Knowl Inf Syst (2006)
DOI 10.1007/s10115-006-0031-9

Knowledge and Information Systems

REGULAR PAPER

Daxin Jiang · Jian Pei · Murali Ramanathan · Chuan Lin · Chun Tang · Aidong Zhang

Mining gene–sample–time microarray data: a coherent gene cluster discovery approach

Received: 17 September 2004 / Revised: 14 March 2005 / Accepted: 27 May 2005

Abstract Extensive studies have shown that mining microarray data sets is important in bioinformatics research and biomedical applications. In this paper, we explore a novel type of gene–sample–time microarray data sets that records the expression levels of various genes under a set of samples during a series of time points. In particular, we propose the mining of *coherent gene clusters* from such data sets. Each cluster contains a subset of genes and a subset of samples such that the genes are coherent on the samples along the time series. The coherent gene clusters may identify the samples corresponding to some phenotypes (e.g., diseases), and suggest the candidate genes correlated to the phenotypes. We present two efficient algorithms, namely the *Sample-Gene Search* and the *Gene–Sample Search*, to mine the complete set of coherent gene clusters. We empirically evaluate the performance of our approaches on both a real microarray data set and synthetic data sets. The test results have shown that our approaches are both efficient and effective to find meaningful coherent gene clusters.

Keywords Bioinformatics · Clustering · Microarray data

D. Jiang(✉)
School of Computing Engineering,
Nanyang Technological University,
Blk N4, 2a-32, Nanyang Avenue,
639798 Singapore
E-mail: dxjiang@ntu.edu.sg

J. Pei
Simon Fraser University, Canada
E-mail: jpei@cs.sfu.ca

M. Ramanathan · C. Lin · C. Tang · A. Zhang
State University of New York at Buffalo, Buffalo, New York, USA
E-mail: murali@acsu.buffalo.edu, {chuanlin, chuntang, azhang}@csu.buffalo.edu

1 Introduction

The microarray technology can measure the expression levels of thousands of genes simultaneously. It is an important research problem in bioinformatics and clinical research to explore the patterns in microarray data sets. For example, in drug development, gene expression patterns may reflect gene-level responses to different drug treatments and provide deep insights into the nature of the diseases.

Most microarray data sets[1] can be divided into two categories.

– The *gene–time data sets* record the expression levels of various genes during important biological processes over a series of time points.
– The *gene–sample data sets* account the expression levels of various genes across related samples.

Both gene–time data sets and gene–sample data sets can be represented by an $n \times l$ *gene expression matrix*, where each row is a gene and each column is either a sample (in a gene–sample data set) or a time instant (in a gene–time data set). Each cell in the matrix represents the expression level of a certain gene on a certain sample or at a certain time point.

With the latest advances in the microarray technology, the expression levels of a set of genes under a set of samples can be monitored synchronically during a series of time points [39]. Different from the previous gene–time or gene–sample microarray data sets, these new data sets have three types of variables: *genes*, *samples*, and *time*. We call such data *gene–sample–time microarray data*, or *GST data* for short. Figure 1a elaborates the structure of a *GST* microarray data set.

In general, each cell $m_{i,j}^{k}$ in a *GST* data set represents the expression level of gene g_i under sample s_j at time point t_k. Interestingly, a *GST* data set can also be viewed as an $n \times l$ matrix such that each cell $m_{i,j}$ contains the time series with respect to gene g_i under sample s_j, as shown in Fig. 1b.

The previous studies on gene–sample microarray data (e.g., [1, 2, 13, 33]) indicate that high correlations may exist between the gene expression patterns and some diseases. It is natural to extend the similar analysis to *GST* microarray data. That is, it is interesting to identify a subset of genes G and a subset of samples S in a *GST* microarray data set such that each gene $g \in G$ has coherent patterns across the samples in S during the time series. For example, in Fig. 1, gene g_{i1}, g_{i2}, and g_{i3} show coherent patterns across samples s_{j1}, s_{j2}, and s_{j3}, respectively. We call such subsets of genes and samples a *coherent gene cluster*.

The computational model of *coherent gene clusters* addresses a significant problem in the clinical use of a variety of drug therapies. For example, IFN-β is the most widely prescribed immunomodulatory therapy for multiple sclerosis (an autoimmune disease of the brain and spinal cord). The therapy is known to exert all its biological effects via gene transcription but there are no validated markers for its long-term efficacy in multiple sclerosis. Although double blind, randomized, placebo-controlled clinical trials have established that IFN-β treatment reduces the progression of disability in multiple sclerosis, only 30–40% of patients respond well to the therapy. To define the mechanism of IFN-β and investigate the partial responsiveness of various patients, the expression levels of large numbers

[1] In this paper, the term "microarray data" is used to refer to *gene expression* microarray data.

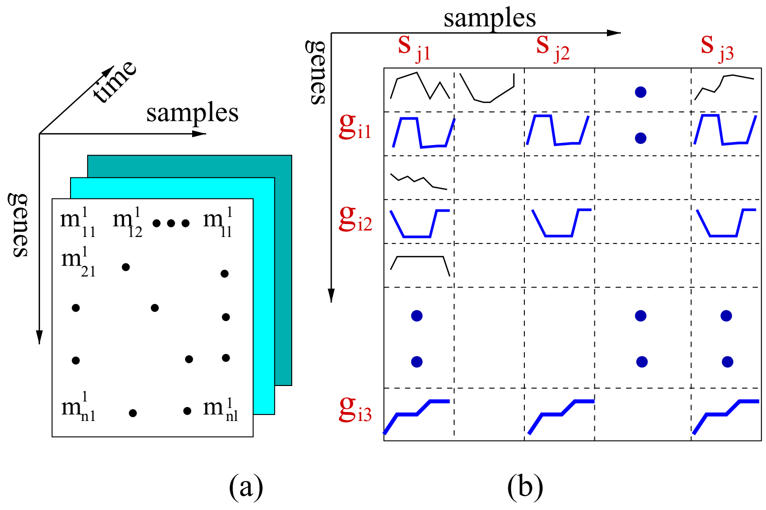


Fig. 1 The structure of *GST* microarray data

of genes were monitored for 13 multiple sclerosis patients during a 10-point time series [39].

There is considerable inter-individual heterogeneity in responsiveness to IFN-β. In other words, patients can differ both in the magnitude and rate of their gene expression profiles. However, the underlying mechanisms are not fully characterized. The *coherent gene cluster* model is directly relevant to characterizing this underlying heterogeneity of treatment responses to IFN-β, since it is capable of identifying patients whose responses are similar and defining the time courses of genes that distinguish these patient subsets. Moreover, the genes in the clusters may suggest the candidate genes correlated to the response.

In general, the coherent gene clusters may provide valuable hypotheses for biologists. The sample sets in clusters may correspond to some phenotypes (e.g., the diseased/healthy patients or patients responding differently to a treatment), while the corresponding set of genes may suggest the candidate genes correlated to the phenotypes.

The functions of genes in an organism are highly complicated. There are typically multiple coherent clusters in a data set. Different clusters may correlate to different phenotypes such as age and gender. Therefore, to avoid missing any valuable hypothesis, it is necessary to mine *all* the coherent clusters in a data set.

Many previous studies investigate the mining of interesting patterns from microarray matrices. For example, various clustering algorithms can identify the co-expressed genes showing coherent patterns during the time series (e.g., [19, 27, 33, 35]). Moreover, both supervised and unsupervised approaches are proposed to partition the samples into homogeneous groups (e.g., [5, 10, 24, 34]). Additionally, statistical approaches are proposed to validate the significance of the mining results (e.g., [22, 37]). However, all those previous studies target at conventional gene–time or gene–sample microarray data sets. The models of clusters in those previous studies are different from our coherent gene clusters, which disclose the correlation among genes, samples, and time points. Therefore, those algorithms cannot be extended directly to solve our problem.

Recently, the *pattern-based* clustering approaches (e.g., [38]) have been developed to discover subsets of objects following similar patterns on subsets of attributes. Conceptually, a pattern-based cluster is a coherent gene cluster. If we treat the *GST* microarray data sets as an $n \times l$ matrix of time series, as shown in Fig. 1b, then pattern-based clusters and coherent gene clusters may have some similarity at the first look. However, in some pattern-based approaches, a cluster requires that each pair of objects in the cluster must be coherent on each pair of attributes. Such a requirement is often too strong in practice. Our coherent gene clustering relaxes the constraints among the objects. Therefore, each traditional pattern-based cluster is a coherent gene cluster, but the other way is not necessarily true.

In this paper, we tackle the problem of *mining coherent patterns from gene–sample–time microarray data sets* and make the following contributions.

First, we propose a model of coherent gene clusters in *GST* microarray data sets. We justify that the model is meaningful for biomedical research.

Second, we identify the computational challenges and conduct a systematic research on mining coherent gene clusters from *GST* microarray data sets. We develop two approaches, namely the *Gene–Sample Search* and the *Sample–Gene Search*, to mine the complete set of coherent gene clusters. We illustrate and compare the efficiency and scalability of both approaches.

Last, we conduct an extensive empirical evaluation on both real data sets and synthetic data sets. Our results show that our proposed methods can find coherent gene clusters that are of interest to biomedical research from real data sets. The results on synthetic data sets also show that our algorithms are both efficient and scalable.

The rest of the paper is organized as follows. Section 2 defines the problem. Section 3 describes the preprocessing step of computing the maximal coherent sample sets for each individual gene. Section 4 presents two algorithms to mine coherent gene clusters. In Sect. 5, our methods are evaluated using both real and synthetic data sets. The related work is discussed in Sect. 6. We discuss other possible coherent clusters for GST data sets in Sect. 7. Section 8 concludes the paper.

2 Problem description

Given a set of n genes $G\text{-}Set = \{g_1, \ldots, g_n\}$ and a set of l samples $S\text{-}Set = \{s_1, \ldots, s_l\}$, we can measure the expression levels of the genes on the samples. The results form a conventional $n \times l$ microarray matrix $M = \{m_{i,j}\}$, where $m_{i,j}$ is the expression level of gene g_i $(1 \leq i \leq n)$ on sample s_j $(1 \leq j \leq l)$. If such microarray experiments are conducted synchronically on all genes and all samples at time instants $t_1, \ldots, t_T$, the results form an $n \times l \times T$ *GST microarray matrix* $M = \{m^t_{i,j}\}$, where $(1 \leq t \leq T)$.

A *GST* microarray matrix $M = \{m^t_{i,j}\}$ can also be viewed as an $n \times l$ matrix $M = \{m_{i,j}\}$ where $m_{i,j}$ is a vector of $\langle m^1_{i,j}, \ldots, m^T_{i,j}\rangle$. Hereafter, we do not strictly distinguish the two notations $M = \{m^t_{i,j}\}$ and $M = \{m_{i,j}\}$. Instead,

Table 1 Notations used in the paper

M	Gene expression data matrix
$G\text{-}Set$	Set of genes in M
$S\text{-}Set$	Set of samples in M
G, G'	Subsets of $G\text{-}Set$
S, S'	Subsets of $S\text{-}Set$
$(G \times S)$	Submatrix of M
$m_{i,j}$	Measure of gene i at sample j. It is a real *number* for conventional expression data matrix, while a real *vector* for *GST* data matrix
n	Number of genes in M
l	Number of samples in M
T	Number of time points in M
δ	User-specified coherence threshold
$\min_g$	User-specified minimum number of genes
$\min_s$	User-specified minimum number of samples

whenever $m_{i,j}$ is written, the corresponding vector $\langle m^1_{i,j}, \ldots, m^T_{i,j}\rangle$ is referred to. Table 1 list the notations used in this paper.

In this paper, we are interested in finding those genes that are coherent on a subset of samples during the whole time series. There are various methods to measure the correlation between two time series. However, for gene expression data, users are often interested in the overall trends of the expression levels instead of the absolute magnitudes. Therefore, we choose the *Pearson's correlation coefficient* as the coherence measure, since it is robust to shifting and scaling patterns [41]. Specifically, given two vectors m_{i,j_1} and m_{i,j_2} of gene g_i, the coherence is defined as

$$\rho(m_{i,j_1}, m_{i,j_2}) = \frac{\sum_{t=1}^{T}(m^t_{i,j_1} - \overline{m_{i,j_1}})(m^t_{i,j_2} - \overline{m_{i,j_2}})}{\sqrt{\sum_{t=1}^{T}(m^t_{i,j_1} - \overline{m_{i,j_1}})^2}\sqrt{\sum_{t=1}^{T}(m^t_{i,j_2} - \overline{m_{i,j_2}})^2}}, \tag{1}$$

where $\overline{m_{i,j}} = \sum_{t=1}^{T}(m^t_{i,j})/T$ is the mean of the expression levels of gene g_i on sample s_j. The correlation coefficient ranges between -1 and 1. The larger the value, the more coherent are the two vectors.

Definition 2.1 (Coherent gene submatrix) Given a *GST* data matrix M, a gene $g_i \in G\text{-}Set$ is *coherent* across a subset of samples $S \subseteq S\text{-}Set$, if for any given pair of samples $s_{j_1}, s_{j_2} \in S$, $\rho(m_{i,j_1}, m_{i,j_2}) \geq \delta$, where δ is a *minimum coherence threshold* specified by the user. A subset of genes $G \subseteq G\text{-}Set$ is *coherent* across a subset of samples $S \subseteq S\text{-}Set$, if every gene $g_i \in G$ is coherent across samples in S. We call a submatrix $(G \times S)$ a *coherent gene submatrix* if G is coherent across S. A coherent gene submatrix having u genes and v samples is said a (u, v)-coherent gene submatirx.

Proposition 2.1 (Trivial coherent gene submatrices) *For any gene g_i and any sample s_j, $(\{g_i\} \times \{s_j\})$ is a $(1, 1)$-coherent gene submatrix and $(G\text{-}Set \times \{s_j\})$ and $(\{g_i\} \times S\text{-}Set)$ are $(|G\text{-}Set|, 1)$- and $(1, |S\text{-}Set|)$-coherent gene submatrices, respectively.*

Proof The proposition follows the definition of coherent gene submatrix immediately. □

To avoid the triviality indicated in Proposition 2.1, we require that a coherent gene submatrix should consist of at least two genes and two samples.

Proposition 2.2 (Anti-monotonicity) *Let $(G \times S)$ be a coherent gene submatrix. Then, for any subsets $G' \subseteq G$ and $S' \subseteq S$, $(G' \times S')$ is also a coherent gene submatrix.*

Proof For any gene pair in G' and any sample pair in S', they must satisfy the coherence requirement since they are in the cluster $(G \times S)$. □

The anti-monotonicity of coherent submatrices brings a lot of redundancy. To avoid such redundancy, a user may only want the maximal submatrices. A coherent gene submatrix $(G \times S)$ is *maximal* if there exists no any other coherent gene submatrix $(G' \times S')$ such that $G \subseteq G'$, $S \subseteq S'$. Moreover, a user may not be interested in very small clusters, which are often formed by chance. Thus, a user can specify the minimum numbers of genes and samples in a submatrix. Generally, given $\min_g$ and $\min_s$ as user defined minimum gene size and minimum sample size thresholds, a submatrix $(G \times S)$ is called *significant* if $|G| \geq \min_g$ and $|S| \geq \min_s$.

Definition 2.2 (Coherent gene cluster) Given a *GST* microarray matrix M, a minimum coherence threshold δ, a minimum gene size threshold $\min_g$ and a minimum sample size threshold $\min_s$, a submatrix $(G \times S)$ of M is a *coherent gene clusters* if it satisfies the following constraints: (1) $(G \times S)$ is a coherent gene submatrix; (2) $(G \times S)$ is maximal; and (3) $|G| \geq \min_g$ and $|S| \geq \min_s$.

The problem of *mining coherent gene clusters* is to find the complete set of coherent gene clusters in the given data set with respect to the user-specified parameters.

3 Maximal coherent sample sets

We propose two algorithms to compute maximal coherent gene clusters. In both algorithms, to compute coherent gene clusters, we need to check whether a subset of genes are coherent on a subset of samples. To facilitate the tests, for each gene g_k, we compute the sets of samples S such that (1) $|S| \geq \min_s$; (2) g_k is coherent on S; and (3) there exists no proper superset $S' \supset S$ such that g_k is also coherent on S'. S is called a *maximal coherent sample set* of g_k. Please note that, in general, a gene may have more than one maximal coherent sample set.

For a gene g_k, all of its maximal coherent sample sets can be computed efficiently using the following 2-step process.

In the first step, we test whether gene g_k is coherent on each pair of samples (s_i, s_j). A binary triangle matrix $\{c_{i,j}\}$ is populated, where $1 \leq i < j \leq |S\text{-}Set|$. We set $c_{i,j} = 1$ if gene g_k is coherent on samples s_i and s_j, i.e., $\rho(m_{k,i}, m_{k,j}) \geq \delta$, otherwise, $c_{i,j} = 0$.

Once the matrix $\{c_{i,j}\}$ is populated, the problem of finding g_k's maximal coherent sample sets can be reduced to the problem of finding all maximal cliques of size at least $\min_s$ in graph $\mathcal{G}_k = (S\text{-}Set, E)$, where (s_i, s_j) is an edge in the graph if and only if $c_{i,j} = 1$. Here, we follow the terminology that a clique is a set of vertices such that the induced subgraph is a complete graph. A clique S is called *maximal* if there exists no any other clique S' such that $S \subset S'$. Please note that there may exist more than one maximal clique in a graph.

Unlike the conventional clique problem where the clique of the maximal size is found, here, we need to find the complete set of maximal cliques in the graph.

Theorem 1 (Complexity of preprocessing) *The problem of computing the complete set of maximal cliques that have at least* $\min_s$ *vertices is NP-hard.*

Proof It is well known that the conventional clique problem is NP-complete. Therefore, the counting problem of finding the complete set of cliques of size at least $\min_s$ is in #P. Since a #P problem corresponding to any NP-complete problem must be NP-hard, the problem of computing the complete set of maximal cliques in a graph is NP-hard. □

Fortunately, the real *GST* microarray data sets are often sparse and the number of samples is typically below 100. For each gene, the number of maximal cliques is quite small and the samples can often be partitioned into exclusive small subsets. Our experimental results show that, with efficient search and pruning techniques that will be introduced soon, it is still practical to find the complete set of maximal cliques. In the following, we will show how to find the maximal cliques of a sample set by a depth-first search in a sample set enumeration tree.

Given a set of samples $S = \{s_1, \ldots, s_l\}$, the set 2^S (i.e., all combinations of samples) can be enumerated systematically. For example, consider a set of samples $S = \{a, b, c, d\}$. The complete set of nonempty combinations of samples can be divided into 4 exclusive subsets: (1) the ones having sample a; (2) the ones having sample b but no a; (3) the ones having sample c but no a or b; and (4) $\{d\}$. They are shown as the immediate children of the root in Fig. 2.

These subsets can be further partitioned. For example, the first subset can be further divided into three exclusive sub-subsets: (1) the ones having samples a and b; (2) the ones having samples a and c but no b; and (3) $\{a, d\}$.

The tree shown in Fig. 2 is called a *set enumeration tree* [28] with respect to $\{a, b, c, d\}$. It provides a conceptual tool to enumerate the complete set of combinations systematically.

We can conduct a *recursive*, *depth-first search* of the sample set enumeration tree to detect the maximal cliques of the samples. Given a set of samples S, the set enumeration tree has $2^{|S|}$ nodes. However, we never need to materialize such a

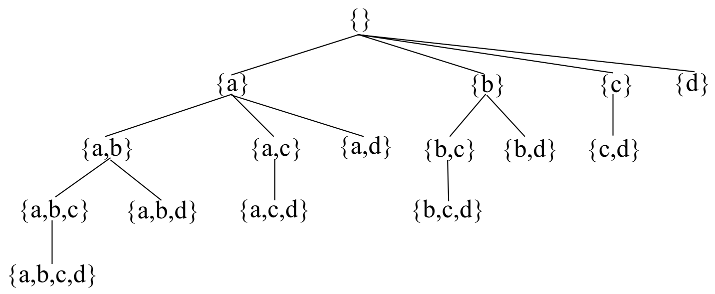


Fig. 2 Enumeration of combinations of samples

tree. Instead, we only need to keep a path from the root of the tree to the node that we are working on as a working set. Such a path contains at most $(|S| + 1)$ nodes.

Clearly, in the set enumeration tree, each node contains a unique subset of samples. Thus, we can use the subset of samples to refer to the node. At node $\{s_{i_1}, \ldots, s_{i_k}\}$ $(1 \leq i_1 < \cdots < i_k \leq l)$, we also keep a list *Tail*, which contains the samples that can be used to extend the node to some larger subsets of samples in the subtree. We have the following result.

Lemma 3.1 *At node $v = \{s_{i_1}, \ldots, s_{i_k}\}$ of the sample set enumeration tree, where $(1 \leq i_1 < \cdots < i_k \leq l)$, a sample $s_j \notin Tail$ if (1) $j \leq i_k$; or (2) there exists some $1 \leq r \leq k$ such that $c_{i_r,j} = 0$. Moreover, for v's parent node $v' = \{s_{i_1}, \ldots, s_{i_{k-1}}\}$, v's Tail is a subset of that of v'.*

Proof The first claim of the lemma follows the definition of set enumeration tree. If $j \leq i_k$, s_j cannot appear in any node of the subtree of v. Furthermore, if $c_{i_r,j} = 0$, then the gene is not coherent on c_j and $c_{i_r,j}$.

To show the second claim, we only need to note that v contains a superset of samples in v'. If the gene is coherent on every sample in v and a sample s that is not in the tail of v, it must also be coherent on every sample in v' and s. □

Clearly, we can prune any subtree that cannot lead to a coherent sample set of at least $\min_s$ samples.

Pruning-rule 3.1 (Pruning small sample sets) *At a node $v = \{s_{i_1}, \ldots, s_{i_k}\}$, the subtree of v can be pruned if $(k + |Tail|) < \min_s$.*

For example, for a set of l samples, even the complete set of sample combinations can be divided into l exclusive subsets as shown before, we only need to search the first $(l - \min_s + 1)$ subsets, since each of the last $(\min_s - 1)$ subsets contains less than $\min_s$ samples.

Moreover, if the samples at the current node and its *Tail* are subsumed by some maximal coherent sample set found so far, then the recursive search can also be pruned, since it cannot lead to any new maximal coherent sample set.

Pruning-rule 3.2 (Pruning subsumed sets) *At a node $v = \{s_{i_1}, \ldots, s_{i_k}\}$, if $\{s_{i_1}, \ldots, s_{i_k}\} \cup Tail$ is a subset of some maximal coherent sample set, then the subtree of the node can be pruned.*

Based on the previous lemma and pruning rules, the preprocessing algorithm is presented in Fig. 3. For the readers familiar with the techniques of depth-first mining of maximal/closed frequent patterns, the ideas of pruning here share the similar spirit with the pruning in frequent closed item set mining (e.g., [25]). However, one key difference is that the frequent pattern mining conducts counting on databases, and we do not need to scan the database for any counting once the triangle matrix $\{c_{i,j}\}$ is materialized.

4 The mining algorithms

A naïve method to find the maximal coherent gene clusters is to test every possible combination of genes and samples thoroughly. After all the coherent gene clusters

```
Input: the GST data set, the coherence threshold $\delta$, and the minimum sample size
    threshold $min_s$;
Output: the maximal coherent sample sets for each gene;
Method:
    for each gene $g_k$ do
        generate matrix $c_{i,j}$ for $g_k$;
        for $i = 1$ to $(l - min_s + 1)$ call search-clique($\{s_i\}, \{s_{i+1}, \ldots, s_l\}$); end-for
    end-for

Procedure: search-clique($head, tail$) // $head$ records the samples in the current node
    suppose $s_i$ is the last sample in $head$, remove samples $s_j$ from $tail$ such that $c_{i,j} = 0$;
    // Lemma 3.1
    if ($|head \cup tail| < min_s$) // Pruning 3.1
        or ($head \cup tail \subset S$) s.t. $S$ is a maximal clique // Pruning 3.2
    then return;
    if $tail = \emptyset$ then output a maximal clique;
    else do
            let $j = \min\{k|s_k \in tail\}$, $tail = tail - \{s_j\}$; call search-clique($head \cup \{s_j\}, tail$);
    until $tail = \emptyset$;
    return;
```

Fig. 3 Preprocessing: computing maximal coherent sample sets

are found, we can identify and report the maximal ones. The naïve method is very costly and thus infeasible for real data sets. For example, suppose we have 1000 genes and 20 samples. The naïve method may have to search up to $(2^{1000} - 1) \times (2^{20} - 1 - 20) \approx 1.12 \times 10^{307}$ combinations!

How can we search the huge space efficiently and prune unpromising subspaces sharply? When computing the maximal coherent sample sets (Fig. 3), we systematically enumerate combinations of samples in a recursive depth-first search and develop techniques to prune unpromising subspaces aggressively. Stimulated by the similar spirit, here we can also systematically enumerate the combinations of genes and samples and prune the unfruitful combinations.

Basically, we have two options. On the one hand, we can enumerate all combinations of samples systematically. Then, for each subset of samples, we can find the maximal subsets of genes that form coherent gene clusters on the samples and check whether the clusters are maximal. This method is called the *Sample–Gene Search*. On the other hand, we can let the gene enumeration go first. For each subset of genes, we find the maximal subsets of samples that form coherent gene clusters with the genes and check whether the clusters are maximal. The method is called the *Gene–Sample Search*.

The frameworks of the *Sample–Gene Search* and the *Gene–Sample Search* are shown in Fig. 4. Proper pruning techniques should be developed to prune the unpromising combinations and search branches as early as possible.

```
The Sample-Gene Search
    depth-first enumerate subsets of samples
    for each subset of samples $S$ do
        find the maximal subsets of genes $G$ s.t. $G \times S$ is a coherent gene cluster;
        test whether $(G \times S)$ is a maximal coherent gene cluster;
    end-for

The Gene-Sample Search
    depth-first enumerate subsets of genes
    for each subset of genes $G$ do
        find the maximal subsets of samples $S$ s.t. $G \times S$ is a coherent gene cluster;
        test whether $(G \times S)$ is a maximal coherent gene cluster;
    end-for
```

Fig. 4 The frameworks of the *Sample–Gene Search* and the *Gene–Sample Search*

A first look at Fig. 4 may suggest that the two methods are symmetric. However, since genes and samples are not symmetric in the problem, the technical details are in fact substantially different. We are mining coherent gene clusters on samples. As long as the genes coherently respond on the same subset of samples, they belong to the same cluster. However, the expression patterns of different genes in the same cluster on one sample can be very different.

4.1 Sample–Gene Search

In the *Sample*–Gene Search, we need to address the following issues.

- As we enumerate the combinations of samples systematically, *for each subset of samples*, *how can we find the maximal sets of genes such that the genes are coherent on the samples*?
- *During the sample set enumeration*, *which sample sets can be pruned*?
- Similar to the situation in Pruning rule 3.2 *can we identify and prune the searches that cannot lead to any potential maximal coherent clusters*?
- *How can we determine whether a coherent gene cluster is subsumed by the others*?

We answer the aforementioned questions in this section.

4.1.1 Maximal coherent gene sets for sample sets

For each combination of samples S, we need to compute the *maximal coherent gene set* G_S such that the genes in G_S are coherent on S and no proper superset $G' \supset G_S$ also has this property.

Clearly, for a gene g, if there exists a maximal coherent sample set S_g such that $S \subseteq S_g$, then $g \in G_S$. In other words, G_S can be derived by one scan of the

Gene	Maximal coherent sample sets
g_1	$\{s_1, s_2, s_3, s_4, s_5\}$
g_2	$\{s_1, s_2, s_4\}, \{s_1, s_5\}$
g_3	$\{s_1, s_2, s_3, s_4, s_5\}$
g_4	$\{s_1, s_2, s_3\}, \{s_5, s_6\}$
g_5	$\{s_1, s_5, s_6\}$

(a)

Sample	The inverted list
s_1	$\{g_1.b_1, g_2.b_1, g_2.b_2, g_3.b_1, g_4.b_1, g_5.b_1\}$
s_2	$\{g_1.b_1, g_2.b_1, g_3.b_1, g_4.b_1\}$
s_3	$\{g_1.b_1, g_3.b_1, g_4.b_1\}$
s_4	$\{g_1.b_1, g_2.b_1, g_3.b_1\}$
s_5	$\{g_1.b_1, g_2.b_2, g_3.b_1, g_4.b_2, g_5.b_1\}$
s_6	$\{g_4.b_2, g_5.b_1\}$

(b)

Fig. 5 The maximal coherent sample sets and the inverted lists: **a** The maximal coherent sample sets for genes. **b** The inverted lists for samples.

maximal coherent sample sets of all genes. If a maximal coherent sample set is a superset of S, then the corresponding gene g is inserted into G_S.

It is expensive to scan the complete list of maximal coherent sample sets of all genes once for every combination of samples. An efficient solution is to use an *inverted list*.

Let us illustrate the idea using an example. Suppose we have 5 genes and 6 samples. The maximal coherent sample sets for each gene are listed in Fig. 5a.

We label each maximal coherent sample set by the gene g_k, and the set-id, b_j, in the gene. For example, gene g_2 has two maximal coherent sample sets, $g_2.b_1 = \{s_1, s_2, s_4\}$ and $g_2.b_2 = \{s_1, s_5\}$.

For each sample s, we make up the *inverted list* L_s as the list of all maximal coherent sample sets containing s, as shown in Fig. 5b.

Now, when we want to compute the maximal coherent gene sets for a subset of samples, say $\{s_1, s_2, s_3\}$, we do not need to search the complete list in Fig. 5a. Instead, we only need to get the intersection of the inverted lists of the samples s_1, s_2, and s_3, which is $\{g_1.b_1, g_3.b_1, g_4.b_1\}$. By this intersection, we know that $\{g_1, g_3, g_4\}$ is the maximal coherent gene set.

4.1.2 Pruning irrelevant samples

For a combination of samples $S = \{s_{i_1}, \ldots, s_{i_k}\}$, where $i_1 < \cdots < i_k$, let S_{tail} be the set of samples that can be used to extend S to a larger set $S' \subset S \cup S_{\text{tail}}$ such that there are at least $\min_g$ genes coherent on S'. Clearly, a sample $s_j \notin S_{\text{tail}}$

if $j \leq i_k$. Moreover, if the maximal coherent gene set of $S \cup \{s_j\}$ contains less than $\min_g$ genes, then $s_j \notin S_{\text{tail}}$, either. This is in the similar spirit of Lemma 3.1. For example, in our running example (Fig. 5), sample s_6 cannot be used to extend sample set $S = \{s_2\}$, since there is no gene coherent on both s_2 and s_6.

Moreover, similar to Pruning rule 3.1, if $|S| + |S_{\text{tail}}| < \min_s$, then S cannot lead to any coherent gene cluster having $\min_s$ or more samples, and thus can be pruned.

4.1.3 Pruning unpromising coherent gene clusters

Similar to the situation in Pruning rule 3.2, we can prune the unpromising combinations that cannot lead to any new maximal coherent gene cluster. Here, two pruning techniques can be applied.

In our running example (Fig. 5), suppose we find the maximal coherent gene cluster $(\{g_1, g_3\} \times \{s_1, s_2, s_3, s_5\})$ before we search sample set $S = \{s_1, s_3\}$. For $S = \{s_1, s_3\}$, $S_{\text{tail}} = \{s_5\}$ and $G_{S \cup S_{\text{tail}}} = \{g_1, g_3\}$. That is, both $S \cup S_{\text{tail}}$ and $G_{S \cup S_{\text{tail}}}$ are subsumed by the maximal coherent gene cluster. The recursive search of S cannot lead to any maximal coherent gene cluster and thus can be pruned.

In general, suppose we are searching a sample combination S'. If there exists a maximal coherent gene cluster $(G \times S)$ found before such that $S' \cup S'_{\text{tail}} \subseteq S$ and $G_{S' \cup S'_{\text{tail}}} \subseteq G$, then any recursive search from S' results in a coherent gene cluster subsumed by $(G \times S)$, and thus can be pruned.

Moreover, if there exists a maximal coherent gene cluster $(G \times S)$ found before such that $S' \subseteq S$ and every maximal coherent sample set containing S' also contains S, then the recursive search of S' cannot lead to any maximal coherent gene cluster either, and thus can be pruned. For example, suppose we search the sample set $S' = \{s_2\}$ after we find the maximal coherent gene cluster $(\{g_1, g_2, g_3, g_4\} \times \{s_1, s_2\})$ in our running example (Fig. 5). From Fig. 5a, we can see that every maximal coherent sample set containing s_2 also contains s_1. In other words, there exists no maximal coherent gene cluster containing s_2 but no s_1. Thus, the search of S' can be pruned.

4.1.4 Determining maximal coherent gene clusters

When we search a combination of samples S, we need to check whether $G_S \times S$ is a maximal coherent gene cluster. We examine those maximal coherent gene clusters $(G' \times S')$ such that $S' \supset S$. Clearly, since we conduct depth-first search in the set enumeration tree, such maximal coherent gene clusters should be reported either before S is searched, or in the subtree rooted at S.

4.1.5 The Sample–Gene Search *algorithm*

Based on the previous discussion, we have the *Sample–Gene Search* algorithm in Fig. 6.

```
Input: the maximal coherent samples sets for genes // from the algorithm in Figure 3
Output: the complete set of coherent gene clusters
Method:
    generate the inverted list for samples as described in Section 4.1.1;
    for $i = 1$ to $(|S\text{-}Set| - min_s)$ do
        let $S = \{s_i\}$ and $S_{tail} = \{s_{i+1}, \ldots, s_{|S\text{-}Set|}\}$; call recursive-search$(S, S_{tail})$;
    end-for

Procedure: recursive-search$(S, S_{tail})$
    remove irrelevant samples from $S_{tail}$ as described in Section 4.1.2;
    if $(|S| + |S_{tail}| < min_s)$ then return;
    derive the intersection of inverted lists for samples in $S$ as described in Section 4.1.1;
    if $S$ can be pruned by the criteria in Section 4.1.3 then return;
    while $S_{tail} \neq \emptyset$ do
        let $i = \min\{j | s_j \in S_{tail}\}$; let $tail = tail - \{s_i\}$; call recursive-search$(S \cup \{s_i\}, S_{tail})$;
    end-while
    derive the maximal coherent gene set $G_S$;
    output $(G_S \times S)$ as a maximal coherent gene cluster if it is not subsumed by any maximal
    coherent gene cluster found before
End
```

Fig. 6 The *Sample–Gene Search* algorithm

4.2 Gene–Sample Search

Although the computational details of *Sample–Gene Search* and *Gene–Sample Search* are substantially different, the overall structures of these two algorithms are symmetric one to the other. In Gene–Sample Search, we enumerate the combinations of genes systematically. For each combination of genes, we compute the maximal sets of samples that the genes are coherent on. Many pruning techniques in *Sample–Gene Search* have the symmetric versions in *Gene–Sample Search*. Limited by space, we omit the details here. Instead, we only focus on the differences between the two approaches.

4.2.1 Determining coherent gene clusters

The concept of coherent sample sets for a gene can be generalized for a set of genes. Given a set of genes G, a *maximal coherent sample set* with respect to G is a set of samples S_G such that (1) genes in G are coherent on S_G; and (2) there exists no $S' \supset S_G$ that samples in G are also coherent on S'. Please note that there can be more than one maximal coherent sample set for a given set of genes.

How can we compute the maximal coherent sample sets efficiently? Interestingly, S_G can be computed by some simple intersection operations. For example, suppose $S_{\{g_1\}} = \{(s_1, s_2, s_3)\}$ and $S_{\{g_2\}} = \{(s_2, s_3, s_4), (s_5, s_7)\}$. Then, $\{s_2, s_3\}$ is

Function *find-max-coherent-sample-sets*

Input: gene sets G_1 and G_2, sets of maximal coherent sample sets S_{G_1} and S_{G_2};

Output: the maximal coherent sample sets w.r.t. $G_1 \cup G_2$

Method:

```
let $S_{G_1 \cup G_2} = \emptyset$;
for each maximal coherent sample set $S_i \in S_{G_1}$ do
    for each maximal coherent sample set $S_j \in S_{G_2}$ do
        let $S_k = S_i \cap S_j$;
        if $|S_k| \geq min_s$ then insert $S_k$ into $S_{G_1 \cup G_2}$;
    end-for
end-for
for each $S_k \in S_{G_1 \cup G_2}$ do
    if $S_k$ is a proper subset of $S' \in S_{G_1 \cup G_2}$
        then $S_{G_1 \cup G_2} = S_{G_1 \cup G_2} - \{S_k\}$;
end-for
output $S_{G_1 \cup G_2}$;
```

Fig. 7 Computing maximal coherent sample set for a set of genes

the only maximal sample set that both g_1 and g_2 are coherent on. That is, $S_{\{g_1,g_2\}} = \{(s_2, s_3)\}$. In other words, we can derive $S_{\{g_1,g_2\}}$ from $\{s_1, s_2, s_3\} \cap \{s_2, s_3, s_4\}$ and $\{s_1, s_2, s_3\} \cap \{s_5, s_7\}$.

In general, if gene set $G = G_1 \cup G_2$, then S_G can be derived from S_{G_1} and S_{G_2} by the function *find-max-coherent-sample-sets* in Fig. 7.

4.2.2 Pruning irrelevant genes and unpromising coherent gene clusters

Similar to the idea in Sect. 4.1.2, we can prune genes that cannot be used to extend the current combination of genes. For a given set of genes $G = \{g_{i_1}, \ldots, g_{i_k}\}$, where $1 \leq i_i < \cdots < i_k$, a gene g_j cannot be used to extend G to a larger set of genes if $j \leq i_k$ or none of the maximal coherent sample set with respect to $G \cup \{g_j\}$ has at least $\min_s$ samples. Moreover, let G_{tail} be the set of genes that can be used to extend G. If $|G| + |G_{\text{tail}}| < \min_g$, then G should be pruned.

Based on the same idea in Sect. 4.1.3, we can use the maximal coherent gene clusters to prune the unpromising coherent gene clusters. Suppose we are searching a gene combination G_1. Let S_1 be one maximal coherent sample set with respect to G_1, i.e., $S_1 \in S_{G_1}$. If there exists a maximal coherent gene cluster $(G \times S)$ such that $S_1 \subseteq S$ and $G_1 \subseteq G$, then S_1 should be removed from the list of the maximal coherent sample sets S_{G_1}, since it cannot lead to any maximal coherent gene cluster. Moreover, if S_{G_1} becomes empty after the pruning, then G_1 should be pruned, since any recursive search from G_1 cannot lead to any maximal coherent gene cluster.

4.2.3 Merging coherent genes in the tail list

In our running example, the maximal coherent sample sets with respect to gene g_1 and g_3 are identical. Then, for any coherent gene cluster $(G \times S)$ such that $g_1 \in G$, $(G \cup \{g_3\} \times S)$ must also be a coherent gene cluster. Thus, we can search $\{g_1, g_e\}$ in a shoot.

In general, we have the following result.

Lemma 4.1 *When search a combination of genes G, if there exist genes $\{g_{j_1}, \ldots, g_{j_k}\} \subseteq G_{\text{tail}}$ such that they are coherent on every maximal coherent sample set of G, then there exists no maximal coherent gene cluster containing G but no $\{g_{j_1}, \ldots, g_{j_k}\}$.*

Proof Suppose we have such a maximal coherent gene cluster C containing G but no $\{g_{j_1}, \ldots, g_{j_k}\}$. Let the set of samples in C be S. It is easy to see that $G \cup \{g_{j_1}, \ldots, g_{j_k}\}$ are also coherent on sample set S. That is, $(G \cup \{g_{j_1}, \ldots, g_{j_k}\} \times S)$ is a coherent gene cluster. Since $G \subset (G \cup \{g_{j_1}, \ldots, g_{j_k}\} \times S)$, C cannot be a maximal coherent gene cluster. A contradiction. $\square$

Based on Lemma 4.1, we can immediately merge genes $\{g_{j_1}, \ldots, g_{j_k}\}$ to G at the current node, and thus shrink the number of recursions. The computation time is saved as well, since we only need to check the coherent gene clusters, prune the irrelevant genes or unpromising gene clusters for all these genes in one shoot.

In our experiments on real data sets, we observe many genes can be merged by Lemma 4.1. The real-world *GST* microarray data sets are typically sparse and genes are coherent on a quite small number of sample sets. As a consequence, the performance of *Gene–Sample Search* can be improved substantially by this optimization.

One may ask, "*Do we have a symmetric pruning for* Sample–Gene Search?" We can apply the similar optimization technique for sample–gene. That is, a sample s_j is merged into current combination of samples S as long as the inverted list of S is a subset of that of s_j. However, it is rare in practice that the maximal coherent gene sets with respect to two different samples sets S_1 and S_2 are identical. Therefore, there are few cases when this rule can be applied in real applications.

4.2.4 The Gene–Sample Search Algorithm

Based on the previous discussion, we have the *Gene-Sample Search* algorithm as shown in Fig. 8.

5 Experimental results

We implemented and tested our approaches on both a real *GST* microarray data set and synthetic data sets. The system is implemented in Java. The tests are conducted on a Sun Ultra 10 work station with a 440 MHz CPU and 256 MB main memory.

```
Input and output: same as the Sample-Gene Search algorithm (Figure 6)
Method:
    for $i = 1$ to $(|G\text{-}Set| - min_g)$ do
        let $G = \{g_i\}$ and $G_{tail} = \{g_{i+1}, \ldots, g_{|G\text{-}Set|}\}$;
        call recursive-search$(G, G_{tail})$;
    end-for

Procedure: recursive-search$(G, G_{tail})$
    remove irrelevant genes from $G_{tail}$ as described in Section 4.2.2;
    if $(|G| + |G_{tail}| < min_g)$ then return;
    merge coherent genes in $G_{tail}$ as described in Section 4.2.3;
    for each maximal coherent sample set $S_i \in S_G$ do
        if $S_i$ can be pruned by the second criteria in Section 4.2.2 then remove $S_i$ from $S_G$;
    end-for
    if $(S_G = \emptyset)$ then return;
    while $G_{tail} \neq \emptyset$ do
        let $i = \min\{j | g_j \in G_{tail}\}$;
        let $G_{tail} = G_{tail} - \{g_i\}$;
        call recursive-search$(G \cup \{g_i\}, G_{tail})$;
    end-while
    for each sample set $S_i$ in $S_G$ do
        output $(G \times S_i)$ as a maximal coherent gene cluster if it is not subsumed by any
        maximal coherent gene cluster found before
    end-for
End
```

Fig. 8 The *Gene–Sample Search* algorithm

5.1 The data sets

5.1.1 The real data set

We use the real gene–sample–time microarray data set reported in [39]. It consists of the microarray measurements of 4324 genes in 13 multiple sclerosis (MS) patients before and at 1, 2, 4, 8 h, 1, 2, 5, 7 days, and 3 months after IFN-β treatments. MS patients show heterogeneous responses to IFN-β treatments. For example, the patients with relapsing MS respond better to IFN-β treatments than the patients with progressive disease do. However, relapsing MS patients also exhibit considerable inter-individual heterogeneity in their clinical responses to IFN-β therapies. So far, the effects of IFN-β treatment at the genomic level in humans are poorly understood. Researchers are interested in distinguishing the heterogeneous clinical response to IFN-β therapy among the patients. Moreover, characterized gene expression dynamics correlated to the heterogeneous responses potentially help in exploring the causing mechanisms at the molecular level.

5.1.2 Synthetic data

We observe that the preprocessing, i.e., mining the maximal coherent sample sets for each individual gene, is relatively fast. The major bottleneck in mining coherent gene clusters is in the latter part. Therefore, instead of generating synthetic *GST* data sets, we simulate the table of maximal coherent sample sets for genes such as in Fig. 5a. Initially, an empty table is created. Then, a certain number of coherent gene clusters ($G \times S$) are randomly generated. For each $g \in G$, S is inserted into the table as one maximal coherent sample set with respect to g. In addition to the size of the synthetic data set, i.e., the total number of genes in *G-Set* and the number of samples in *S-Set*, the synthetic data generator takes the following parameters: (1) k, the number of coherent gene clusters in the data set; (2) $\max_{\text{gene}}$ and $\min_{\text{gene}}$, the maximal and minimal numbers of genes in a coherent gene cluster, respectively; and (3) $\max_{\text{sample}}$ and $\min_{\text{sample}}$, the maximal and minimal numbers of samples in a cluster, respectively.

We generate the data sets by setting $\min_{\text{gene}} = 10$ and $\min_{\text{sample}} = 5$. $\max_{\text{sample}}$ is set to the same value of $|S\text{-}Set|$, and $\max_{\text{gene}}$ is set to 1000. In practice, only a small number of genes are correlated with a phenotype [13]. When the size of the data set grows, we expect to see more coherent gene clusters. To simulate the situation, we set k to $(|G\text{-}Set| \cdot |S\text{-}Set|)/(3000)$.

5.2 Results on the MS microarray data

The original MS microarray data contain outliers, missing values, and experimental bias. We first choose a global normalization strategy to filter out the outliers [39], estimate the missing values using KNN impute [36], and standardize the data set such that the gene expression levels of each patient at each time point have a mean of zero and a standard deviation of one. We then apply the principle component analysis (PCA) [18] to remove the systematic variation caused by experimental bias. Moreover, we filter the genes which exhibit “flat patterns” across the whole set of samples. That is, a gene will be removed from the data set if its expression level do not change significantly (e.g., 10%) during the whole time series on any sample. The rationale is that these genes are probably irrelevant to IFN-β response under investigation.

After the data pre-processing steps mentioned earlier, we apply our algorithm to the data with $\min_s = 3$, $\min_g = 50$, and $\delta = 0.8$. From the mining results, we systematically select top 25 high-quality coherent gene clusters using the method in [20]. Please note that for each cluster $C = (G \times S)$, the genes showing “flat patterns” across S are removed. The clusters are reported at http://www.cse.buffalo.edu/DBGROUP/bioinformatics/GST.

In the following, we will first test the statistical significance of the reported clusters. To better understand the biological functions of the genes in clusters, we then investigate the enrichment of clusters in terms of Gene Ontology [6]. Finally, we discuss the correlation between the identified clusters and the known aspects of IFN-β pharmacology.

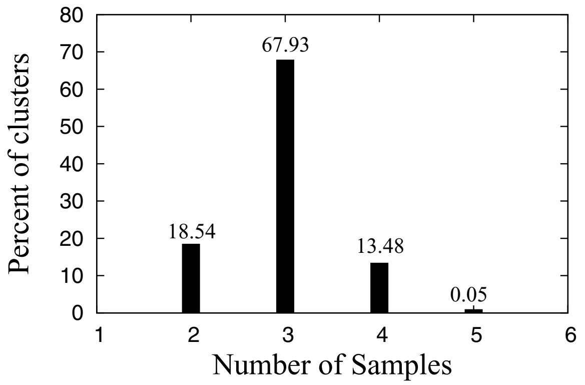


Fig. 9 The percentage of clusters with respect to the number of samples in clusters

5.2.1 Statistical significance of clusters

To estimate the statistical significance of clusters, we generate 100 permutated data sets from the original data. Since clusters containing only one gene or one sample are trivial (Proposition 2.1), we set $\min_s = 2$, $\min_g = 2$, and $\delta = 0.8$, and apply our algorithm to each permutated data set. Suppose $\mathcal{C}$ records all the clusters generated from the 100 permutated data sets. We calculate the histograms of the number of samples and genes in clusters from $\mathcal{C}$. Figure 9 illustrates the distribution of the percentage of clusters in $\mathcal{C}$ with respect to the number of samples in clusters. For example, 18.54% of the clusters in $\mathcal{C}$ contains two samples. Please note no clusters have more than five samples.

Figure10 shows the distribution of the number of clusters in $\mathcal{C}$ with respect to the number of genes in clusters given a specific number of samples. For example, in Fig. 10a, we can see all the clusters containing only two samples have 160–260 genes, and the distribution is approximately Gaussian. However, when the number of samples in clusters increases, the number of genes in clusters decreases dramatically. For example, the number of genes in clusters ranges between 2 and 24 when $|S| = 3$ (Fig. 10b) and the range drops to 2–6 for $|S| = 4$ (Fig. 10c) and 2–3 for $|S| = 5$ (Fig. 10d). This means that by chance, fewer genes will form coherent patterns across a larger subset of samples.

Let c denote the random variable of cluster, and g and s be the number of genes and samples in clusters, respectively. The probability of c is a joint distribution $P(c) = P(g, s) = P(s) \cdot P(g|s)$. We define the p-Value of a given cluster $C = (G \times S)$ as

$$p\text{-}Value(C) = P(s \geq |S|) \cdot P(g \geq |G| \mid s = |S|). \quad (2)$$

Clearly, the meaning of $p\text{-}Value(C)$ is how likely the cluster C with at least $|G|$ genes and $|S|$ samples is formed by chance. The smaller the *p-Value*, the higher the statistical significance. In our case, we set $\min_s = 3$ and $\min_g = 50$, according to Fig. 10b, the $p\text{-}Value$ of a cluster with at least three samples and 50 genes is approximately zero. Therefore, the clusters reported by our algorithm are statistically significant.

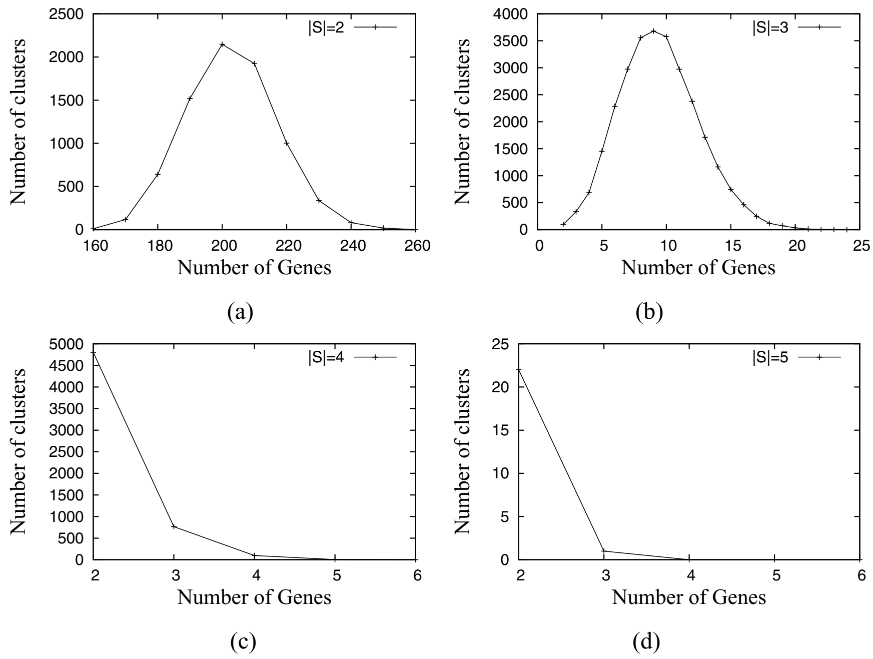


Fig. 10 The number of clusters with respect to the number of genes in clusters when: **a** $|S| = 2$, **b** $|S| = 3$, **c** $|S| = 4$, and **d** $|S| = 5$

5.2.2 Gene annotation of clusters

We use the Gene Ontology (GO) [6] to annotate the genes in clusters. GO is organized as a hierarchical direct acyclic graph (DAG). There are three major parts of GO, which form three independent ontologies describing the attributes of molecular function, biological process, and cellular component for a gene product. GO is a rapid growing collection with more than 11,000 terms so far. A significant portion of genes (3501 out of 4324) in our MS data set has been annotated by GO terms.

Suppose the total number of genes in the data set associated with a GO term T is M. If we randomly draw p genes from the complete gene set $G\text{-}Set$, the probability that q of the selected p genes are associated with T can be approximated by the hypergeometric distribution [35]

$$P(q \mid |G\text{-}Set|, M, p) = \frac{\binom{M}{q}\binom{|G\text{-}Set|-M}{p-q}}{\binom{|G\text{-}Set|}{p}},$$

and the p-Value of a cluster $C = (G \times S)$ with q genes in term T is

$$p\text{-}Value(C, T) = \sum_{i=q}^{Min(M,|G|)} \frac{\binom{M}{i}\binom{|G\text{-}Set|-M}{|G|-i}}{\binom{|G\text{-}Set|}{|G|}}. \tag{3}$$

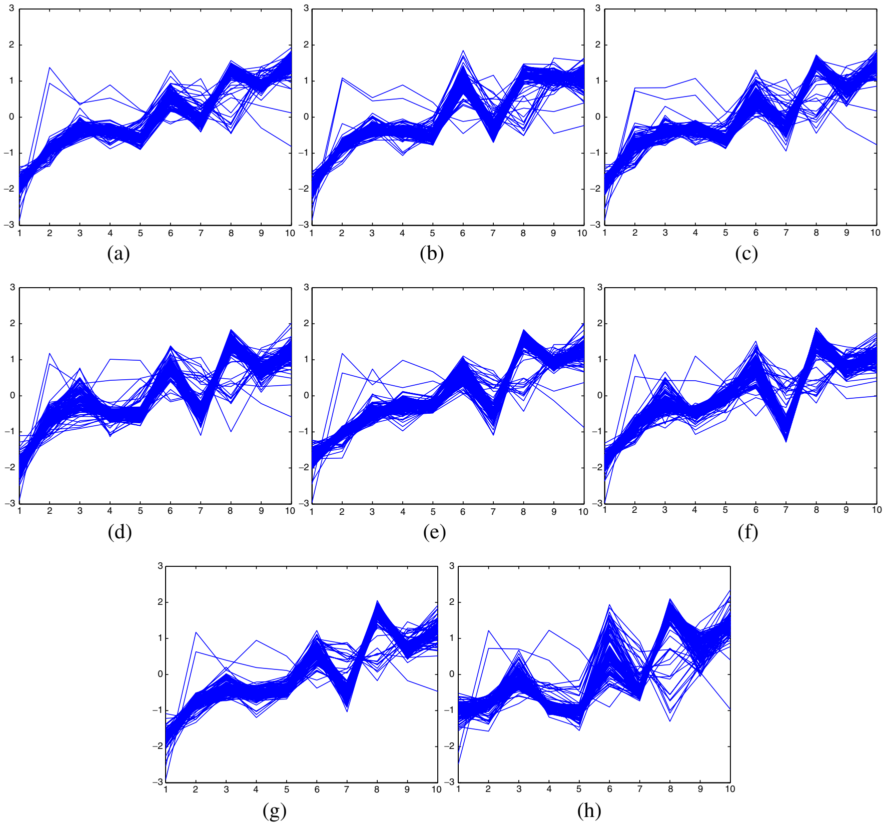


Fig. 11 The temporal expression patterns of genes on individual samples in Cluster 2

Among the 25 clusters reported by our algorithm, we find 15 clusters are significantly enriched in one of the second-level GO terms regarding biological process: 6 clusters are strongly correlated with "response to external stimuli"; 9 clusters are associated with "cell communication." We also examine the remaining 10 clusters that are not enriched in any second-level GO terms, we find that some clusters contain a large percent of novel genes which have not been annotated. For example, only 19 out of 107 genes in Cluster 2 are annotated. However, the 107 genes in Cluster 2 exhibit coherent patterns across a wide range of eight samples (Fig. 11). This cluster may provide valuable information for function prediction of those unknown genes. Moreover, among the 10 clusters that are not enriched in any second-level Go terms, we also find some clusters are enriched in the third-level GO terms. For example, Cluster 11 is enriched in "transport" (p-Value $= 0.0184$) and Cluster 24 is enriched in "oxygen and reactive oxygen species metabolism" (p-Value $= 0.00648$). The expression patterns as well as the detailed gene annotation for all the clusters are reported at http://www.cse.buffalo.edu/DBGROUP/bioinformatics/GST. Since IFN-β has anti-viral, anti-proliferative and immunomodulatory effects, the clustering results are biological feasible.

5.2.3 Interesting observations on clusters

To assess the usefulness of the *coherent gene cluster* model, we initially focus on "benchmark" mRNAs that are known to be IFN-β responsive. For example, 80 out of the 659 genes in Cluster 1 are associated with the response to external stimuli (p-Value 0.0262). We find this cluster contains several genes known to be induced by IFN-β treatment [8, 32]. A few examples of such gene of particular interest are signal transducer and activator of transcription 1 (STAT-1, Hs. 479043), which is associated with the IFN-β receptor and forms part of the transcription factor complex that binds the interferon-responsive promoter sequence; guanylate binding protein-1 (Hs. 62661), myxovirus resistance protein-2 (Hs. 926) and double stranded RNA-dependent protein kinase (PKR, Hs. 131431), all of which are involved in the anti-viral response.

As another example, 46 out of 174 genes in cluster 12 are involved in cell communication (p-Value 0.00749) and 13 are known to associate with cell–cell signaling (p-Value 0.00487). Those genes, together with the other genes in the same cluster that are unknown or poorly understood, may serve as switches in the genetic network and hence play an essential role in the biological processes. Thus, studying the time series of the genes in the coherent gene clusters may greatly help people understand the regulatory mechanisms behind the response to IFN-β treatment.

Interestingly, we find a only few classes of temporal patterns within each coherent gene cluster (e.g., only one major pattern in Cluster 2 (Fig. 11)), even though our computational model allows genes with diverse temporal profiles to be present in the same cluster as long as their profiles are similar across the subset of patients. This suggests that there are groups of genes with similar temporal profiles that are activated in patients. The emergence of significant p-Values in most clusters suggests that the genes groups are functionally coordinated. Critical analysis of the evidence for functional coordination is currently underway in our laboratory. Biologically, this finding is potentially very valuable because the promoter sequences of these gene groups can be analyzed to determine whether they share common regulatory pathways. Further analysis that include clinical information could potentially reveal whether the subsets of patients differ in their clinical phenotypes.

5.3 Effects of the parameters

The maximal coherent gene cluster is defined with respect to three parameters, i.e., the minimum number of genes $\min_g$, the minimum number of samples $\min_s$, and the coherence threshold δ. We test the effect of the parameters on the real *GST* data set. Figure 12a shows the number of coherent gene clusters when $\min_g$ varies from 5 to 100, $\min_s = 3$ and $\delta = 0.8$. Clearly, the number of coherent gene clusters in the data set decreases when $\min_g$ increases. The result concurs the intuition: with a lower $\min_g$ value, we can catch more clusters with more or less genes. As a matter of fact, with fixed $\min_s$ and δ, let $\mathcal{B}_i$ be the complete set of coherent blocks when $\min_g = i$. Then, we can show $\mathcal{B}_1 \supseteq \cdots \mathcal{B}_i \supseteq \cdots \mathcal{B}_n$.

Figure 12b shows the number of coherent gene clusters with respect to various $\min_s$ when $\min_g$ and δ are fixed to 10 and 0.8, respectively. This result can be

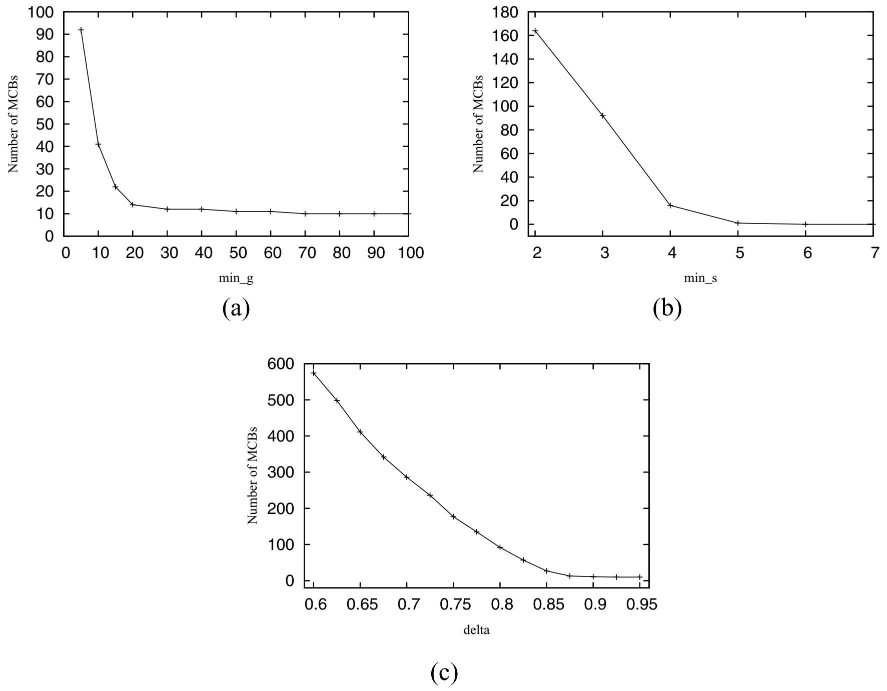


Fig. 12 The effects of the parameters on the number of clusters. (**a**) Number of clusters vs. $\min_g$ $\min_s = 3$, $\delta = 0.8$. (**b**) Number of clusters vs. $\min_s$ $\min_g = 10$, $\delta = 0.8$. (**c**) Number of clusters vs. δ $\min_g = 10$, $\min_s = 3$

explained in a way similar to the situation of $\min_g$. Figure 12c shows the effect of δ on the number of coherent gene clusters in the data set, with $\min_g = 10$ and $\min_s = 3$. When we lower the coherence threshold, more combinations of samples are "coherent" by chance with respect to a minimum of $\min_g$ genes.

Interestingly, the three curves in Fig. 12 share similar trends. That is, when the value of the parameter (represented by the X-axis) increases, the number of coherent gene clusters (represented by the Y-axis) goes down. The curve drops sharply until a "knot" is met, then the curve goes stably to the right. For example, we can see the "knots" of $\min_g = 20$ in Fig. 12a, $\min_s = 5$ in Fig. 12b and $\delta = 0.85$ in Fig. 12c. We examine the "knot" ($\min_\mathrm{s} = 5$) in Fig. 12a with the histogram in Fig. 9, and find that $P(s \geq 5) \simeq 0$. We also check the "knot" ($\min_g = 20$) with the histogram in Fig. 10b, similarly, we find $P(g \geq 20 \mid s = 3) \simeq 0$. The consistency between the "knots" in Fig. 12 and the distribution of clusters in Figs. 9 and 10 suggests that the "knots" indicate that there exist stable and significant coherent gene clusters in the real data set.

In practice, users can choose the threshold $\min_s$ from the values $\{s \mid P(s) \neq 0\}$ based on the histogram in Fig. 9. Users can then set up a confidence level and determine the threshold $\min_g$ according to the histogram in Fig. 10 and the $p\text{-}Value$ defined by Eq. (2). Finally, users can tune the parameter δ based on Fig. 12c. A "knot" in the figure may suggest an appropriate δ value. As shown by our empirical study, such parameter settings often generate highly coherent and statistically significant clusters. For example, Fig. 11 shows the temporal patterns within a cluster with $\delta = 0.8$.

5.4 Scalability

We first test the efficiency of the preprocessing (algorithm in Fig. 3) on various random subsets (by sampling) of the real microarray data set. The size of the subsets varies from 500 to 4324 genes, and all the samples are included. For each size, we sampled 30 subsets and calculate the average runtime. Figure 13a illustrates the scalability for the preprocessing step. As we discussed in Sect. 3, the real *GST* microarray data sets are often sparse. With the efficient pruning techniques, the preprocessing algorithm is linearly scalable to the size of the data sets.

We then test the scalability of both *Gene–Sample Search* and *Sample–Gene Search* on synthetic data sets. We set $\min_s = 5$, $\min_g = 10$, and $\delta = 0.8$. We first fix the number of samples to 30, and report the runtime with respect to number of genes (Fig. 13b). We can see both approaches show an approximately linear scalability with respect to the number of genes. Figure 13c shows the scalability for both approaches under different sizes of sample sets (from 30 to 100), when the number of genes is fixed to 3000. We can see both approaches scale well with respect to the number of samples. Please note that *Sample–Gene Search* and

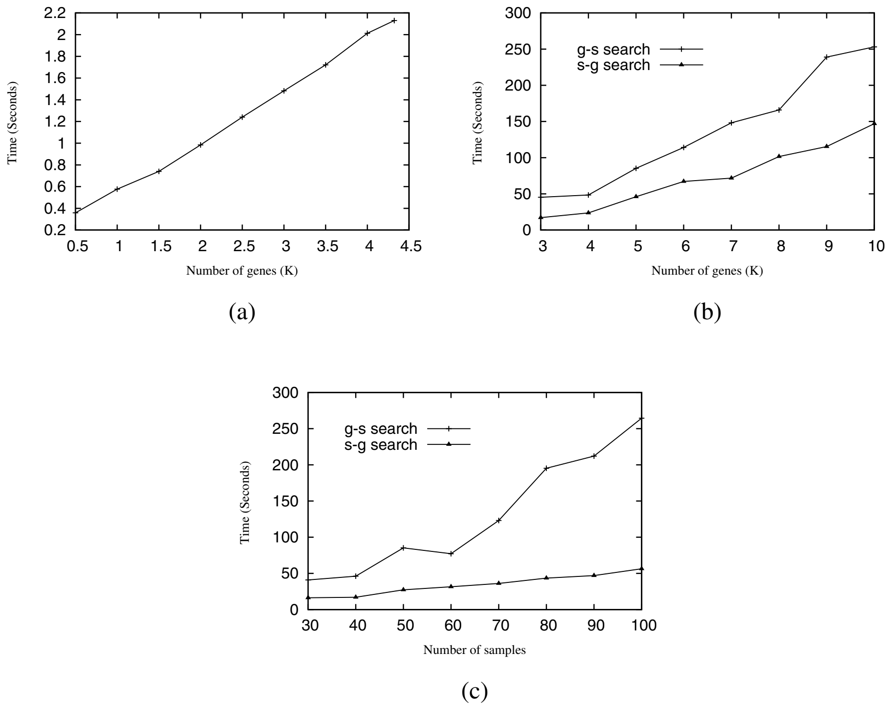
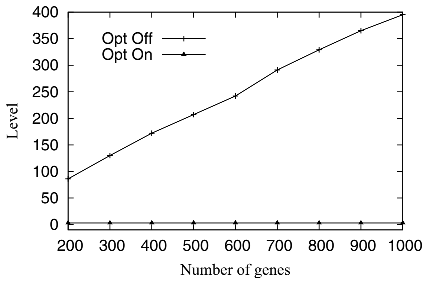


Fig. 13 Scalability on large data sets. (**a**) Preprocessing. (**b**) Scalability w.r.t. number of genes (number of samples: 30). (**c**) Scalability w.r.t. number of samples (number of genes: 3, 000).

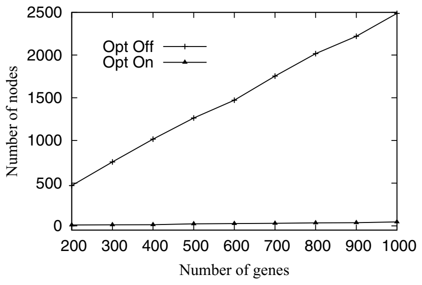
Gene–Sample Search always report the same set of clusters, since both approaches find the unique *complete set of maximal coherent gene clusters* in the data set. However, since the number of genes for a microarray data is typically by far larger than that of the samples, the enumeration of genes is much more expensive than the enumeration of samples. This explains why the *Sample–Gene Search* is faster than the *Gene–Sample Search*.

5.5 The effect of Lemma 4.1

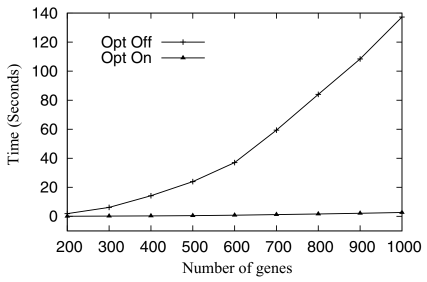
Lemma 4.1 can identify the genes g_i that can be merged into the current combination of genes, and thus can reduce the number of recursions in the mining. We use some samples of the real microarray data set (each subset contains 100–1000 genes and 12 patients) to compare the performance of the *Gene–Sample Search* with and without the optimization. The comparison is conducted in three aspects: (1) the maximal number of recursion levels in the *Gene–Sample Search*; (2) the number of gene combinations in the *Gene–Sample Search*; and (3) the runtime. Figure 14 shows the results. We can clearly see that (1) the maximal number of recursion levels can be reduced substantially (Fig. 14a); (2) with the optimization, the total number of gene combinations needed to be checked goes down sharply Fig. (14b); and (3) the runtime is much shorter when the optimization is applied Fig. (14c). The results strongly confirm that the optimization is effective for *Gene–Sample Search*.



(a)



(b)



(c)

Fig. 14 Effect of Lemma 4.1 for *Gene–Sample Search*. (**a**) Maximal number of recursion levels (**b**) Number of gene combinations searched (**c**) Runtime

We also apply the spirit of Lemma 4.1 on the *Sample–Gene Search*, and conduct similar tests. However, we can hardly see any significant improvement brought by the optimization. As we discussed in Sect. 4.2.3, due to the sparsity of the microarray data, many genes can be merged because they are coherent on the same sample set. However, few samples would be merged together since usually the maximal coherent gene sets with respect to two different sample sets are not identical.

6 Related work

This research is related to the previous work on clustering conventional gene–time and gene–sample microarray data, pattern-based clustering, and frequent item set mining.

As explained in Sect. 1, there have been two categories of conventional microarray data sets: gene–time data sets and gene–sample data sets. For gene–time microarray data, various algorithms (e.g., [2, 11, 19, 33, 35]) have focused on clustering the genes. That is, co-expressed genes are grouped based on their expression patterns during the time series. However, different approaches (e.g., [3, 5, 9, 34, 40]) have been proposed to partition the sample sets to find their macroscopic phenotypes as well as to detect informative genes which manifest the sample partition. However, all the cluster models in those previous studies are substantially different from our coherent gene clusters. As a consequence, those algorithms cannot be extended directly to solve our problem.

Recently, pattern-based clustering has been proposed and applied for mining microarray data. This study is directly stimulated by this category of research.

The general idea of pattern-based clustering is to identify groups of genes that follow similar patterns on subsets of samples. Different from the traditional clusters such as subspace clusters, the direct distance between the genes may not be short due to the shifting of the patterns. Moreover, the pattern-based clusters may have overlap: a gene can participate in different biological processes and thus more than one pattern-based cluster.

Cheng and Church [7] introduced the concept of *bicluster* to measure the coherence between genes and conditions (either time series or samples). For a given set of genes and a given set of conditions, a bicluster is a subset of genes coherent with a subset of conditions. Yang et al. [41] proposed a move-based algorithm to find biclusters more efficiently. Both algorithms [7, 41] adopt heuristic search approaches, and thus cannot guarantee to find the complete set of biclusters in the data set.

Wang et al. [38] proposed the model of pattern-based cluster. For a given subset of objects O and a subset of attributes A, pair (O, A) forms a pattern-based cluster if for any pair of objects $x, y \in O$, and any pair of attributes $a, b \in A$, the difference of change of values on attributes a and b between objects x and y is smaller than a threshold δ. In a recent study [26], Pei et al. addressed the redundancy due to the anti-monotonicity of pattern-based clusters. The concept of *maximal pattern-based clusters* was developed and an efficient algorithm, *MaPle*, was proposed to mine the complete set of maximal pattern-based clusters.

Liu and Wang [23] proposed another pattern-based clustering model called *order-preserving clustering* (*OP-cluster* for short). A group of genes form an

OP-cluster on a permutation of a subset of samples if each gene has increasing expression levels on the subset of samples.

We borrow some important ideas from previous studies on frequent item set mining. First, the framework of our approaches is similar in spirit to that of pattern-growth methods for frequent pattern mining. The idea of using set enumeration tree in frequent item set mining was first proposed in [4]. The pattern-growth framework for mining frequent item sets was presented in [14], and was extended to mining frequent closed item sets in [25]. Moreover, enumerating samples in gene microarray data sets was first proposed in [26, 38], and was stimulated by the fact that the number of samples is often 1–2 orders of magnitudes less than the number of genes in microarray data sets.

Second, the pruning techniques in our approaches share some interesting similarities with the methods of mining frequent closed item sets (e.g., [25, 43]). However, there are two essential differences between the frequent pattern mining methods and the approaches developed in this paper. First, the coherent gene clusters are inherently different from frequent item sets. Thus, the similarity between the two categories of methods is only at the level of spirit (e.g., set enumeration and pruning). The technical details are dramatically different. Second, several new techniques such as the inverted lists are adopted to tackle the particular microarray data.

7 Other interesting coherent clusters in GST data

In this paper, we focus on mining *coherent gene clusters* in GST data. That is, the set of genes in a cluster are constrained to exhibit coherent expression patterns across the set of samples in the cluster. Nevertheless, two other types of clusters, *coherent sample clusters* and *coherent gene–sample clusters*, may also be interesting.

Figure 15a illustrates the structure of a coherent sample cluster $(\{g_{i1}, g_{i2}, g_{i3}\} \times \{s_{j1}, s_{j2}, s_{j3}\})$. We can see that for each sample in the cluster, the expression patterns of genes are similar with each other. In other words, genes are *co-expressed* across the samples. In practice, co-expressed genes may belong to the same or similar function categories, and co-expression may indicate co-regulation as well. Although numerous studies have aimed at finding co-expressed genes from traditional gene–time data sets (e.g., [2, 11, 12, 15–17, 19, 27, 29–31, 33, 35, 42]), due to the high noise ratio of microarray data, a group of genes which consistently demonstrate co-expression across multiple samples are usually more reliable.

Figure 15b shows an example of coherent gene–sample cluster $(\{g_{i1}, g_{i2}\} \times \{s_{j1}, s_{j2}\})$. Each cell in the cluster carries the similar trend of expression levels during the time series. Clearly, a coherent gene–sample cluster is both a coherent gene cluster and a coherent sample cluster. Therefore, the biological meaning of the coherent gene–sample cluster is twofold: the samples in the cluster may correspond to some phenotype, while the genes in the cluster may not only correlate to the phenotype but also share similar functions.

In earlier discussion, the coherence among genes and/or samples is measured with respect to the whole time series. However, a specific cellular process may last for only a sub-interval within the whole time series of the microarray exper-

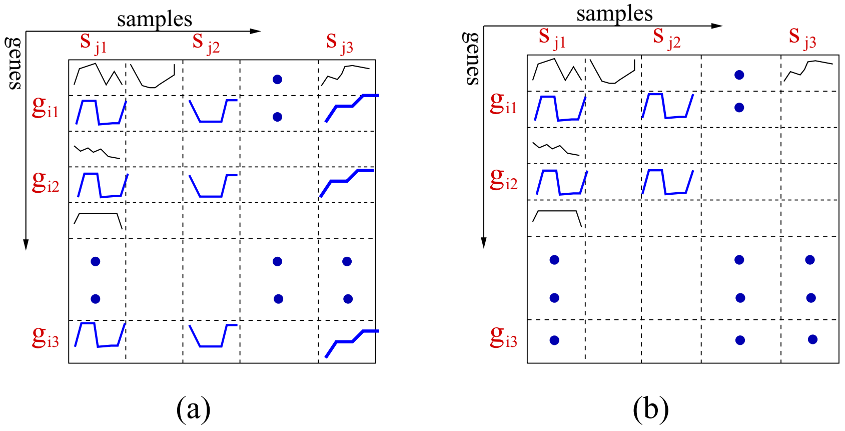


Fig. 15 Other types of coherent clusters. (**a**) Coherent sample cluster. (**b**) Coherent gene sample cluster

iment. In particular, a gene may participate in one cellular process together with a set of genes G_1 at time point t_1 but join another process with a different set of genes G_2 at the next time point t_2. In the future, we will find all the three types of coherent clusters on a subset of continuous time points. Moreover, we will develop effective indices on the coherent clusters to describe the dynamics of genes.

8 Conclusions

In this paper, we investigate a novel type of gene–sample–time microarray data sets and propose a new problem of mining coherent gene clusters from such data sets. We conduct a systematic study to develop two mining methods: the *Sample–Gene Search* and the *Gene–Sample Search*. Our extensive performance study on both a real microarray data set and synthetic data sets shows that there exist interesting and significant coherent gene clusters in the real data set, and both algorithms have good performance. Despite that both search methods return the unique complete set of *maximal coherent gene clusters*, the *Sample–Gene Search* is usually more efficient than the *Gene–Sample Search* since the number of genes in the microarray data is typically by far larger than the number of samples.

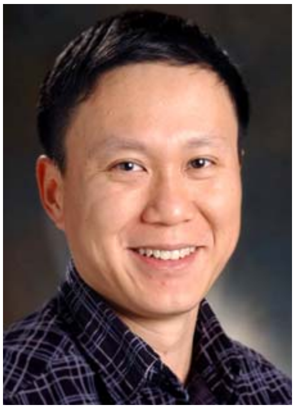
Acknowledgements This research is partly supported by NSF grants DBI-0234895 and IIS-0308001 and NIH grant 1 P20 GM067650-01A1. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. We would like to thank the anonymous reviewers for their insightful comments on our KDD'04 submission and this journal submission, which help to improve the quality of the paper.

References

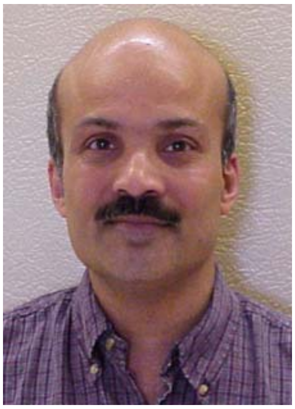
1. Alizadeh AA, Eisen MB, Davis RE et al (2000) Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature 403:503–511
2. Alon U, Barkai N, Notterman DA et al (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. Proc Natl Acad Sci USA 96(12):6745–6750
3. Alter O, Brown PO, Bostein D (2000) Singular value decomposition for genome-wide expression data processing and modeling. Proc Natl Acad Sci USA 97(18):10101–10106
4. Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proceeding of the 1998 ACM-SIGMOD international conference management of data (SIGMOD'98), Seattle, WA, pp 85–93
5. Ben-Dor A, Friedman N, Yakhini Z (2001) Class discovery in gene expression data. In: Proceeding of the fifth annual international conference on computational molecular biology (RECOMB 2001) ACM Press, pp 31–38
6. Blake JA, Harris M (2003) The gene ontology project: structured vocabularies for molecular biology and their application to genome and expression analysis. In: Current protocols in bioinformatics Wiley, New York
7. Cheng Y, Church GM (2000) Biclustering of expression data. Proc ISMB'00 8:93–103
8. Der SD, Zhou A, Williams BR, Silverman RH (1998) Identification of genes differentially regulated by interferon alpha, beta, or gamma using oligonucleotide arrays. Proc Natl Acad Sci USA 95(26):15623–15628
9. Ding C (2002) Analysis of gene expression profiles: class discovery and leaf ordering. In: Proceeding of the international conference on computational molecular biology (RECOMB). Washington, DC, pp 127–136
10. Dudoit S, Fridlyand J, Speed TP (2002) Comparison of discrimination methods for the classification of tumors using gene expression data. J Am Stat Assoc 77–87
11. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci USA 95(25):14863–14868
12. Fraley C, Raftery AE (1998) How many clusters? Which clustering method? Answers via model-based cluster analysis. Comput J 41(8):578–588
13. Golub TR, Slonim DK, Tamayo P et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(15):531–537
14. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceeding of 2000 ACM-SIGMOD international conference management of data (SIGMOD'00), Dallas, TX, pp 1–12
15. Hartuv E, Shamir R (2000) A clustering algorithm based on graph connectivity. Inf Process Lett 76(4–6):175–181
16. Herrero J, Valencia A, Dopazo J (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics 17:126–136
17. Heyer LJ, Kruglyak S, Yooseph S (1999) Exploring expression data: identification and analysis of coexpressed genes. Genome Res 9(11):1106–1115
18. Holter NS, Mitra M, Maritan A, Cieplak M, Banavar JR, Fedoroff NV (2000) Fundamental patterns underlying gene expression profiles: simplicity from complexity. Proc Natl Acad Sci USA 97(15):8409–8414
19. Jiang D, Pei J, Zhang A (2003) Interactive exploration of coherent patterns in time-series gene expression data. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'03), Washington, DC, USA
20. Jiang D, Pei J, Zhang A (2005) A general approach to mining quality pattern-based clusters from gene expression data. In: Proceedings of the 10th international conference on database systems for advanced applications (DASFAA'05), Beijing, China
21. Jiang D, Pei J, Ramanathan M, Tang C, Zhang A (2004) Mining coherent gene clusters from gene–sample–time microarray data. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD'04) ACM Press, pp 430–439
22. Kerr K, Churchill G (2001) Statistical design and the analysis of gene expression microarrays. Genet Res 77:123–128

23. Liu J, Wang W (2003) Op-cluster: clustering by tendency in high dimensional space. In: Proceedings of the third IEEE international conference on data mining (ICDM'03), IEEE, Melbourne, Florida
24. Moler EJ, Chow ML, Mian IS (2000) Analysis of molecular profile data using generative and discriminative methods. Physiol Genomics 4(2):109–126
25. Pei J, Han J, Mao R (2000) CLOSET: an efficient algorithm for mining frequent closed itemsets. In: Proceeding of 2000 ACM-SIGMOD international workshop data mining and knowledge discovery (DMKD'00), Dallas, TX, pp 11–20
26. Pei J, Zhang X, Cho M, Wang H, Yu PS (2003) MaPle: a fast algorithm for maximal pattern-based clusterin. In: Proceedings of the third IEEE international conference on data mining (ICDM'03)
27. Ralf-Herwig PA, Muller C, Bull C, Lehrach H, O'Brien J (1999) Large-scale clustering of cDNA-fingerprinting data. Genome Res 9:1093–1105
28. Rymon R (1992) Search through systematic set enumeration. In: Proceeding of 1992 international conference principle of knowledge representation and reasoning (KR'92), Cambridge, MA, pp 539–550
29. Seo J, Shneiderman B (2002) Interactively exploring hierarchical clustering results. IEEE Comput 35(7):80–86
30. Shamir R, Sharan R (2000) Click: a clustering algorithm for gene expression analysis. In: Proceedings of ISMB '00
31. Smet FD, Mathys J, Marchal K et al (2002) Adaptive quality-based clustering of gene expression profiles. Bioinformatics 18:735–746
32. Stark GR, Kerr IM, Williams BR, Silverman RH, Schreiber RD (1998) How cells respond to interferons. Ann Rev Biochem 67:227–264
33. Tamayo P, Solni D, Mesirov J et al (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proc Natl Acad Sci USA 96(6):2907–2912
34. Tang C, Zhang A, Pei J (2003) Mining phenotypes and informative genes from gene expression data. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD'03), Washington, DC, USA
35. Tavazoie S, Hughes D, Campbell MJ et al (1999) Systematic determination of genetic network architecture. Nature Genet 281–285
36. Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman R (2001) Missing value estimation methods for DNA microarrays. Bioinformatics 17(6):520–525
37. Tusher VG, Tibshirani R, Chu G (2001) Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci USA 98(9):5116–5121
38. Wang W, Yang J, Wang H, Yu PS (2002) Clustering by pattern similarity in large data sets. In: Proceeding of 2002 ACM-SIGMOD international conference on management of data (SIGMOD'02), Madison, WI
39. Weinstock-Guttman B, Badgett D, Patrick K, Hartrich L, Hall D, Baier M, Feichter J, Ramanathan M (2003) Genomic effects of interferon-b in multiple sclerosis patients. J Immun 171(5):2694–2702
40. Xing EP, Karp RM (2001) Cliff: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. Bioinformatics 17(1):306–315
41. Yang J, Wang W, Wang H, Yu PS (2002) δ-cluster: capturing subspace correlation in a large data set. In: Proceedings of 18th international conference on data engineering (ICDE 2002), pp 517–528
42. Yeung KY, Fraley C, Murua A, Raftery AE, Ruzzo WL (2001) Model-based clustering and data transformations for gene expression data. Bioinformatics 17:977–987
43. Zaki MJ, Hsiao CJ (2002) CHARM: an efficient algorithm for closed itemset mining. In: Proceeding of 2002 SIAM international conference on data mining, Arlington, VA, pp 457–473

Daxin Jiang received the Ph.D. degree in computer science and engineering from the State University of New York at Buffalo in 2005. He received the B.S. degree in computer science from the University of Science and Technology of China. From 1998 to 2000, he was a M.S. student in Software Institute, Chinese Academy of Sciences. He is currently an assistant professor at the School of Computer Engineering, Nanyang Technology University, Singapore. His research interests include data mining, bioinformatics, machine learning, and information retrieval.



Jian Pei received the Ph.D. degree in computing science from Simon Fraser University, Canada, in 2002, under Dr. Jiawei Han's supervision. He also received the B.Eng. and the M.Eng. degrees from Shanghai Jiao Tong University, China, in 1991 and 1993, respectively, both in Computer Science. He is currently an assistant professor of computing science at Simon Fraser University. His research interests include developing effective and efficient data analysis techniques for novel data intensive applications. He is currently interested in various techniques of data mining, data warehousing, online analytical processing, and database systems, as well as their applications in bioinformatics. His current research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the National Science Foundation (NSF) of the United States. Since 2000, he has published over 70 research papers in refereed journals, conferences, and workshops, has served in the organization committees and the program committees of over 60 international conferences and workshops, and has been a reviewer for some leading academic journals. He is a member of the ACM, the ACM SIGMOD, and the ACM SIGKDD.



Murali Ramanathan is an associate professor of pharmaceutical sciences and neurology. He received the B.Tech. (Honors) in chemical engineering from the Indian Institute of Technology, India, in 1983. After a 4-year stint in the chemical industry, he obtained the M.S. degree in chemical engineering from Iowa State University, Ames, IA, in 1987, and the Ph.D. degree in bioengineering from the University of California-San Francisco and University of California-Berkeley Joint Program in Bioengineering in 1994. Dr. Ramanathan research interests are primarily focused on the treatment of multiple sclerosis (MS), an inflammatory-demyelinating disease of the central nervous system that affects over 1 million patients worldwide. MS is a complex, variable disease that causes physical and cognitive disability and nearly 50% of patients diagnosed with MS are unable to walk after 15 years. The etiology and pathogenesis of MS remains poorly understood. Dr. Ramanathan's research interests include stochastic modeling of pharmaceutical systems and novel approaches to analyzing and using genetic and genomic data for improving patient care and optimizing therapy.

Chuan Lin is currently a Ph.D. student in the Department of Computer Science and Engineering, State University of New York at Buffalo. She received the B.E. and the M.S. degrees in computer science and technology from Tsinghua University in China. Her research interests include bioinformatics, data mining, and machine learning.

Chun Tang received the B.S. and M.S. degrees from Peking University, China, in 1996 and 1999, respectively, and the Ph.D. degree from State University of New York at Buffalo, USA, in 2005, all in computer science. Currently, she is a postdoctoral associate of Center for Medical Informatics, Yale University. Her research interests include bioinformatics, data mining, machine learning, database, and information retrieval.

Aidong Zhang received the Ph.D. degree in computer science from Purdue University, West Lafayette, Indiana, in 1994. She was an assistant professor from 1994 to 1999, an associate professor from 1999 to 2002, and has been a professor since 2002 in the Department of Computer Science and Engineering at State University of New York at Buffalo. Her research interests include multimedia systems, content-based image retrieval, bioinformatics, and data mining. She is an author of over 140 research publications in these areas. Dr. Zhang's research has been funded by NSF, NIH, NIMA, and Xerox. Zhang serves on the editorial boards of *International Journal of Bioinformatics Research and Applications* (*IJBRA*), *ACM Multimedia Systems*, *International Journal of Multimedia Tools and Applications*, and *International Journal of Distributed and Parallel Databases*. She was the editor for ACM SIGMOD DiSC (Digital Symposium Collection) from 2001 to 2003. She was co-chair of the technical program committee for ACM Multimedia in 2001. She has also served on various conference program committees. Dr. Zhang is a recipient of the National Science Foundation CAREER award and SUNY Chancellor's Research Recognition award.