SIAM J. COMPUT.
Vol. 38, No. 4, pp. 1504–1518

APPROXIMATION ALGORITHMS FOR BICLUSTERING PROBLEMS*

LUSHENG WANG†, YU LIN‡, AND XIAOWEN LIU†

Abstract. One of the main goals in the analysis of microarray data is to identify groups of genes and groups of experimental conditions (including environments, individuals, and tissues) that exhibit similar expression patterns. This is the so-called biclustering problem. In this paper, we consider two variations of the biclustering problem: the *consensus submatrix problem* and the *bottleneck submatrix problem*. The input of the problems contains an $m \times n$ matrix A and integers l and k. The *consensus submatrix problem* is to find an $l \times k$ submatrix with $l < m$ and $k < n$ and a consensus vector such that the sum of distances between the rows in the submatrix and the consensus vector is minimized. The *bottleneck submatrix problem* is to find an $l \times k$ submatrix with $l < m$ and $k < n$, an integer d and a center vector such that the distance between every row in the submatrix and the vector is at most d and d is minimized. We show that both problems are NP-hard and give randomized approximation algorithms for special cases of the two problems. Using standard techniques, we can derandomize the algorithms to get polynomial time approximation schemes for the two problems. To the best of our knowledge, this is the first time that approximation algorithms with guaranteed ratios are presented for microarray data analysis.

Key words. approximation algorithms, computational biology, microarray data analysis, genes

AMS subject classifications. 68W25, 90C05, 90C27, 92B05

DOI. 10.1137/060664112

1. Introduction.

In the past few years, microarray techniques have been widely used in biological research. Microarray techniques have helped to illuminate mechanisms of diseases, identify disease subphenotypes, predict disease progression, assign functions to previously unannotated genes, group genes into functional pathways, and predict activities of new compounds [1]. Microarray data analysis is an important problem in computational biology [2]. For this large-scale data, classifying genes into different groups under certain conditions is a first step to gain more sophisticated knowledge of different biological pathways or functions. Several clustering or classification techniques such as k-means [3, 4], self-organizing maps [5, 6], hierarchical clustering [7, 8, 9], principal component analysis, and singular value decomposition [10, 11, 12] have been extensively applied to identify groups of similarly expressed genes and conditions from gene expression data.

It is known that many activation patterns are common to a group of genes only under specific experimental conditions. We should expect subsets of genes to be coregulated and coexpressed only under certain experimental conditions, but to behave almost independently under other conditions, according to our general understanding of cellular processes [22, 23, 24]. The fact is that we need to discover local patterns in the microarray matrix. The basic model for biclustering is as follows: given an

*Received by the editors July 3, 2006; accepted for publication (in revised form) August 16, 2007; published electronically September 25, 2008. This work was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project CityU 120905).

http://www.siam.org/journals/sicomp/38-4/66411.html

†Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong (lwang@cs.cityu.edu.hk, liuxw@cs.cityu.edu.hk).

‡Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China (linyu@cs.cityu.edu.hk).

$m \times n$ matrix A, where each element $a_{i,j} \in \{0, 1\}$, the problem here is to find an $l \times k$ submatrix with all elements identical to 1 such that $l \times k$ is maximized.

Let $\Sigma = \{\pi_1, \pi_2, \ldots, \pi_{|\Sigma|}\}$ be a fixed size alphabet of symbols. A vector over Σ is a sequence of symbols in Σ. Let A be an $m \times n$ matrix, where each row corresponds to a gene and each column corresponds to a condition. Each element $a_{i,j}$ in A represents the expression level of gene i under condition j. Such a matrix A is defined by its set of m rows, $X = \{x_1, x_2, \ldots, x_m\}$, and its set of n columns, $Y = \{y_1, y_2, \ldots, y_n\}$. Let $P = \{p_1, \ldots, p_l\}$ be a subset of $\{1, 2, \ldots, m\}$ indicating rows in X and $Q = \{q_1, \ldots, q_k\}$ be a subset of $\{1, 2, \ldots, n\}$ indicating columns in Y. The $l \times k$ submatrix $A_{P,Q}$ induced by the pair (P, Q) contains the elements $a_{i,j}$, where $i \in P$ and $j \in Q$. We treat each row in the matrix or submatrix as a vector over Σ. Define $x_i|^Q = a_{i,q_1} a_{i,q_2} \ldots a_{i,q_k}$. Let p and p' be two vectors of the same length over Σ. We use $d(p, p')$ to denote the number of mismatches between the two vectors. Throughout this paper, we will study the following two problems:

1. *The consensus submatrix problem*: Given an $m \times n$ matrix A, and integers l and k, find a subset P of l rows, a subset Q of k columns in matrix A, and a consensus vector z of length k such that the consensus score $\sum_{i=1}^{l} d(x_{p_i}|^Q, z)$ is minimized.
2. *The bottleneck submatrix problem*: Given an $m \times n$ matrix A, and integers l and k, find a subset P of l rows, a subset Q of k, columns in matrix A, a center vector z of length k, and an integer d such that for every $p_i \in P$, $d(x_{p_i}|^Q, z) \leq d$ and the bottleneck score d is minimized.

In some applications, the number of conditions/samples ranges from 50 to 550 and the number of genes is about a few thousand. The interesting submatrices (biclusters) may contain tens of conditions/samples. In this case, we have $k = \Omega(n)$. In practice, there are errors in the microarray data. In [22, 29], the input matrix contains 12.3% missing values (see Data Preparation in [22]). In the $l \times k$ submatrix, if we assume that 12.3% values are missing, then the total consensus score $\sum_{i=1}^{l} d(x_{p_i}|^Q, z)$ is $\Omega(lk)$ and the bottleneck score d is $\Omega(k)$. Moreover, the expression levels of coregulated and coexpressed genes in the submatrices are not 100% numerically identical. Therefore, in some cases we can assume that for the consensus submatrix problem $\sum_{i=1}^{l} d(x_{p_i}|^Q, z) = \Omega(lk)$ and for the bottleneck submatrix problem $d = \Omega(k)$.

Throughout this paper, we assume that for the consensus submatrix problem $\sum_{i=1}^{l} d(x_{p_i}|^Q, z) = \Omega(lk)$ and for the bottleneck submatrix problem $d = \Omega(k)$. Moreover, we further assume that $k = \Omega(n)$.

1.1. Real examples. Here we present a few real examples to show that our assumptions on d and k are reasonable in some real applications.

Example 1. The first paper using the biclustering approach for microarray data analysis is [22]. In this paper, the authors analyze the human data originated from [29]. There are $4,026$ genes and 96 conditions with 12.3% missing values in the input matrix. (For more on microarray data errors, see [13, 14, 15, 16, 17, 18, 19, 20].) For the interesting submatrices, the number of conditions ranges from 13 to 96 (most of them > 20). (See the caption of Figure 5 in [22].) Here the assumption that $k = \Omega(n)$ does hold. Moreover, with 12.3% missing values (thus $\sum_{i=1}^{l} d(x_{p_i}|^Q, z) = \Omega(lk)$ and $d = \Omega(k)$), they can still find some biologically interesting submatrices.

Example 2. Koyutürk, Szpankowski, and Grama study a dataset containing 84 samples for human breast cancer [28]. They obtain a biologically interesting submatrix containing 62 samples and 141 genes. Figure 1 shows the submatrix after binary quantization. From Figure 1, we can see that in the gray (red in the color version)

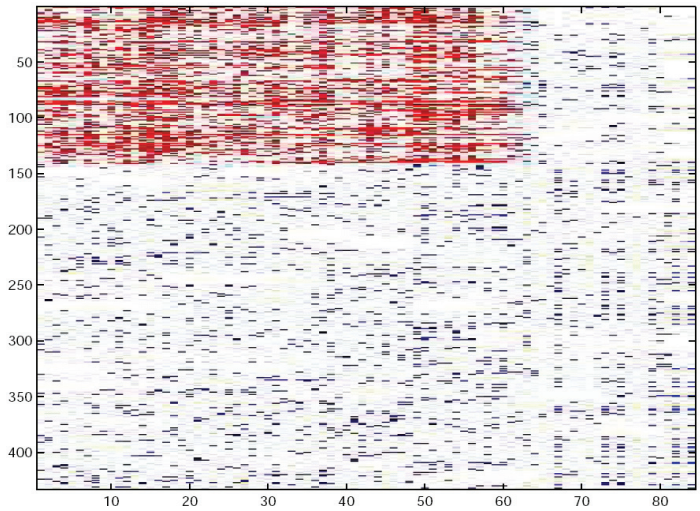

FIG. 1. *The interesting submatrix originally from* [*M. Koyutürk, W. Szpankowski, and A. Grama, Biclustering gene-feature matrices for statistically significant dense patterns, in Proceedings of the* 2004 *Annual IEEE Bioinformatics Conference,* 2004*, pp.* 480–484]. ©2004 *IEEE.*

submatrix, there are still quite a lot of dark and white (blue and white in the color version) dots indicating that the expression levels of genes are not completely identical. (This may be caused by the performance of genes or by data errors.) Based on Figure 1, it is easy to see that there are at least 10% errors. Again, this example shows that our assumptions on d and k are reasonable in some applications.

1.2. Relations to previous work. The basic model for biclustering is to find a submatrix $A_{P,Q}$ with all elements identical to a constant value [23]:

$$a_{i,j} = \mu \quad \text{for all} \quad i \in P, j \in Q.$$

If the submatrix is error free, both the consensus score and the bottleneck score are clearly 0 for the new problems that we propose in the paper.

In practice, it is interesting to find submatrices such that all elements in a row have the same constant value [21, 24]. That is,

$$a_{i,j} = c_i \text{ for } \quad j \in Q.$$

In this case, all the columns in the submatrix are identical. Again, it is clear that both the consensus score and the bottleneck score are 0 if the submatrix is error-free.

For the additive model [22, 23], we are interested in submatrices satisfying

$$a_{i,j} = a_{i',j} + c(i,i') \quad \text{for all} \quad i,i' \in P, \quad j \in Q. \tag{1}$$

That is, for two elements $a_{i,j}$ and $a_{i',j}$ in row i and row i', the difference is a constant $c(i',i)$.

Now we will show that our model can also handle the additive model. Let r be a row in the error-free submatrix. We construct a new matrix A' as follows:

$$a'_{i,j} = a_{i,j} - a_{r,j} \quad \text{for all} \quad i \in X, \quad j \in Y.$$

Then, the error-free submatrix is converted into a new submatrix $A'_{P,Q}$ with the element

$$\begin{aligned} a'_{i,j} &= a_{i,j} - a_{r,j} \\ &= c(i,r) \quad \text{for all} \;\; i \in P, \;\; j \in Q. \end{aligned}$$

That is, in the resulting submatrix, all elements in a row have the same value. Thus, the additive model degenerates to the second case. Therefore, our models can also handle the additive model by trying all rows in A as row r.

Cheng and Church were the first to use the biclustering approach for microarray data analysis [22]. They introduced the *mean squared residue score* H to measure the coherence of the rows and columns in the submatrix.

$$H(P,Q) = \frac{1}{|P||Q|} \sum_{i \in P, j \in Q} (a_{i,j} - a_{i,Q} - a_{P,j} + a_{P,Q})^2,$$

where

$$a_{i,Q} = \frac{1}{|Q|} \sum_{j \in Q} a_{i,j}, \quad a_{P,j} = \frac{1}{|P|} \sum_{i \in P} a_{i,j}, \quad \text{and} \quad a_{P,Q} = \frac{1}{|P||Q|} \sum_{i \in P, j \in Q} (a_{i,j}).$$

Clearly, the H score is 0 for the first two cases if the submatrix is error free. We can show that for the additive model, the H score is also 0 if the submatrix is error free.

In this paper, we design randomized approximation algorithms for both problems. We have a new idea to randomly select $\Theta(\log m)$ columns in the optimal set of columns $Q_{opt} \subseteq Y$ when Q_{opt} is not known. For the bottleneck submatrix problem, we use linear programming and randomized rounding to successfully select a good approximation Q of Q_{opt} and set the letters for the center vector at the columns in Q. Using standard techniques, we can derandomize the randomized algorithms to get polynomial time approximation schemes (PTAS) for the two problems. To the best of our knowledge, this is the first time that approximation algorithms with guaranteed ratios have been presented for microarray analysis.

The paper is organized as follows. In section 2, we prove that both problems are NP-hard. In section 3, we give the algorithm for the consensus submatrix problem. The algorithm for the bottleneck submatrix problem is given in section 4.

2. NP-hardness result. In this section, we will show that both the consensus submatrix problem and the bottleneck submatrix problem are NP-hard. The reduction is from the maximum edge biclique problem. The maximum edge biclique problem was proved to be NP-hard in [25]. A biclique is a complete bipartite subgraph, where every vertex of the first set is connected to every vertex of the second set.

The maximum edge biclique problem. Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a positive integer t, does G contain a biclique with t edges?

THEOREM 1. *The consensus submatrix problem and the bottleneck submatrix problem are NP-hard.*

Proof. We use a reduction from the maximum edge biclique problem. Given a graph $G = (V_1 \cup V_2, E)$ and a positive integer t, where $|V_1| = m$, $|V_2| = n$, we construct an $m \times n$ matrix $A = \{a_{i,j}\}$ as follows: $a_{i,j} = 1$ if and only if $(v_1(i), v_2(j)) \in E$, and otherwise $a_{i,j} = 0$. After constructing A, we get a new matrix A' of size $2m \times n$ by

adding m rows, each containing n 1's, to A. Thus, for an element $a'_{i,j}$ in A', $a'_{i,j} = a_{i,j}$ if $1 \leq i \leq m$. Otherwise, $a'_{i,j} = 1$. (Here we need A' to ensure that every element in the consensus/center vector is 1.)

The input of both the consensus submatrix and the bottleneck submatrix problems contains the matrix A' and two integers $l+m$ and k with $t = l \times k$. The number of pairs for l and k is at most t^2.

It is easy to see that there exists an $l \times k$ biclique for the maximum edge biclique problem if and only if there is a size $l \times k$ submatrix of A in which every element is 1.

Now, we want to show that there is an $l \times k$ submatrix of A such that every element is 1 if and only if there is an $(l+m) \times k$ submatrix of A' such that every element is 1.

(if) Given an $l \times k$ submatrix of A such that every element is 1, we can obtain an $(l+m) \times k$ submatrix of A' such that every element is 1 by adding the m newly added rows with every element being 1 in A'.

(only if) Consider an $(l+m) \times k$ submatrix of A' such that every element is 1. There are at least l rows in the $(l+m) \times k$ submatrix that are also in A since there are m newly added rows in A'. Thus, we can select l rows in the $(l+m) \times k$ submatrix that are also in A. In this way, we get an $l \times k$ submatrix of A such that every element is 1.

Obviously, every element in the $(l+m) \times k$ submatrix of A' is 1 if and only if both the consensus score and the bottleneck score for the submatrix are 0 and every element in the consensus/center vector is 1. □

The proof also suggests that it is NP-hard to decide whether the score is 0 in both problems. Therefore, there is no approximation algorithm with *any* guaranteed ratio for both problems when the optimal score is 0.

3. The consensus submatrix problem. In this section, we will present the randomized approximation algorithm for the consensus submatrix problem. Let P_{opt}, Q_{opt}, and z_{opt} be the set of rows, the set of columns, and the consensus vector of an optimal solution. The optimal consensus score is H_{opt}. By assumption, $H_{opt} = \Omega(kl)$, i.e., there is a constant c' such that $H_{opt} \times c' = kl$. Again, by assumption, $k = \Omega(n)$, i.e., there is a constant c such that $k \times c = n$.

Before we present the algorithm, we first introduce the basic ideas of the algorithm. By enumerating all size k subsets of Y and all length k vectors, we can find Q_{opt} and z_{opt} at some moment. It is easy to see that if we know exactly Q_{opt} and z_{opt}, then we can find the corresponding P_{opt} in polynomial time to minimize the consensus score. However, this straightforward approach costs exponential time. Here we use a random sampling technique to randomly select $\Theta(\log m)$ columns in Q_{opt}, and enumerate all possible vectors of length $\Theta(\log m)$ for those columns. (If $n < \Theta(\log m)$, we will select all the n columns and the running time is still polynomial in terms of the input size.) At some point, we know $\Theta(\log m)$ bits of z_{opt} and we can use the partial z_{opt} to select the l rows which are closest to z_{opt} in those $\Theta(\log m)$ bits. After that we can construct a consensus vector z as follows. For each column, choose the (majority) letter that appears the most in each of the l letters in the l selected rows. Then for each of the n columns, we can calculate the number of mismatches between the majority letter and the l letters in the l selected rows. By selecting the best k columns, we can get a good solution.

The remaining difficulty is how to randomly select $\Theta(\log m)$ columns in Q_{opt} while Q_{opt} is unknown. Our new idea is to randomly select a set B of $\lceil (c+1)(\frac{4 \log m}{\epsilon^2} + 1) \rceil$ columns from A and enumerate all size $\lceil \frac{4 \log m}{\epsilon^2} \rceil$ subsets of B in time $O(m^{\frac{4(c+1)}{\epsilon^2}})$,

Algorithm 1 for the Consensus Submatrix Problem

Input: an $m \times n$ matrix A, integers l and k, and a number $\epsilon > 0$.

Output: a size l subset P of rows, a size k subset Q of columns and a length k consensus vector z.

Step 1: Randomly and independently select a set B of $\lceil (c+1)(\frac{4\log m}{\epsilon^2}+1)\rceil$ columns from A. (If $n < \lceil (c+1)(\frac{4\log m}{\epsilon^2}+1)\rceil$, we will select all the n columns and the running time is still polynomial in terms of the input size.)

(1.1) **for** every size $\lceil \frac{4\log m}{\epsilon^2}\rceil$ subset R of B **do**

(1.2) **for** every $z|^R \in \Sigma^{|R|}$ **do**

(a) Select the best l rows $P = \{p_1, \ldots, p_l\}$ that minimize $d(z|^R, x_i|^R)$.

(b) **for** each column j **do**

Compute $f(j) = \sum_{i=1}^{l} d(s_j, a_{p_i,j})$, where s_j is the majority element of the l rows in P in column j.

Select the best k columns $Q = \{q_1, \ldots, q_k\}$ with minimum value $f(j)$ and let $z(Q) = s_{q_1} s_{q_2} \ldots s_{q_k}$.

(c) Calculate $H = \sum_{i=1}^{l} d(x_{p_i}|^Q, z)$ of this solution.

Step 2: Output P, Q and z with minimum H.

FIG. 2. *Algorithm* 1.

which is polynomial in terms of the input size $O(mn)$. We can show that with high probability, we can get a set of $\lceil \frac{4\log m}{\epsilon^2}\rceil$ columns randomly selected from Q_{opt}.

Now we describe the complete algorithm in Figure 2.

The following lemma, originally from [26], will be used in our proofs.

LEMMA 1. *Let $X_1, X_2, \ldots, X_n$ be n independent random* 0-1 *variables, where X_i takes* 1 *with probability p_i, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then for any $0 < \epsilon \le 1$,*

$$\mathbf{Pr}(X > \mu + \epsilon\, n) < \exp\left(-\frac{1}{3} n\epsilon^2\right),$$

$$\mathbf{Pr}(X < \mu - \epsilon\, n) \le \exp\left(-\frac{1}{2} n\epsilon^2\right).$$

LEMMA 2. *With probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}}$, no subset R of size $\lceil \frac{4\log m}{\epsilon^2}\rceil$ used in Step* 1 *of Algorithm* 1 *satisfies $R \subseteq Q_{opt}$.*

Proof. Obviously, if the subset B in Step 1 of Algorithm 1 contains at least $\lceil \frac{4\log m}{\epsilon^2}\rceil$ columns in Q_{opt}, there is a size $\lceil \frac{4\log m}{\epsilon^2}\rceil$ subset $R \subseteq Q_{opt}$. Now we consider the probability that the subset B contains less than $\lceil \frac{4\log m}{\epsilon^2}\rceil$ columns in Q_{opt}. Let b be the number of columns in B that are also in Q_{opt}. Recall that $k \times c = n$. If we randomly select a column, the probability that the column is in Q_{opt} is $\frac{1}{c}$. Let μ be the expected number of columns in B that are in Q_{opt}. We have $\mu = \frac{|B|}{c} = \frac{1}{c}\lceil (c+1)(\frac{4\log m}{\epsilon^2}+1)\rceil$. Let $X_1, X_2, \ldots, X_{|B|}$ be $|B|$ independent random 0/1 variables, where $X_i = 1$ with probability $\frac{1}{c}$ indicating that the selected column is in Q_{opt}. Thus,

$$b = \sum_{i=1}^{|B|} X_i \tag{2}$$

and

$$\mu = E\left(\sum_{i=1}^{|B|} X_i\right) = \frac{1}{c}\left\lceil (c+1)\left(\frac{4\log m}{\epsilon^2}+1\right)\right\rceil. \tag{3}$$

From the algorithm,

$$|B| = \left\lceil (c+1)\left(\frac{4\log m}{\epsilon^2}+1\right)\right\rceil. \tag{4}$$

Based on Lemma 1, we have

$$\begin{aligned}
\mathbf{Pr}\left(b < \left\lceil \frac{4\log m}{\epsilon^2}\right\rceil\right) &\le \mathbf{Pr}\left(b < \frac{4\log m}{\epsilon^2}+1\right)\\
&\le \mathbf{Pr}\left(b < \left(\frac{1}{c}-\frac{1}{c(c+1)}\right)\left\lceil (c+1)\left(\frac{4\log m}{\epsilon^2}+1\right)\right\rceil\right)\\
&= \mathbf{Pr}\left(\sum_{i=1}^{|B|} X_i < \mu - \frac{1}{c(c+1)}|B|\right) \quad \text{(from (2), (3), and (4))}\\
&\le \exp\left(-\frac{1}{2c^2(c+1)^2}|B|\right)\\
&\le \exp\left(-\frac{1}{2c^2(c+1)^2}(c+1)\left(\frac{4\log m}{\epsilon^2}\right)\right)\\
&= \exp\left(-\frac{2\log m}{\epsilon^2 c^2(c+1)}\right) = m^{-\frac{2}{\epsilon^2 c^2(c+1)}}.
\end{aligned}$$

Therefore, with probability at most $m^{-\frac{2}{\epsilon^2 c^2(c+1)}}$, no subset R of size $\lceil \frac{4\log m}{\epsilon^2}\rceil$ used in Step 1 of Algorithm 1 satisfies $R \subseteq Q_{opt}$. ∎

LEMMA 3. *Assume* $|R| = \lceil \frac{4\log m}{\epsilon^2}\rceil$ *and* $R \subseteq Q_{opt}$. *Let* $\rho = \frac{k}{|R|}$. *With probability at most* m^{-1}, *there is a row* x_i *in* X *satisfying*

$$\frac{d(z_{opt}, x_i|^{Q_{opt}}) - \epsilon k}{\rho} > d(z_{opt}|^R, x_i|^R). \tag{5}$$

With probability at most $m^{-\frac{1}{3}}$, *there is a row* x_i *in* X *satisfying*

$$d(z_{opt}|^R, x_i|^R) > \frac{d(z_{opt}, x_i|^{Q_{opt}}) + \epsilon k}{\rho}. \tag{6}$$

Proof. We will calculate $\mathbf{Pr}(d(z_{opt}|^R, x_i|^R) < \frac{d(z_{opt}, x_i|^{Q_{opt}})-\epsilon k}{\rho})$ first. From Algorithm 1, R is a subset of B, a set of randomly independently selected columns. Thus, R is also a set of randomly independently selected columns. Therefore, we can treat $d(z_{opt}|^R, x_i|^R)$ as the sum of $|R|$ independent random 0/1 variables $\sum_{j=1}^{|R|} X_j$. As $R \subseteq Q_{opt}$, $X_j = 1$ indicates a mismatch between z_{opt} and x_i at the jth position in R. Clearly $E[d(z_{opt}|^R, x_i|^R)] = d(z_{opt}, x_i|^{Q_{opt}})/\rho$. From Lemma 1, we have

$$\begin{aligned}
&\mathbf{Pr}\left(d(z_{opt}|^R, x_i|^R) < \frac{d(z_{opt}, x_i|^{Q_{opt}}) - \epsilon k}{\rho}\right)\\
&= \mathbf{Pr}(d(z_{opt}|^R, x_i|^R) < E[d(z_{opt}|^R, x_i|^R)] - \epsilon|R|)\\
&\le \exp\left(-\frac{1}{2}\epsilon^2|R|\right) \le m^{-2},
\end{aligned}$$

where the last inequality is due to the setting $|R| = \lceil \frac{4 \log m}{\epsilon^2} \rceil$ in Step 1 of Algorithm 1. Considering all the m x_i's, the probability that an $x_i \in X$ satisfies (5) is at most $m \times m^{-2} = m^{-1}$.

Similarly, we have

$$\mathbf{Pr}\left(d(z_{opt}|^R, x_i|^R) > \frac{d(z_{opt}, x_i|^{Q_{opt}}) + \epsilon k}{\rho} \right) < m^{-\frac{4}{3}}.$$

Considering all the m x_i's, the probability that an $x_i \in X$ satisfies (6) is at most $m \times m^{-\frac{4}{3}} = m^{-\frac{1}{3}}$. ∎

LEMMA 4. *When $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $2m^{-\frac{1}{3}}$, the set of rows $P = \{p_1, \ldots, p_l\}$ selected in Step* 1(a) *of Algorithm* 1 *satisfies $\sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) > H_{opt} + 2\epsilon kl$.*

Proof. Let $P_{opt} = \{p_1^*, \ldots, p_l^*\}$ be the l rows in the optimal solution; we have

$$\sum_{i=1}^{l} d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) = H_{opt}. \tag{7}$$

From Lemma 3, with probability at most m^{-1}, a row $x_{p_i} \in \{p_1, p_2, \ldots, p_l\}$ satisfies

$$\sum_{i=1}^{l} d(z_{opt}|^R, x_{p_i}|^R) < \sum_{i=1}^{l} \frac{d(z_{opt}, x_{p_i}|^{Q_{opt}}) - \epsilon k}{\rho}. \tag{8}$$

Again from Lemma 3, with probability at most $m^{-\frac{1}{3}}$, a row $x_{p_i^*} \in \{p_1^*, p_2^*, \ldots, p_l^*\}$ satisfies

$$\sum_{i=1}^{l} d(z_{opt}|^R, x_{p_i^*}|^R) > \sum_{i=1}^{l} \frac{d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) + \epsilon k}{\rho}. \tag{9}$$

By trying all length $|R|$ vectors in Step 1 of Algorithm 1, we can assume that we know $z_{opt}|^R$. In Step 1(a), we select the best set $P = \{p_1, \ldots, p_l\}$ with minimum $d(z_{opt}|^R, x_i|^R)$. Thus, for any $R \subseteq Q_{opt}$, we have

$$\sum_{i=1}^{l} d(z_{opt}|^R, x_{p_i}|^R) \leq \sum_{i=1}^{l} d(z_{opt}|^R, x_{p_i^*}|^R). \tag{10}$$

From (8), (9), and (10), if $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $m^{-1} + m^{-\frac{1}{3}} \leq 2m^{-\frac{1}{3}}$, there exists a set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step 1(a) of Algorithm 1 such that

$$\begin{aligned} \sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) &> \sum_{i=1}^{l} (d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) + 2\epsilon k) \\ &= H_{opt} + 2\epsilon kl. \end{aligned}$$

The last inequality is from (7). ∎

THEOREM 2. *For any $\delta > 0$, with probability at least $1 - m^{-\frac{8c'^2}{\delta^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}}$, Algorithm* 1 *outputs a solution with consensus score at most $(1+\delta)H_{opt}$ in $O(nm^{O(\frac{1}{\delta^2})})$ time.*

Proof. When $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, in Step 1(b), we can construct a Q and $z(Q)$ such that

$$\sum_{i=1}^{l} d(z(Q), x_{p_i}|^Q) \leq \sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}). \tag{11}$$

From Lemma 2, we know that with probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}}$, there is no subset R with size $\lceil \frac{4 \log m}{\epsilon^2} \rceil$ in Step 1 of Algorithm 1 such that $R \subseteq Q_{opt}$. Combining with Lemma 4, we know that with probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} + 2m^{-\frac{1}{3}}$, in the execution of Algorithm 1, any set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step 1(a) satisfies

$$\sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) > H_{opt} + 2\epsilon kl.$$

In other words, with probability at least $1 - m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}}$, in the execution of Algorithm 1, we can get a set of rows $P = \{p_1, \ldots, p_l\}$ in Step 1(a) that satisfies $\sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) \leq H_{opt} + 2\epsilon kl$.

From (11), we have

$$\sum_{i=1}^{l} d(z(Q), x_{p_i}|^Q) \leq \sum_{i=1}^{l} d(z_{opt}, x_{p_i}|^{Q_{opt}}) \leq H_{opt} + 2\epsilon kl.$$

Recall that $H_{opt} \times c' = kl$. Set $\epsilon = \frac{\delta}{2c'}$. So with probability at least $1 - m^{-\frac{8c'^2}{\delta^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}}$, Algorithm 1 outputs a solution with consensus score at most $(1+\delta)H_{opt}$.

For the time complexity, Step 1(a), Step 1(b), and Step 1(c) take $O(mn)$ time. Step 1.1 is repeated $O(2^{\frac{4(c+1)\log m}{\epsilon^2}}) = O(m^{O(\frac{1}{\epsilon^2})}) = O(m^{O(\frac{1}{\delta^2})})$ times. Step 1.2 is repeated $O(m^{O(\frac{\log |\Sigma|}{\epsilon^2})}) = O(m^{O(\frac{1}{\delta^2})})$ times as $\epsilon = \frac{\delta}{2c'}$ and $|\Sigma|$ is a fixed constant. Thus, the total running time is $O(nm^{O(\frac{1}{\delta^2})})$. ∎

THEOREM 3. *There exists a PTAS for the consensus submatrix problem.*

Proof. Algorithm 1 can be derandomized by the standard method. For instance, instead of randomly and independently choosing $\Theta(\log m)$ columns from the n columns in Step 1, we can pick the vertices encountered on a random walk of length $\Theta(\log m)$ on a constant degree expander [27]. Obviously, the number of such random walks on a constant degree expander is polynomial in terms of m. Thus, by enumerating all random walks of length $\Theta(\log m)$, we have a polynomial time deterministic algorithm. (Also see [30].) ∎

4. The bottleneck submatrix problem. In this section, we present the randomized approximation algorithm for the bottleneck submatrix problem. Let P_{opt}, Q_{opt}, and z_{opt} be the set of rows, the set of columns, and the center vector of an optimal solution. The optimal bottleneck score is d_{opt}. By assumption, $d_{opt} = \Omega(k)$ and $k = \Omega(n)$, i.e., there are constants c'' and c such that $d_{opt} \times c'' = k$ and $k \times c = n$.

Similar to Algorithm 1, we can use a random sampling technique to know $\Theta(\log m)$ bits of z_{opt}. Then we can use the partial z_{opt} to select the l rows which are closest to z_{opt} in those $\Theta(\log m)$ bits as in Step 1(a) of Algorithm 1. From Lemma 3, we know that using $\Theta(\log m)$ bits in R, we can get a good estimation of $d(z_{opt}, x_i|^{Q_{opt}})$ for each

x_i in X. Thus, if we can correctly select Q_{opt} from the n given columns, then we can get a good approximation solution. However, Step 1(b) in Algorithm 1 does not work for the bottleneck score in selecting a good approximation of Q_{opt}. Thus, we use the linear programming and randomized rounding technique to select k columns in the matrix.

Linear programming formulation. Given a set of rows $P = \{p_1, \ldots, p_l\}$, we want to find a set of k columns Q and a vector z such that the bottleneck score is minimized. This problem is equivalent to the following optimization problem:

$$\begin{cases} \min d; \\ d(z, x_{p_i}|^Q) \le d, \quad i = 1, 2, \ldots, l, \quad Q \subseteq Y, \quad |Q| = k, z \in \Sigma^k. \end{cases} \tag{12}$$

Let $\Sigma = \{\pi_1, \pi_2, \ldots, \pi_{|\Sigma|}\}$. We introduce 0/1 variable $y_{i,j}$ $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, |\Sigma|)$ to indicate the membership of column i in Q and the corresponding bit of z. We have $y_{i,j} = 1$ if and only if column i is in Q and the corresponding bit in z is π_j. For any $a, b \in \Sigma$, $\chi(a, b) = 0$ if $a = b$ and $\chi(a, b) = 1$ if $a \ne b$. We can formulate (12) as 0/1 integer linear programming:

$$\begin{cases} \min d; \\ \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{|\Sigma|} y_{i,j} = k, \\ \displaystyle\sum_{j=1}^{|\Sigma|} y_{i,j} \le 1, \quad i = 1, 2, \ldots, n, \\ \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y_{i,j} \le d, \quad s = 1, 2, \ldots, l. \end{cases} \tag{13}$$

Here $y_{i,j}$ is used to achieve two tasks: (1) to decide whether column i is selected and (2) if column i is selected, to choose the letter in the center vector z on this column.

We can obtain a fraction solution $y_{i,j} = \overline{y_{i,j}} (i = 1, 2, \ldots, n, j = 1, 2, \ldots, |\Sigma|)$ for (13) in polynomial time. After we get the fraction solution, we do randomized rounding to get an integer solution.

Randomized rounding. Given a fraction solution $y_{i,j} = \overline{y_{i,j}}$ $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, |\Sigma|)$ with cost $\overline{d}$, for each $1 \le i \le n, 1 \le j \le |\Sigma|$, we randomly select column i to Q with probability $\sum_{j=1}^{|\Sigma|} \overline{y_{i,j}}$ and randomly set the bit of z in this column according to the distribution $\overline{y_{i,j}}$ for $j = 1, 2, \ldots, |\Sigma|$. In terms of programming, we can generate a random number ρ in (0,1) for every column i. If $\rho < \sum_{j=1}^{|\Sigma|} \overline{y_{i,j}}$, we select this column into Q and let the bit of z corresponding to this column be π_t if and only if $\sum_{j=1}^{t-1} \overline{y_{i,j}} \le \rho < \sum_{j=1}^{t} \overline{y_{i,j}}$. If $\rho \ge \sum_{j=1}^{|\Sigma|} \overline{y_{i,j}}$, this column is not selected. Hence we get a 0/1 integer solution $y' = \{y'_{1,1}, \ldots, y'_{1,|\Sigma|}, \ldots, y'_{n,1}, \ldots, y'_{n,|\Sigma|}\}$.

In this randomized rounding process, we have to do two things: (1) select k' columns, where $k' \ge k - \delta d_{opt}$, and (2) get the integral value for each $y_{i,j}$ such that the distance (restricted on the k' selected columns) between any row in P and the center vector thus obtained is at most γd_{opt}. Here $\delta > 0$ and $\gamma > 0$ are two parameters used to control the errors.

LEMMA 5. *When* $\frac{n\gamma^2}{3(cc')^2} \ge 2 \log m$, *for any* $\gamma, \delta > 0$, *with probability at most* $\exp(-\frac{n\delta^2}{2(cc')^2}) + m^{-1}$, *the integer solution* $y' = \{y'_{1,1}, \ldots, y'_{1,|\Sigma|}, \ldots, y'_{n,1}, \ldots, y'_{n,|\Sigma|}\}$

violates at least one of the following inequalities,

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} y'_{i,j}\right) > k - \delta d_{opt}, \tag{14}$$

and for every row $x_{p_s}(s = 1, 2, \ldots, l)$*,*

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}\right) < \overline{d} + \gamma d_{opt}. \tag{15}$$

Proof. From $k \times c = n$ and $d_{opt} \times c'' = k$, we have $d_{opt} \times cc'' = n$. For each $1 \le i \le n$, the randomized rounding process ensures that at most one $y'_{i,a} = 1$ for $\pi_a \in \Sigma$. Since the rounding is independent for different i's, $(\sum_{j=1}^{|\Sigma|} y'_{i,j})$'s are independent 0/1 random variables for $1 \le i \le n$, and $(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j})$'s are independent 0/1 random variables for $1 \le i \le n$ in every row x_{p_s}. It is easy to see that

$$E\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} y'_{i,j}\right)\right) = k,$$

and for every row $x_{p_s}(s = 1, 2, \ldots, l)$,

$$E\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}\right)\right) \le \overline{d}.$$

From Lemma 1, we have

$$\begin{aligned}\mathbf{Pr}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} y'_{i,j}\right) < k - \delta d_{opt}\right) &= \mathbf{Pr}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} y'_{i,j}\right) < E\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} y'_{i,j}\right)\right) - \frac{\delta n}{cc''}\right) \\ &\le \exp\left(-\frac{n\delta^2}{2(cc'')^2}\right),\end{aligned}$$

and for every row $x_{p_s}(s = 1, 2, \ldots, l)$,

$$\begin{aligned}&\mathbf{Pr}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}\right) > \overline{d} + \gamma d_{opt}\right) \\ &\le \mathbf{Pr}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}\right) > E\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}\right)\right) + \frac{\gamma}{cc''} n\right) \\ &< \exp\left(-\frac{n\gamma^2}{3(cc'')^2}\right).\end{aligned}$$

Considering all l rows in P, the probability that at least one row in P satisfies $\sum_{i=1}^{n}(\sum_{j=1}^{|\Sigma|} \chi(\pi_j, x_{p_s,i}) y'_{i,j}) > \overline{d} + \gamma d_{opt}$ is at most $l \times \exp(-\frac{n\gamma^2}{3(cc'')^2}) \le m \times \exp(-\frac{n\gamma^2}{3(cc'')^2})$. Thus, with probability at most $\exp(-\frac{n\delta^2}{2(cc'')^2}) + m \times \exp(-\frac{n\gamma^2}{3(cc'')^2})$, the integer solution $y' = \{y'_{1,1}, \ldots, y'_{1,|\Sigma|}, \ldots, y'_{n,1}, \ldots, y'_{n,|\Sigma|}\}$ violates at least one of (14) and (15). When $\frac{n\gamma^2}{3(cc'')^2} \ge 2 \log m$, we get the desired probability. ∎

Algorithm 2 for the Bottleneck Submatrix Problem

Input: a matrix $A \in \Sigma^{m\times n}$, integers l, k, a row $z \in \Sigma^n$ and numbers $\epsilon > 0$, and $\gamma > 0$.

Output: a size l subset P of rows, a size k subset Q of columns and a length k center vector z.

if $\frac{n\gamma^2}{3(cc'')^2} \le 2\log m$ **then**

try all size k subset Q of the n columns and all center vectors of length k to solve the problem.

if $\frac{n\gamma^2}{3(cc'')^2} > 2\log m$ **then**

Step 1: randomly and independently select a set B of $\lceil \frac{4(c+1)\log m}{\epsilon^2} \rceil$ columns from A.

for every $\lceil \frac{4\log m}{\epsilon^2} \rceil$ size subset R of B **do**

for every $z|^R \in \Sigma^{|R|}$ **do**

(a) Select the best l rows $P = \{p_1, \ldots, p_l\}$ that minimize $d(z|^R, x_i|^R)$.

(b) Solve the optimization problem (12) by linear programming and randomized rounding to get Q and z.

Step 2: Output P,Q and z with minimum bottleneck score d.

FIG. 3. *Algorithm* 2.

When $\frac{n\gamma^2}{3(cc'')^2} < 2\log m$, we try all subsets of X with size k and all length k vectors in polynomial time and get the best solution.

From Lemma 5, we know that in the randomized rounding process, with high probability, we selected k' columns in Q, where $(1-\epsilon)k \le k'$. Our aim is to exactly select k columns. If $k' > k$, we can arbitrarily delete $k'-k$ columns from Q and obtain the set of k columns $Q' \subseteq Q$. If $k' < k$, we can arbitrarily select $k-k'$ columns outside Q and add them to Q to get a set of k columns $Q' \supset Q$. By doing so, the extra error introduced is at most ϵk. Since $d = \Omega(n)$, the error ϵk is small and we still can get a PTAS.

Now we describe the complete algorithm in Figure 3.

Similar to Lemma 4, we have the following lemma.

LEMMA 6. *When $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $2m^{-\frac{1}{3}}$, the set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step* 1(a) *of Algorithm* 2 *satisfies $d(z_{opt}, x_{p_i}|^{Q_{opt}}) > d_{opt} + 2\epsilon k$ for some row $x_{p_i}(1 \le i \le l)$.*

Proof. Let $P_{opt} = \{p_1^*, \ldots, p_l^*\}$ be the l rows in the optimal solution. We have

$$\max_{p_i^* \in P_{opt}} d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) = d_{opt}. \tag{16}$$

From Lemma 3, with probability at most m^{-1}, a row $x_{p_i} \in \{p_1, p_2, \ldots, p_l\}$ satisfies

$$d(z_{opt}|^R, x_{p_i}|^R) < \frac{d(z_{opt}, x_{p_i}|^{Q_{opt}}) - \epsilon k}{\rho}. \tag{17}$$

Again from Lemma 3, with probability at most $m^{-\frac{1}{3}}$, a row $x_{p_i^*} \in \{p_1^*, p_2^*, \ldots, p_l^*\}$ satisfies

$$d(z_{opt}|^R, x_{p_i^*}|^R) > \frac{d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) + \epsilon k}{\rho}. \tag{18}$$

By trying all length $|R|$ vectors in Step 1 of Algorithm 2, we can assume that we know $z_{opt}|^R$. In Step 1(a), we select the best set $P = \{p_1, \ldots, p_l\}$ with minimum $d(z_{opt}|^R, x_i|^R)$. Thus, for any $R \subseteq Q_{opt}$, for every $p_i \in P$, we have

$$d(z_{opt}|^R, x_{p_i}|^R) \leq \max_{p_i^* \in P_{opt}} d(z_{opt}|^R, x_{p_i^*}|^R). \tag{19}$$

From (17), (18), and (19), if $R \subseteq Q_{opt}$ and $z|^R = z_{opt}|^R$, with probability at most $m^{-1} + m^{-\frac{1}{3}} \leq 2m^{-\frac{1}{3}}$, there exists a set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step 1(a) of Algorithm 2 such that for some row $x_{p_i} (1 \leq i \leq l)$,

$$\begin{aligned} d(z_{opt}, x_{p_i}|^{Q_{opt}}) &> \max_{p_i^* \in P_{opt}} d(z_{opt}, x_{p_i^*}|^{Q_{opt}}) + 2\epsilon k \\ &= d_{opt} + 2\epsilon k. \end{aligned}$$

The last inequality is from (16). ∎

From Lemmas 2, 5, and 6, we have the following theorem.

THEOREM 4. *With probability at least* $1 - m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}} - \exp(-\frac{n\delta^2}{2(cc'')^2}) - m^{-1}$, *Algorithm* 2 *runs in time* $O(n^{O(1)} m^{O(\frac{1}{\epsilon^2} + \frac{1}{\gamma^2})})$ *and obtains a solution with the bottleneck score at most* $(1 + 2c''\epsilon + \gamma + \delta) d_{opt}$ *for any fixed* $\epsilon,\ \gamma,\ \delta > 0$.

Proof. From Lemma 2, we know that with probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}}$, there is no subset R with size $\lceil \frac{4 \log m}{\epsilon^2} \rceil$ in Step 1 of Algorithm 2 such that $R \subseteq Q_{opt}$. Combining with Lemma 6, we know that with probability at most $m^{-\frac{8c''^2}{\epsilon'^2 c^2 (c+1)}} + 2m^{-\frac{1}{3}}$, in the execution of Algorithm 2, for any set of rows $P = \{p_1, \ldots, p_l\}$ obtained in Step 1(a), problem (12) has a solution $Q = Q_{opt}$ and $z = z_{opt}$ with the bottleneck score $d_P > (1 + 2c''\epsilon) d_{opt}$ as $k = 2c'' d_{opt}$. In Step 1(b), we can get a fraction solution $y_{i,j} = \overline{y_{i,j}}$ $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, |\Sigma|)$ with cost $\overline{d} < d_P$. Thus, with probability at most $m^{-\frac{8c''^2}{\epsilon'^2 c^2 (c+1)}} + 2m^{-\frac{1}{3}}$,

$$\overline{d} > (1 + 2c''\epsilon) d_{opt}. \tag{20}$$

From Lemma 5, when $\frac{n\gamma^2}{3(cc'')^2} \geq 2 \log m$, we know that with probability at most $\exp(-\frac{n\delta^2}{2(cc'')^2}) + m^{-1}$, the integer solution $y' = \{y'_{1,1}, \ldots, y'_{1,|\Sigma|}, \ldots, y'_{n,1}, \ \ldots, y'_{n,|\Sigma|}\}$ violates at least one of (14) and (15). Thus the bottleneck score

$$d' > (\delta + \gamma) d_{opt} + \overline{d}. \tag{21}$$

From (20) and (21), when $\frac{n\gamma^2}{3(cc'')^2} \geq 2 \log m$, with probability at most $m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} + 2m^{-\frac{1}{3}} + \exp(-\frac{n\delta^2}{2(cc'')^2}) + m^{-1}$,

$$d' > (1 + 2c''\epsilon + \gamma + \delta) d_{opt}.$$

In other words, when $\frac{n\gamma^2}{3(cc'')^2} \geq 2 \log m$, with probability at least $1 - m^{-\frac{2}{\epsilon^2 c^2 (c+1)}} - 2m^{-\frac{1}{3}} - \exp(-\frac{n\delta^2}{2(cc'')^2}) - m^{-1}$, in the execution of Algorithm 2, we can get a solution in Step 1(b) with the bottleneck score at most $(1 + 2c''\epsilon + \gamma + \delta) d_{opt}$.

For the time complexity, Step 1(a), Step 1(b), and Step 1(c) take $O((mn|\Sigma|)^{O(1)})$ time. Step 1 is repeated at most $O(m^{O(\frac{\log |\Sigma|}{\epsilon^2})}) = O(m^{O(\frac{1}{\epsilon^2})})$ times. Thus, the total time required is $O(n^{O(1)} m^{O(\frac{1}{\epsilon^2})})$.

When $\frac{n\gamma^2}{3(cc')^2} < 2\log m$, we can solve the problem in $O(n^{O(1)}m^{O(\frac{1}{\gamma^2})})$ time by enumerating all possible sets Q and all possible center vectors z. □

THEOREM 5. *There exists a PTAS for the bottleneck submatrix problem.*

Proof. For Step 1(b), we can use the technique in [26] to derandomize it. The derandomization of the random sampling step is the same as in Algorithm 1. □

5. Conclusion. We have designed PTASs for both the consensus submatrix and the bottleneck submatrix problems. To the best of our knowledge, this is the first time that approximation algorithms with guaranteed performance ratios have been presented for microarray data analysis. It is important to point out that the running time here is very high and the algorithms may not work in practice.

Acknowledgment. We thank the referees for their helpful suggestions.

REFERENCES

[1] R. B. STOUGHTON, *Applications of DNA microarrays in biology*, Annu. Rev. Biochem, 74 (2005), pp. 53–82.

[2] D. B. ALLISON, X. CUI, G. P. PAGE, AND M. SABRIPOU, *Microarray data analysis: From disarray to consolidation and consensus*, Nature Rev. Genetics, 7 (2006), pp. 55–65.

[3] S. TAVAZOIE, J. D. HUGHES, M. J. CAMPBELL, R. J. CHO, AND G. M. CHURCH, *Systematic determination of genetic network architecture*, Nature Genetics, 22 (1999), pp. 281–285.

[4] F. X. WU, W. J. ZHANG, AND A. J. KUSALIK, *A genetic K-means clustering algorithm applied to gene expression data*, Lecture Artificial Intelligence, 2671 (2003), pp. 520–526.

[5] P. TAMAYO, D. SLONIM, J. MESIROV, Q. ZHU, S. KITAREEWAN, E. DMITROVSKY, E. S. LANDER, AND T. R. GOLUB, *Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation*, Proc. Natl. Acad. Sci. USA, 96 (1999), pp. 2907–2912.

[6] H. RESSOM, D. WANG, AND P. NATARAJAN, *Clustering gene expression data using adaptive double self-organizing map*, Physiol. Genomics, 14 (2003) pp. 35–46.

[7] M. B. EISEN, P. T. SPELLMAN, P. O. BROWN, AND D. BOTSTEIN, *Cluster analysis and display of genome-wide expression patterns*, Proc. Natl. Acad. Sci. USA, 95 (1998), pp. 14863–14868.

[8] V. R. IYER, M. B. EISEN, D. T. ROSS, G. SCHULER, T. MOORE, J. C. F. LEE, J. M. TRENT, L. M. STAUDT, J. HUDSON, M. S. BOGUSKI, D. LASHKARI, D. SHALON, D. BOTSTEIN, AND P. O. BROWN, *The transcriptional program in the response of human fibroblasts to serum*, Science, 283 (1999), pp. 83–87.

[9] J. QIN, D. P. LEWIS, AND W. S. NOBLE, *Kernel hierarchical gene clustering from microarray expression data*, Bioinformatics, 19 (2003), pp. 2097–2104.

[10] O. ALTER, P. O. BROWN, AND D. BOTSTEIN, *Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms*, Proc. Natl. Acad. Sci. USA, 100 (2003), pp. 3351–3356.

[11] N. S. HOLTER, M. MITRA, A. MARITAN, M. CIEPLAK, J. R. BANAVAR, AND N. V. FEDOROFF, *Fundamental patterns underlying gene expression profiles: Simplicity from complexity*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 8409–8414.

[12] K. C. LI, M. YAN, AND S. S. YUAN, *A simple statistical model for depicting the cdc*15*-synchronized yeast cell-cycle regulated gene expression data*, Statist. Sinica, 12 (2002), pp. 141–158.

[13] B. TJADEN, *An approach for clustering gene expression data with error information*, BMC Bioinform., 7 (2006), p. 17.

[14] B. H. MECHAM, D. Z. WETMORE, Z. SZALLASI, Y. SADOVSKY, I. KOHANE, AND T. J. MARIANI, *Increased measurement accuracy for sequence-verified microarray probes*, Physiol. Genomics, 18 (2004), pp 308–315.

[15] D. M. ROCKE AND B. DUBIN, *A model for measurement error for gene expression arrays*, J. Comput. Biol., 8 (2001), pp. 557–569.

[16] S. DRAGHICI, P. KHATRI, A. C. EKLUND, AND Z. SZALLASI, *Reliability and reproducibility issues in DNA microarray measurements*, TRENDS in Genetics, 22 (2006), pp. 101–109.

[17] J. P. BRODY, B. A. WILLIAMS, B. J. WOLD, AND S. R. QUAKE, *Significance and statistical errors in the analysis of DNA microarray data*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 12975–12978.

[18] E. Purdom and S. P. Holmes, *Error distribution for gene expression data*, Statist. Appl. Genetics Molecular Biol., 4 (2005).
[19] H. Cho and J. K. Lee, *Bayesian hierarchical error model for analysis of gene expression data*, Bioinformatics, 20 (2004), pp. 2016–2025.
[20] X. Liu and L. Wang, *Computing the maximum similarity bi-clusters of gene expression data*, Bioinformatics, 23 (2007), pp. 50–56.
[21] G. Getz, E. Levine, and E. Domany, *Coupled two–way clustering analysis of gene microarray data*, Proc. Natl. Acad. Sci. USA, 2000, pp. 12079–12084.
[22] Y. Cheng and G. M. Church, *Biclustering of expression data*, in Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB), 2000, pp. 93–103.
[23] S. C. Madeira and A. L. Oliveira, *Biclustering algorithms for biological data analysis: A survey*, IEEE/ACM Trans. Comput. Biol. Bioinform., 1 (2004), pp. 24–45.
[24] S. Lonardi, W. Szpankowski, and Q. Yang, *Finding biclusters by random projections*, in Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, Istanbul, Turkey, 2004, pp. 102–116.
[25] R. Peeters, *The maximum edge biclique problem is NP-complete*, Discrete Appl. Math., 131 (2003), pp. 651–654.
[26] M. Li, B. Ma, and L. Wang, *On the closest string and substring problems*, J. ACM, 49 (2002), pp. 157–171.
[27] D. Gillman, *A Chernoff bound for random walks on expander graphs*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 680–691.
[28] M. Koyutürk, W. Szpankowski, and A. Grama, *Biclustering gene-feature matrices for statistically significant dense patterns*, in Proceedings of the 2004 Annual IEEE Bioinformatics Conference, 2004, pp. 480–484.
[29] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, *Systematic determination of genetic network architecture*, Nature Genetics, 22 (1999), pp. 281–285.
[30] S. Arora, D. Karger, and M. Karpinski, *Polynomial time approximation schemes for dense instances of NP-hard problems*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, ACM, New York, 1995, pp. 284–293.