JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 10, Numbers 3–4, 2003

Pp. 373–384

Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem

AMIR BEN-DOR,[1] BENNY CHOR,[2] RICHARD KARP,[3] and ZOHAR YAKHINI[4]

ABSTRACT

This paper concerns the discovery of patterns in gene expression matrices, in which each element gives the expression level of a given gene in a given experiment. Most existing methods for pattern discovery in such matrices are based on clustering genes by comparing their expression levels in all experiments, or clustering experiments by comparing their expression levels for all genes. Our work goes beyond such global approaches by looking for local patterns that manifest themselves when we focus simultaneously on a subset G of the genes and a subset T of the experiments. Specifically, we look for *order-preserving submatrices* (OPSMs), in which the expression levels of all genes induce the same linear ordering of the experiments (we show that the OPSM search problem is NP-hard in the worst case). Such a pattern might arise, for example, if the experiments in T represent distinct stages in the progress of a disease or in a cellular process and the expression levels of all genes in G vary across the stages in the same way. We define a probabilistic model in which an OPSM is hidden within an otherwise random matrix. Guided by this model, we develop an efficient algorithm for finding the hidden OPSM in the random matrix. In data generated according to the model, the algorithm recovers the hidden OPSM with a very high success rate. Application of the methods to breast cancer data seem to reveal significant local patterns.

Key words: gene expression, data analysis, local structure, local pattern, non-parametric methods.

1. INTRODUCTION

THE ADVENT OF DNA MICROARRAY TECHNOLOGIES has revolutionized the experimental study of gene expression. Thousands of genes are routinely probed in a parallel fashion, and the expression levels of their transcribed mRNA are reported. By repeating such experiments under different conditions (e.g., different patients, different tissues, or variation of the cells' environment), data from tens to hundreds of experiments can be gathered. The analysis of the resulting large datasets poses numerous algorithmic challenges.

[1]Agilent Laboratories, 12741 NE 39th Street, Bellevue, WA 98005.
[2]School of Computer Science, Tel-Aviv University, Tel Aviv 69978, Israel.
[3]International Computer Science Institute, 194 Center Street, Berkeley, CA 94704.
[4]Agilent Laboratories, 2 Hashlosha Street, Tel Aviv 67060, Israel.

So far, the main approach taken for analyzing gene expression data is *clustering* (and variants thereof). Clustering methods have indeed proved successful in many contexts. There is a very large body of literature on clustering in general and on applying clustering techniques to gene expression data in particular. The following list of examples represents the viewpoint of the authors and is not comprehensive or representative. The pioneering paper (Eisen *et al.*, 1998) develops an initial approach to analyzing expression data, mostly adapting hierarchical clustering techniques for this purpose. These methods are successfully applied to yeast cell cycle data in Spellman *et al.* (1998). In Ben-Dor *et al.* (1999) and Sharan and Shamir (2000), more direct approaches to clustering are taken, using graph theoretic methods. Studies where combinations of clustering and classification methods were applied are reported by Golub *et al.* (1999), Bittner *et al.* (2000), and Alizadeh *et al.* (2000). Surveys of analysis approaches appear in Bassett *et al.* (1999) and Gaasterland and Bekiranov (2000).

A major drawback of clustering, however, is the difficulty in identifying patterns that are common to only a part of the expression data matrix. Based on general understanding of cellular processes, we expect subsets of genes to be coregulated and coexpressed under certain experimental conditions, but to behave almost independently under other conditions. Discovery of such local expression patterns may be the key to uncovering many genetic pathways that are not apparent otherwise. It is, therefore, highly desirable to move beyond the clustering paradigm and to develop algorithmic approaches capable of discovering local patterns in microarray data.

Ben-Dor *et al.* (2001) and Xing and Karp (2001) discuss approaches to unsupervised identification of patterns in expression data that distinguish two subclasses of tissues on the basis of a supporting set of genes. "Supporting" here means that high-success-rate classification can be performed based on this set of genes. Ben-Dor *et al.* (2001) describe subclasses that correlate with lymphoma prognosis, based on analyzing data reported (Alizadeh *et al.*, 2000). The present work extends the class discovery task to a progression of more than just two stages.

The first to consider local patterns (*biclusters*) in gene expression data sets were Cheng and Church (2000). Their biclusters are based on uniformity criteria, and they use a greedy algorithm to find them. The *plaid model* (Lazzeroni and Owen, 2000) is a statistical model for gene expression and other data. The plaid model describes two-sided clusters where overlap is allowed. Like our model, their two-sided clusters are not necessarily supported on the entire set of either genes or tissues. The plaid model seeks submatrices that have almost uniform entries. It also affords the identification of submatrices where, over a prescribed set of tissues, genes differ in their expression levels by an almost constant vector. Our work focuses on the uniformity of the relative order of the tissues rather than on the uniformity of the actual expression levels as in the plaid model. This approach is potentially more robust to the stochastic nature of the expression levels and to the variation caused by the measurement process.

In this work, we address the identification and statistical assessment of coexpressed patterns for large sets of genes. For example, in expression data that comes from a population of patients (such as in Bittner *et al.* [2000]), it is reasonable to expect that each individual is in a particular stage of the disease. There is a set of genes that are coexpressed with this progression, and we therefore expect the data to contain a set of genes and a set of patients such that the genes are identically ordered on this set of patients. The same situation occurs when considering data from nominally identical exposure to environmental effects, data from drug treatment, data representing some temporal progression, etc. In many cases, the data contains more than one such pattern. For example, in cancer data, patients can be staged according to the disease progression, as well as according to the extent of genetic abnormalities. These two orders on some subset of tissues are not necessarily correlated. Therefore, even in data where some nominal order is given a priori, we are seeking related or unrelated hidden orders and the sets of genes that support them. In this work, we take first steps towards automatically performing this task.

2. GOAL OF THE PAPER

The readout of a DNA chip containing n genes consists of n real numbers that represent the expression level of each gene, either as an absolute or as a relative quantity (with respect to some reference). When the readouts for m experiments (tissues) are combined, each gene yields a vector of m real numbers. To make our results independent of the scaling of the data, we consider only the relative ordering of the expression

TABLE 1. THE RANKS OF THE THREE GENES g_1, g_2, g_3 INDUCE A COMMON PERMUTATION WHEN RESTRICTED TO COLUMNS t_1, t_2, t_3, t_4, t_5

Gene \ tissue	t_1	t_2	t_3	t_4	t_5
g_1	7	13	19	2	50
g_2	19	23	39	6	42
g_3	4	6	8	2	10
Induced permutation	2	3	4	1	5

levels for each gene, as opposed to the exact values. This motivates us to consider the permutation induced on the m numbers by sorting them. Thus, we view the expressed data matrix, D, as an n-by-m matrix, where each row corresponds to a gene and each column to an experiment. The m entries in each row are a permutation of the numbers $\{1, \ldots, m\}$. The (i, j) entry is the rank of the readout of gene i in tissue j, out of the m readouts of this gene. Typical values for n and m are in the ranges $500 \leq n \leq 15{,}000$ and $10 \leq m \leq 150$.

We are seeking a biological progression that is represented as a "hidden" k-by-s submatrix $G \times T$ inside the data matrix D. The k genes from G are coexpressed in the s tissues from T. This means that the expression levels of all the genes in G move up and down together within the set T. Consider, for example, three genes $g_1, g_2, g_3 \in G$ and the three rows in D corresponding to g_1, g_2, g_3, restricted to the columns in $T = \{t_1, \ldots, t_s\}$. The s ranks in each row correspond to a partial permutation of $\{1, \ldots, m\}$. By projecting the three partial permutations on the subset $\{1, \ldots, s\}$, we get three identical permutations. For a concrete example, see Table 1, where $s = 5$ and $m \geq 50$.

The computational task we address is the identification of large *order-preserving submatrices* (OPSMs) in an $n \times m$ matrix D. A submatrix is order preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. In the case of expression data, such a submatrix is determined by a set of genes G and a set of tissues T such that, within the set of tissues T, the expression levels of all the genes in G have the same linear ordering.

To motivate our heuristic approach to the OPSM problem, we first show that the OPSM problem is NP-Hard, and thus we cannot hope to solve it efficiently in the worst case scenario. The goal of the present paper is to develop an algorithm for finding large submatrices having the strict OPSM condition and to report the performance of the algorithm on both real and simulated data. We begin by formulating a probabilistic model of the expression data to be analyzed. The data consists of an $n \times m$ matrix D, where the rows correspond to genes and the columns to tissues (or, more generally, to experimental samples). Each row of the matrix is a permutation of $\{1, 2, \ldots, m\}$, giving the linear ordering of the expression levels of one gene over all the tissues. We assume that within the matrix D there is a hidden planted submatrix $G \times T$ determined by a set of rows G, a set of columns T, and a linear ordering of the columns of T. Within each row of $G \times T$, the ordering of the entries is consistent with the linear ordering of T. The parameters of the model are n, m, s, and p, where s is the number of elements in T and p is the probability that any given row belongs to G. The s-element set T and the linear ordering of T are randomly chosen. The parameters s and p are not known to the algorithm.

In crafting the algorithm, we are strongly guided by the properties of the probabilistic model. Of course, we do not expect real expression data to conform in detail to such a simple model, but we expect the performance of the algorithm to be robust enough to apply to real data, and preliminary indications are that this is the case.

2.1. Organization

The remainder of this paper is organized as follows. In Section 3, we show that the OPSM problem is NP-hard. In Section 4, we describe the probabilistic model used in the simulation. This model motivates our algorithm design and is used in the simulations. In Section 5, our algorithm is presented. Section 6 contains the results of running our algorithm on simulated and real data. Finally, Section 7 contains some concluding remarks and directions for further research.

3. OPSM IS NP-COMPLETE

In this section we show that the decision version of OPSM is NP-complete. We consider the following decision problem:

Instance: a real valued n-by-m matrix, A, and two integers, k and s.

Question: In A, is there an order-preserving submatrix of size k-by-s? That is, is there a set of row indices $K = \{r_1, \ldots, r_k\}$ and a sequence of column indices $S = (c_1, \ldots, c_s)$ such that $A(r_i, c_j) < A(r_i, c_{j+1})$ for all $1 \leq i \leq k, 1 \leq j \leq s - 1$?

Theorem 3.1. *OPSM is NP-complete.*

Proof. The hardness proof is by reduction from the *balanced complete bipartite subgraph* problem that is known to be NP-complete (Garey and Johnson, 1979):

Instance: bipartite graph $G = (V, U, E)$, positive integer k.

Question: Are there two disjoint subsets $X \subset V, Y \subset U$ such that $|X| = |Y| = k$, and for all $x \in X$, and $y \in Y$, $(x, y) \in E$?

The reduction: Given a bipartite graph $G = (V, U, E)$, define the matrix $A = \{A_{ij}\}$ as follows: if $(v_i, u_j) \in E$, then $A_{i,j} = j$; otherwise, $A_{i,j} = -1$. To finish the reduction, we add an extra column to A consisting of "-1" entries. Thus, the size of A is $|V|$-by-$(|U| + 1)$.

We now show that G contains a balanced complete bipartite graph of size k if and only if the matrix A contains an order preserving submatrix of size k-by-$(k + 1)$. The theorem follows. The first direction follows by construction, if G contains a balanced complete subgraph of size k, then at the same indices we have an order-preserving submatrix of size k-by-k. Note that we can extend this submatrix by the "-1" column to get a k-by-$(k + 1)$ order-preserving submatrix.

To show the other direction, assume that there exists an OPSM B of size k-by-$(k + 1)$ in the matrix A. Note that at most one column of B can contain a "-1" entry. Otherwise, we contradict the order preserving property. Thus, A contains a k-by-k order-preserving submatrix that consists of only positive numbers. This matrix corresponds to a complete bipartite graph in G (at the same indices). ■

4. THE STOCHASTIC MODEL

We model the gene expression data set by a random data matrix D in which an unknown order-preserving submatrix $G \times T$ has been planted. The process of generating a data matrix with a planted order-preserving submatrix consists of three stochastic steps. First, we choose at random the indices for the planted rows and columns. Second, we choose a random ordering for the planted columns. Finally, we assign ranks at random to the data matrix in a way which is consistent with the planted submatrix. More formally, the parameters of the stochastic process are n (number of genes), m (number of experiments), s (size of T), and p (probability that a row i is in the subset G).

1. To determine the set of genes G, toss iid coins X_i for every i $(i = 1, 2, \ldots, n)$, with probability p of coming up heads $(X_i = 1)$. The set G is the set of indices i with $X_i = 1$, and the *expected size* of G equals $p \cdot n$. For the set of experiments, we choose a subset $T \subset \{1, \ldots, m\}$ of size s uniformly at random.
2. Pick uniformly at random a linear ordering $t_1, t_2, \ldots, t_s$ of the elements of T.
3. For every row i, assign the m entries in the i-th row of D independently by a random permutation of $\{1, \ldots, m\}$.
4. For each row i with $X_i = 1$ $(i \in G)$, rearrange the ranks in the columns corresponding to T: The entry in column t_1, $D[i, t_1]$, will be assigned the lowest rank among these s entries, the entry in column t_2 will be assigned the second rank among the entries corresponding to T, and so on. The entry $D[i, t_s]$ will be assigned the highest rank among the entries of T.

At the completion of these three steps, the data matrix D with the planted submatrix $G \times T$ is determined. Note that in addition to the set of planted rows, G, every nonplanted row has a probability of $\frac{1}{s!}$ to satisfy the same ordering constraints as the planted rows. Given D and T, those "spuriously planted" rows are indistinguishable from the "genuinely planted" rows. Thus, the algorithmic goal is, for a given D, to recover the set of planted columns T and their planted linear order π. The set of rows supporting this model ("genuinely planted" together with the "spuriously planted") is then uniquely defined.

5. ALGORITHM

5.1. Complete models

Let $T \subset \{1, \ldots, m\}$ be a set of size s. Let $\pi = (t_1, t_2, \ldots, t_s)$ be a linear ordering of T. The pair (T, π) is called a *complete OPSM model* or simply a *complete model* . We say that a row $i \in \{1, \ldots, n\}$ *supports* (T, π) if the s corresponding entries, ordered according to the permutation π, are monotonically increasing, namely, $D[i, t_1] < D[i, t_2] < \cdots < D[i, t_s]$. Given a complete model (T, π), we can efficiently find out which rows support it (in time $O(n \cdot m)$). Intuitively, our algorithm aims at finding a complete model with the maximum number of rows supporting it. Obviously, there always is a model with $s = 2$ that is supported by at least $n/2$ rows. In general, the absolute number of rows we expect to support a complete model decreases with the model size s. Therefore, rather than a maximum support, our algorithm actually aims at finding a complete model with highest *statistically significant support.* To assess the significance, we compute an upper bound on the probability that a random dataset of size n-by-m (i.e., a dataset in which each row is an independent random permutation of $\{1, 2, \ldots, m\}$) will contain a complete model of size s with k or more rows supporting it.

For a given value of s, the probability that a random row supports a given model (T, π) is $(1/s!)$. As the rows are assumed to be independent, the probability of having at least k rows supporting a model (T, π) is the k-tail of the $(n, (1/s!))$ binomial distribution, namely, $\sum_{i=k}^{n} \binom{n}{i} \left(\frac{1}{s!}\right)^i \left(1 - \frac{1}{s!}\right)^{(n-i)}$. As there are $m_s = m(m-1)\cdots(m-s+1)$ ways to choose a complete model of size s, the following expression $U(s, k)$ is an upper bound on the probability of having a model of size s with support k or greater:

$$U(s, k) = m \cdots (m - s + 1) \sum_{i=k}^{n} \binom{n}{i} \left(\frac{1}{s!}\right)^i \left(1 - \frac{1}{s!}\right)^{(n-i)}.$$

We use this bound as our estimate of the significance of a given model of size s. To account for the fact that s is unknown, we could try all values of s ($s = 2, 3, \ldots, m$), find the best complete model for each, and take the one with the largest statistical significance, namely, the one with the *smallest* $U(\cdot, \cdot)$.

5.2. Partial models

To find the best model for a given s, an exhaustive algorithm could try all $m_s = m(m-1)\cdots(m-s+1)$ complete models. This approach yields an $O(nm^{s+1})$ time algorithm, which is infeasible for $s \geq 4$ and realistic values of the parameters m and n. Instead , our approach is to "grow partial models" iteratively, with the goal of "converging" to the best complete model. A partial model of order (a, b) specifies, in order, the indices of the a "smallest" elements $\langle t_1, \ldots, t_a \rangle$ and the indices of the b "largest" elements $\langle t_{s-b+1}, \ldots, t_s \rangle$ of a complete model (T, π). A partial model also specifies the size s of the complete model.

Let $\theta = \{\langle t_1, \ldots, t_a \rangle, \langle t_{s-b+1}, \ldots, t_s \rangle, s\}$ be a partial model. If $a + b < s$, then such a θ can be extended to several complete models. What we do next is to describe a way to evaluate the quality of a given partial model. Once we do that, our algorithm will be the following: Start by trying all $m(m-1)$ partial models of order $(1, 1)$. Pick the best ℓ partial models of order $(1, 1)$, and for each of them try all $m - 2$ extensions to partial models of order $(2, 1)$. Record the best ℓ of these and for each one try all $m - 3$ extensions to partial models of order $(2, 2)$. Pick the best ℓ of these and keep going this way until we reach ℓ models of order $(\lceil s/2 \rceil, \lfloor s/2 \rfloor)$. These are full models, so we output the best one. The computational complexity of this approach is $O(m^2 + ms\ell)$ times the complexity of evaluating the

quality of a partial model. Our algorithm takes $O(n \cdot s)$ for this subtask, so its overall complexity is $O(ns(m^2 + ms\ell)) = O(nm^2 s + nms^2\ell) = O(nm^3\ell)$.

In defining partial models, we chose to focus on columns at the extremes of the ordering, leaving the internal columns unspecified. This is done because, in a planted row, the distribution of ranks in an extreme column is more concentrated than the distribution in an interior column and more distinguishable from the rank distribution in a nonplanted column. Thus, the extreme columns are more useful in identifying planted rows.

To further illustrate the power of the extreme columns to distinguish the correct partial model from incorrect ones, we define a simple statistic associated with a partial model $\theta = \{\langle t_1, \ldots, t_a\rangle, \langle t_{s-b+1}, \ldots, t_s\rangle, s\}$ and a row i. This statistic is defined as follows:

$$range(i, \theta) = max(0, D[i, t_{s-b+1}] - D[i, t_a)].$$

It is closely related to the *gap* defined in Section 5.3.

For the correct partial model of order (a, b), the expected value of $range(i, \theta)$ is $\frac{m(s+1-a-b)}{s+1}$ for a planted row and $\frac{m}{(a+b+1)!}$ for a nonplanted row (both values are approximations that are highly accurate when m is large). Thus, the correct partial model tends to draw a sharp distinction between planted and nonplanted rows. For a partial model of order (a, b) which has no columns in common with the correct partial model of order (a, b), the expected value of $range(i, \theta)$ is $\frac{m}{(a+b+1)!}$ for both planted and nonplanted rows, and such a model exhibits no distinction between planted and nonplanted rows. Also, the expected valued of $\sum_i range(i, \theta)$ is significantly higher for the correct model than for an incorrect one unless k, the number of planted rows, is very small. This comparison can be extended to partial models that are incorrect but have some columns in common with the correct one. The results support the claim that it should be possible to distinguish the correct partial model from incorrect partial models of the same order.

5.3. Scoring partial models

In this section, we explain in detail the objective function employed to determine the quality of a partial model, θ. Let us denote by τ the underlying (hidden) complete model used to generate the data matrix D, and let p denote the (unknown) probability for rows to be planted with respect to τ. To score the partial model θ, we assume that τ is an extension of θ and estimate p. We use the estimated p as the quality measure of θ—the more rows seem to be planted with respect to θ, the more confident we are in θ.

Let $D_\theta(i)$ denote the vector of ranks in the i-th row within the columns specified by the partial model θ. Let

$$A_i = Prob[D_\theta(i)|X_i = 1],$$

and let

$$B_i = Prob[D_\theta(i)|X_i = 0].$$

That is, A_i denotes the probability of observing the ranks $D_\theta(i)$, given that i is a planted row, and B_i denotes the probability of observing $D_\theta(i)$, given that row i is not planted. We show in Section 5.4 how to compute A_i and B_i. We are interested in the probability that a given row is planted (by a complete model extending θ) after observing $D_\theta(i)$. This probability can be computed using Bayes' Theorem:

$$Prob[X_i = 1|D_\theta(i)] = \frac{A_i p}{A_i p + B_i(1 - p)}. \tag{1}$$

Consider a partial model θ. Assuming that the observed data was generated by some full model extending θ and by planting rows with probability p $(0 < p < 1)$, we have two ways of evaluating the expected number of planted rows:

1) $n \cdot p$,

2) $\sum_{i=1}^{n} Prob[D_\theta(i)|X_i = 1]$.

Recalling the formula for $Prob[X_i = 1|D_\theta(i)]$ (Equation [1]), we get the implicit equation

$$\sum_{i=1}^{n} \frac{A_i p}{A_i p + B_i(1-p)} = np. \tag{2}$$

This equation always has $p = 0$ as a solution, but we are interested in strictly positive solutions. Cancelling p from both sides, we obtain $\sum_{i=1}^{n} \frac{A_i}{A_i p + B_i(1-p)} = n$. The left-hand side of this equation is a sum of convex functions, so it also is convex. In Appendix I, we further show that the equation always has at most a single solution p, which we find numerically. Denote the resulting solution by p_θ.

Thus, given a partial model θ assumed to be compatible with the hidden complete model, the expected number of rows supporting the hidden complete model is np_θ. Accordingly, p_θ measures the quality of the partial model θ, and we can rank partial models of order (a, b) according to this measure.

5.4. *Computing* $Prob[X_i = 1|D_\theta(i)]$

Let θ be a partial model of rank(a, b), assumed to be correct, $\theta = \{\langle t_1, \ldots, t_a\rangle, \langle t_{s-b+1}, \ldots, t_s\rangle, s\}$, and let $D_\theta(i)$ denote the corresponding vector of ranks in the i-th row,

$$D_\theta(i) = (D[i, t_1], \ldots, D[i, t_a], D[i, t_{s-b+1}], \ldots, D[i, t_s]).$$

In this section, we show how to compute $Prob[X_i = 1|D_\theta(i)]$, the probability that the i-th row is planted given the rank vector. By Equation 1, this reduces to computing A_i and B_i.

In a planted row, per the stochastic model (Section 4), the rank vector $D_\theta(i)$ is in ascending order. Define the *gap* at row i with respect to θ as $g_i^\theta = D[i, t_{s-b+1}] - D[i, t_a] - 1$. That is, g_i^θ is the number of "unused" ranks lying between the a-leftmost planted column and the b-rightmost planted column. Following the analysis described in Appendix II, we obtain

$$A_i = Prob[D_\theta(i)|X_i = 1] = \frac{\binom{g_i^\theta}{s-(a+b)}}{\binom{m}{s}}. \tag{3}$$

In a nonplanted row, all $m!$ possible linear orderings of the m columns are equally likely. Thus, in particular, if we consider the ranks in $a + b$ specific columns, as indicated by θ, the probability of observing any particular sequence of $a + b$ ranks is equally likely as other sequences. Thus,

$$B_i = B = \frac{1}{\binom{m}{(a+b)}(a+b)!}.$$

Combining it all together, we get

$$Prob[X_i = 1|D_\theta(i)] = \frac{A_i p}{A_i p + B(1-p)}$$

$$= \frac{\frac{\binom{g_i^\theta}{s-(a+b)} p}{\binom{m}{s}}}{\frac{\binom{g_i^\theta}{s-(a+b)} p}{\binom{m}{s}} + \frac{1}{\binom{m}{(a+b)}(a+b)!}(1-p)}.$$

TABLE 2. PROBABILITIES OF IDENTIFYING THE PLANTED TISSUES (IN THE CORRECT ORDER)[a]

$p \setminus s$	3	4	5	7	10
0.025	0.0	0.01	0.21	0.72	0.92
0.05	0.17	0.7	0.94	1.0	1.0
0.075	0.69	0.98	1.0	1.0	1.0
0.1	0.92	1.0	1.0	1.0	1.0

[a]For all experiments $n = 1{,}000, m = 50$. Probabilities are based on 100 simulations per entry. Pool size (number of partial models maintained) equals $\ell = 100$.

6. RESULTS

We have implemented our algorithm (in Matlab). We then ran it on simulated as well as real data. In this section, we report the outcomes of these runs. For simulated data, all our datasets were 1,000-by-50 matrices. The number of planted columns s varied over the five values $s = 3, 4, 5, 7, 10$. The number of rows in G was determined by flipping a p biased coin per row, where p varied over the four values $p = 0.025, 0.05, 0.075, 0.1$. Table 2 reports the probabilities that the algorithm recovers correctly the set of planted columns and their correct internal order. (All these probabilities are over finite spaces, representing the proportion of certain cases out of the total number of cases.) Each entry of the table is based on one hundred random datasets. The running time of the algorithm (for one dataset) is 23 seconds (running in Matlab on a 500 MHz PC).

In cases where the algorithm fails to recover the planted submatrix, we compared the size (number of rows) of the recovered submatrix to the size of the planted submatrix. For certain values of the parameters, the algorithm recovered submatrices that are larger than the planted one. Clearly, those cases should not be considered as failures of the algorithm, but rather as indication that for the simulation parameters applied there is no hope of recovering the planted submatrix. We report in Table 3 the failure rate of the algorithm, i.e., cases in which the search returned a smaller (less significant) submatrix than the planted one. The success rate of the algorithm (proportion of cases where the algorithm produced a submatrix at least as large as the planted one) is obviously 1 minus the failure rate. It is typically very high and was below 0.5 for only one entry in the parameters' space. The maximal failure rate (0.57) occurred for $s = 5$ and $p = 0.025$. Our interpretation is that 25-by-5 OPSMs do not occur at random. However, the 25 (expected) planted rows do not give enough advantage to the correct partial model (1,1); it is *not* among the top $\ell = 100$ pairs, and therefore the submatrix will not be recovered. Increasing ℓ would clearly improve results but will come at a cost of a higher running time. We have kept ℓ to low values to allow extensive simulations. Running the same parameter with $\ell = 500$, we get a success rate of 0.7. For real data (that needs to be analyzed only once), we can afford much longer running time than the 23 seconds our algorithm currently utilizes on problems of this size, so it would be possible to set a much higher value for ℓ.

In addition to simulated datasets, we also ran our algorithm on a breast tumor dataset (Chen *et al.*, 2001). This dataset has $n = 3{,}226$ genes and $m = 22$ samples: 8 with brca1 mutations, 8 with brca2 mutations, and 6 sporadic breast tumors. (Of course, this information is not known to the algorithm). We found several statistically significant order-preserving submatrices in this dataset. One such OPSM had $s = 4$ tissues

TABLE 3. FAILURE PROBABILITIES: IDENTIFYING A SUBMATRIX LESS SIGNIFICANT THAN THE PLANTED ONE[a]

$p \setminus s$	3	4	5	7	10
0.025	0.0	0.06	0.57	0.28	0.08
0.05	0.0	0.18	0.06	0.0	0.0
0.075	0.0	0.02	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0

[a]All parameters identical to previous table.

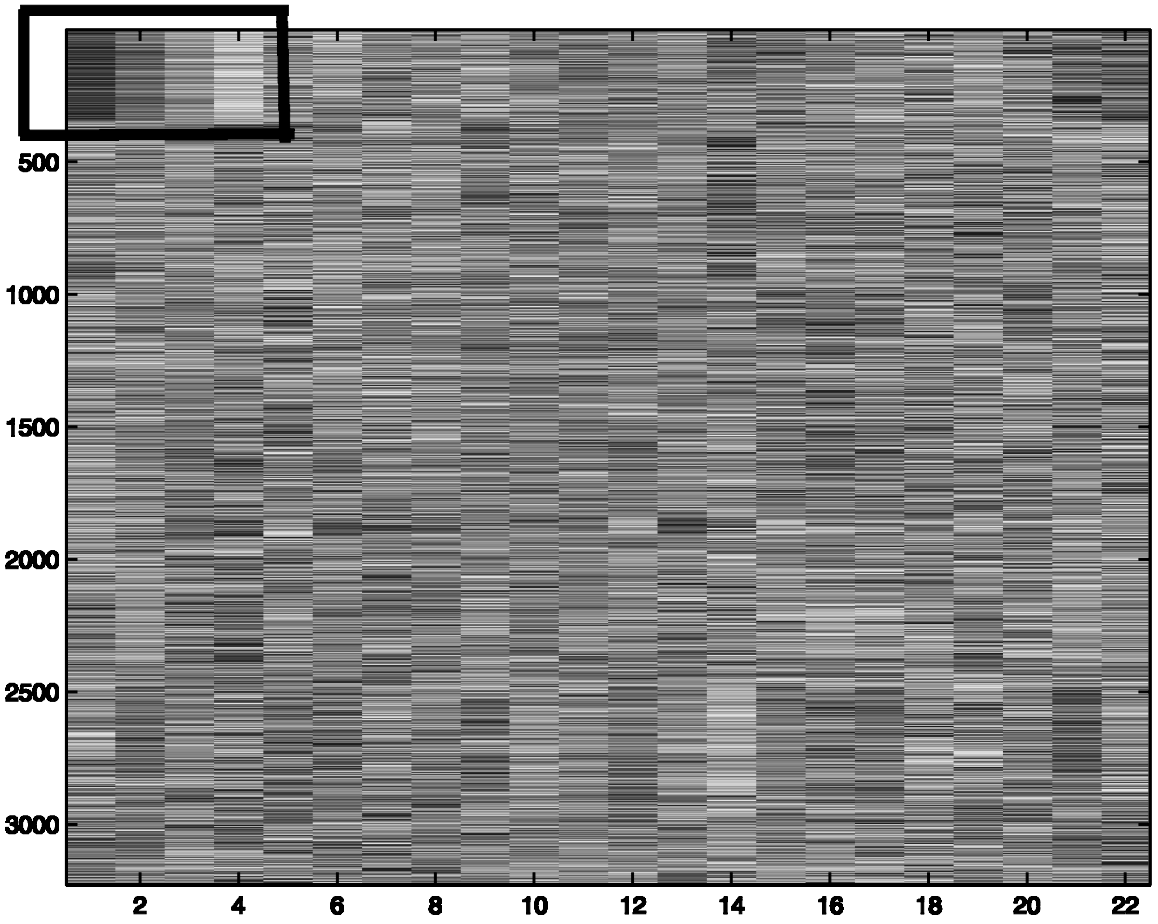

FIG. 1. An order-preserving submatrix identified in the breast cancer data, consisting of 347 genes and 4 tissues.

supported by $k = 347$ genes. This pattern is statistically significant since we would expect to see only $3{,}226/4! = 134$ genes that support such a pattern at random, and the overabundance of supporting genes suggests biological relevance. Interestingly, the first three tissues are all brca2 mutations, while the last one (largest expression levels) is sporadic. Figure 1 depicts this result. The region marked by thick lines indicates the 347-by-4 order-preserving submatrix. The upper bound on the significance of this OPSM is $U(4, 347) = 8.83 \cdot 10^{-51} < 10^{-50}$. We remark that it certainly is *not* the case that all the genes in one of these four tissues have higher expression levels than their counterparts in one of the other tissues. This can be verified from the figure, where the expression levels are coded using gray levels (darker shade corresponds to lower expression level). Other highly significant patterns are a 42-by-6 OPSM (five brca2 mutations, followed by one brca1 mutation). This OPSM has $U(6, 42) = 8.85 \cdot 10^{-19} < 10^{-18}$. Finally, the algorithm discovered a 7-by-8 OPSM. The upper bound on its significance level, $U(8, 7) = 0.0497$, is not that impressive. But the order imposed on the tissues seems interesting, as we have four brca2 mutants, followed by three brca1 mutants, followed by one sporadic cancer sample. Note that the upper bound $U(s, k)$ is not tight, and overcounting due to the simple union bound is worse as s increases.

7. CONCLUDING REMARKS AND FUTURE DIRECTIONS

Our goal in this work is to find submatrices $G \times T$ in which the rows have a significant tendency to be similarly ordered. It is perhaps too optimistic to expect submatrices to be identically ordered as the OPSM model requires, first because biological patterns are not always neat, and secondly because of error in the microarray measurements. One way to relax the requirement is to introduce an additional

corruption probability α which will influence the generation of the probabilistic model. As before, the model generates a hidden submatrix $G \times T$ with a specified ordering of its columns. But now each entry of the hidden submatrix is exempted from respecting the ordering with probability α and, in each row, only the entries that have not been exempted are permuted according to the prescribed ordering. Another interesting relaxation is to require that, in each row in the submatrix $G \times T$, the ordering of the ranks is compatible with a given partial ordering of the columns. Of particular interest are *layered partial orderings*, in which T is partitioned into subsets $T_1, T_2, \ldots, T_r$ and, for $i = 1, 2, \ldots, r-1$, every element of T_i precedes every element of T_{i+1}. Each of the sets T_i can be interpreted as a set of tissues representing a stage in a cellular process or in the progression of a disease.

One of the major design goals of our algorithm was speed, especially as we tested it by running numerous simulations. For real biological data, which takes more time to generate than simulated data, we would be willing to sacrifice some speed for higher accuracy and reliability. One way to achieve this is to start with partial models of order $(2, 1)$, instead of $(1, 1)$ as was done here. This increases (by a factor of m) the number of models that are initially examined, but reduces the chances that the correct partial model of order $(1, 1)$ will be mistakenly dismissed at the first stage of the algorithm.

Note that the reduction employed in our NP-completeness proof used tractability of OPSMs in close to square matrices. Thus, it is possible that the problem of finding hidden OPSMs in a different regime, in terms of the matrix shape, is not NP-hard. Proving otherwise requires a proof directed at such instances.

We have demonstrated that our algorithm for the OPSM problem works well on data generated according to the probabilistic model and have given an illustration of its ability to find a highly statistically significant order-preserving submatrix in a biological data set. Of course, the ultimate test of the OPSM approach will be its usefulness in uncovering hidden local structure in microarray data and finding functional, clinical or biochemical interpretations of that structure.

APPENDICES

Appendix I: Uniqueness of solution

We show that the equation

$$G_{\theta,D}(p) \stackrel{\triangle}{=} \sum_{i=1}^{n} \frac{A_i}{A_i p + B(1-p)} = n$$

has at most a single solution p. Each summand of $G_{\theta,D}(p)$ has the form $\frac{A_i}{A_i p + B(1-p)}$. Since A_i is non-negative and B is positive, $\frac{A_i}{A_i p + B(1-p)}$ is a convex cup function of p in the range $0 \le p \le 1$. Therefore, $G_{\theta,D}(p)$ is also a convex cup function of p. If $G_{\theta,D}(p)$ is either monotonically increasing or decreasing, then the uniqueness claim is obvious. Otherwise, there could be at most two solutions to the equation, and $G_{\theta,D}(p)$ attains a minimum at a point p_{min} satisfying $0 < p_{min} < 1$. As a convex cup function in the range $0 \le p \le 1$, the function $G_{\theta,D}(p)$ attains its maximum value(s) at either end of the interval, namely either at $p = 0$ or at $p = 1$. At $p = 1$, the term $\frac{A_i}{A_i p + B(1-p)}$ attains the value 1 for every summand with $A_i > 0$; thus, $G_{\theta,D}(1)$ equals the number of terms with $A_i > 0$. The total number of terms is n, and therefore $G_{\theta,D}(1) \le n$. Equality is attained only if no term A_i vanishes. Recalling the definition $(A_i = Prob[X_i = 1 | D_\theta(i)])$, it is extremely unlikely that all A_i will be nonzero. (Even for $a = b = 1$, the probability that $n - k$ random rows will all have $g \ge 1$ is $2^{-(n-k)} \ll 1$.) Therefore, $G_{\theta,D}(1) < n$ almost surely. This means that $G_{\theta,D}(p) = n$ can have at most one solution p_θ, satisfying $0 \le p_\theta < p_{min}$. Furthermore, the equation has a solution if and only if $G_{\theta,D}(0) = \sum_{i=1}^{n} \frac{A_i}{B} \ge n$.

Appendix II: Partially ordered random permutations

In this appendix, we investigate the stochastic properties of partially ordered permutations. Fix $s \le m$. Consider the probability measure μ induced on S_m by the following process:

1. Uniformly draw $\pi = (\pi(1), \pi(2) \ldots, \pi(m)) \in S_m$.
2. Sort the first s entries $(\pi(1), \pi(2) \ldots, \pi(s))$ of π.

For any permutation $\pi \in S_m$

$$\mu(\pi) = \begin{cases} \dfrac{1}{\binom{m}{s}(m-s)!} & \text{if } \pi(1) < \pi(2) < \cdots < \pi(s) \\ 0 & \text{otherwise.} \end{cases}$$

Let a, b be positive integers such that $a + b \leq s$. Consider a vector $\vec{v}$ containing $a + b$ distinct entries, all integers in the range $[1, m]$, $\vec{v} = (v(1), \ldots, v(a), v(a+1), \ldots, v(a+b))$. We say that $\vec{v}$ is *consistent* if $v(1) < \cdots < v(a) < v(a+1) < \cdots < v(a+b)$. Define

$$E_v = \{\pi \in S_m \mid \pi(1) = v(1), \ldots, \pi(a) = v(a), \pi(s-b+1) = v(a+1), \ldots, \pi(s) = v(a+b)\}.$$

We are interested in computing $\mu(E_v)$ for any $\vec{v}$. We start with a simple case : Assume that $\vec{v}$ has s entries ($a + b = s$). If $\vec{v}$ is not consistent, then E_v contains no permutation π with positive $\mu(\pi)$, so $\mu(E_v) = 0$. If $\vec{v}$ is consistent, then it specifies the first s entries of π, but nothing else. So E_v contains exactly $(m-s)!$ permutations with positive μ, and $\mu(E_v) = \frac{(m-s)!}{\binom{m}{s}(m-s)!} = \frac{1}{\binom{m}{s}}$.

In the general case ($a + b \leq s$), inconsistency also implies $\mu(E_v) = 0$. For consistent $\vec{v}$, denote $gap(\vec{v}) = v(a+1) - v(a) - 1$. Then $\vec{v}$ determines the first a entries of π and the entries $s-b+1, \ldots, s$ of π. In order for π to have a positive measure under μ, the $s - (a+b)$ entries $\pi(a+1), \ldots, \pi(s-b)$ may be any integers in the range $(v(a), v(a+1))$, provided these entries are in increasing order. The other $m - s$ entries of π are under no additional constraints. Thus, E_v contains $(m-s)!\binom{gap(v)}{s-(a+b)}$ permutations with positive measure μ, and therefore

$$\mu(E_v) = \begin{cases} \dfrac{\binom{gap(v)}{s-(a+b)}}{\binom{m}{s}} & \text{if } \vec{v} \text{ is consistent,} \\ 0 & \text{otherwise.} \end{cases}$$

REFERENCES

Alizadeh, A.A., Eisen, M.B., Davis, R.E., Ma, C., Lossos, I.S., Rosenwald, A., Boldrick, J.C., Sabet, H., Tran, T., Yu, X., Powell, J.I., Yang, L., Marti, G.E., Moore, T., Hudson, J., Lu, L., Lewis, D.B., Tibshirani, R., Sherlock, G., Chan, W.C., Greiner, T.C., Weisenburger, D.D., Armitage, J.O., Warnke, R., Staudt, L.M. 2000. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 403(6769), 503–511.

Bassett, D., Eisen, M., and Boguski, M. 1999. Gene expression it's all in your mine. *Nature Genet.* 21, 3–4.

Ben-Dor, A., Friedman, N., and Yakhini, Z. 2001. Class discovery in gene expression data. *RECOMB*.

Ben-Dor, A., Shamir, R., and Yakhini, Z. 1999. Clustering gene expression patterns. *J. Comp. Biol.* 6(3–4), 281–297.

Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, J., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D., and Sondak, V. 2000. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature* 406(6795), 536–540.

Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Raffeld, M., Yakhini, Z., Ben-Dor, A., Dougherty, E., Kononen, J., Bubendorf, L., Fehrle, W., Pittaluga, S., Gruvberger, S., Loman, N., Johannsson, O., Olsson, H., Wilfond, B., Sauter, G., Kallioniemi, O.-P., Borg, A., Trent, J., Hedenfalk, I., and Duggan, D. 2001. Gene-expression profiles in hereditary breast cancer. *NEJM* 344, 539–548.

Cheng, Y., and Church, G.M. 2000. Biclustering of expression data. *Proc. 8th Int. Conf. on Intelligent Systems for Molecular Biology.*

Eisen, M.B., Spellman, P.T., Brown, P.O., and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95(25), 14863–14868.

Gaasterland, T., and Bekiranov, S. 2000. Making the most of microarray data. *Nature Genet.* 24(3), 204–206.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 196, Freeman, San Francisco.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., and Lander, E.S. 1999. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286(5439), 531–537.

Lazzeroni, L., and Owen, A. 2000. Plaid models for gene expression data. *www-stat.stanford.edu/~owen/plaid/.*

Sharan, R., and Shamir, R. 2000. Click: A clustering algorithm with applications to gene expression analysis. *Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology*, 307–316. *www.math.tau.ac.il/~roded/click.html.*

Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast *Sacccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell* 9, 3273–3297.

Xing, E.P., and Karp, R.M. 2001. Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Proc. 9th Int. Conf. Intelligent Systems for Molecular Biology.*

Address correspondence to:
Amir Ben-Dor
Agilent Laboratories
12741 NE 39th Street
Bellevue, WA 98005

E-mail: amir_ben-dor@agilent.com