Discovering Significant Relaxed Order-Preserving Submatrices

Qiong Fang Wilfred Ng
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{fang,wilfred}@cse.ust.hk

Jianlin Feng
School of Software
Sun Yat-Sen University
Guangzhou, China
fengjlin@mail.sysu.edu.cn

ABSTRACT

Mining order-preserving submatrix (OPSM) patterns has received much attention from researchers, since in many scientific applications, such as those involving gene expression data, it is natural to express the data in a matrix and also important to find the order-preserving submatrix patterns. However, most current work assumes the noise-free OPSM model and thus is not practical in many real situations when sample contamination exists.

In this paper, we propose a relaxed OPSM model called ROPSM. The ROPSM model supports mining more reasonable noise-corrupted OPSM patterns than another well-known model called AOPC (approximate order-preserving cluster). While OPSM mining is known to be an NP-hard problem, mining ROPSM patterns is even a harder problem. We propose a novel method called OPSM-Growth to mine ROPSM patterns. Specifically, two pattern growing strategies, such as column-centric strategy and row-centric strategy, are presented, which are effective to grow the seed OPSMs into significant ROPSMs. An effective median-rank based method is also developed to discover the underlying true order of conditions involved in an ROPSM pattern. Our experiments on a biological dataset show that the ROPSM model better captures the characteristics of noise in gene expression data matrix compared to the AOPC model. Importantly, we find that our approach is able to detect more quality biologically significant patterns with comparable efficiency with the counterparts of AOPC. Specifically, at least 26.6% (75 out of 282) of the patterns mined by our approach are strongly associated with more than 10 gene categories (high biological significance), which is 3 times better than that obtained from using the AOPC approach.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*

KDD'10, July 25–28, 2010, Washington, DC, USA.

General Terms

Algorithm, Experimentation, Performance

Keywords

biclustering, order-preserving submatrices, backbone order, relaxed order-preserving submatrics

1. INTRODUCTION

The advent of DNA microarray technologies has greatly advanced the studies of gene expression. Gene expression data are usually arranged in a matrix, with each row corresponding to one gene, each column to one condition, and each entry in the matrix representing the expression level of a gene under a specific condition. In the area of gene expression analysis, an important research problem is to discover submatrix patterns in the gene expression matrix. We follow the convention in [3] and call this problem *biclustering* and the discovered patterns *biclusters*. A bicluster consists of a set of genes that have similar expression levels under a set of conditions. Based on different similarity functions, different bicluster models can be formulated. One typical example is the *order-preserving submatrix model* (or the OPSM for short), proposed by Ben-Dor et al. [1].

Table 1: An example of order-preserving submatrix

Genes \ Conds	t_1	t_2	t_3	t_4	t_5
g_1	**7**	**13**	**16**	**2**	30
g_2	**9**	**16**	**23**	**6**	0
g_3	**4**	**6**	**8**	**3**	5
g_4	8	6	2	13	5

An order-preserving submatrix consists of a subset of genes and a subset of experimental conditions such that the expression levels of every gene induce the same linear order of the conditions. This linear order of the conditions represents the consensus trend that the expression levels of all the genes in the subset follow. Considering the gene-condition matrix shown in Table 1, there is a submatrix (P, Q) with $P = \{g_1, g_2, g_3\}$ and $Q = \{t_1, t_2, t_3, t_4\}$, as highlighted in bold font. For any gene g_i in P, we order its expression levels under conditions in Q in ascending order and then replace the values by their corresponding condition labels. We find that all the genes in P induce an identical linear order on Q, i.e., $[t_4 \succ t_1 \succ t_2 \succ t_3]$. Thus, the submatrix (P, Q) is an order-preserving submatrix.

While it has been shown in extensive research work that the adoption of the OPSM model promotes the detection of important biological associations in gene expression analysis [1, 12, 8, 7, 10], researchers also find that it may not be realistic to assume that, as required by the OPSM model, all the rows in the submatrix induce the same linear orders in applications. The reason is that, due to the instrumental limitations or measurement errors, it is inevitable that the data in gene expression matrix are corrupted by noise [1, 12]. In this case, the strict OPSM model prohibits the discovery of larger (usually more significant) but noise-contaminated OPSMs. Therefore, it is necessary to relax the strict OPSM model in order to allow more significant order-preserving submatrices to be discovered.

Recently, Zhang et al. [12] proposed a relaxation to the OPSM model with the general idea as follows: a pre-specified fraction of rows in the bicluster are required to induce the same linear order of columns, and this linear order is called the *core order* of the bicluster, while every other row is only required to induce a similar order with the core order. Such a bicluster is called an *approximate order preserving cluster* (or AOPC for short). The core order is used to capture the consensus trend that all the rows in the AOPC follow. This relaxation has enabled the discovery of more biologically significant biclusters.

In this paper, we propose a new bicluster model as follows. We assume there exactly exists a *backbone order* associated with a bicluster and require that all the rows in the bicluster be similar enough to the backbone order. In contrast to the AOPC model, our model removes the *fraction requirement* that a pre-specified fraction of the rows involved in an AOPC are required to induce exactly the same order. In fact, our relaxation is motivated by the observation that two valid AOPCs can be merged into a bigger bicluster which is biologically more significant, yet the bigger bicluster is not a valid AOPC pattern because it fails to satisfy the fraction requirement, as demonstrated by the example in Section 2.3. We call the proposed model the *relaxed order-preserving submatrix* (ROPSM for short). Our empirical studies show that our ROPSM model is more effective in finding significant biclusters including many that are missed by the AOPC model.

As it is already known that mining strict OPSMs is an NP-hard problem [1], mining ROPSMs, which is a more general problem, is even more difficult to resolve. There are also other challenges as follows. First, the OPSMs patterns hold some nice properties like anti-monotonicity, which can reduce the search space when the OPSMs are mined. However such a property is no longer held by ROPSM patterns. Second, due to the presence of noise, it is very difficult to identify the backbone order of the ROPSM patterns. The AOPC mining method assumes that the core order of an AOPC pattern is maintained by a fraction of rows, and is passed on through the merging process. However, this method cannot be used for mining ROPSM patterns, since we cannot expect that any one of the rows in the ROPSM should retain the backbone order of the pattern. Thus, we adopt a new strategy for mining ROPSM patterns, which takes seed patterns as input and expands them until maximal ROPSM patterns are reached.

In summary, the main contributions of this paper are as follows.

1. We propose a new relaxation model, called *ROPSM*, to lift the restriction of the OPSM model. We demonstrate that our model is effective to capture the characteristics of noisy OPSM patterns, and to detect significant patterns from real biological data.

2. We propose an ROPSM mining method, which starts with OPSM patterns, and adopts efficient strategies to expand those OPSM patterns into maximal ROPSM patterns. A median-rank based method is also proposed to identify the backbone order of the ROPSM patterns.

3. Extensive empirical studies have been conducted using real biological data sets. The experiment results show that our method significantly outperforms existing work as evidenced by the fact that more biologically significant patterns can be discovered.

The organization of the rest of this paper is as follows. In Section 2, we first introduce some notations that are used throughout the paper, and then propose our new ROPSM model. Following that, a comparison with an existing noisy OPSM model is presented. In Section 3, an ROPSM mining method is introduced. Experiments on real gene expression datasets are presented in Section 4. In Section 5, related works are discussed. Finally, we conclude the paper in Section 6.

2. THE ROPSM MODEL

In this section, we first introduce some notations that are used throughout the paper. Then, we give formal definitions of our ROPSM model. A comparison with another relaxed OPSM model is also presented to further illustrate the modeling capabilities of ROPSM.

2.1 Basic Notations

We denote a gene expression matrix with n genes and m conditions as $M(G, T)$, where the gene set is given by $G = \{g_1, \ldots, g_n\}$ and the condition set is given by $T = \{t_1, \ldots, t_m\}$. Given $P \subseteq G$ and $Q \subseteq T$, $M'(P, Q)$ is called a submatrix of $M(G, T)$. We may use a lighter notation (P, Q) in subsequent discussion to mean $M'(P, Q)$ whenever no ambiguity arises.

Given a gene $g_i \in G$ and a set of conditions Q, we order g_i's expression values under conditions in Q in ascending order and then replace the values by their corresponding condition labels. The resultant linear order of condition labels in Q, denoted as $o_Q^{(g_i)}$, is called the *induced order* of g_i on Q, e.g., $o_Q^{(g_1)} = [t_4 \succ t_1 \succ t_2 \succ t_3]$ in Table 1. We denote by $O_{(P,Q)}$ the set of all $o_Q^{(g_i)}$ for $g_i \in P$ and call $O_{(P,Q)}$ the induced order set of the submatrix (P, Q). In particular, if all the orders in $O_{(P,Q)}$ are identical, we call the submatrix (P, Q) an order-preserving submatrix of M. For example, the submatrix $(\{g_1, g_2, g_3\}, \{t_1, t_2, t_3, t_4\})$ is an OPSM pattern of the matrix shown in Table 1.

2.2 The Model

In this subsection, we define our ROPSM model, which tries to capture the characteristics of noisy OPSM patterns.

We adopt a similarity function defined based on the concept of *longest common subsequence* (or simply LCS), and the similarity function is formally defined as follows.

DEFINITION 2.1. ***the LCS similarity.*** *Given two orders o_1 and o_2, the LCS similarity between o_1 and o_2, denoted as $d_{LCS}(o_1, o_2)$, is defined as*

$$d_{LCS}(o_1, o_2) = \frac{|LCS(o_1, o_2)|}{|T(o_1) \cup T(o_2)|},$$

where $|LCS(o_1, o_2)|$ is the length of the longest common subsequence between o_1 and o_2, and $T(o_i)$ is the set of items involved in o_i.

For example, given two orders $o_1 = [t_1 \succ t_2 \succ t_3 \succ t_4]$ and $o_2 = [t_2 \succ t_1 \succ t_3 \succ t_4]$, the longest common subsequences between o_1 and o_2 are $[t_1 \succ t_3 \succ t_4]$ and $[t_2 \succ t_3 \succ t_4]$, both of which have the length of 3. Therefore, the LCS similarity between o_1 and o_2 is $\frac{3}{4}$. The LCS similarity and its variants are widely used in molecular biology [2], and can be computed by using dynamic programming in $O(n^2)$ time.

We now define our relaxed OPSM model, called *ROPSM*, as follows.

DEFINITION 2.2. ***Relaxed Order-Preserving SubMatrix (ROPSM).*** *Given a similarity threshold α, a submatrix (P, Q) is an ROPSM if there exists a linear order τ_Q of Q such that the LCS similarity between τ_Q and the induced order of every row in P is larger than or equal to α. The linear order τ_Q is called the backbone order of the ROPSM (P, Q).*

Since an ROPSM (P, Q) is always associated with a backbone order τ_Q, we also denote the ROPSM as $(P, Q : \tau_Q)$. Consider the matrix shown in Table 1 again. Suppose that the entry (g_2, t_4) in the matrix is corrupted and its value changes from 6 to 10, and assume that the similarity threshold α is set to be 0.75. Although the submatrix $(\{g_1, g_2, g_3\}, \{t_1, t_2, t_3, t_4\})$ is not an OPSM, it is an ROPSM satisfying α, with $[t_4 \succ t_1 \succ t_2 \succ t_3]$ as the backbone order. Note that OPSM is a special case of ROPSM, i.e., ROPSM satisfying the threshold $\alpha = 1$.

An ROPSM $(P, Q : \tau_Q)$ is said to be *maximal* if there does not exist any other ROPSM $(P', Q' : \tau_{Q'})$ such that $P \subseteq P'$, $Q \subseteq Q'$, and τ_Q is a subsequence of $\tau_{Q'}$.

Having defined the ROPSM model, we then formulate the *ROPSM mining problem* as follows:

DEFINITION 2.3. ***ROPSM mining problem.*** *Given a data matrix M and a similarity threshold α, the goal is to mine from M the maximal ROPSMs that satisfy the threshold α.*

2.3 Comparison with the AOPC Model

The AOPC model was proposed in [12]. A submatrix (P, Q) is called an AOPC if the following two requirements are satisfied. First, at least $\sigma_s \cdot |P|$ rows induce an identical linear order of Q. This order is called the *core order* of the AOPC and is denoted as π_Q. Second, the linear orders induced from the other $(1 - \sigma_s) \cdot |P|$ rows share with π_Q a longest common subsequence with the length of at least $\sigma_c * |Q|$.

While both the ROPSM model and the AOPC model adopt a similarity threshold, i.e., α and σ_c, to restrict the amount of noise existing in the pattern, our ROPSM model does not assume the core order and removes the first requirement of the AOPC model. Such relaxation is not trivial, in the sense that more significant biclusters, which are not discovered based on the AOPC model, now becomes valid ROPSMs. This point was testified in our experiments, and can be seen in the following example illustrated in Table 2.

Table 2: Significance comparison of AOPCs

	AOPC M_1' (P_1, Q_1)	AOPC M_2' (P_2, Q_2)	Merged $M'(P, Q)$
Number of columns	7	7	7
Number of rows (Number of main_rows)	121(30)	98 (30)	159 (30)
Number of strongly associated categories	12	8	15
The smallest p-value	10^{-20}	10^{-15}	10^{-24}

Since the similar dataset is used, we implement the AOPC mining method and mine the AOPCs by setting σ_s and σ_c the same values as used in [12], i.e., $\sigma_s = 0.2$ and $\sigma_c = 0.6$. Among the set of AOPCs that are generated, there are two AOPCs, respectively denoted as $M_1'(P_1, Q_1)$ and $M_2'(P_2, Q_2)$, whose information is listed in Table 2. These two AOPCs contain the same set of columns, i.e., $Q_1 = Q_2$, and the core orders of the two AOPCs, i.e., π_{Q_1} and π_{Q_2}, are also the same. In AOPC M_1', 30 out of 121 rows induce identical orders as the core order, while 30 out of 98 rows in AOPC M_2' induce identical orders as the core order. We call this portion of rows *maintaining rows* (*main_rows* for short). Then, we try to merge these two AOPCs into a bigger submatrix, denoted as $M'(P, Q)$, such that $P = P_1 \cup P_2$ and $Q = Q_1 \cup Q_2$. The merged submatrix M' contains 7 columns with the same core order as π_{Q_1} and π_{Q_2}, and 159 rows with 30 maintaining rows. M' is not a valid AOPC because the fraction of maintaining rows, i.e., 30/159, is smaller than the pre-specified threshold $\sigma_s = 0.2$. However, when we check the biological significance of the two AOPCs and the merged submatrix M' using the widely-used p-value measure [10, 6], the two AOPCs have 12 and 8 strongly associated gene categories (with p-value $\leq 10^{-9}$ [12]) respectively, and have the smallest p-value of 10^{-20} and 10^{-15} with their most strongly associated categories. In contrast, the biological significance of the merged submatrix M' is strongly associated with 15 categories and has the smallest p-value of 10^{-24} with the most strongly associated category, which means that the merged submatrix is more biologically significant than the two AOPCs.[1]

Under the ROPSM model, the merged submatrix M' is an ROPSM pattern that satisfies the similarity threshold $\alpha = 0.6$, although it is not a valid AOPC.

3. THE OPSM-GROWTH ALGORITHM

In this section, we propose an efficient algorithm called OPSM-Growth for mining ROPSM patterns.

Notably, given a data matrix $M(G, T)$, for a set of columns Q with $Q \subseteq T$ and a linear order τ_Q, we can simply check every row in G to see if it *supports* τ_Q or not. A row in G is said to *support* τ_Q if its induced order on Q has the LCS similarity with τ_Q no smaller than α. Let P be the set of rows that supports τ_Q. We can see that P can be deterministically identified and that (P, Q) is an ROPSM with the backbone order τ_Q. Thus, it appears that a straightforward way of mining ROPSM patterns is to exhaustively search all possible linear orders of combinations of columns in T. However, as discussed in [1], the cost of such an exhaustive search

[1]Detailed information about this example is available at `http://www.cse.ust.hk/~fang/aopc-example.html`.

is prohibitively expensive and is infeasible when the number of columns in T is larger than or equal to 4. On the other hand, we should avoid searching some linear orders which may finally lead to *insignificant patterns*. While we aim to discover patterns that show strong biological significance, a consensus is that, considering the size of the patterns, for a fixed number of columns (or rows), a larger number of rows (or columns) usually leads to more significance [7]. Therefore, when mining ROPSM patterns, we focus on searching some linear orders that likely lead to patterns with larger size.

The backbone orders of OPSM patterns are good candidates for such linear orders. Given an OPSM $(P, Q : \tau_Q)$ and the similarity threshold α, we can always expand the OPSM by $(\frac{1}{\alpha} - 1)|Q|$ columns, denoted as the column set ΔQ, and get a submatrix $(P, Q \cup \Delta Q)$. We then get a linear order of $(Q \cup \Delta Q)$, denoted as $\tau_{Q\cup\Delta Q}$, such that τ_Q is a subsequence of $\tau_{Q\cup\Delta Q}$. Since the induced order of every row in the submatrix $(P, Q \cup \Delta Q)$ shares with $\tau_{Q\cup\Delta Q}$ a longest common subsequence with the length of at least $|Q|$, the LCS similarity between them is at least α. Therefore, the submatix $(P, Q \cup \Delta Q)$ is an ROPSM with $\tau_{Q\cup\Delta Q}$ as the backbone order.

On the other hand, intuitively, a significant ROPSM very likely contains at least one order-preserving submatrix. Considering a noise-contaminated data matrix, it is reasonable to assume the probability that an entry in the matrix is corrupted by noise is less than 50%. Otherwise, we cannot distinguish whether the mined patterns really show the association among the involved genes or they are simply formed because of noise. Moreover, noise does not always affect the formation of OPSM patterns. For example, in Table 1, even if the entry (g_2, t_4) changes to 7 due to some noise, the induced order of row g_2 on $\{t_1, t_2, t_3, t_4\}$ remains the same, i.e., $[t_4 \succ t_1 \succ t_2 \succ t_3]$. Thus, the submatrix $(\{g_1, g_2, g_3\}, \{t_1, t_2, t_3, t_4\})$ is still an OPSM pattern. Therefore, only the noise that causes the entries of the matrix deviating from respecting the original induced order of the corresponding row may finally affect the discovery of OPSM patterns. The probability that such noise exists, however, should be even smaller. Thus, it is reasonable to assume that *a significant ROPSM pattern may still contain a non-contaminated order-preserving submatrix*.

Motivated by the above ideas, we propose an ROPSM mining method as follows: we first mine a set of OPSMs that satisfy some size thresholds; then we take those OPSMs as seeds and expand them until maximal ROPSMs patterns are reached.

3.1 Mining Seed OPSMs

We adopt an existing OPSM mining method, called the OPC-Tree [8], to mine seed OPSMs. The OPC-Tree method was proposed to exhaustively mine all OPSMs that satisfy two size thresholds r_{min} and c_{min}, which means that the OPSMs contain at least r_{min} rows and at least c_{min} columns. The general idea of the OPC-Tree method is that, for every induced linear order of columns, its subsequences that may lead to a valid OPSM are enumerated and organized in a compact prefix-sharing tree structure. Then, the size thresholds are used to further prune the tree. Finally, a linear scan over the tree is conducted to output all OPSMs that satisfy the thresholds.

Here, we adapt this method to mining all maximal OPSMs that satisfy the two size thresholds r_{min} and c_{min}. Note that, although our ROPSM mining method also takes OPSMs as an input, which seems to be similar to the AOPC mining method, the resultant ROPSMs mined by our method are hardly influenced by the settings of the thresholds r_{min} and c_{min}, which however have a big effect on the performance of the AOPC mining method.

The AOPC mining method takes the set of OPSMs as an input, and merges all possible pairs of OPSMs that may result in a valid AOPC pattern. Therefore, those rows that appear in some final AOPC patterns should definitely appear in some initial OPSMs. However, it may happen that the setting of the thresholds r_{min} and c_{min} exclude the discovery of some OPSM patterns, which contain correlated rows to a certain significant AOPC pattern. These rows will fail to appear in the AOPC pattern as the final result. To avoid missing promising rows of significant patterns, we search for ROPSMs by expansion with the remaining part of the matrix being probed in an efficient way.

3.2 OPSM-Growth

In this subsection, we introduce the OPSM-Growth algorithm, which grows the seed OPSMs into ROPSM patterns. Given an ROPSM (initially an OPSM), the OPSM-Growth algorithm can expand it column wise and row wise. We respectively implement the columm-wise and row-wise expansion as Col-Expand and Row-Expand procedures. The Col-Expand procedure always takes the best column (in terms of some criterion) to expand the current ROPSM (or OPSM) pattern. The Row-Expand procedure simply scans the remaining rows that are not included in the current pattern, and expand the pattern by those rows which have high enough LCS similarity with its backbone order. When the current pattern is expanded by a new column, the backbone order of the pattern needs to be updated accordingly, while the backbone order does not change during row-wise expansion.

We can grow an OPSM pattern by calling the Col-Expand and Row-Expand procedures in different order, which leads to different pattern growing strategies. Also, different methods can be adopted to update the backbone order during column-wise expansion. Next, we introduce the techniques adopted by OPSM-Growth algorithm that handles pattern growing and backbone order updating.

3.2.1 Pattern Growing Strategies

Combining the Col-Expand and Row-Expand procedures in different order leads to different pattern growing strategies. Here we discuss two representative strategies: *column-centric strategy* and *row-centric strategy*.

a. Column-centric Strategy
The basic idea of the column-centric strategy is that the seed OPSM pattern is repetitively expanded column wise by calling Col-Expand until no more columns can be added. Then, the Row-Expand procedure is called for one time in order to guarantee that all the rows that are similar with the backbone order of the current pattern are included. The column-centric strategy can be demonstrated as follows:

$$(P, Q) \xrightarrow{\text{Col-Expand}} (P, Q\cup\Delta Q) \xrightarrow{\text{Row-Expand}} (P\cup\Delta P, Q\cup\Delta Q),$$

where (P, Q) is an initial seed OPSM.

We can see that the seed OPSM pattern (P, Q) is part of the resultant ROPSM pattern it grows into. Thus, the rows in P should definitely support the backbone order of the resultant ROPSM. Therefore, it is reliable to identify the column set as well as the backbone order of the resultant ROPSM based on the rows of the seed OPSM, which is the motivation of the column-centric strategy.

b. Row-centric Strategy
While the column-centric strategy postpones the row-wise expansion until the last minute, the row-centric strategy calls the Row-Expand procedure whenever there are new rows that can be taken to expand the current pattern. The basic idea of the row-centric strategy is that, every time the current pattern is updated by a new column (note that the backbone order should also be updated accordingly), we compute the LCS similarity between every row in the pattern and the updated backbone order, and denote the smallest LCS similarity as μ^*. Then, Row-Expand expands the current pattern with those rows that have the LCS similarity with the updated backbone order no less than μ^*. Similarly, the row-centric expansion can be demonstrated as follows:

$$(P, Q) \xrightarrow{\text{Col-Expand}} (P, Q \cup \{t_1\}) \xrightarrow{\text{Row-Expand}} (P \cup \Delta P_1, Q \cup \{t_1\}) \xrightarrow{\text{Col-Expand}} \cdots \xrightarrow{\text{Row-Expand}} (P \cup \Delta P, Q \cup \Delta Q)$$

We can see that the threshold μ^* used for row-wise expansion decreases over iterations as the pattern becomes larger. Intuitively, the resultant ROPSM pattern is approached by gradually relaxing the noise-controlling criterion as the size of the pattern becomes larger. When the size of the pattern is small, less noise should be allowed. As the pattern grows larger, more noise will be tolerated.

Note that both the column-centric strategy and the row-centric strategy guarantee that the resultant ROPSMs are maximal.

3.2.2 Backbone Order Updating

In the OPSM-Growth algorithm, when a new column is taken to expand the current pattern, the backbone order of the current pattern should be updated accordingly. We propose a median-rank based method for updating the backbone order, and call the method *MedRank Updating*.

The concept of *median rank* is effective for reducing the effect of noise on rankings, which is adopted in [4, 5] for producing the consensus (linear or bucket) order of a set of permutations. We define a median-rank based score, called *MedScore* (denoted as S_m), and determine the backbone order of a pattern according to the *MedScores* of its columns. The definition of *MedScore* is as follows.

DEFINITION 3.1. ***Median Rank Based Score (MedScore)*** *Given an input matrix $M(G, T)$, and an ROPSM pattern (P, Q), the MedScore of a column $t \in Q$, denoted as $S_m(t)$, is computed as:*

$$S_m(t) = \text{median}\{\ldots, r(o_T^{(g_i)}, t), \ldots | g_i \in P\},$$

where $o_T^{(g_i)}$ is the induced order of row g_i on the column set T, $r(o_T^{(g_i)}, t)$ is the rank of column t in $o_T^{(g_i)}$, and the function median() *returns the median value of the set of ranks.*

We use the following example to further illustrate the computation of *MedScore*. Suppose there is an input matrix (G, T) with $G = \{g_1, \ldots, g_5\}$ and $T = \{t_1, \ldots, t_6\}$, and its induced order set is listed in the second column of Table 3. Given a pattern (P, Q) with $P = \{g_1, g_2, g_3, g_4\}$ and $Q = \{t_1, t_2, t_3, t_4\}$, we compute the *MedScore* of column t_2 based on (P, Q). We first get t_2's ranks in order $o_T^{(g_1)}$, $o_T^{(g_2)}$, $o_T^{(g_3)}$, and $o_T^{(g_4)}$, which respectively are 2, 2, 4, and 3; then the *MedScore* of t_2, i.e., $S_m(t_2)$, is the median of the four ranks which is 2.5. The function *median*() returns the average of the two median ranks when the number of ranks are even.

Table 3: Illustration of MedScore computation

	Induced orders on T	Induced orders on Q
$\boldsymbol{g_1}$	$[t_1 \succ \boldsymbol{t_2} \succ t_5 \succ t_6 \succ \boldsymbol{t_3} \succ t_4]$	$[t_1 \succ t_2 \succ t_3 \succ t_4]$
$\boldsymbol{g_2}$	$[t_1 \succ \boldsymbol{t_2} \succ t_6 \succ t_5 \succ \boldsymbol{t_3} \succ t_4]$	$[t_1 \succ t_2 \succ t_3 \succ t_4]$
$\boldsymbol{g_3}$	$[t_1 \succ t_6 \succ \boldsymbol{t_3} \succ \boldsymbol{t_2} \succ t_5 \succ t_4]$	$[t_1 \succ t_3 \succ t_2 \succ t_4]$
$\boldsymbol{g_4}$	$[t_1 \succ \boldsymbol{t_3} \succ \boldsymbol{t_2} \succ t_6 \succ t_5 \succ t_4]$	$[t_1 \succ t_3 \succ t_2 \succ t_4]$
g_5	$[t_6 \succ t_2 \succ t_4 \succ t_3 \succ t_5 \succ t_1]$	$[t_4 \succ t_3 \succ t_2 \succ t_1]$

Having computed the *MedScores* of all the columns in Q, the columns are ordered in increasing order of their *MedScore*, and the resultant order of the columns is the *MedRank backbone order* of the above pattern (P, Q). For example, the MedRank backbone order of the pattern (P, Q) is $[t_1(1) \succ t_2(2.5) \succ t_3(4) \succ t_4(6)]$, where the number in the bracket is the *MedScore* of the column. (P, Q) is an ROPSM pattern satisfying $\alpha = 0.75$. We expect that the MedRank backbone order could well tolerate the influence of noise and express the underlying true order of the columns in the pattern.

Note that, *the* MedScore *of columns are computed based on their ranks in the induced orders on T instead of the ranks in the induced orders on Q.* In this way, the gaps between the ranks of columns in the induced orders on T can be kept, so that the ordering relationship among columns may be more distinguishable.

Let us consider the example in Table 3 again. The ranks of column t_3 in the induced orders on T are respectively 5, 5, 3, 2, and thus its *MedScore* is $S_m(t_3) = 4$. Since $S_m(t_2)$ is smaller than $S_m(t_3)$, t_2 is ranked higher than t_3 in the MedRank backbone order. The ordering relationship between t_2 and t_3 in the backbone order intuitively well respects the truth that t_2 has high ranks in the majority of the induced orders of (G, T) while t_3 gets comparatively low ranks. However, when we check the ranks of t_2 and t_3 in the induced orders on Q as listed in the third column of Table 3, both t_2 and t_3 have ranks of $2, 2, 3, 3$ and thus they have the same median rank of 2.5. The ordering relationship between t_2 and t_3 cannot be determined in this way.

In addition to the advantage that computing the *MedScore* using the ranks in induced orders on T can better keep the ordering relationship between columns, another benefit of computing *MedScore* in this way is that, those ranks are fixed throughout the expansion process. Thus, an inverted index can be built for fetching the ranks of items efficiently.

Algorithm 1 shows the details of the MedRank-Updating method. Given an ROPSM $(P, Q : \tau_Q)$ and a new column

Algorithm 1: MedRank-Updating

Input: ROPSM $(P, Q : \tau_Q)$ with $\tau_Q = [t_1 \succ \cdots \succ t_{|Q|}]$; column t to be expanded
Output: $\tau_{Q\cup\{t\}}$ - the MedRank backbone order of $(P, Q \cup \{t\})$
Variable: $O_{(P,T)}$ - induced orders from (P, T)

1. $V_{(P,T)}(t) = \{$ranks of t in o_i with $o_i \in O_{(P,T)}\}$;
2. $S_m(t)$ equals to the median of ranks in $V_{(P,T)}(t)$;
3. $\tau_{Q\cup\{t\}} = [t_1 \succ \cdots \succ t_k \succ t \succ t_{k+1} \cdots \succ t_{|Q|}]$ such that $S_m(t_i) \leq S_m(t)$ with $1 \leq i \leq k$ and $S_m(t_i) > S_m(t)$ with $k + 1 \leq i \leq |Q|$;

Algorithm 2: ROW-MED-Growth

Input: a set U of OPSMs; the similarity threshold α
Output: a set V of ROPSMs
Variable: μ^* - current smallest LCS similarity

1. **for** $(P, Q : \tau_Q) \in U$ **do**
2. **if** (P, Q) *is a submatrix of an ROPSM in* V **then**
3. discard $(P, Q : \tau_Q)$;
4. **else**
5. **while** Col-Expand($(P, Q : \tau_Q)$, α, μ^*) = *succeed* **do**
6. Row-Expand($(P, Q : \tau_Q)$, μ^*);
7. **end**
8. **end**
9. **end**
10. Add $(P, Q : \tau_Q)$ to V;

t, the *MedScore* of t, i.e., $S_m(t)$ is firstly computed (Lines 1 and 2). Then, the column t is inserted into τ_Q such that the *MedScore*s of all the columns before t are no larger than $S_m(t)$ (Line 3).

Time Complexity. The time complexity of the MedRank-Updating method is analyzed as follows. Suppose the final ROPSM pattens contain at most k rows and s columns, each call of the MedRank-Updating method costs $O(k \log k)$ time, which is actually the cost of computing the *MedScore* of newly added columns.

3.2.3 The OPSM-Growth Algorithm

Having illustrated the pattern growing strategies and the backbone updating method, we now introduce the OPSM-Growth algorithm. Taking different growing strategies, we devise two variants of the OPSM-Growth algorithm, called COL-MED-Growth and ROW-MED-Growth. The ROW-MED-Growth method adopts the row-centric pattern growing strategy, and its details are shown in Algorithm 2. The two procedures Col-Expand and Row-Expand are shown in Algorithm 3 and 4, respectively.

First, let us see the procedure Col-Expand (Algorithm 3). Suppose $(P, Q : \tau_Q)$ is the current ROPSM pattern. Every remaining column t_j in $(T - Q)$ is taken to update the current backbone order τ_Q (Line 2). Then, both the smallest LCS similarity μ between the updated backbone order and induced orders and the sum of the LCS similarities S_{LCS} are recorded (Lines 3 and 4). Among all the columns in $(T - Q)$, column t_k which has the largest μ value is picked; if there are ties, the column with larger S_{LCS} wins (Line 6). We set μ^* as the μ value of column t_k (Line 7). If μ^* is no

Algorithm 3: Col-Expand

Input: an ROPSM $(P, Q : \tau_Q)$; the similarity threshold α, current smallest LCS similarity μ^*
Output: *succeed* if (P, Q) is updated by a new column; *fail* otherwise
Variable: $O_{(P,Q)}$ - the induced order set of (P, Q)

1. **for** $t_j \in T - Q$ **do**
2. $\tau_{Q\cup\{t_j\}}$ = MedRank-Updating($(P, Q : \tau_Q), t_j$);
3. $\mu(Q, t_j) = \min\{d_{LCS}(\tau_{Q\cup\{t_j\}}, o_i), \forall o_i \in O_{(P,Q\cup\{t_j\})}\}$;
4. $S_{LCS}(Q, t_j) = \sum_{o_i \in O_{(P,Q\cup\{t_j\})}} d_{LCS}(\tau_{Q\cup\{t_j\}}, o_i)$;
5. **end**
6. Pick t_k such that $\mu(Q, t_k) = \max\{\mu(Q, t_j), \forall t_j \in T - Q\}$; ties are broken with larger S_{LCS};
7. $\mu^* = \mu(Q, t_k)$;
8. **if** $\mu^* \geq \alpha$ **then**
9. Update τ_Q to $\tau_{Q\cup\{t_k\}}$; $Q \cup \{t_k\} \rightarrow Q$;
10. Return *succeed*;
11. **end**
12. Return *fail*;

Algorithm 4: Row-Expand

Input: an ROPSM $(P, Q : \tau_Q)$; current smallest LCS similarity μ^*
Variable: $o_Q^{(g_j)}$ - the inducd orders of row g_j on Q

1. $\Delta P = \phi$;
2. **for** *row* g_j *in* $G - P$ **do**
3. **if** $d_{LCS}(\tau_Q, o_Q^{(g_j)}) \geq \mu^*$ **then**
4. $\Delta P \cup \{g_j\} \rightarrow \Delta P$;
5. **end**
6. **end**
7. $P \cup \Delta P \rightarrow P$;

smaller than the similarity threshold α, column t_k is chosen to update the current pattern (P, Q) as well as the backbone order, and the Col-Expand procedure returns *succeed*; otherwise, it returns *fail* (Lines 8 to 11).

The Row-Expand procedure (Algorithm 4) takes the current ROPSM pattern $(P, Q : \tau_Q)$ and the current smallest LCS similarity μ^*, which is passed from Col-Expand, as input parameters. It conducts a linear scan of the remaining rows in the input matrix. Those rows whose induced order has the LCS similarity with τ_Q no smaller than μ^* are chosen to expand the current ROPSM pattern (Lines 2 to 7).

We now introduce the ROW-MED-Growth algorithm as shown in Algorithm 2. For every seed OPSM $(P, Q : \tau_Q)$, we first detect whether it is a submatrix of some ROPSM pattern that is mined. If it is, we discard this OPSM and continue with a new OPSM (Lines 2 to 4). We call this step *early pruning*, which will be explained in more detail later. Then, an OPSM that passes the early pruning is expanded with the row-centric growing strategy being adopted. That is, Col-Expand and Row-Expand are alternately called until a maximal ROPSM is reached (Lines 5 to 7). Finally the newly generated ROPSM is added to V (Line 10).

We omit the details of the algorithm COL-MED-Growth, i.e., the other variant of OPSM-Growth. Generally speaking, the COL-MED-Growth algorithm takes the column-centric

pattern growing strategy. That is, for every seed OPSM that passes the early pruning step, the Col-Expand procedure is repetitively called until no more columns can be added. Finally, one round of Row-Expand is performed.

Early Pruning. We observe that, no matter which variant of the OPSM-Growth algorithm is taken, it is possible that similar OPSMs may finally grow into the same ROPSM pattern. In order to avoid useless expansion that may produce an ROPSM already found, we adopt an early pruning step as follows. Before expanding an OPSM, we first check whether it is a submatrix of some mined ROPSM. The early pruning step is quite efficient, since we only need to check if the backbone order of a given OPSM is a subsequence of the backbone order of some ROPSM. If it is, the OPSM is surely the submatrix of the ROPSM.

It might be argued that some seed OPSMs that will finally lead to new ROPSM patterns may also get pruned during the early pruning step. However, such new ROPSM patterns will be very similar to some other ROPSM patterns, which may, for example, catch similar functional associations among a similar set of genes. Considering real applications, such similar patterns may not be very informative even they are significant. Therefore, we still prefer to adopt the strict early pruning step in order to avoid generating many similar patterns.

Time Complexity. Suppose the input matrix contains n rows and m columns, and the mined ROPSM patterns contain at most p rows and q columns. The running time of the Col-Expand procedure is $O(m(q \log q + q^2) + m)$ where $O(q \log q)$ is spent updating the MedRank backbone order and $O(q^2)$ is spent computing the LCS similarity between orders. The running time of the Row-Expand procedure is $O(nq^2)$. Then the time complexity of the ROW-MED-Growth method is $O(p(m+n)q^2)$. The COL-MED-Growth method calls the procedure Col-Expand p times and calls Row-Expand only once, and therefore its time complexity is $O((pm + n)q^2)$.

Table 4: Data matrix (G, T) with $G = \{g_1, g_2, g_3, g_4\}$ and $T = \{t_1, t_2, t_3, t_4, t_5\}$

Matrix (G,T)						The induced order set
	t_1	t_2	t_3	t_4	t_5	$O_{(G,T)}$
g_1	4	2	3	1	5	$[t_4 \succ t_2 \succ t_3 \succ t_1 \succ t_5]$
g_2	**2**	**3**	**4**	1	5	$[t_4 \succ t_1 \succ t_2 \succ t_3 \succ t_5]$
g_3	**2**	**4**	**5**	3	1	$[t_5 \succ t_1 \succ t_4 \succ t_2 \succ t_3]$
g_4	**2**	**4**	**5**	1	3	$[t_4 \succ t_1 \succ t_5 \succ t_2 \succ t_3]$

Next, we further illustrate the ROW-MED-Growth algorithm in the following example. Suppose there is an input matrix (G, T) as shown in Table 4, and the induced orders of the rows are listed in the last column of the table. The similarity threshold α is 0.75. We start with a seed OPSM (P, Q) where $P = \{g_2, g_3, g_4\}$ and $Q = \{t_1, t_2, t_3\}$. The initial backbone order is $\tau_Q = [t_1(2) \succ t_2(4) \succ t_3(5)]$ where the numbers in the bracket are the *MedScore*s of the columns. First, (P, Q) are expanded column-wise. For the remaining two columns t_4 and t_5, we find that expanding (P, Q) with either one will lead to the μ^* value of 0.75. However, when t_4 is taken for expansion, the sum of LCS similarities (i.e., S_{LCS} value) is larger. Therefore, the OPSM is expanded with column t_4, i.e., $Q = Q \cup \{t_4\}$. Since the *MedScore* of t_4 is 1, the backbone order is updated to $\tau_Q = [t_4(1) \succ t_1(2) \succ t_2(4) \succ t_3(5)]$. Then, the current ROPSM is expanded row-wise. We find that the induced order of g_1 on Q, i.e., $o_Q^{(g_1)}$, is $[t_4 \succ t_2 \succ t_3 \succ t_1]$, and the LCS similarity between $o_Q^{(g_1)}$ and τ_Q is 0.75 which is no smaller than μ^*. Thus, the current ROPSM is expanded by row g_1, and turns out to be $(\{g_1, g_2, g_3, g_4\}, \{t_1, t_2, t_3, t_4\})$. We try to expand the ROPSM column-wise again, and find that the remaining column t_5 cannot be taken for expansion. We therefore stop with the final ROPSM pattern as $(\{g_1, g_2, g_3, g_4\}, \{t_1, t_2, t_3, t_4\} : [t_4 \succ t_1 \succ t_2 \succ t_3])$.

4. EXPERIMENTS

In this section, we study the performance of the two variants of our ROPSM mining algorithm, i.e., COL-MED-Growth and ROW-MED-Growth, through a series of experiments. We also compare them with the AOPC mining algorithm in terms of the biological significance of the mined patterns. All the experiments are conducted on a Macbook Pro with 2.53GHZ CPU and 4G memory.

The AOPC mining method takes a set of OPSMs as an input, and merges all possible pairs of OPSMs that may result in a valid AOPC pattern. The validity of the AOPCs is evaluated using the two thresholds σ_s and σ_c. We respectively set σ_s and σ_c to be 0.2 and 0.6, which are the same as those adopted by the experiments in [12]. To improve the efficiency of the AOPC mining method, the input OPSM set can be divided into groups. The AOPC mining algorithm runs on every group, and the resultant AOPCs produced by each running are gradually combined. Referring to [12], we run the AOPC mining method by respectively setting the initial number of groups to be 1 and 64, and we denote the AOPC mining method with these two different settings respectively as AOPC-1 and AOPC-64.

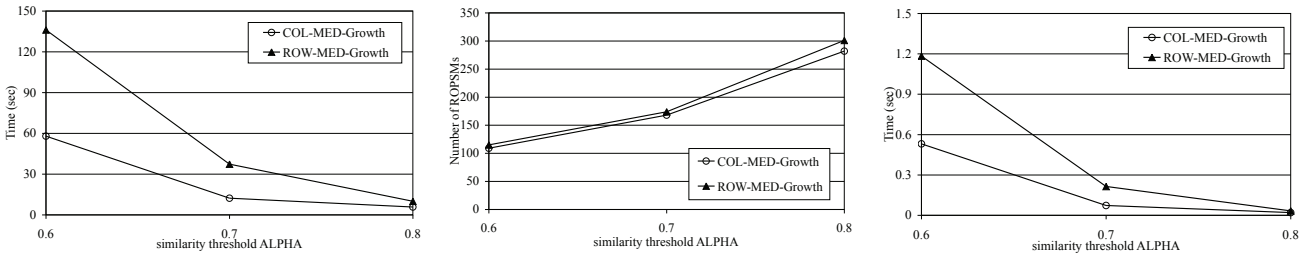
The dataset we use is a real gene expression data set – the yeast cell cycle data from [11], which contains the expression levels of 771 regulated genes across 18 time points. We use the tool Gene Ontology Term Finder (GO-TermFinder)[2] to validate the biological significance of the mined ROPSMs. The GO-TermFinder tool computes the p-values between the mined patterns and known gene categories. A smaller p-value indicates a stronger association between the patterns and the category. In the experiments, for a particular pattern (ROPSM, AOPC, and OPSM), we only count those categories the pattern strongly associates with. We took the same p-value threshold, i.e., 10^{-9}, for strong association as the experiments in [12]. That is, a pattern is said to be strongly associated with a known gene category if the p-value between the pattern and the category is less than 10^{-9}.

We first adopt the OPC-Tree method to mine all maximal OPSMs satisfying the thresholds $l_{min} = 60$ and $s_{min} = 5$. There are totally 595 OPSMs having been mined, which form the input set of OPSMs for both the OPSM-Growth algorithm and the AOPC mining method.

4.1 Effectiveness of OPSM-Growth

In the first set of experiments, we study the effectiveness of the COL-MED-Growth and ROW-MED-Growth methods as the similarity threshold α changes. Intuitively the α value should be larger than 0.5, and thus we respectively set α to be 0.6, 0.7, and 0.8.

[2] `http://search.cpan.org/dist/GO-TermFinder/`



(a) Running time (b) Number of patterns (c) Unit time

Figure 1: Effectiveness of the OPSM-Growth algorithm

Figure 1(a) shows the running time of the two methods, Figure 1(b) shows the number of ROPSM patterns being mined, and Figure 1(c) shows the amount of time that is spent in finding a single pattern, which we call *unit time*. We can see from Figure 1(a) that, the running time of both methods increases as α becomes smaller. The reason is that when more noise is allowed to exist in the ROPSM patterns, more time is needed for each seed OPSM to grow into a bigger ROPSM. On the other hand, when α is smaller, a seed OPSM tends to grow into an ROPSM with larger size, which is accordingly more likely to cover other seed OPSMs and prune them during the early pruning step. Therefore, less ROPSMs are finally mined when α is smaller, as shown in Figure 1(b).

Comparing these two methods, we can see that COL-MED-Growth is more efficient in terms of the running time, and the advantage is more significant when α is small. The reason is that the row-centric growing strategy needs to scan the remaining rows (in Row-Expand) repetitively during the expansion. Although one scan only takes linear time, it is still time-consuming since the number of rows contained in the gene expression matrix is much larger than the number of columns. Considering the number of mined ROPSMs, we can see that ROW-MED-Growth mines slightly more patterns. This is because the column-centric expansion tends to mine patterns with more columns, and thus the seed OPSMs are more likely to be pruned by those previously mined ROPSM patterns. Therefore, less ROPSMs can be finally mined by COL-MED-Growth.

The efficiency and scalability of the COL-MED-Growth are more apparent in Figure 1(c), which shows the unit time of the two methods under differnt α values. When α is 0.6, the COL-MED-Growth method on average spends about 0.53 seconds to find one ROPSM pattern, while the ROW-MED-Growth method needs 1.18 seconds to find one ROPSM pattern, which is more than two times longer.

4.2 Biological Significance of ROPSMs

We conduct the second set of experiments to compare the biological significance of the ROPSMs that are mined by the ROW-MED-Growth method and the COL-MED-Growth method. The experiment results are shown in Figure 2, where the x-axis of the two histograms are the number of categories that an ROPSM strongly associates with, and each bar corresponds to a combination of the ROPSM mining method and the α value. For each method and under every α value, we respectively count the number of ROPSMs that strongly associate with more than a certain number of categories (e.g., $12, 11, \ldots$). The histogram in (a) shows the statistics of the number of significant ROPSMs, and the histogram in (b) shows the statistics of the fraction of significant ROPSMs.

If we consider the ROPSMs that strongly associate with more than 10 categories, we can see that both methods mine a larger percentage (67.9% and 70.0%, respectively) of such ROPSMs when α is 0.6, which means that setting α to be 0.6 better captures the characteristics of noisy OPSM patterns in this data set. However, the total number of ROPSMs mined when α is 0.6 is smaller, as already illustrated in the previous subsection. Both methods mine the most absolute number (98 and 102, respectively) of such ROPSMs when α is 0.7.

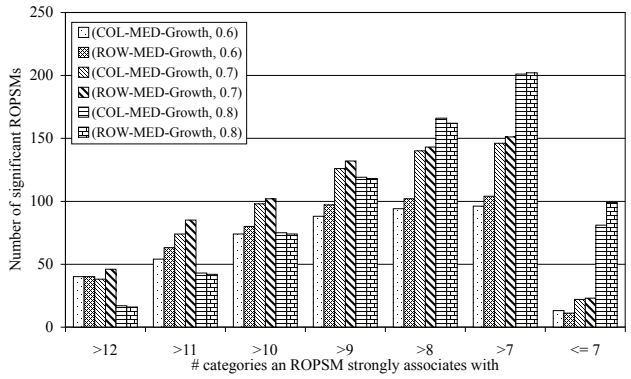
When comparing the COL-MED-Growth method and the ROW-MED-Growth method, we find that, generally, ROW-MED-Growth mines a larger percentage of the significant patterns when α is 0.6; while COL-MED-Growth mines more percentage of significant patterns when α is 0.8. The results imply that the column-centric expansion performs better when less noise is allowed to exist in the ROPSM patterns while the row-centric expansion performs better when more noise is allowed.

4.3 Performance Comparison with AOPC

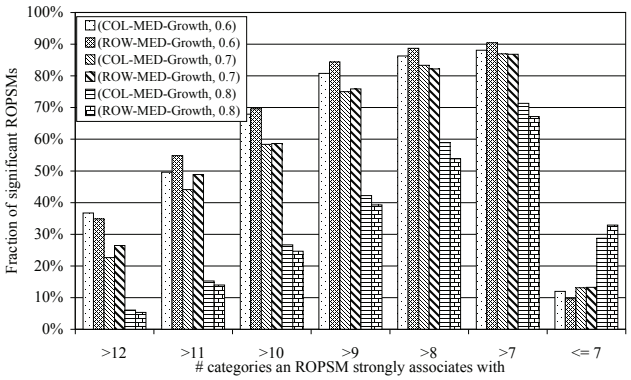
The last set of experiments is conducted to compare the biological significance of the patterns (ROPSM, AOPC, or OPSM) respectively mined by our OPSM-Growth algorithm, the AOPC mining method, and the OPC-Tree method. For the AOPCs, we list the statistical records of AOPCs mined by both AOPC-1 and AOPC-64. The AOPC-64 method runs much faster than AOPC-1, although several significant patterns that strongly associate with more than 12 categories fail to be mined.

Note that our OPSM-Growth algorithm and the AOPC mining method aim at mining different kinds of noisy OPSM patterns, i.e., ROPSMs and AOPCs. To make the results of these two algorithms comparable in some sense, we impose a possible requirement, that is, the amount of time spent in mining one pattern should be roughly the same. According to this requirement, we choose the COL-MED-Growth method for comparison with α set to 0.8. The experiment results are listed in Table 5.

The COL-MED-Growth method spends 5.78 seconds in finding 282 ROPSMs, and thus mining a single ROPSM pattern costs 0.0205 second. In comparison, the AOPC-64 method only spends 0.0167 seconds in finding an AOPC. Although the unit time cost for finding an ROPSM is slightly longer than the unit time cost for finding an AOPC, more number of significant patterns have been found by our COL-MED-



(a) Number of Significant ROPSMs



(b) Fraction of Significant ROPSMs

Figure 2: Biological significance of the ROPSM patterns

Table 5: Comparison with AOPC

	COL-MED-Growth	AOPC-1	AOPC-64	OPSM
#Patterns	282	222	239	595
Running time	5.78 sec	74.49 sec	3.99 sec	–
Unit Time	0.0205 sec	0.3355 sec	0.0167 sec	–
# categories a pattern strongly associate with				
> 12	17(6.03%)	5(2.25%)	2(0.84%)	0
> 11	43(15.25%)	10(4.5%)	11(4.6%)	5(0.84%)
> 10	**75**(**26.60**%)	**19**(**8.56**%)	**20**(**8.37**%)	**16**(**2.69**%)
> 9	119(42.20%)	29(13.06%)	32(13.39%)	35(5.88%)
> 8	166(58.87%)	39(17.57%)	42(17.57%)	44(7.39%)
> 7	201(71.28%)	67(30.18%)	68(28.45%)	75(12.61%)
≤ 7	81(28.72%)	155(69.82%)	171(71.55%)	520(87.39%)

Growth method. For example, considering the patterns that strongly associate with more than 10 gene categories, 75 out of 282 (26.60%) ROPSMs mined by COL-MED-Growth are such patterns. In contrast, neither AOPC-1 nor AOPC-64 mines more than 20 (or more than 8.56%) AOPCs that fulfill such level of significance. Actually, more significant number of patterns are mined by our COL-MED-Growth methods considering any level of significance. It should also be noted that COL-MED-Growth with α to be 0.8 is never the best setting for mining significant patterns. If we consider the trade-off between the unit time and the quality of the mined pattern, COL-MED-Growth with α set to be 0.7 spends 0.0733 seconds in finding one ROPSM which is about 4.3 times longer than the unit time cost by AOPC-64. However, the number of significant ROPSM patterns surely exceeds 4.4 times that of significant AOPC patterns at the same level, especially when those patterns strongly associate with 9 or more categories are considered.

The last column of Table 5 lists the statistical records of the OPSMs that are taken as the input of our OPSM-Growth algorithm. We can see that even less number (or percentage) of OPSMs, i.e., 16 (or 2.69%) strongly associate with more than 10 categories. However, starting from these OPSM patterns with much lower level of significance, our OPSM-Growth algorithm can mine considerably more significant patterns.

5. RELATED WORK

The concept of biclustering was first introduced to discover submatrix patterns (biclusters) in gene expression data matrix by Chen and Church [3]. Madeira et al. [9] classify four major types of biclusters: constant-value bicluster, constant-row (or -column) bicluster, coherent-value bicluster, and coherent-evolution bicluster. The order-preserving submatrix (OPSM) is a typical type of coherent-evolution bicluster, which was defined by Ben-Dor et al. [1].

While the OPSM model is shown to be biologically significant, mining OPSM patterns is very challenging and the problem is NP-hard. Therefore, Ben-Dor et al. first proposed a model-based method that keeps a limited number of partial models which are smaller OPSMs for possible further expansion, and then expand them into full OPSMs. Their method however is designed to identify only a single pattern, and the significance of the pattern is very sensitive to the selection of partial models. Liu et al. later proposed an OPSM mining method that mines multiple OPSMs at the same time. Their method adopts a tree structure that organizes all the candidate OPSMs, with some pruning techniques being also applied. However, the computation cost is still large, especially when the number of columns increases. Gao et al.'s KiWi framework [7] also adopts a tree structure to store all necessary information for searching twig OPSMs, which is characterized by containing a large number of columns and a few rows.

Ben-Dor et al. also noticed that the OPSM model is too strict with the presence of noise, and thus they developed a probabilistic noise model under the assumption of uniformly random noises. Zhang et al. later proposed an intuitive relaxation called the approximate order-preserving cluster (AOPC) [12]. The AOPC model only requires a pre-specified fraction of rows in the bicluster to induce the same linear order of columns, while the remaining rows only

need to be similar enough. Based on the AOPC model, they also proposed an AOPC mining method that merges pairs of smaller AOPCs (initially OPSMs) into bigger ones in a greedy manner. Our ROPSM model further relaxes the AOPC model and demonstrates much better effectiveness in mining quality patterns.

Chen and Church's algorithm (CC's algorithm for short) [3], which similarly adopts column-wise and row-wise expansion or deletion, is designed for finding approximate constant-value and constant-row (or column) biclusters. CC's algorithm adopts a two-phase strategy. First rows and columns are iteratively removed from the original input matrix until a valid pattern is reached. The pattern discovered in the first phase can be regarded as a seed, and it is then iteratively expanded by previously removed columns and rows until a maximal valid pattern is reached. This two-phase method, however, is very time-consuming. Besides, if more patterns are needed, those previously discovered patterns are masked by random values to avoid finding identical patterns, which accordingly influences the quality of the following discovered patterns. In comparison, we take maximal OPSMs as seeds, which is shown to be more likely to obtain significant patterns.

6. CONCLUSION

In this paper, we study the problem of mining noisy OPSM patterns and develop a new relaxed OPSM model called the ROPSM model. In contrast to the strict OPSM model, ROPSM only requires that each gene in the bicluster induces a linear order that is sufficiently similar with respect to the backbone order of the biclusters. Thus, our proposed model is able to better capture the biological fact that correlated genes usually induce similar but not exactly the same orders on the same set of conditions.

We propose a new method called OPSM-Growth to mine ROPSM patterns, which uses an effective median rank based method to generate (or approximate) the backbone order and hence find the ROPSMs. Our experiment on a biological dataset shows that the ROPSM model better captures the characteristics of noise in gene expression data matrix compared to the AOPC model. Importantly, compared with the more effective version of AOPC, our ROPSM mining method mines more quality biologically significant patterns and needs far less time to process the mining than its counterpart. When compared with the more efficient version of AOPC, we need comparable mining time but gain far more significant patterns. Thus, our OPSM-Growth method achieves a better balance between the efficiency in processing the mining and the quality of the mined patterns.

7. ACKNOWLEDGEMENTS

This work is partially supported by China NSF Grant 60970043 and HKUST RGC Grant 618509. We would like to thank KDD reviewers for giving us insightful comments.

8. REFERENCES

[1] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02*, pages 49–57. ACM, 2002.

[2] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIRE 2000*, pages 39–48, 2000.

[3] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, pages 93–103, 2000.

[4] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *SIGMOD '03*. ACM, 2003.

[5] J. Feng, Q. Fang, and W. Ng. Discovering bucket orders from full rankings. In *SIGMOD '08*, pages 55–66. ACM, 2008.

[6] R. A. Fisher. On the interpretation of $\mathcal{X}^2$ from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

[7] B. J. Gao, O. L. Griffith, M. Ester, and S. J. M. Jones. Discovering significant opsm subspace clusters in massive gene expression data. In *SIGKDD '06*, pages 922–928. ACM, 2006.

[8] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. In *ICDM '03*. IEEE Computer Society, 2003.

[9] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM TCBB*, 1(1):24–45, 2004.

[10] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, and et al. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[11] P. T. Spellman, G. Sherlock, M. Q. Zhang, and et al. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[12] M. Zhang, W. Wang, and J. Liu. Mining approximate order preserving clusters in the presence of noise. In *ICDE '08*, pages 160–168, 2008.