000*
k (VOTE)	13	19	24	27	31	34	37	40	43	45
Variance (VOTE)	0.048	0.056	0.030	0.028	0.070	0.083	0.069	0.031	0.047	0.098
k (RMSBE)	13	19	24	29	35	37	41	46	49	55
Variance (RMSBE)	0.016	0.023	0.026	0.022	0.010	0.025	0.024	0.016	0.021	0.013
n	*100*	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1000*
k (VOTE)	12	19	23	27	30	34	37	39	42	44
Variance (VOTE)	0.064	0.070	0.096	0.099	0.12	0.080	0.070	0.15	0.10	0.15
k (RMSBE)	10	15	19	25	27	31	34	38	42	48
Variance (RMSBE)	0.029	0.017	0.013	0.016	0.011	0.014	0.010	0.013	0.016	0.015

The upper half of the table corresponds to the case match score $\geq$80%, and the lower half corresponds to match score $\leq$60%.

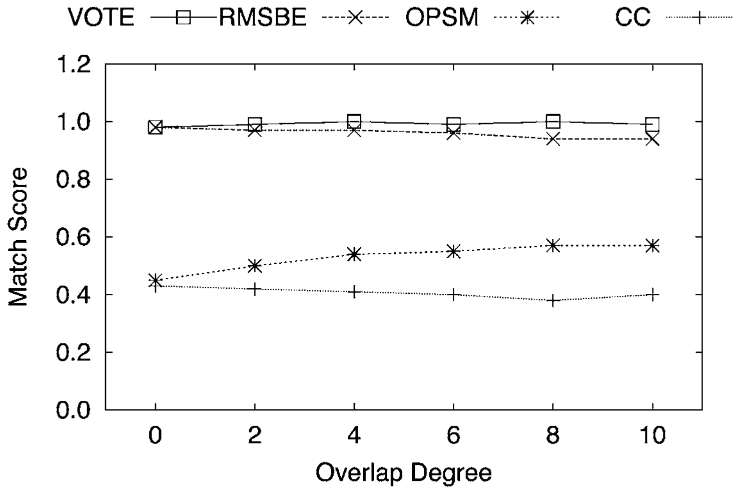

FIG. 4. Performance on overlapping biclusters.

the *overlap degree*. The background matrix size is fixed as 100 × 100. Both the background matrix and the biclusters are generated as before. To find multiple biclusters in a given matrix, some methods, for example, CC, need to mask the previously discovered biclusters with random values. One of the advantages of the approaches based on a reference row, for example, VOTE and RMSBE, is that it is unnecessary to mask previously discovered biclusters. We test the performance of VOTE, RMSBE, CC and OPSM on overlapping biclusters by using 20 × 20 additive biclusters with noise level $\theta = 0.1$ and overlap degree o ranging from 0 to 10. The results are shown in Figure 4. We can see that both VOTE and RMSBE are only marginally affected by the overlap degree of the implanted biclusters. VOTE is slightly better than RMSBE, especially when o increases.

Finding rectangular biclusters. We generate rectangular additive biclusters with different sizes and noise levels. The row and column sizes of the implanted biclusters range from 20 to 50. The noise level θ is from the range $[0, 0.25]$. The background matrix is of size 100 × 100. The results are shown in Figure 5. We can see that the performance of VOTE is not affected by the shapes of the rectangular biclusters. On the other hand, RMSBE can only find near square biclusters (Liu and Wang, 2007), and it has to

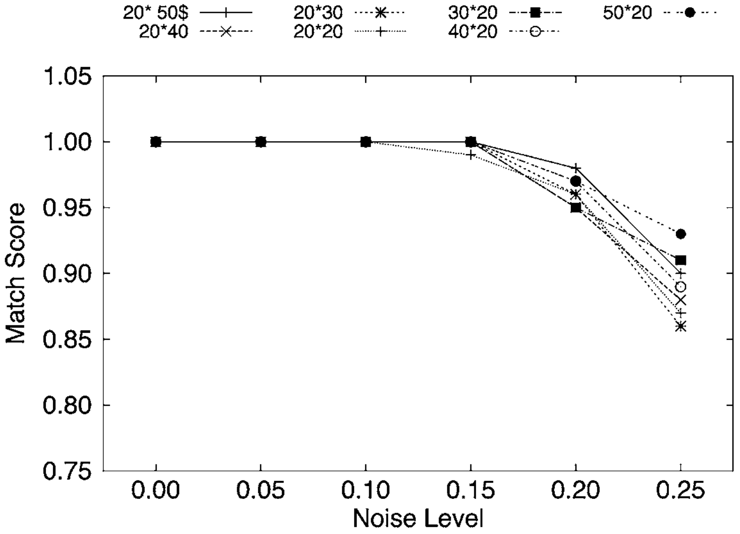

FIG. 5. Performance on rectangular biclusters.

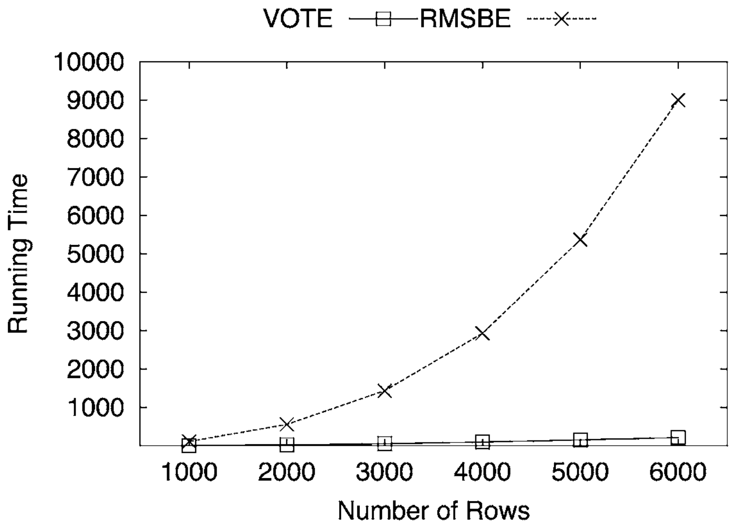

FIG. 6. Speeds of the programs.

be extended to work for general rectangular biclusters. By comparing Figure 5 with the test results in Liu and Wang (2007), we can see that VOTE is better in finding rectangular biclusters than the extended RMSBE.

Running time. To compare the speeds of VOTE and RMSBE, we consider background matrices of 200 columns. The number of rows ranges from 1000 to 6000. The size of the implanted bicluster is 50 × 50. The running time of VOTE and RMSBE is shown in Figure 6. In the test, we let RMSBE randomly select 10% rows as the reference row and 50 columns as the reference column. We can see that VOTE is much faster than RMSBE. Moreover, for the real gene expression data of *S. cerevisiae* provided by Gasch et al. (2000), our algorithm runs in 66 seconds and RMSBE (randomly selecting 300 genes as the reference row and 40 conditions as the reference column) runs in 1230 seconds.

5.2. Real dataset

Gene ontology. Similar to the method used by Tanay et al. (2002) and Prelić et al. (2006), we investigate whether the set of genes discovered by a biclustering method shows significant enrichment with respect to a specific GO annotation provided by the Gene Ontology Consortium (Gasch et al., 2000). We use the web tool funcAssociate of Berriz et al. (2003) to evaluate the discovered biclusters. FuncAssociate first uses Fisher's exact test to compute the hypergeometric functional score of a gene set. Then, it uses the Westfall and Young (1993) procedure to compute the adjusted significance score of the gene set. The analysis is performed on the gene expression data of *S. cerevisiae* provided by Gasch et al. (2000). The dataset contains 2993 genes and 173 conditions. We set $L = 30$ to discretize the data for VOTE. (Here, the value of L is chosen empirically.) For all the programs, we output the best 100 biclusters according to their own criteria. For VOTE, we output the largest 100 biclusters since our algorithm is based on counting. The running time of VOTE on this dataset is 66 seconds. The adjusted significance scores (adjusted p-values) of the 100 best biclusters are computed by using FuncAssociate. Here, we compare the significance scores for RMSBE (Liu and Wang, 2007), OPSM (Ben-Dor et al., 2002), BiMax (Prelić et al., 2006), ISA (Ihmels et al., 2004), Samba (Tanay et al., 2002), and CC (Cheng and Church, 2000). The result is summarized in Figure 7. We can see that 92% of discovered biclusters by VOTE are statistically significant, i.e., with $\alpha \leq 5\%$. Moreover, the performance of VOTE in this regard is comparable to (although slightly worse than) that of RMSBE and is better than those of the other programs compared by Liu and Wang (2007). However, VOTE is much faster than RMSBE since VOTE runs in $O(n^2m)$ time, while RMSBE runs in $O(nm(n+m)^2)$ time in the worse case.

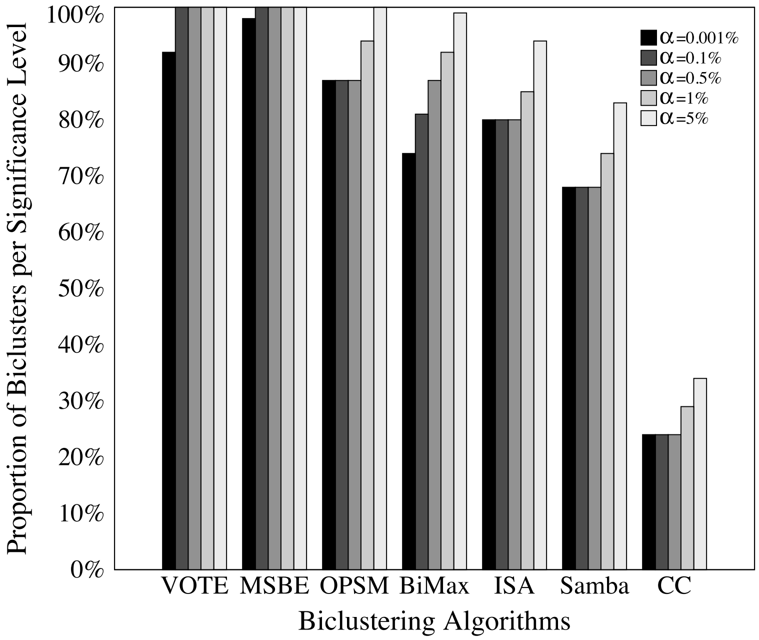

FIG. 7. Proportion of biclusters significantly enriched in a GO category. Here, α is the adjusted significance score of a bicluster.

Colon cancer dataset. Murali and Kasif (2003) used a colon cancer dataset introduced by Alon et al. (1999) to test their biclustering algorithm XMOTIF. The matrix contains 40 colon tumor samples and 22 normal colon samples over about 6500 genes. The dataset is available at *www.weizmann.ac.il/physics/complex/compphys* (Getz et al., 2000). The two best biclusters found by Murali and Kasif (2003) using XMOTIF are B1 and B2 as shown in Table 5. B1 contains 11 genes and 15 samples. Among the 15 samples, 14 of them are tumor samples and 1 is a normal sample. B2 contains 13 genes and 18 samples. Among the 18 samples, 16 of them are normal and 2 are tumor. We use $L = 472$ to run our program VOTE. The two best biclusters that we find are B3 and B4 in Table 2. B3 contains 35 genes and 27 samples, where all of the 27 samples are tumor samples. B4 contains 91 genes and 11 samples. Among the 11 samples, 10 of them are normal and 1 is tumor. Clearly, the bicluster B3 characterizes tumor samples and B4 normal samples. To evaluate the chance of observing such phenotype (tumor or normal) enrichments at random, we compare the output sample subsets to those obtained by randomized selection. Since the number of phenotypes in a random sample subset fits the hypergeometric distribution, the p value can be computed based on the hypergeometric distribution., All the four biclusters found by both XMOTIF and VOTE are statistically significant as shown by their hypergeometric p-values in Table 5. This result shows that VOTE is able to find high quality biclusters on the dataset.

Carcinoma dataset. The dataset is available at *http://microarray.princeton.edu/oncology/*. The matrix contains 18 tumor samples and 18 normal samples over about 7464 genes. We use $L = 400$ to run VOTE. The most tumor-related bicluster contains 152 genes and 14 samples, where all of the 14 samples are tumor samples. The largest normal bicluster contains 1266 genes and 12 samples, where all of the 12 samples are normal samples. This result shows that VOTE can classify the tumor and normal samples well.

TABLE 5. BICLUSTERS FOUND IN THE COLON CANCER DATASET

Bi-cluster	*Method*	*No. of. genes*	*No. of samples*	*No. of tumors*	*No. of normal*	*p-value*
B1	XMOTIF	11	15	14	1	3.6e−2
B2	XMOTIF	13	18	2	16	1.0e−5
B3	VOTE	35	27	27	0	7.3e−6
B4	VOTE	91	11	1	10	3.7e−4

6. CONCLUSION

Based on a simple probabilistic model, we have designed a three-phase voting algorithm to find implanted additive biclusters. We proved that when the size of the implanted bicluster is $\Omega(\sqrt{m}\log m)$, the voting algorithm can correctly find the implanted bicluster with high probability. We have also implemented the voting algorithm as a software tool, VOTE, for finding novel biclsuters in real microarray gene expression data. Our extensive experiments on simulated datasets demonstrate that VOTE performs very well in terms of both accuracy and speed. Future work includes testing VOTE on more real datasets, which could be a bit challenging since true biclusters for most gene expression datasets are unknown. Another direction is to extend the probability model to include small deviations of the values in the input matrix. Perhaps, this will lead to algorithms that work much better in practice. Finally, we note that the automatic selection of parameters in our algorithm is a nontrivial issue and requires further research.

7. APPENDIX: THE MISSING PROOFS

Proof of Lemma 5. Let $x_{i,j}$ be a 0/1 random variable, where $x_{i,j} = 1$ if $a_{i,j}$ is unchanged in generating B', and $x_{i,j} = 0$, otherwise. Consider a column $j \in J_B$. Let $|I_0| = l$. The expectation for $\sum_{i \in I_0} x_{i,j}$ is $(1-\theta)l$. By Lemma 2 and the fact that $l \geq \frac{k}{L}$,

$$\begin{aligned} Pr\left(\sum_{i \in I_0} x_{i,j} \leq \frac{l}{2}\right) &= Pr\left(\sum_{i \in I_0} x_{i,j} \leq (1-\theta)l - \left(\frac{1}{2}-\theta\right)l\right) \\ &\leq \exp\left(-\frac{1}{2}l\left(\frac{1}{2}-\theta\right)^2\right) \\ &\leq e^{-\frac{(1-2\theta)^2}{8L}k}. \end{aligned} \tag{4}$$

Note that for all $i \in I_0$, $c_{i,i^*} = 0$. Therefore, the probability that $f^*(I_0, j) \leq \frac{l}{2} = \frac{|I_0|}{2}$ is also at most $e^{-\frac{(1-2\theta)^2}{8L}k}$. The probability that column $j \in J_B$ is added into J_1 is at least $1 - e^{-\frac{(1-2\theta)^2}{8L}k}$. The probability that all columns in J_B are added into J_1 is at least $1 - ke^{-\frac{(1-2\theta)^2}{8L}k}$.

For a column $j \in J - J_B$ and an integer $u \in [a_{i^*,j} - L + 1, a_{i^*,j}]$, the expectation for $f(I_0, j, u)$ is $\frac{l}{L}$. By Lemma 2 and $l \geq \frac{k}{L}$,

$$\begin{aligned} Pr\left(\sum_{i \in I_0} x_{i,j} \geq \frac{l}{2}\right) &= Pr\left(\sum_{i \in I_0} x_{i,j} \geq \frac{l}{L} + \left(\frac{1}{2}-\frac{1}{L}\right)l\right) \\ &\leq \exp\left(-\frac{1}{3}l\left(\frac{1}{2}-\frac{1}{L}\right)^2\right) \\ &\leq e^{-\frac{(L-2)^2}{12L^3}k}. \end{aligned} \tag{5}$$

For $j \in J - J_B$, the probability that there exists an integer $u \in [a_{i^*,j} - L + 1, a_{i^*,j}]$ such that $f(I_0, j, u) \geq \frac{l}{2} = \frac{|I_0|}{2}$ is at most $Le^{-\frac{(L-2)^2}{12L^3}k}$. The probability that column j is not added into J_1 is at least $1 - Le^{-\frac{(L-2)^2}{12L^3}k}$. The probability that no column in $J - J_B$ is added into J_1 is at least $1 - L(m-k)e^{-\frac{(L-2)^2}{12L^3}k}$. Therefore, the probability that $J_1 = J_B$ is at least $1 - ke^{-\frac{(1-2\theta)^2}{8L}k} - L(m-k)e^{-\frac{(L-2)^2}{12L^3}k}$. ■

Proof of Lemma 6. Let $x_{i,j}$ be a 0/1 random variable, where $x_{i,j} = 1$ if $a_{i,j}$ is unchanged in generating B', and $x_{i,j} = 0$ otherwise. Consider a row $i \in I_B$. The expectation for $\sum_{j \in J_1} x_{i,j}$ is $(1-\theta)k$.

By Lemma 2,

$$\begin{aligned} Pr\left(\sum_{j\in J_1} x_{i,j} \le \frac{k}{2}\right) &= Pr\left(\sum_{j\in J_1} x_{i,j} \le (1-\theta)k - \left(\frac{1}{2}-\theta\right)k\right) \\ &\le \exp\left(-\frac{1}{2}k\left(\frac{1}{2}-\theta\right)^2\right) \\ &= e^{-\frac{(1-2\theta)^2}{8}k}. \end{aligned} \tag{6}$$

Note that, the distance between row i and i^* in $A(I_B, J_B)$ is a constant c_{i,i^*}. Since the submatrix $D(I, J_1)$ has been corrected, $f^*(i, J_1) \ge \sum_{j\in J_1} x_{i,j}$. Thus, the probability that $f^*(i, J_1) \le \frac{k}{2} = \frac{|J_1|}{2}$ is also at most $e^{-\frac{(1-2\theta)^2}{8}k}$. That is, the probability that row i is added into I_1 is at least $1 - e^{-\frac{(1-2\theta)^2}{8}k}$. By considering all the k rows in I_B, the probability that all the rows in I_B are added into I_1 is at least $1 - ke^{-\frac{(1-2\theta)^2}{8}k}$.

For a row $i \in I - I_B$ and an integer $u \in [-L+1, L-1]$, the expectation for $f(i, J_1, u)$ is no more than $\frac{k}{L}$.

By Lemma 2,

$$\begin{aligned} Pr\left(f(i, J_1, u) \ge \frac{k}{2}\right) &= Pr\left(f(i, J_1, u) \ge \frac{k}{L} + \left(\frac{1}{2}-\frac{1}{L}\right)k\right) \\ &\le \exp\left(-\frac{1}{3}k\left(\frac{1}{2}-\frac{1}{L}\right)^2\right) \\ &= e^{-\frac{(L-2)^2}{12L^2}k}. \end{aligned} \tag{7}$$

In the algorithm, the probability that there exists an integer $u \in [-L+1, L-1]$ such that $f(i, J_B, u) \ge \frac{k}{2} = \frac{|J_1|}{2}$ is at most $2Le^{-\frac{(L-2)^2}{12L^2}k}$.

Therefore, the probability that row $i \in I - I_B$ is not added into I_1 is at least $1 - 2Le^{-\frac{(L-2)^2}{12L^2}k}$. The probability that no row in $I - I_B$ is added into I_1 is at least $1 - 2L(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$. With the above analysis, with probability at least $1 - ke^{-\frac{(1-2\theta)^2}{8}k} - 2L(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$, we have $I_1 = I_B$. ■

Proof of Lemma 7. For any column $j \in J - J_B$, similar to Lemma 5, we can prove that with probability $1 - Lne^{-\frac{(L-2)^2}{12L^3}k}$, the column j is not added into J_1 in any of the n rounds of Algorithm 1. Since $|J - J_B| = m - k$, with probability at least $1 - Ln(m-k)e^{-\frac{(L-2)^2}{12L^3}k}$, no column in $J - J_B$ are added into J_1. In other words, no column other than those of J_B are output by the algorithm.

For any row $i \in I - I_B$, similar to Lemma 6, we can prove that with probability $1 - 2Lne^{-\frac{(L-2)^2}{12L^2}k}$, the row i is not added into I_1 in any of the n rounds of Algorithm 1. Since $|I - I_B| = n - k$, with probability at least $1 - 2Ln(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$, no row in $I - I_B$ are added into I_1. In other words, no row other than those of I_B are output by the algorithm.

The above analysis shows that, with probability at least $1 - Ln(m-k)e^{-\frac{(L-2)^2}{12L^3}k} - 2Ln(n-k)e^{-\frac{(L-2)^2}{12L^2}k}$, no row or column other than those in $A'(I_B, J_B)$ will be output by Algorithm 1. ■

ACKNOWLEDGMENTS

We wish to thank the anonymous referees for their many helpful suggestions and constructive comments. The research of JX was supported in part by the National Natural Science Foundation of China (grant 60553001) and the National Basic Research Program of China (grant 2007CB807900). LW was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project no. CityU 121207). The research of TJ was supported by the NSF (grant IIS-0711129), the NIH (grant LM008991), the National Natural Science Foundation of China (grant 60528001), and a Changjiang Visiting Professorship at Tsinghua University.

DISCLOSURE STATEMENT

No conflicting financial interests exist.

REFERENCES

Alon, N., Krivelevich, M., and Sudakov, B. 1998. Finding a large hidden clique in a random graph. *Random Struct. Algorithms* 13, 457–466.

Alon, U., Barkai, N., Notterman, D.A., et al. 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* 96, 6745–6750.

Barkow, S., Bleuler, S., Prelić, A., et al. 2006. BicAT: a biclustering analysis toolbox. *Bioinformatics* 22, 1282–1283.

Ben-Dor, A., Chor, B., Karp, R., et al. 2002. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Proc. RECOMB* 45–55.

Ben-Dor, A., Shamir, R., and Yakhini, Z. 1999. Clustering gene expression patterns. *J. Comput. Biol.* 6, 281–297.

Berriz, G.F., King, O.D., Bryant, B., et al. 2003. Charactering gene sets with FuncAssociate. *Bioinformatics* 19, 2502–2504.

Cheng, Y., and Church, G.M. 2000. Biclustering of expression data. *Proc. ISMB-00* 93–103.

Feige, U., and Krauthgamer, R. 2000. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms* 16, 195–208.

Gasch, A.P., Spellman, P.T., Kao, C.M., et al. 2000. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell* 11, 4241–4257.

Getz, G., Levine, E., Domany, E., et al. 2000. Super-paramagnetic clustering of yeast gene expression profiles. *Physica A* 279, 457–464.

Hartigan, J.A. 1972. Direct clustering of a data matrix. *J. Am. Statist. Assoc.* 67, 123–129.

Ihmels, J., Bergmann, S., and Barkai, N. 2004. Defining transcription modules using large-scale gene expression data, *Bioinformatics* 20, 1993–2003.

Kluger, Y., Basri, R., Chang, J., et al. 2003. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.* 13, 703–716.

Kucera, L. 1995. Expected complexity of graph partitioning problems. *Discrete Appl. Math.* 57, 193–212.

Li, H., Chen, X., Zhang, K., et al. 2006. A general framework for biclustering gene expression data. *J. Bioinform. Comput. Biol.* 4, 911–933.

Li, M., Ma, B., and Wang, L. 2002. On the closest string and substring problems. *J. ACM* 49, 157–171.

Liu, X., and Wang, L. 2007. Computing the maximum similarity biclusters of gene expression data. *Bioinformatics* 23, 50–56.

Lonardi, S., Szpankowski, W., and Yang, Q. 2004. Finding biclusters by random projections. *Proc. 15th Annu. Symp. Combin. Pattern Matching* 102–116.

Madeira, S.C., and Oliveira, A.L. 2004. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 1, 24–45.

Motwani, R., and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK.

Murali, T.M., and Kasif, S. 2003. Extracting conserved gene expression motifs from gene expression data. *Pac. Symp. Biocomput.* 8.

Peeters, R. 2003. The maximum edge biclique problem is NP-complete. *Discrete Appl. Math.* 131, 651–654.

Prelić, A., Bleuler, S., Zimmermann, P., et al. 2006. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics* 22, 1122–1129.

Tanay, A., Sharan, R., and Shamir, R. 2002. Discovering statistically significant biclusters in gene expression data. *Bioinformatics* 18, Suppl 1, 136–144.

Westfall, P.H., and Young, S.S. 1993. *Resampling-Based Multiple Testing*. Wiley, New York.

Yang, J., Wang, W., Wang, H., et al. 2002. δ-clusters: capturing subspace correlation in a large data set. *Proc. 18th Int. Conf. Data Eng.* 517–528.

Address reprint requests to:
Dr. Tao Jiang
Department of Computer Science and Engineering
University of California
Riverside, CA 92521

E-mail: jiang@cs.ucr.edu