Biclustering in Gene Expression Data by Tendency

Jinze Liu[1], Jiong Yang[2], and Wei Wang[1]
[1]Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599
{liuj, weiwang }@cs.unc.edu
[2]Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801
jioyang@cs.uiuc.edu

Abstract

The advent of DNA microarray technologies has revolutionized the experimental study of gene expression. Clustering is the most popular approach of analyzing gene expression data and has indeed proven to be successful in many applications. Our work focuses on discovering a subset of genes which exhibit similar expression patterns along a subset of conditions in the gene expression matrix. Specifically, we are looking for the Order Preserving clusters (OP-Cluster), in each of which a subset of genes induce a similar linear ordering along a subset of conditions. The pioneering work of the OPSM model[3], which enforces the strict order shared by the genes in a cluster, is included in our model as a special case. Our model is more robust than OPSM because similarly expressed conditions are allowed to form order equivalent groups and no restriction is placed on the order within a group. Guided by our model, we design and implement a deterministic algorithm, namely OPC-Tree, to discover OP-Clusters. Experimental study on two real datasets demonstrates the effectiveness of the algorithm in the application of tissue classification and cell cycle identification. In addition, a large percentage of OP-Clusters exhibit significant enrichment of one or more function categories, which implies that OP-Clusters indeed carry significant biological relevance.

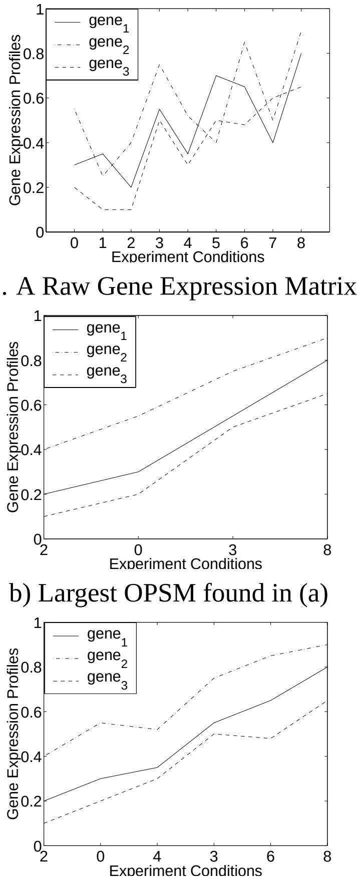
Keyword: Gene expression data, Microarray data, Biclustering, Order Preserving.

1 Introduction

Modern technology provides efficient methods for data collection. The advent of DNA microarray technologies has revolutionized the experimental study of gene expression. Thousands of genes are routinely probed in a parallel fashion. The expression levels of their transcribed mRNA are reported. By repeating such experiments under different conditions (e.g, different patients, different tissues, or varying cells' environments), data from tens to hundreds of experiments can be gathered. The analysis of these large datasets poses numerous algorithmic challenges.

Clustering is the most popular approach of analyzing gene expression data and has proven successful in many applications, such as discovering gene pathway, gene classification, and function prediction. There is a very large body of literature on clustering in general and on applying clustering techniques to gene expression data in particular. Several representative algorithmic techniques have been developed and experimented in clustering gene expression data, which include but are not limited to hierarchical clustering [7], self-organizing maps [11], and graphic theoretic approaches (e.g., CLICK [16]).

The traditional clustering algorithm, however, is incapable of discovering the gene expression pattern visible in only a subset of experimental conditions. In fact, it is common that a subset of genes are co-regulated and co-expressed under a subset of conditions, but behave independently under other conditions. Recently, *bicluster* has been studied to discover the local structure inside the gene expression matrix. Cheng and Church [6] are among the pioneers in introducing this concept. Their biclusters are based on uniformity criteria, and a greedy algorithm is developed to discover them. Plaid [13] is another model to capture the approximate uniformity in a submatrix in gene expression data and look for patterns where genes differ in their expression levels by a constant factor. Ben-Dor *et al.* [2] discussed approaches to identify patterns in expression data that distinguish two subclasses of tissues on the basis of a supporting set of genes that results in high classification accuracy. Segal *et al.* [15] described rich probabilistic models for relations between expressions, regulatory motifs and gene annotations. Its outcome can be interpreted as a collection of disjoint biclusters generated in a supervised manner. Tanay *et al.* [18] defined a bicluster as a subset of genes that jointly respond across a subset of conditions, where a gene is termed responding under some condition if its expression level changes significantly under that condition with respect to its normal level.



a). A Raw Gene Expression Matrix

b) Largest OPSM found in (a)

c). Largest OP-Cluster found in (a)

Figure 1. Comparison of clusters discovered by OPSM and OP-Clustering within noisy expression data

Ben-Dor *et al.* introduced the model of OPSM (order preserving submatrix) [3] to discover a subset of genes identically ordered among a subset of conditions. It focuses on the coherence of the relative order of the conditions rather than the coherence of actual expression levels. For example, in the gene expression data of patients with the same disease, the genes interfering with the progression of this disease shall behave similarly in terms of relative expression levels on this set of patients. These types of pattern can be observed in data from nominally identical exposure to environmental effects, data from drug treatment, and data representing some temporal progression, etc. The OPSM problem was proven to be NP-hard in [3]. A stochastic model was developed to discover the best row supported submatrix given a fixed size of conditions. However, one major drawback of the pioneering work is the strict order of the conditions enforced by the OPSM model. In this paper, we propose a more general model, namely OP-Cluster (Order Preserving Cluster). It includes OPSM model as a special case but allows controlled flexibility in the ordering of the conditions in a cluster.

Let M be an $n \times m$ data matrix obtained from the readout of DNA chips, where n is the number of rows (genes) and m is the number of columns (conditions). Let d_{ij} be the entry of the matrix, where $0 < i \leq n$ and $0 < j \leq m$. We view n rows of the data matrix as n sequences of the column labels. Our model allows a subset of adjacent column labels in a sequence to be grouped as an order equivalent group if their values are similar. Within the group, no strict order of the columns is imposed. The sequence of each row corresponds to a permutation of all column labels, under which the order equivalent groups are arranged in a non-decreasing order. This flexibility can be very important in biological progression. For example, when more than one condition may correspond to the same stage of a disease progression, the order of those similarly expressed conditions is not important. On the contrary, applying the strict order to those conditions may even worsen the problem by introducing inconsistent permutations of conditions.

Given a gene expression matrix M, we are seeking the biological progression that can be represented by a common subsequence S of length s shared by k genes. The k genes from M are coexpressed in the s conditions. We propose a deterministic biclustering model, namely OP-Cluster, to capture the set of general tendencies exhibited by a subset of genes along a subset of conditions in M. Compared with the OPSM model, our model can tolerate a high degree of noise. For example, given a 3×9 gene expression matrix shown in Figure 1 a), the largest OPSM and the largest OP-Cluster that can be discovered are presented in Figure 1 b) and c) respectively. While both of the OPSM and OP-Cluster exhibit strong tendency within the cluster, the OP-Cluster is much more significant in size by having 6 common conditions.

Besides the robustness of the OP-Cluster model, we also design a novel deterministic algorithm to automatically reveal the complete set of the OP-Clusters in one run. A compact OPC-Tree is built to represent all tendencies exhibited in the expression data. By recursively mining and developing the OPC-Tree, we are able to generate the entire set of OP-Clusters in an efficient way. Our algorithm is superior to the OPSM algorithm in both quality and efficiency. The major portion of the OPSM algorithm is the candidate model generation. While removing more partial models or candidate models can improve the efficiency of the algorithm, it is at the risk of missing many significant clusters. Although both algorithms are exponential in nature, our algorithm achieves an outstanding performance by discovering all the valid OP-clusters in parallel with a compact data structure, while the efficiency of OPSM algorithm is at the expense of the quality of the OPSMs.

Our experiments on two large real datasets demonstrate that OP-Clusters can help with the identification of the discriminating genes in tissue classification and the identification of cell cycles in the time-series data. In addition, the result of functional enrichment of OP-Clusters again highlights the fact that OP-Clusters carry significant biological meaning.

The remainder of the paper is organized as follows. Section 2 defines the model proposed in the paper. Section 3 presents the algorithm in detail. An extensive performance study is reported in Section 4. Section 5 concludes the paper and discusses some future work.

2 OP-Cluster Model

In this section, we define the OP-Cluster model for mining genes that exhibit tendencies on a set of conditions.

2.1 Definitions and Problem Statement

Let $\mathcal{D}$ be a set of genes, where each gene is associated with a set of experiment conditions $\mathcal{A}$. We are interested in the subsets of genes that exhibit a coherent tendency on a subset of conditions of $\mathcal{A}$. The tendency between a pair of conditions is defined in terms of the relative order of the corresponding values and can be of the following three types: equivalent, higher and lower. We first introduce the concept of order equivalent group of conditions.

Definition 2.1 *Assume that $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is the condition set. Let o be a gene, $o \in \mathcal{D}$. Let $E : \mathcal{D} \times \mathcal{A} \rightarrow \Re$ be a function that returns the expression level of a gene under a given condition. Given a grouping threshold δ, $\delta \geq 0$, we say that the conditions in $\mathcal{A}'$, $\mathcal{A}' \subseteq \mathcal{A}$ form an* **order equivalent group** *for gene o, if*

$$\max_{a_i, a_{i'} \in \mathcal{A}'} |E(o, a_i) - E(o, a_{i'})| < \delta \times \min_{a_j \in \mathcal{A}'} E(o, a_j) \quad (1)$$

and

$$\forall a_i, a_i \in \mathcal{A}', \min_{a_{i'} \in \mathcal{A}'} |E(o, a_i) - E(o, a_{i'})| < \min\nolimits_{a_{i''} \in (\mathcal{A} - \mathcal{A}')} |E(o, a_i) - E(o, a_{i''})|, \quad (2)$$

The intuition behind the first criterion (Inequality 1) is that, if the difference between the values under two conditions is insignificant, we consider the two conditions to be 'equivalent' and no order is placed on the conditions. For example, in gene expression data, several conditions with similar expression levels might belong to the same group corresponding to a stage or time point in the progression of a disease or a type of genetic abnormality. In this case, no strict order should be enforced within the group.

The right part of Inequality 1 is the maximum difference allowed within a group. Although there are multiple ways to define it based on the characteristic of the data distribution [14], the maximum difference allowed within a group is defined as a percentage of the minimum value of the group in this paper since the expression levels usually follow a skewed distribution.

The second criterion (Inequality 2) guarantees that, for each condition, the minimum difference between its expression level and the rest of its order equivalent group is always smaller than the difference with the expression levels outside its group. In other words, a condition is always grouped with its closest neighbor if the difference between them is within the maximum difference allowed. For example, assume that a gene from four conditions $\{a, b, c, d\}$ has expression level $\{100, 110, 145, 155\}$. Given $\delta = 50\%$, both b and c are in the maximum allowed difference of a. However, the closest neighbor of c is d, not b. Therefore, instead of grouping the three conditions (abc), we group (ab) and (cd) separately.

In the rest of paper, we use $\approx$ to denote the relationships of being in the same equivalent group. For example, if a_1 and a_2 are in the same equivalent group, we say, $a_1 \approx a_2$.

Two genes are coherent under two conditions if their expression values under the two conditions have the same relationships, i.e, $>$, $<$, or $\approx$. The formal definition of coherent tendency is presented in Definition 2.2.

Definition 2.2 *Given two genes $o_i, o_{i'} \in \mathcal{D}$ and two conditions $a_j, a_{j'} \in \mathcal{A}$, o_i and $o_{i'}$ have* **coherent tendency** *on a_j and $a_{j'}$ if one of the following is true:*

$$\begin{array}{l}(1)\forall o, o \in \{o_i, o_{i'}\}, E(o_i, a_j) > E(o_i, a_{j'});\\ (2)\forall o, o \in \{o_i, o_{i'}\}, E(o_i, a_j) < E(o_i, a_{j'});\\ (3)\forall o, o \in \{o_i, o_{i'}\}, a_j \approx a_{j'};\end{array} \qquad (3)$$

Definition 2.3 *Let $\mathcal{O}$ be a subset of genes in the database, $\mathcal{O} \subseteq \mathcal{D}$. Let $\mathcal{T}$ be a subset of conditions in $\mathcal{A}$. ($\mathcal{O}$, $\mathcal{T}$) forms an* **OP-Cluster(Order Preserving Cluster)** *if every pair of genes in $\mathcal{O}$ have coherent tendency for every pair of conditions in $\mathcal{T}$.*

A less strict model of OP-Cluster may allow that each pair of conditions within a cluster has either the combination of the relationships (1) and (3) or the combination of (2) and (3) defined in Definition 2.2.

Suppose that we have two genes o_1 and o_2 measured under four experiment conditions $\{a, b, c, d\}$. The expression levels are $\{401, 281, 120, 298\}$ and $\{280, 318, 37, 215\}$, respectively. With $\delta = 0.1$, (bd) is an order equivalent group for o_1 and no order equivalent group exists for o_2. According to Definition 2.3, with o_1 and o_2 form an OP-Cluster on the condition set $\{a, c, d\}$. Essentially, each OP-Cluster captures the consistent tendency exhibited by a subset of genes in a subset of conditions. In the following sections, since the input data is a matrix, we refer to genes as rows and conditions as columns.

Lemma 2.1 *Given a matrix of size $n \times m$, the probability of finding a submatrix of size $n_c \times n_r$ is*

$$p(n_c, n_r) = \frac{m!}{n_c!} \sum_{i=n_r}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) (\frac{1}{n_c!})^i (1 - \frac{1}{n_c!})^{n-i} \qquad (4)$$

The above probability originally discussed in [3] measures the significance of a submatrix with size $(n_c \times n_r)$. Hence, given the size of the OP-cluster, we will be able to determine the significance of the cluster. This can be used during the postprocessing in order to select the most significant clusters.

Problem Statement Let $\mathcal{D}$ be a database with gene set $\mathcal{O}$ and condition set $\mathcal{A}$. Given a group threshold δ, the minimum number of rows nc, the minimum number of columns allowed within a cluster, the problem is to find all maximal OP-Clusters $(\mathcal{O}, \mathcal{T})$ according to Definition 2.3.

Mathematically, the problem of finding maximum OP-Clusters can be transformed to the following problem: after representing each row of the matrix by an ordered sequence of columns and order equivalent groups, identify the longest common subsequences (if the length is longer than n_c) for any subset of at least n_r rows.

3 OPC-Tree Algorithm

In this section, we present the algorithm to generate OP-Clusters. The algorithm consists of two steps: (1) preprocess each row of the data matrix into a sequence of groups by Definition 2.1; (2) mine the subsets of rows containing frequent subsequences. We design a novel compact structure OPC-Tree to organize the sequences and to guide the pattern generation in the second step. The OPC-Tree algorithm gains its advantage by sharing the same prefixes with a subset of genes avoiding repeated work in the future.

3.1 Preprocessing

To preprocess the data, each row in the database will be converted into an ordered sequence of columns. Given a row o, the ordered sequence will be generated by the following approach. First, sort all columns (conditions) in non-descending order of their values: Then, iterative scans through this sequence will be undertaken to identify the order equivalent groups. During the first scan, the pairs of conditions whose expression levels are closest and within the maximum allowed difference are identified and grouped together. For example, for gene o in Figure 2, given $\delta = 100\%$, after the first scan of non-descending ordered values, four pairs are grouped together. The next scan through the sequence merges the groups which can result in minimum intervals within the groups, which is less than the maximum allowed distance. The interval within the group is the distance between minimum and maximum value. For the four groups in the second line of Figure 2, the first group cannot merge with the second because of the violation of maximum allowed difference inside a group. The merge of the group (1.5, 1.7) and the group (2.3, 2.5) can meet the requirement of the maximum distance allowed. However, the interval [1.5, 2.5] with width 1 is wider than the interval of merging the group (2.3, 2.5) and group (2.9,3.1), which is 0.8. The same procedure is applied to the grouped sequence iteratively until no groups can be combined together. The example in Figure 2 stops after the third scan because of no merging can meet the maximum difference allowed. The iterative approach can guarantee that the closest neighbors within the maximum allowed difference always stay in the same group.

o: [0.5 0.6, 0.8, 1.5, 1.7, 2.3, 2.5, 2.9, 3.1]

1st scan o: [(0.5 0.6, 0.8), (1.5, 1.7), (2.3, 2.5), (2.9, 3.1)]

2nd scan o: [(0.5 0.6, 0.8), (1.5, 1.7), (2.3, 2.5, 2.9, 3.1)]

Figure 2. An example for identifying equivalent groups, δ=100%.

3.2 OPC-Tree

In the above subsection, each row in the matrix has been converted into a sequence of column labels. The goal in the next step is to discover all frequent subsequences in the generated sequences. Our algorithm uses a compact tree structure to store the crucial information used in mining OP-Clusters. The discovery of frequent subsequences and the association of rows with frequent subsequences are performed simultaneously. Sequences sharing the same prefix are gathered and recorded in the same branch. Hence, further operations along the shared prefix will be performed only once for all rows sharing it. Pruning techniques can also be applied easily in the OPC-Tree structure. *To make the algorithm more scalable with respect to the number of columns, the original OPC-Tree algorithm can be improved by collapsing all nodes along a single path [14].*

Before we define the OPC-Tree algorithm formally, we first give the following example.

Example 3.1 *Given the sequences in the last column of Figure 3 (O), given n_c=n_r=3, the OPC-Tree algorithm makes a pre-order depth-first traversal of the tree and works in the following steps.*

Step 1: *Create the root -1(NULL) and insert all sequences into the tree.* This is shown in Figure 3 (A). Notice that all rows sharing the same prefix fall on the same branch of the tree. The gene IDs are stored in the leaves. This is the *initial* OPC-Tree on which a recursive procedure (Algorithm *growTree*) depicted in Steps 2-5 is performed to fully develop the tree.

Step 2: *For each child of the root, insert all suffixes in its subtree to the root's child that has a matching label.* In Figure 3 (B), c is a child of the root -1. In this subtree, the suffix subtree starting at d (for Sequences $3, 4$) is inserted into the child d of root -1. If the same subtree exists in the destination node, the gene IDs associated with the suffixes are combined with existing IDs in the destination node. Otherwise, a new subtree will be created in the destination node. In the case where a suffix is too short to satisfy the inequality *current depth* + *length of the suffix* $> n_c$, the suffix will not be inserted. For example, ba in sequence 3 is also a suffix, it is not to be inserted because *depth 0* + *length of* $ba < n_c$.

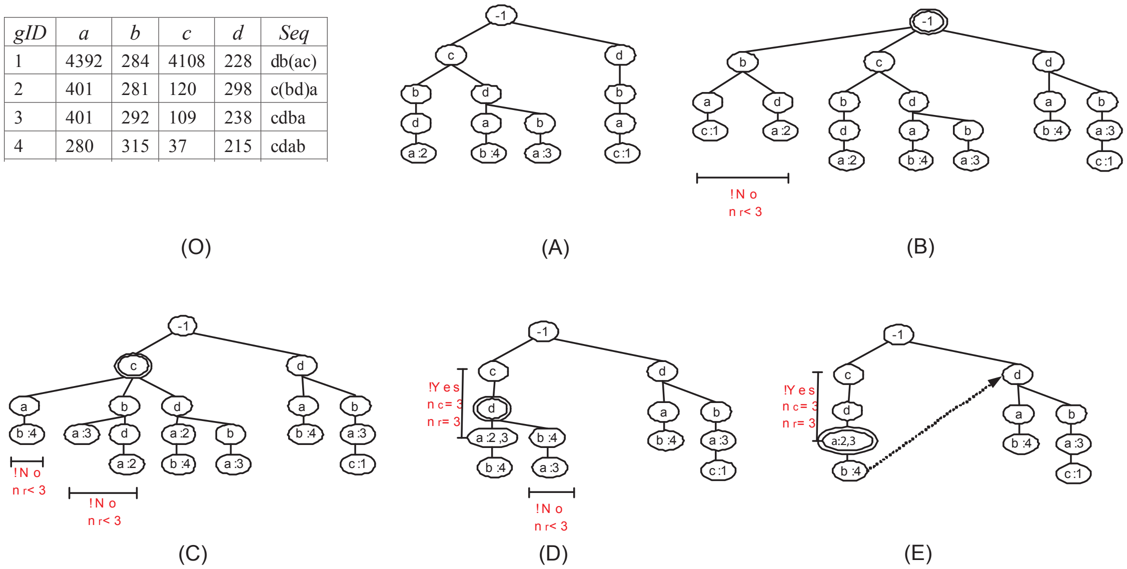
Step 3: *Prune current root's children.* If the number of rows that fall in a subtree is smaller than n_r, the subtree will be deleted because no further development can generate a cluster with more than n_r rows. For example, the subtree leading from $-1b$ in Figure 3 (B) is deleted in Figure 3 (C) since there are only two sequences falling in this subtree.

Step 4: *Repeat Step 2-Step 5 on the root's first child and its subtree recursively*. For example, c is the first child of root -1. Therefore, the same procedure in Step 2 is applied to c first. The suffixes of c's subtree d, such as ba and ab are inserted into c's subtrees b and a respectively. Since there are less than three sequences falling on c's subtrees a and b, the branches $-1ca-$ and $-1cb-$ are deleted. Following the same procedure, we develop c's only subtree $-1cd-$, which is shown in Figure 3(D).

Step 5 : *Follow the sibling link from the first child and repeat Step 2-Step 5 on each sibling node recursively.* For example, after developing the subtree of -1c-, the next subtree to be developed is its sibling $-1d-$.

Definition 3.1 OPC-tree (Order Preserving Clustering tree). *An OPC-Tree is a tree structure defined below.*
1. It consists of a root labeled as "-1" and a set of subtrees as the children of the root;
2. Each node in the subtrees has four entries: entry value, a link to its first children node, a link to its next sibling node, and the list of sequences, each of which has a suffix corresponding to the path from the root to this node. In other words, the gene IDs are only recorded at the node that marks the end of a common subsequence.

Analysis of OPC-Tree construction Only one scan of the entire data matrix is needed during the construction of the OPC-Tree. Each row is converted into a sequence of column labels. The sequences are then



gID	a	b	c	d	Seq
1	4392	284	4108	228	db(ac)
2	401	281	120	298	c(bd)a
3	401	292	109	238	cdba
4	280	315	37	215	cdab

Figure 3. An example of construction of OPC-Tree. O) holds the original raw data set. The last column in O) presents the converted sequence of columns of each row. The trees in A),B),C),D),E) show the procedure of OPC-Tree step by step. The label in the oval node represents the column label. The numbers following ':' are the gene IDs. The node with double oval is the node currently being visited(active) in the depth first traversal. '!No' means that the subtree must be pruned. '!Yes' means that the subtree corresponds an OP-Cluster. A) Initiate the tree with all rows. B) The active node is -1. Insert the suffix of -1's subtrees to node -1. C) The active node is -1c-. Insert and Prune the subtree ($n_r < 3$). D) The active node is -1cd-. Identify the first OP-Cluster. E) Finish growing the -1's first subtree-1c. The next subtree is -1d.

inserted into the OPC-Tree. In the initial OPC-Tree structure, sequences that have the same prefix naturally fall onto the same path from the root to the node corresponding to the end of prefix. To save space, the gene IDs associated with each path are only recorded at the node marking the end of the longest common prefix shared by these sequences. To find OP-Clusters using the OPC-Tree, the common subsequences are developed by adding suffixes of each sub-tree as the tree's children, via a pre-order traversal of the OPC-Tree.

Space Complexity: Each node in the tree corresponds to a subsequence leading from root to this node. Therefore, the total number of nodes is equal to the total number of subsequences in the tree. Given an $n \times m$ matrix, the total number of sequences of length m is limited by the minimum of n and $m!$. Each sequence of length m may have $\binom{m}{1}$ subsequences of length $(m-1)$. For all sequences of length m, if all subsequences of length $(m-1)$ are different, the number of subsequences of length $(m-1)$ will be $\binom{m}{1} \times \min(n, (m-1)!)$. Therefore, the upper bound of the maximal number of subsequences with minimum length n_c will be $\sum_{s=n_c}^{m} \binom{m}{s} \min(n, s!)$ which is also the worst case space complexity. However, since we use the depth-first traversal of the tree and the part of tree that has been traversed will not be needed for future mining, they can be deleted and reused. At depth d $(d \neq 0)$, we only need to keep $(m - d + 1)$ nodes all from one of the nodes at depth $(d-1)$. Therefore, the maximal space to be allocated during the running tree will be limited to $O(n \sum_{i=1}^{m} (m-i+1)) = O(n \times m^2)$.

Time Complexity: In the worst case, the algorithm has to visit every possible node. The time complexity for the insertion operation is $O(n \times m^2)$. According to the total number of nodes in the tree analyzed above, the time complexity is bounded by $nm^2 \sum_{s=n_c}^{m} \binom{m}{s} \min(n, s!)$.

```
Algorithm growTree(T, $\sigma$,depth)
Input: $T$: the root of the initial tree, $\sigma$
Output: OP-Cluster existed in $T$
(* Grow patterns on the initial OPC-Tree $T$ *)
    if $T$ = nil
            return;
    $T_{child} \leftarrow T$'s first child;
    for any sub-tree $subT$ of $T$
        do insertSubTree($subT$, $T$);
    pruneTreeNode($T$);
```

```
        growTree($T_{child}$, $\sigma$, $depth + 1$);
        growTree($T$'s next sibling, $\sigma$, $depth$);
        **return**.
```

We can observe that, in the worst case scenario, both the time and space complexities of the algorithm increase exponentially with the number of columns. However, according to the algorithm, we should also note that the complexity largely depends on the number of distinct nodes in the tree. The more the number of sequences sharing a branch of the tree, the smaller the tree and the less the time and space complexity. Therefore, our algorithm will perform better in cluster-rich real dataset than in any random dataset.

Lemma 3.1 *Given a matrix M, a grouping threshold δ, the initialized prefix tree contains all the information of matrix M.*

Rationale: Based on the first step of the algorithm, each row in the matrix is translated into a sequence which is then mapped onto one path in the OPC-Tree. The row IDs and the order of the columns are completely stored in the initial tree.

3.2.1 Mining OP-Cluster Using OPC-Tree

Lemma 3.2 *Given a prefix tree representing a set of ordered sequences, the OPC-Tree discovers all frequent subsequences in D.*

Rationale: Given any sequence $S = x_1x_2x_3x_4 \ldots x_n$, we want to show that every subsequence of S can be found in a path starting from the root. Through the initiation of OPC-Tree, we know that S is present in the initial OPC-Tree. Then given any subsequence $SS = x_ix_j \ldots x_s, (1 \leq i, s \leq n)$, we can obtain SS by the following steps. First, at node x_i, insert suffix $x_ix_{i+1} \ldots x_n$. Now in the subtree of x_i, node x_j can be found by traversing the path $x_ix_{i+1} \ldots x_n$ inserted in the first step. Similarly, we insert the suffix $x_j \ldots x_n$. As a result, we get the path $x_ix_jx_{j+1} \ldots x_n$. By repeating the same procedure until we insert the suffix starting with x_s, we get the path $x_ix_j \ldots x_s$. Because all suffixes are inserted in the OPC-Tree, the OPC-Tree contains all subsequences presented in the original OPC-Tree.

Each frequent subsequence corresponds to the path from the root to a node in the OPC-Tree, which represents a particular order of a subset of columns preserved by a subset of rows. We can conclude that the OPC-Tree contains all clusters. This leads to the following lemma.

Lemma 3.3 *The OPC-Tree contains all OP-Clusters, each of which corresponds to a node in the OPC-Tree.*

3.2.2 Pruning OPC-Tree

Without any pruning, the whole OPC-Tree fits well into memory when we have a matrix of small to medium size (15 columns by 3000 rows). However, for large matrices, some pruning strategies have to be employed to minimize the size of the OPC-Tree. The pruning techniques used in our implementation mainly utilize the two parameters n_c and n_r. The suffixes to be inserted that are shorter than n_c will not be considered; the visited node in the OPC-Tree with the row support below n_r will be eliminated together with all of its subtrees. These pruning techniques will not harm the correctness of the result.

3.3 Merging the overlapping clusters

So far, the clusters discovered by our algorithms are strictly defined in Definition 2.3, which means that any row in the cluster has the same subsequence as others. However, because of the noisy nature of the microarray data, it would be rational to loosen the restriction and allow some degree of disorder between the columns in the subsequences representing a cluster. In addition, since a large number of subsequences can overlap heavily with each other with minor differences, our algorithm may create a large number of clusters with a large percentage of rows appearing in more than one clusters. To tackle these problems, we apply a hierarchical approach to merge the clusters with high correlations in the set of columns appearing in at least one of these clusters.

Given a subset of columns $S = \{a_1, a_2, ..., a_d\}$, the ordered subsequence of the columns for each row can be equally represented by the rank list of columns, which is a permutation of $\{1...d\}$. Assume that we have a cluster of k genes over S. Each gene is associated with a rank list $l_i, (0 < i \leq k)$, and let $R_j (0 < j \leq n)$ denote the sum of the actual a_j's value of all rows. The Kendall coefficient of concordance of the k genes are defined as $\frac{12}{d^2(k^3-k)}(\sum_{i=1}^{k} R_i^2 - \frac{kd^2(k+1)^2}{4})$ It measures the communality of k genes under the d conditions [12].

Given two clusters with two different condition sets S_1 and S_2, a third union cluster S_3 of the two clusters is first created. Rank information of S_3 is then computed or retrieved and the rank correlation is computed according to the above formula.

The merge of all discovered clusters is done in a level-wise hierarchical manner. Given a constant factor θ, starting from the original clusters in the lowest level, the pair of clusters with the highest rank correlation is merged if their rank correlation is higher than $1 - \theta$. With the rank correlation threshold $1 - \theta$, two clusters are merged if their rank correlation is the highest and is higher than the correlation threshold. This step repeats until no cluster can be merged. Then the current correlation threshold is loosened by θ, the next level will be built upon the current level with the same procedure. The hierarchy construction is complete when the proper correlation value or cluster size is reached.

4 Results on Gene Expression Data

In this section, we use two real datasets to evaluate our algorithm. The two datasets are the breast tumor data from Cheng *et al.* [5] and the yeast cell cycle data from Spellman *et al.* [17]. Our experiments demonstrate the power of the OP-Cluster model in clustering biologically related genes. We compared our clustering result with the OPSM model using breast cancer data. For yeast cell cycle data, the generated clusters are evaluated against gene annotations [22] and the P-value measuring the significance of function enrichment in a cluster is calculated to assess the effectiveness of the model. The OP-Clustering algorithm was implemented in C and the experiments were executed on a Linux machine with a 700 MHz CPU and 2G main memory. With various parameter settings, the running time of the algorithm usually takes less than 5 minutes on both datasets.

4.1 Breast Tumor Dataset

We ran our first set of experiments on the breast tumor dataset. This dataset has 3226 genes and 22 tissues. Among the 22 tissues, there are 7 brca1 mutations, 8 brca2 mutations and 7 sporadic breast tumors. We compare our experiment result with the OPSM algorithms.

First, we report the significance of our clusters in Table 1. For this set of experiments, we set $\delta = 0$. Therefore, the clusters will be identical to OPSM. In practice, our clustering algorithm generates much more significant clusters than the clusters reported by the OPSM clustering algorithm. For example, one significant cluster found by OPSM has 4 tissues and is supported by 347 genes. However, our OP-Clustering algorithm was able to find a cluster with 4 tissues supported by 690 genes, which doubles the number of genes.

Number of Tissues	Max Support(OPC)	Max Support(OPSM)
4	690	304
6	126	42
7	47	8
8	32	N/A
9	9	N/A
10	6	N/A

Table 1. Comparison with Original OPSM algorithms, n_c is the number of tissues, n_r is the minimum number of genes in a cluster. N/A: no result available.

Secondly, we present several representative patterns discovered by the OP-Clustering algorithm. The goal of the heredity cancer expression analysis is to identify the set of genes whose variations in expression best differentiates among different types of breast tumors(BRCA1, BRCA2, Sporadic). Our algorithm is able to find significant clusters which exhibit patterns discriminating the three types of tissues in a more consistent way. The original algorithm in paper [5] discovered 51 genes discriminating the three types of tissues (BRCA1, BRCA2, Sporadic) and 176 genes discriminating the two types of tumors (BRCA1, BRCA2). Our OP-Clustering algorithm not only is able to recover most of the previously identified genes but may suggest more genes that may serve the same purpose as well. Two examples of OP-Clusters with completely opposite trends along the tree types of tissues (BRCA1, BRCA2, Sporadic) are presented in Figure 4. The expression levels of Sporadic tissues are always in the middle of the other two types of tissues that may have completely different orders themselves. This observation is consistent with the pattern presented in [5]. Figure 5 presents two example clusters with completely opposite trends. The genes in Figure 5(A) have lower expression profiles in BRCA1 tissues than BRCA2 tissues while the genes in (B) supports the opposite trend. The discriminating genes identified in [5] were retrieved and compared with the genes in each OP-cluster. The comparison shows that each OP-Cluster can identify part of these identified genes. We do not expect OP-Cluster to discover most of them because the model of OP-Cluster is stricter because it not only applies order constraints among different types of tissues but requires a consistent order among the tissues of the same type as well.

However, if we collect the genes in all clusters that contain discriminating genes, we are able to recover most of the 176 genes. For example, if we set $n_c = 7, n_r = 20$, and δ = 10, we are able to detect 88% of 176 genes presented in [5]. There are 8 out of 20 genes in Figure 5 (A) and 7 out of 22 genes in Figure 5 (B), which are from the 176 genes. This means that our algorithm is capable of detecting relevant genes for discriminating different types of tissues.

Besides, OP-Clustering also reveals additional genes that may be used for the same discriminating purpose. Figure 6(A) presents a cluster with a consistent trend in 3 BRCA2 tissues followed by 3 BRCA1 tissues. Figure 6(B) includes the same set of genes in (A) side by side with the expression profiles along all the tissues in BRCA2 and BRCA1. According to (B), clear distinction between these two types of tissues exists in the gene expression profiles including those not identified previously. The OP-clusters do not require all tissues of the same type, but rather represent consistent trends demonstrated by majority of the tissues of different types.

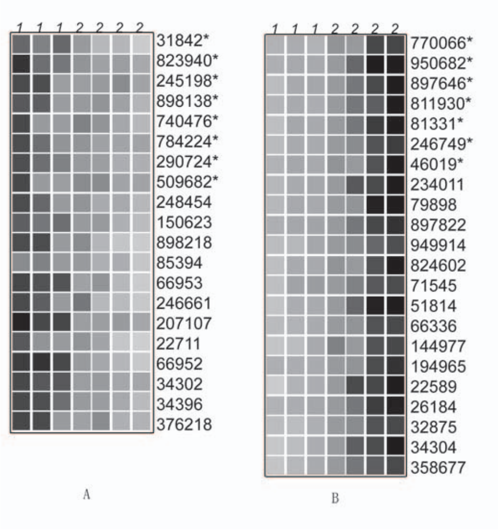


Figure 5. Two examples of OP-Clusters with opposite trends across two types of tissues{BRCA1, BRCA2}, * marks genes that are among the previously identified 176 genes

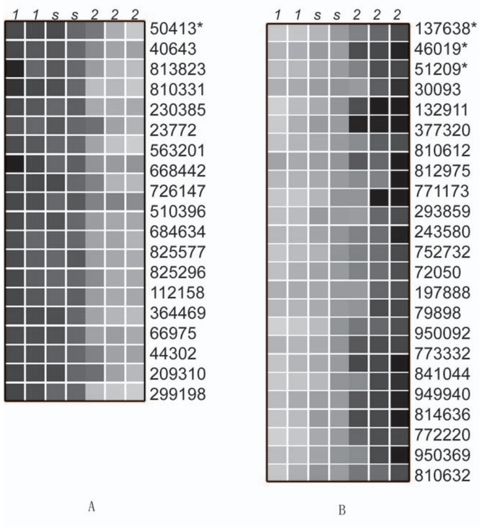


Figure 4. Two examples of OP-Clusters with opposite trends across three types of tissues{BRCA1, Sporadic, BRCA2}. * marks genes that are among the previously identified 51 genes

4.2 Gene Annotation and P-value

The hypergeometric distribution is used to model the probability of observing at least k ORFs from a cluster of size n by chance in a category containing f ORFs from a total genome size of g ORFs. The P-value is given by $P = 1 - \sum_{i=0}^{k} \frac{\binom{f}{i}\binom{g-f}{n-i}}{\binom{g}{n}}$. The test measures whether a cluster is enriched with genes from a particular category to a greater extent than that would be expected by chance. For example, if the majority of genes in a cluster appear from one category, then it is unlikely that this happens by chance and the category's P-value would be close to 0. Adopting the Bonferroni correction for multiple independent hypotheses, $\frac{0.01}{Na}$ is used as the defaut threshold to measure the significance of the P-value in our experiments. We expect a large fraction of the clusters to conform to the known classification.

4.3 Yeast Cell Cycle Datasets

The OPC-Tree algorithm was also tested on the yeast cell cycle data of Spellman *et al.*(1998). The study monitored the expression levels of 6,218 S. cerevisiae putative gene transcripts (ORFs) measured at 10-minute intervals over two cell cycles (160 minutes) with 18 time points. Spellman *et al.* identified 799 genes that are cell cycle regulated. We used the expression levels of the 799 genes across 18 time points as the original input matrix. The OP-Cluster procedure groups together genes on the basis of their common expression tendency across a subset of time points.

By taking $n_c = 7$, $n_r = 20$ and grouping threshold $\delta = 10\%$, the OPC-Tree algorithm outputs 225 original clusters with average size 22.5. Since the

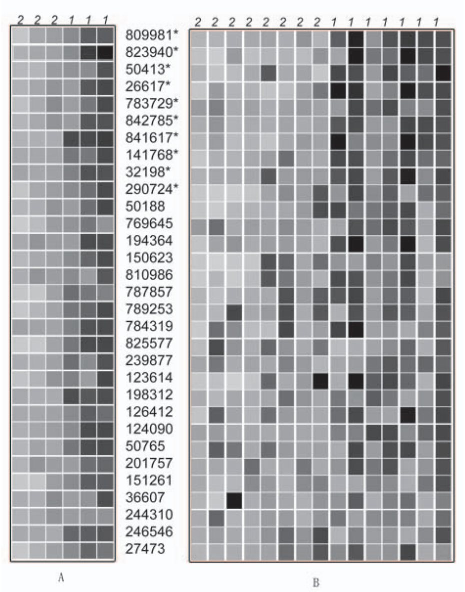


Figure 6. An example OP-Cluster and its corresponding full dimensional profiles along two types of tissues{BRCA1, BRCA2}, * marked genes are among previously identified 176 genes

total number of genes covered by all the clusters is 667, the average number of clusters a gene might exist is larger than $\frac{22.5 \times 225}{667} = 7.5$, which may indicate a high degree of overlapping between clusters. To minimize the overlap, we use the hierarchical merging algorithm described in Section 3. Given $\theta = 0.1$, we obtain 12 merged clusters with an average size of 89.

To assess the classification capability of the clusters, we use gene ontology information of each gene to evaluate whether the cluster has significant enrichment of one or more function groups. The ontology of the 799 yeast genes is downloaded from gene ontology consortium [22] in July, 2003. We use 104 different function categories below ontology level 2 and with a family size at least 8. The discovered OP-Clusters in each level of the hierarchy are evaluated for enrichment with any of those function categories. Table 2 shows the details of several clusters with enriched function groups.

Cluster 1 in Table 2 contains 53 genes. In Cluster 1, one of the function groups enriched is the pre-replicative complex group. The algorithm discovers 6 out of 8 genes in the pre-replicative complex group which is presumably to help set up origins for the next cell cycle. The six genes are $\{MCM2$, $MCM3$, $MCM4, MCM5, MCM7$, $CDC6\}$ plotted in Figure 7 (b) while all 8 genes in the same function group are plotted in Figure 7 (a). A consistent tendency along the time course shared by the six genes can be observed. The remaining two genes $\{MCM6, CDC45\}$ in the pre-replicative complex group do not induce the same tendency and hence are excluded from the OP-Cluster. The remaining three clusters in Table 2 are presented in Figure 8. According to the three expression graphs, we can observe significant similarity of the cell cycle among genes within the same cluster and clear distinction of cell cycle patterns between different clusters. For example, cluster 2 peaks first at the 4th time point and peaks again at the 10th time point. The first peak of Cluster 3 occurs one time point later than Cluster 1 while Cluster 4 occurs even later in both two peaks. Significant function groups closely related to the regulation of cell cycle [17] have also been discovered in the three clusters(Table 2). Those function groups include, but are not limited to, cell cycle, DNA replication, DNA repair, mitotic cell cycle, glycoprotein biosynthesis and cykoskeleton.

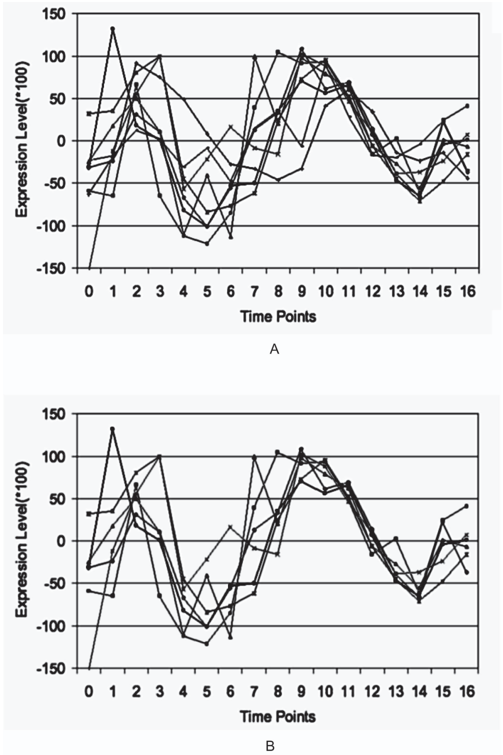


Figure 7. Expression Levels of Cluster 1 in Table 2 . (a) shows the expressions of all the genes in pre-replicative complex function family. (b) shows the genes in pre-replicative complex group in clustered by Cluster 1

Cluster	Number of genes	Enriched functional Category(total genes)	Clustered genes within the category(k)	$-log_{10}$ (P-value)
1.	53	pre-replicative complex(8)	6	5
		ATP dependent DNA helicase activity(10)	4	3
2.	109	cell proliferation(77)	33	13
		Cell cycle(66)	28	11
		DNA replication and chromosome cycle (38)	17	6
		Mitotic cell cycle(41)	17	6
3.	66	Nuclear chromatin(9)	7	5
		Cytoskeleton(27)	11	6
4.	70	Cytoplasm(97)	27	11
		DNA binding(27)	11	6
		glycoprotein biosynthesis(6)	5	5

Table 2. Enrichment of OP-Clusters by at least one function category

It is also observed in several studies that co-expressed genes tend to share common regulatory elements in their promoter regions [19]. We use GeneSpring software to find common motif appearing in the promoter regions(500 bases upstream of the translation start sites) of the genes in Cluster 2, 3 and 4. Significant motifs have been discovered. For example, the motif ACGCGT which was shown to be a perfect MCB element in [17] is located in 74 genes in Cluster 2.

5 Conclusions and Future Work

To discover clusters representing consistent tendencies exhibited by a subset of conditions in gene expression data, we introduce a new model named OP-Cluster and devise a depth-first algorithm that can efficiently and effectively discover all OP-Clusters satisfying some user-specified threshold. Our OP-Cluster model extends the original OPSM by relaxing strict orders among conditions to allow equivalent groups defined on similar expression levels.

Meanwhile, we also investigate the overlap among OP-clusters. Overlap is usually unavoidable due to the fact that genes belonging to different function categories may appear in more than one clusters. One approach to minimize the number of clusters is to merge clusters guided by the principle of maximizing the similarity between them but minimizing the similarity with respect to the rest. We build a hierarchy of clusters. This may serve as one step toward the goal of building a computational framework to systematically analyze and predict genes' function categories.

Still, there are several extensions we can make based on our model. Although we've relaxed the strict order requirement among the conditions by introducing order equivalent groups, it is still too optimistic to expect the order among groups are always meaningful, in the presence of normal biological variations and noises in microarray data. Therefore, one extension of the current model is to explore similar but not exact orders among a subset of conditions. The similarity measure of two sequences can base on the number of reverse pairs. For example, two sequences $abcd$ and $dcba$ have six reverse pairs, $\{\{a,b\}$, $\{a,c\}$, $\{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}\}$ while $abcd$ and $acbd$ only have one reverse pairs, $\{b,c\}$. If the similarity threshold is 3, we can take $abcd$ and $acbd$ as two similarly ordered sequences. Our algorithm can also be adapted to accommodate this by implementing a similarity check at each branch during the depth-first development.

Besides the above extensions and improvements, we will continue to investigate the functional, clinical or biochemical interpretations of OP-Clusters.

References

[1] A. Ben-Dor, R.Shamir, and Z.Yakhini. Clustering gene expression patterns. in *J Comput Biol* 6(3-4):281-97.

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

[3] A. Ben-Dor, B. Chor, R.Karp, and Z.Yakhini. Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem. In *RECOMB* 2002.

[4] M.P.S. Brown, W.N. Grundy, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler. Knowledge-basedanalysis of microarray gene expression data by using support vector machine. *proc. Natl Acad. Sci.* USA, 97, 262-267.

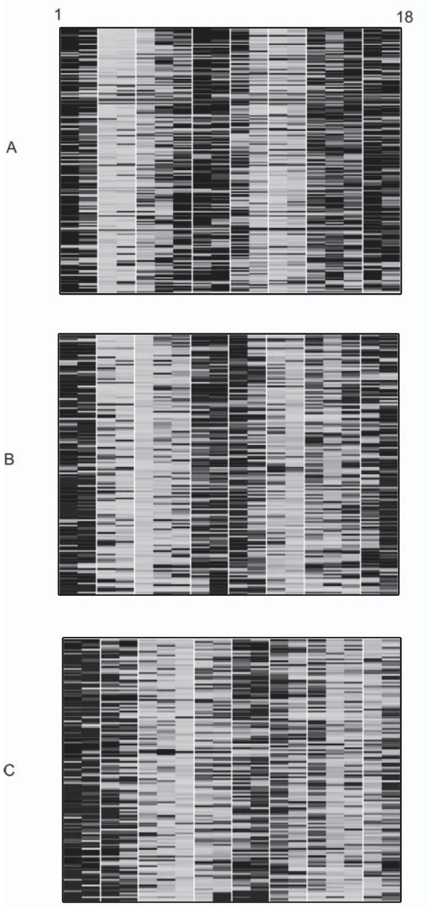


Figure 8. Expression Levels of Enriched Clusters of Table 2. Figure A, B and C represent Cluster 2, 3, and 4 in Table 2 respectively. Gene expression levels during the yeast cell cycle of the three Clusters 2,3 and 4. The columns in each graph correspond to 18 time points while the rows correspond to the genes in the cluster.

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

[6] Y. Cheng and G. Church. Biclustering of expression data. In *ISMB*, 2000.

[7] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proc Natl Acad Sci U S A*, 95(25):14863-8, 1998.

[8] A. V. Heydebreck, W. Huber, A. Poustka, and M. Vingron. Identifying splits with clear separation: a new class discovery method for gene expression data. *Bioinformatics*, Vol.17 Suppl.1, 2001.

[9] J. Hartigan. *Clustering Algorithms*. Wiley.

[10] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl Acad. Sci.* USA, 97, 12079-12084, 2000.

[11] S. Kaski, J. Nikkil, and G. Wong. Analysis And Visualization Of Gene Expression Data Using Self-Organizing Maps, *Proceedings of NSIP-01, IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing, 2001.*

[12] M. Kendall and J. D. Gibbons. *Rank Correlation methods*. 1990.

[13] L. Lazzeroni and A. Owen. Plaid models for gene expression data. http://www-stat.stanford.edu/ owen/plaid/,2000

[14] J. Liu and W. Wang. Subspace clustering by tendency in high dimensional space. In *ICDM 2003*.

[15] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17, S243-S252, 2001.

[16] R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. In *ISMB*, pages 307-216, 2000.

[17] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Lyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microaray hybidization. *Molecular Biology of the Cell*, 9:3273-2297, 1998.

[18] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, Vol.18, pages S136-S144, 2002.

[19] S. Tavazoie, J. D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture, *Nature Genetics*, 22: 281-285, 1999.

[20] S. Tavazoie, J. Hughes, M.Campbell, R. Cho, and G. Church. Yeast micro data set. In *http://arep.med.harvard.edu/biclustering/yeast.matrix*, 2000.

[21] L.F. Wu, T.R. Hughes, A.P. Davierwala, M.D. Robinson, R. Stoughton, and S.J. Altschuler. Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters.*Nature Genetics* 2002, 31:255-265.

[22] Gene Ontology Consortium, www.geneontology.org.