Finding Additive Biclusters with Random Background (Extended Abstract)

Jing Xiao[1], Lusheng Wang[2], Xiaowen Liu[3], and Tao Jiang[4]

[1] Department of Computer Science and Technology, Tsinghua University
xiaojing00@mails.tsinghua.edu.cn
[2] Department of Computer Science, City University of Hong Kong, Hong Kong
lwang@cs.cityu.edu.hk
[3] Department of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7
liuxiaowencs@gmail.com
[4] Department of Computer Science and Engineering, University of California, Riverside
jiang@cs.ucr.edu

Abstract. The biclustering problem has been extensively studied in many areas including e-commerce, data mining, machine learning, pattern recognition, statistics, and more recently in 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 under various mathematical models. In this paper, we study a probabilistic model of the implanted additive bicluster problem, where each element in the $n \times m$ background matrix is a random number from $[0, L-1]$, 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 $O(n^2 m)$ time voting algorithm to solve the problem. We show that for any constant δ such that $(1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$, when $k \geq \max\left\{\frac{8}{\alpha}\sqrt{n \log n},\ \frac{8 \log n}{c} + \log(2L)\right\}$, where c is a constant number, 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 software tool for finding novel biclusters in microarray gene expression data, called VOTE. The implementation incorporates several nontrivial ideas for estimating the size of an implanted bicluster, adjusting the threshold in voting, dealing with small biclusters, and dealing with multiple (and 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: bicluster,Chernoff bound, polynomial-time algorithm, probability model, computational biology, gene expression data analysis

1 Introduction

Biclustering has proved extremely useful for exploratory data analysis. It has important applications in many fields, *e.g.*, e-commerce, data mining, machine learning, pattern

recognition, statistics, and computational biology [24]. Data arising from text analysis, market-basket data analysis, web logs, microarray experiments *etc.* are usually arranged in a co-occurrence table or a matrix, such as 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 becomes an important approach to microarray gene expression data analysis [5]. The underlying bases for using biclustering in the analysis of gene expression data are (i) similar genes may exhibit similar behaviors only under a subset of conditions, not 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, *e.g.*, "co-clustering", "two-way clustering", and "direct clustering". The problem was first introduced by Hartigan in the 70's [8]. Since then, it has been extensively studied in many areas. 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 [16, 19].

Let $A(I, J)$ be an $n \times m (n \geq m)$ matrix, where $I = \{1, 2, \ldots, n\}$ is the set of rows and $J = \{1, 2, \ldots, m\}$ is the set of columns. 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. For 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 with a row set I' and a single column j. There are several ways to model the relationship between objectives (or genes) [24].

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

The additive model is a general model of biclusters that covers several other popular models as its special cases. See [17] for details. This model has many applications and has been extensively studied [2, 11, 13, 15–17, 19–21, 24]. In this paper, we will focus on the additive model. In particular, we study a probabilistic model of implanted additive biclusters that has recently been used in the literature for evaluating biclustering algorithms [15, 20].

The probabilistic additive model: 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 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 implanced 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 implanted additive bicluster as described above, find the implanted additive bicluster B'.

Based on the above probabilistic model, we propose an $O(n^2 m)$ time voting algorithm for finding the implanted bicluster. We show that for any constant δ such that $(1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$, when $n < m^3$ and $k \geq \max\left\{\frac{8}{\alpha}\sqrt{m \log m},\ \frac{8 \log m}{c} + \log(2L)\right\}$, where $c = \min\{\frac{(1-\theta)\delta^2 k}{2L}, \frac{(1-2\theta)^2}{8L}, \frac{(L-2)^2}{12L^3}\}$, the voting algorithm can correctly find the implanted bicluster with probability at least $1 - 9m^{-2}$. We also implement our algorithm into a software tool, called VOTE. In order to make tool applicable in a real setting, the implementation has to incorporate several nontrivial ideas for estimating the size of an implanted bicluster, adjusting the threshold in voting, dealing with small biclusters, and dealing with multiple and overlapping biclusters. Our extensive 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 comparable performance/accuracy as the best programs compared in [20, 15], but much faster speed.

We note in passing that a closely related problem of finding an implanted clique/ distribution in a random graph has been studied in the graph theory community [1, 6, 12]. In [12], Kucera claimed that when the size of the implanted clique is at least $\Omega(\sqrt{m \log m})$, where m is the number of vertices in the input random graph, a simple approach based counting the degrees of vertices can find the clique with a high probability. Alon *et al.* gave an improved algorithm that can find implanted cliques of sizes at least $\Omega(\sqrt{m})$ with a high probability [1]. Feige and Krauthgamer gave an algorithm that can find implanted cliques of similar sizes in semi-random graphs [6]. 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.

In the rest of the paper, we first present the voting algorithm and analyze its theoretical performance on the above probabilistic model. We then describe the

implementation of VOTE, and the experimental results. Due to the page limit, the proofs will be omitted in this extended abstract but will be provided in the full paper.

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 identifical. That is, 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 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_1 in I_C, the expectation of the number of* 0*'s in row i_1 of $D(I, J)$ is at least $\frac{m-k}{L} + (1-\theta)^2 k$. For each row i_2 in $I_B - I_C$, the expectation of the number of* 0*'s in row i_2 of $D(I, J)$ is at most $\frac{m-k}{L} + \frac{2\theta k}{L}$. For each row i_3 in $I - I_B$, the expectation of the number of* 0*'s in row i_3 of $D(I, J)$ is at most $\frac{m-k}{L} + \frac{k}{L}$.*

Based on the observation, if the reference row i^* is in I_C, we can find the rows with the most 0's in the distance matrix to obtain a row set I_0 by using 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 a high probability, the row set I_0 is equal to I_C. The proof will be given in the next section.

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

Observation 3 *For a column j_1 in J_B, the expectation of the number of occurrences of the element that appears the most in $D(I_C, j_1)$ is $(1 - \theta)|I_C|$. For a column j_2 in $J - J_B$, the expectation of the number of occurrences of an element u in $D(I_C, j_1)$ is $\frac{1}{L}|I_C|$.*

With a 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 a 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 a 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 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.

Observation 4 *For a row i_1 in I_B, the expectation of the number of occurrences of the element that appears the most in $D(i_1, J_B)$ is at least $(1 - \theta)k$. For a row i_2 in $I - I_B$, the expectation of the number of occurrences of the element that appears the most in $D(I_2, j_B)$ is $\frac{k}{L}$.*

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

The Three Phase Voting Algorithm

Input: An $n \times m$ matrix $A'(I, J)$, an integer k, noise level θ, 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.

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 a high probability, I_1 is equal to the implanted column 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 a 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 $O(n^2m)$.

3 Analysis of the Algorithm

In this section, we will prove that, with a 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 a 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 [18, 14].

Lemma 1. [18] *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) $\mathbf{Pr}(X > (1+\delta)\mu) < \left[\frac{\mathbf{e}^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$,
(2) $\mathbf{Pr}(X < (1-\delta)\mu) \leq e^{-\frac{1}{2}\mu\delta^2}$.

Lemma 2. [14] *Let $X_i, 1 \leq i \leq n$, X and μ be defined as in Lemma 1. Then for any $0 < \epsilon < 1$,*

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

These two lemmas will be used to establish the next lemma.

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* $A'(i, J_B)$.

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 a 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 a 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 above. When* $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ *and* $k \geq \frac{8}{\alpha}\sqrt{m \log m}$*, 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* $i \in I - I_C$.

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 a high probability. Next, we prove that, with a 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.

Similarly, we can prove that, with a 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.

Finally, we can prove that, with a high probability, no columns or rows 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 columns or rows of* $A'(I, J)$ *other than those in* $A'(I_B, J_B)$ *will be output by the Algorithm 1.*

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

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

Corollary 1. *When* $\alpha = (1-\delta)(1-\theta)^2 - \frac{1}{L} > 0$ *and* $k \geq \max\left\{\frac{8}{\alpha}\sqrt{n\log n}, \frac{8\log n}{c} + \log(2L)\right\}$, *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 1 allows the parameter k to be smaller (*i.e.* it works for smaller implanted biclusters) than Corollary 1, although it assumes a slightly more complicated condition ($n < m^3$) 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 retangular biclusters. Many interesting biclusters in the practice of microarray gene expression data are non-square. To deal with such rectangular biclusters, where $|I_B| \neq |J_B|$, we first try to obtain a square bicluster in the first phase voting (assuming $|I_B| \geq |J_B|$) and then use the k rows in I_0 for the second phase voting. The third phase voting may in fact generate a rectangular bicluster with unequal numbers of rows and columns.

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 $Pr\left(d_{i,j} = l\right)$ for each $l \in [-L+1, L+1]$ in the input 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 easily 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.

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 3.0G CPU and 512M memory running Windows operating system.

To evaluate the performance of different methods, we use a measure (called *match score*) similar to the score introduced in Prelić *et al.* [20]. 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 in [15, 20], 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

Table 1. Parameter settings for different biclustering methods.

Method	Type of Bicluster	Parameter Setting
BiMax	Constant	minimum number of genes and chips: 4
ISA	Constant/Additive	$t_g = 2.0, t_c = 2.0, seeds = 500$
CC	Constant	$\delta = 0.5, \alpha = 1.2$
CC	Additive	$\delta = 0.002, \alpha = 1.2$
RMSBE	Constant/Additive	$\alpha = 0.4, \beta = 0.5, \gamma = \gamma_e = 1.2$
OPSM	Additive	$l = 100$

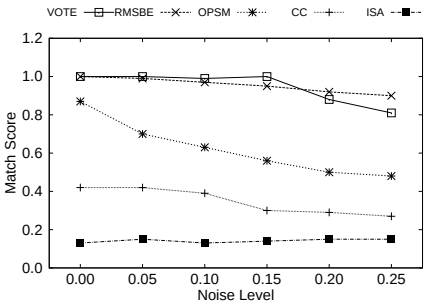

Fig. 2. Performance on small additive biclusters.

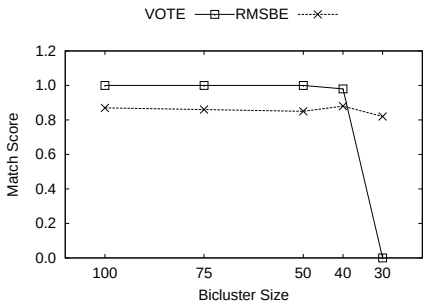

Fig. 3. Performance on biclusters of different sizes.

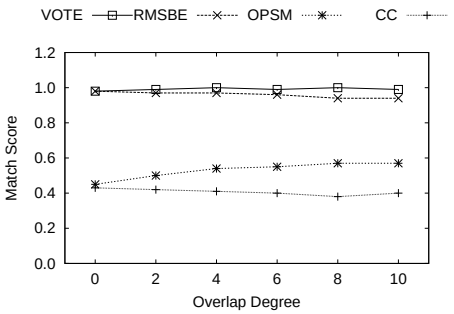

Fig. 4. Performance on overlapping biclusters.

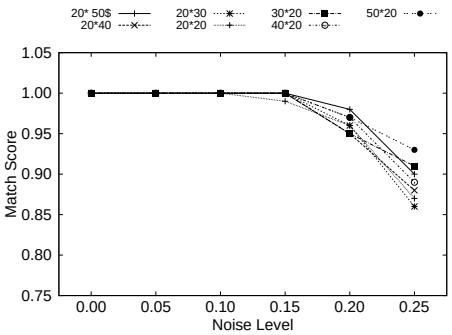

Fig. 5. Performance on rectangular biclusters.

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

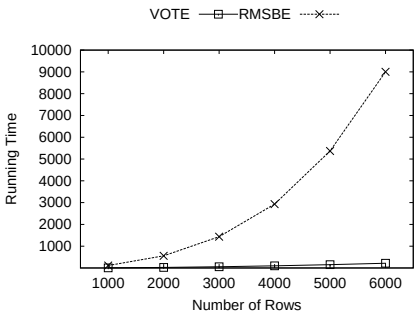

Fig. 6. Speeds of the programs.

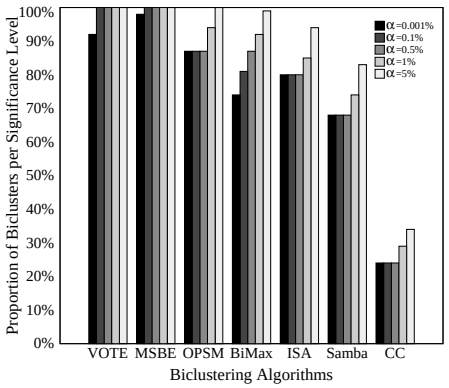

Fig. 7. Proportion of biclusters significantly enriched by a GO category. Here, α is the adjusted significance score of a bicluster.

ISA, CC, OPSM, and RMSBE [3, 5, 9, 10, 15]. The parameter settings of different methods are listed in Table 1.

Testing the performance on small biclusters. First, we test how well the programs are able to find 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 $[0, 0.25]$. Figure 2 shows that VOTE and RMSBE outperform CC, OPSM and ISA with on 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 $\theta = 0.2$. The sizes of the implanted (square) biclusters vary from 30×30 to 100×100 and the background matrix is of size 500×500. As illustrated in Figure 3, VOTE outperforms RMSBE when the size of the square bicluster is greater than 40, while RMSBE is more powerful in finding small biclusters.

Finding multiple biclusters. To test the ability of finding multiple biclusters, we first generate two $b \times b$ additive biclusters with o overlapped rows and columns. The parameter o is called 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, *e.g.*, CC, needs to mask the previously discovered biclusters with random values. One of the advantages of the approaches based on a reference row, *e.g.*, 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. Since RMSBE can only find near square biclusters, we compare the performance of VOTE with that of an extension of RMSBE. Comparing Figure 5 with the test results given in [15], our algorithm is better in finding rectangular biclusters.

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.* [7], 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

Similar to the method used by Tanay et al. [22] and Prelić *et al* [20], 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 [7]. We use the web tool funcAssociate of Berriz *et al.* [4] 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 procedure [23] 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.* [7]. The dataset contains 2993 genes and 173 conditions. We set $L = 30$, filter out the biclusters with over 25% overlapped elements, and output the largest 100 biclusters. The running time of VOTE on this dataset is 66 seconds. The adjusted significance scores (adjusted p-values) of the 100 biclusters are computed by using FuncAssociate. Here, we compare the significance scores for RMSBE, OPSM, BiMax [20], ISA, Samba [22], and CC obtained from Figure 7 in Liu *et al.* [15]. 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 that of RMSBE and is better than those of the other programs compared in [15].

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 a 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.

Acknowledgments

JX's research is supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901, LW's research is supported by a grant from City University of Hong Kong [Project No. 7001996], and TJ's research is supported by NSF grant IIS-0711129, NIH grant LM008991-01, National Natural Science Foundation of China grant 60528001, and a Changjiang Visiting Professorship at Tsinghua University.

References

1. N. Alon, M. Krivelevich, and B. Sudakov, Finding a Large Hidden Clique in a Random Graph, *Random Structures and Algorithms*, **13** (3-4), pp. 457-466, 1998.
2. S. Barkow, S. Bleuler, A. Prelić, P. Zimmermann, and E. Zitzler, BicAT: a biclustering analysis toolbox. *Bioinformatics,* **22**(10), pp. 1282–1283, 2006.
3. A. Ben-Dor, B. Chor, R. Karp and Z. Yakhini, Discovering local structure in gene expression data: the order-preserving submatrix problem, *Proceedings of Sixth International Conference on Computational Molecular Biology (RECOMB)*, ACM Press, pp. 45–55, 2002.
4. G.F. Berriz, O.D. King, B. Bryant, C. Sander and F.P. Roth, Charactering gene sets with FuncAssociate. *Bioinformatics*, **19**, pp. 2502-2504, 2003.
5. Y. Cheng and G.M. Church, Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00),* AAAI Press, Menlo Park, CA, pp. 93–103, 2000.
6. U. Feige and R. Krauthgamer, Finding and certifying a large hidden clique in a semirandom graph, *Random Structures and Algorithms*, **16**(2), pp. 195-208, 2000.
7. A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein and P.O. Brown, Genomic expression programs in the response of yeast cells to enviormental changes. *Molecular Biology of the Cell*, **11**, pp. 4241-4257, 2000.
8. J.A. Hartigan, Direct clustering of a data matrix, *J. of the American Statistical Association* **67**, pp. 123-129, 1972.
9. J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv and N. Barkai, Revealing modular organization in the yeast transcriptional network, *Nature Genetics*, **31**, pp. 370–377, 2002.
10. J. Ihmels, S. Bergmann and N. Barkai, Defining transcription modules using large-scale gene expression data, *Bioinformatics*, **20**(13), pp. 1993–2003, 2004.
11. Y. Kluger, R. Basri, J. Chang, and M. Gerstein, Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research,* **13**, pp. 703-716, 2003.
12. L. Kucera, Expected complexity of graph partitioning problems, *Disc. Appl. Math.* **57**, pp. 193-212, 1995.
13. H. Li, X. Chen, K. Zhang and T. Jiang, A general framework for biclustering gene expression data. *Journal of Bioinformatics and Computational Biology*, **4**(4), pp. 911-933, 2006.
14. M. Li, B. Ma, and L. Wang, On the closest string and substring problems, *J. ACM*, **49**(2), pp. 157-171, 2002.
15. X. Liu and L. Wang, Computing the maximum similarity biclusters of gene expression data. *Bioinformatics*, **23**(1), pp. 50-56, 2007.
16. S. Lonardi, W. Szpankowski and Q. Yang, Finding biclusters by random projections. In *Proceedings of the Fifteenth Annual Symposium on Combinatorial Pattern Matching*, pp. 102-116, 2004.
17. S. C. Madeira, and A.L. Oliveira, Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* **1**(1), pp. 24–45, 2004.
18. R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press. 1995.
19. R. Peeters, The maximum edge biclique problem is NP-complete. *Disc. Appl. Math.* **131**(3), pp. 651 - 654, 2003.
20. A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics,* **22**(9), pp. 1122–1129, 2006.
21. R. Shamir, A. Maron-Katz, A. Tanay, C. Linhart, I. Steinfeld, R. Sharan, Y. Shiloh, and R. Elkon, EXPANDER - an integrative program suite for microarray data analysis. *BMC Bioinformatics,* **6**:232, 2005.
22. A. Tanay, R. Sharan and R. Shamir, Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, **18**, Suppl. 1, pp. 136-144, 2002.
23. P.H. Westfall and S.S. Young, *Resampling-based multiple testing*, Wiley, New York, 1993.
24. J. Yang, W. Wang, H. Wang, and P. Yu, δ-clusters: capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering*, pp. 517-528, 2002.