A General Approach to Mining Quality Pattern-Based Clusters from Microarray Data*

Daxin Jiang[1], Jian Pei[2], and Aidong Zhang[1]

[1] State University of New York at Buffalo, USA
{djiang3, azhang}@cse.buffalo.edu
[2] Simon Fraser University, Canada
jpei@cs.sfu.ca

Abstract. Pattern-based clustering has broad applications in microarray data analysis, customer segmentation, e-business data analysis, etc. However, pattern-based clustering often returns a large number of highly-overlapping clusters, which makes it hard for users to identify interesting patterns from the mining results. Moreover, there lacks of a general model for pattern-based clustering. Different kinds of patterns or different measures on the pattern coherence may require different algorithms. In this paper, we address the above two problems by proposing a general quality-driven approach to mining top-k quality pattern-based clusters. We examine our quality-driven approach using real world microarray data sets. The experimental results show that our method is general, effective and efficient.

1 Introduction

Clustering is an important data mining problem. For a set of objects, a clustering algorithm partitions the objects into a set of *clusters*, such that objects within a cluster are similar to each other, and objects in different clusters are dissimilar. While many traditional clustering methods often assume that the clusters are mutually exclusive and rely on metric distance between objects, some recently emerging applications, such as those in bio-informatics and e-business, post the challenges of mining non-exclusive, non-distance-based clusters in various subspaces from large databases.

As a typical application, a microarray data set can be modelled as a numerical data matrix recording the expression levels of genes on samples. An important task of analyzing microarray data is to find co-expressed genes and phenotypes. A group of *co-expressed genes* are the ones that demonstrate similar expression

* This research is partly supported by NSF grants DBI-0234895 and IIS-0308001, NIH grant 1 P20 GM067650-01A1, the Endowed Research Fellowship and the President Research Grant from Simon Fraser University. 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.

L. Zhou, B.C. Ooi, and X. Meng (Eds.): DASFAA 2005, LNCS 3453, pp. 188–200, 2005.

patterns over a substantial subset of samples, and the subset of samples may correspond to some *phenotype*.

Moreover, given a microarray data set, a gene can belong to more than one co-expressed gene group, since it may correlate to more than one phenotype; and a sample can manifest more than one phenotype, such as tumor vs. normal tissues and male vs. female samples. To address the novel requirements, recently, a new theme of *pattern-based clustering*, is being developed [1, 5, 6, 9, 10] (Please see Section 2 for a brief review).

As indicated by the previous studies, pattern-based clustering is effective for mining non-exclusive, non-distance-based clusters. However, the state-of-the-art methods for pattern-based clustering are still facing the following two serious challenges, which will be addressed in this paper.

Challenge 1: Pattern-based clustering may return a large number of highly-overlapping clusters.
To filter out trivial clusters, most of the pattern-based clustering methods adopt some thresholds, such as the minimum number of objects in a cluster, the minimum number of attributes in a cluster, and the minimum degree of coherence of a cluster. Since too tight threshold values may prune out most of the clusters, including those bearing interesting patterns, loose threshold values are usually preferred.

However, pattern-based clustering will return the complete set of possible combinations of objects and attributes that pass the thresholds. When loose threshold values are specified, thousands or tens of thousands of clusters will be reported. Moreover, since the microarray data are typically highly-connected [3], the reported clusters may be often highly overlapping. For example, our empirical study has shown that the average overlap among the clusters returned by a representative pattern-based clustering algorithm may be as high as 79% (Please see Section 5 for details). Clearly, it is hard for users to identify useful patterns from such voluminous and redundant mining results.

Can we develop an effective method that can automatically focus on finding a small set of representative clusters with respect to loose threshold values?

Our Contribution. In this paper, we propose a theme of *mining top-k quality pattern-based clusters*, based on a user specified quality/utilization function. In particular, the top-k clusters are sorted according to their quality, and the clusters with higher quality are reported before those with lower quality. We show that, by intuitive quality functions, highly overlapping clusters can be avoided.

Challenge 2: There are numerous pattern-based clustering models due to various definitions of patterns and coherence measures.
For example, Cheng and Church [1] measured the coherence of clusters by the *mean squared residue score*. Wang et al. [9] introduced the notion of *pScore* to measure the similarity between the objects in clusters. Liu and Wang [5] defined patterns by ordering attributes in value ascending order. Jiang et al. [4] constrained the coherence within groups of samples by the *minimum coherence threshold*. Different algorithms are proposed to handle specific models. Even

with a minor change to the specific pattern-based clustering model, such as the definition of coherence function, we may have to write a new algorithm.

Given that pattern-based clustering methods share essential intuitions and principles, can we have a general approach such that many different pattern-based clustering models can be handled consistently?

Our Contribution. In this paper, we develop a general model for pattern-based clustering to address the above challenge. Our new pattern-base clustering model is a generalization of several previous models, including *bi-Cluster* [1], *δ-pCluster* [9], *OP-Cluster* [5] and *coherent gene cluster* [4]. We study how to mine top-k quality pattern-based clusters under the general model, and give a general and efficient algorithm.

The remainder of the paper is organized as follows. In Section 2, we review the related work, and also clarify the novel progress that we make in this paper comparing to our previous studies on mining microarray data. A general quality-driven model is introduced in Section 3. A general approach to mining top-k quality pattern-based clusters is presented in Section 4. We report the experimental results in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

Our research is highly related to pattern-based clustering. Cheng and Church [1] introduced *bi-cluster* model. Given a subset of objects I and a subset of attributes J, the coherence of the submatrix (I, J) is measured by the *mean squared residue score*.

$$r_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2, \tag{1}$$

where a_{ij} is the value of object i on j, a_{iJ} is the average value of row i, a_{Ij} is the average value of column j, and a_{IJ} is the average value of the submatrix (I, J). The problem of *bi-clustering* is to mine submatrices with low mean squared residue scores. Yang et al. [10] proposed a move-based algorithm to find biclusters more efficiently. The algorithms in [1] and [10] adopt heuristic search strategies, and thus cannot guarantee to find the optimal biclusters in a data set.

In [9], Wang et al. proposed the model of *δ-pCluster*. A subset of objects O and a subset of attributes A form 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, i.e., $|(x.a - y.a) - (x.b - y.b)| \leq \delta$. In a recent study [6], Pei et al. developed *MaPle*, an efficient algorithm to mine the complete set of maximal pattern-based clusters (i.e., non-redundant pattern-based clusters).

In [5], Liu and Wang presented the model of *OP-Cluster*. Under this model, two objects g_i, g_j are similar on a subset of attributes S if the values of these two objects induce the same relative order of those attributes. An efficient algorithm, *OPC-Tree*, was developed.

2.1 New Progress in This Paper

Since 2002, we have been systematically developing pattern-based clustering methods for mining microarray data, e.g., [6, 4, 3]. For example, we proposed a model for coherent clusters, a specific type of pattern-based clusters, in the novel gene-sample-time series microarray data sets, and developed algorithms *Sample-Gene Search* and *Gene-Sample Search* [4]. *Sample-Gene Search* was shown more efficient.

This paper is critically different from [4] and other previous studies on pattern-based clustering in the following perspectives. First, the methods discussed in [4] enumerate *all* pattern-based clusters. As discussed before, although *MaPle*, *OPC-Tree*, *Gene-Sample Search* and *Sample-Gene Search* can find the complete set of the pattern-based clusters in a data set, they may not be effective to handle the two challenges discussed in Section 1. In this paper, we address the challenges by proposing a general quality-driven pattern-based clustering framework. Instead of enumerating all the pattern-based clusters, we mine only the *top-k clusters* here according to a quality/utilization function specified by users. All existing methods cannot mine such top-k clusters.

Second, [4] studies a specific type of microarray data stes. In this paper, we do not focus on a specific model. Instead, we generalize several previously proposed pattern-based clustering models and propose a general approach.

Last, [4] and this paper share the framework of pattern-growth approaches, i.e., both methods conduct depth-first search. However, due to the quality-driven mining requirements, in this paper, we develop techniques to prune futile search subspaces using the quality criteria (e.g., Section 4.1 and Rule 3). The algorithm developed in this paper inherits and generalizes the technical merits from [4, 6].

3 Mining Quality Pattern-Based Clusters

For a set of n genes $G\text{-}Set = \{g_1, \ldots, g_n\}$ and a set of m samples $S\text{-}Set = \{s_1, \ldots, s_m\}$, the expression levels of the genes on the samples form a 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 $(1 \leq j \leq m)$. A *cluster* is a submatrix $C = (G, S)$ of M, i.e., $G \subseteq G\text{-}Set$ and $S \subseteq S\text{-}Set$, such that C is coherent. Here, the coherence of C describes how coherently the genes in G exhibit expression patterns on the set of samples S.

The measure of coherence varies in different specific pattern-based clustering models. In this paper, we are interested in constructing a general model instead of proposing another measure of coherence. Thus, we assume that the coherence of a submatrix is given by a function $cScore$ such that (1) $cScore(C) \geq 0$ for any submatrix C; and (2) for submatrices C_1 and C_2, if $cScore(C_1) > cScore(C_2)$, then C_1 is more coherent than C_2.

For a specific model, it is easy to revise the coherence measure to satisfy the above two requirements. For example, the *bi-Cluster* model [1] minimizes the *mean squared residue score* r_{IJ} (Equation 1). Since the score is always greater than or equals to 0, minimizing r_{IJ} is equivalent to maximizing $\frac{1}{r_{IJ}}$. Thus, we can use the following $cScore()$ function.

$$cScore(C) = \frac{1}{\sum_{i \in I, j \in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2} \tag{2}$$

For δ-*pCluster*, we can use the following function.

$$cScore(C) = \begin{cases} 1 & \text{if } pScore(X) \leq \delta \text{ for any } 2 \times 2 \text{ sumbmatrix } X \text{ of } C \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

For *OP-Cluster*, we have

$$cScore(C) = \begin{cases} 1 & \text{if patterns in } C \text{ follow the same ordering} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Moreover, for *coherent gene cluster* [4], we can specify the *cScore* function as follows.

$$cScore(C) = \begin{cases} 1 & \text{if in } C \text{ each gene is coherent across the samples} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In real applications, users often have a preference among the clusters. For example, in mining gene expression data, clusters with a high coherence score and a large number of genes and samples are strongly preferred. Accordingly, we define the quality measure of clusters as follows.

Definition 1 (Quality of a cluster). Let $C = (G, S)$ be a submatrix of a microarray data set M, the *quality* of C is defined as $quality(C) = size(C) \cdot cScore(C)$, where $size(C) = |G| \cdot |S|$ and *cScore* is the coherence function. ∎

For a set of clusters that have no overlap, the quality of the set of clusters is simply the sum of the quality of each cluster. However, when there exist some overlaps, we have to make sure that each overlapping cell contributes to the total quality only once, and the contribution goes to the most quality cluster that contains the overlap.

Definition 2 (Quality of a set of clusters). Let Ω be a set of submatrices. The *quality* of Ω is defined as $quality(\Omega) = \sum_{m_{i,j} \in \cup_{C \in \Omega} C} Q(m_{i,j})$, where $Q(m_{i,j}) = \max\{cScore(C) | (C \in \Omega) \wedge (m_{i,j} \in C)\}$. ∎

Suppose a user wants a set of k clusters that have the best quality, the problem can be formulated as to compute a set $\Omega = \{C_1, \ldots, C_k\}$ of k submatrices such that $quality(\Omega)$ is globally maximized. However, given different numbers of clusters k and k' such that $k < k'$, the corresponding optimized sets of clusters Ω and Ω' may not be consistent. In other words, since we maximize the quality function on a global level, a quality cluster $C \in \Omega$ may not necessarily appear in Ω'. The inconsistency among the mining results with respect to different numbers of clusters is undesirable, since the number of clusters k is usually unknown a priori.

To address this problem, we turn to a greedy framework. The main idea is that we compute a series of k clusters $\Omega = \{C_1, \ldots, C_k\}$ such that (1) C_1 is the cluster with the highest quality; and (2) for C_i ($i \geq 2$), C_i is a cluster maximizing the "quality improvement" with respect to $C_1, \ldots, C_{i-1}$. In this way, for any two numbers of clusters $k < k'$, we have $\Omega \subset \Omega'$. Then the user can choose the number of clusters in an incremental manner. At first, the user can choose a small value of k, if all the clusters reported are with high quality, the user can ask for more clusters until the quality of the latest reported cluster is not satisfactory. We formulate the idea as follows.

Definition 3 (Quality gain). Let $C = (G, S)$ be a submatrix of a gene expression matrix M, and $cScore$ be a coherence function. For a set of submatrices $\Omega = \{C_1, \ldots, C_k\}$, the *quality gain* of C (against Ω) is defined as $quality(C|\Omega) = |C - overlap(C, \Omega)| \cdot cScore(C)$, where function $overlap(C, \Omega) = \{m_{i,j} | (m_{i,j} \in C) \wedge (\exists C' \in \Omega : m_{i,j} \in C')\}$ returns the set of cells in C that overlap with some clusters in Ω. ∎

Problem of Mining Top-k Quality Clusters. Given a gene expression matrix M, a coherence function $cScore(\cdot)$ and a positive integer k. The *problem of mining top-k quality pattern based clusters* is to compute a series of k submatrices $C_1, \ldots, C_k$ such that (1) $quality(C_1)$ is the maximum; and (2) for $i \geq 2$, $quality(C_i|\{C_1, \ldots, C_{i-1}\})$ is the maximum. ∎

Our general model of mining quality pattern-based clusters has the following distinct features. First, *our model can generate a list of clusters in quality descending order*. Many previous approaches such as [4, 5, 6, 9] report all the pattern-based clusters without any indication of the significance of the clusters. It is often tedious to select the interesting clusters from those trivial ones. Second, *our model is a generalization of* bi-cluster*,* δ-pCluster*,* OP-Cluster*, and* coherent gene cluster. As shown before, we can easily assign coherence functions to those specific models.

In many applications, a user has several basic constraints to avoid trivial clusters. The constraints can be specified using the following three thresholds. (1) *Minimum number of genes* min_g; (2) *Minimum number of samples* min_s; and (3) *Minimum coherence* δ. A submatrix $C = (G, S)$ will be reported as a cluster only if it satisfies the constraints: $|G| \geq min_g$, $|S| \geq min_s$ and $cScore(C) \geq \delta$.

Moreover, in some pattern-based clustering models, such as *δ-pCluster*, *OP-Cluster*, and *coherent gene cluster*, an anti-monotonicity holds: a coherence function $cScore()$ is *anti-monotonic* if for any two clusters $C_1 = (G_1, S_1)$ and $C_2 = (G_2, S_2)$ such that $G_1 \subseteq G_2$ and $S_1 \subseteq S_2$, $cScore(C_1) \geq cScore(C_2)$. The anti-monotonicity captures a natural assumption: the coherence of a submatrix monotonically decreases as more genes and/or more samples are included. In our general model, we also assume that the anti-monotonicity holds for the coherence function.

4 The Mining Algorithm

In this section, we will present a general approach to mine top-k quality clusters that satisfy the thresholds. Basically, we find the top-k clusters iteratively, one at a time. We will address the following two issues.

- In the i-th iteration ($1 \leq i \leq k$), how can we find cluster C_i that maximizes the quality gain against the set of clusters $\{C_1, \ldots, C_{i-1}\}$?
- How to search the huge space of all possible submatrices efficiently and prune unpromising subspace sharply?

4.1 Mining a Cluster Maximizing Quality Gain

A naïve method to find a cluster that has the maximum quality gain is to test every possible submatrix and its quality gain. A submatirx can be viewed as a combination of genes and samples. Therefore, the problem can be reduced to enumerating all possible combinations of genes and samples.

A systematic way to tackle an enumeration problem is to use enumeration tree [7]. Figure 1 shows the enumeration tree of a four-element set $\{a, b, c, d\}$. It provides a conceptual tool to enumerate all the subsets of $\{a, b, c, d\}$ systematically.

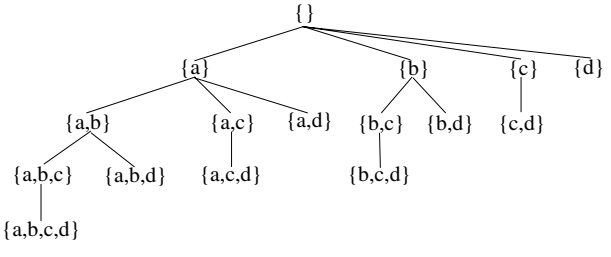


Fig. 1. Enumeration of combinations of samples

Basically, we can enumerate all the subsets of samples first. For each subset of samples S, we enumerate all subsets of genes G, and test the quality gain of (G, S). We only need to keep the submatrix $C = (G, S)$ that satisfies the thresholds and achieves the best quality gain in the current iteration. This method is called the *Sample-Gene Search*.[1]

Why do we enumerate subsets of samples first and then subsets of genes later, but not in the reverse way?

In gene expression data, the number of genes is typically by far larger than the number of samples. In other words, the number of combinations of genes

[1] The initial idea of enumerating samples instead of genes in microarray data sets to find pattern-based clusters was firstly proposed by Wang et al. [9], and further systematically developed in [6].

is often dramatically larger than the number of combinations of samples. With our pruning rules in Section 4.2, if the Sample-Gene Search is adopted, once a subset of samples and its descendants are pruned, all searches of related subsets of genes are pruned as well. Heuristically, the Sample-Gene Search may bring a better chance to prune a more bushy search sub-tree than the Gene-Sample Search for gene expression data.

When we enumerate the subsets of samples or genes, we can conduct a *recursive, depth-first search* of the set enumeration tree. Given a data set of m samples and n genes, the set enumeration tree has 2^{m+n} nodes. However, we never need to materialize such a tree. Instead, we only need to keep a path from the root of the tree to the node we are searching as a working set, which contains at most $m + n + 1$ nodes. Besides, proper pruning techniques will be developed to prune unpromising branches as early as possible.

4.2 The Rules for Pruning

In this subsection, we develop efficient rules to prune unpromising subspaces using the thresholds and/or the anti-monotonicity of the coherence function.

For the Sample-Gene Search, each node on the set enumeration tree contains a unique submatrix. Thus we will use the submatrix to refer to the node. At node $C = (G, S)$ of the set enumeration tree, where $G = \{g_{i_1}, \ldots, g_{i_k}\}$ $(1 \leq i_1 < \cdots < i_k \leq n)$, we keep a list $gTail$ of genes. A gene $g_j \in G\text{-}Set$ is included in list $gTail$ if (1) $j \geq i_k$ and (2) the coherence score of $C' = (G \cup \{g_j\}, S)$ is no less than minimum coherence threshold δ. We have the following result, which generalized some of the pruning techniques in the existing pattern-based clustering methods (e.g., [6, 4]).

Rule 1 (Pruning irrelevant genes). *For a node C in the set enumeration tree, only the genes in list $gTail$ should be used to construct super clusters of C.*

Rationale. Suppose gene $g_j \notin gTail$ of $C = (G, S)$, where $G = \{g_{i_1}, \ldots, g_{i_k}\}$ $(1 \leq i_1 < \cdots < i_k \leq n)$. Two situations may happen. First, $j \leq i_k$. Second, $C' = (G \cup \{g_j\}, S)$ violates the coherence constraint. For the first situation, g_j cannot be used to expand C according to the structure of the set enumeration tree. For the second situation, since any descendant C'' of C' is a submatrix of C', according to the anti-monotonic property, C'' also violates the coherence constraint. Therefore, we can prune the genes not in the $gTail$ list. ∎

Similarly, we can maintain a list $sTail$ of samples for node C, and prune the samples not in $sTail$ when we search the subtree of C. Due to the limit of space, we omit the details here. Moreover, for any descendant node C' of C, the $gTail$ and $sTail$ lists of C' are subsets of those lists of C, respectively.

Since the $gTail$ and $sTail$ lists of the current node C tell us which genes and samples can be used to further expand the subtree of C, they actually provide us the a priori information about the subtree of C. Based on such information, we can prune the unpromising descendants of C early.

Rule 2 (Pruning small submatrices). *For a node $C = (G, S)$, the subtree of C can be pruned if $(|G| + |gTail|) < min_g$ or $(|S| + |sTail|) < min_s$.*

Rationale. Since we only use the genes and samples in $gTail$ and $sTail$ lists to expand the subtree of C, for any descendant node $C' = (G', S')$ of C, we have $|G'| \leq (|G| + |gTail|)$ and $|S'| \leq (|S| + |sTail|)$. If for node C, $(|G| + |gTail|) < min_g$ or $(|S| + |sTail|) < min_s$, then none of the descendants of C will satisfy the size constraint. Therefore, the subtree of C can be pruned. ∎

Rules 1 and 2 are essential for pattern-based clustering (as well as frequent itemset mining). The similar idea has been studied before extensively (e.g., [4, 6]). The quality mining inherits them. To push the quality requirement into the mining, the following lemma gives the upper bound of the quality gain that can be achieved in a subtree.

Input: the gene expression data set M
Output: the top-k clusters Ω
Method:

```
let Ω = ∅ // the set of top clusters already found
for num = 1 to k do
    let maxQ = −1, maxCluster = null
    for each subset of samples S
        if |S| < min_s continue
        for i = 1 to (|G-Set| − min_g) do
            let G = {g_i}, C = (G, S); compute gTail
            call recursive-search(C, gTail)
        end for // end the enumeration of genes
    end for // end the enumerate of samples
    let Ω = Ω ∪ {maxCluster}
end for
```

Procedure: recursive-search(C, $gTail$)

```
if (|G| + |gTail|) < min_g then return
calculate the quality upper bound of C's descendants according to Lemma 1
if C can be pruned by Pruning Rule 3 then return
while (gTail ≠ ∅) do
    let i = min{j| g_j ∈ gTail}
    let C' = (G ∪ {g_i} × S); compute gTail'
    call recursive-search(C', gTail')
end while
if ((|G| ≥ min_g) && (|S| ≥ min_s)) then
    if (quality(C|Ω)) > maxQ
    then let maxQ = quality(C|Ω), let maxCluster = C
    end if
end if
```

Fig. 2. Algorithm *Q-Clustering* for mining top-k quality clusters

Lemma 1. *Let Ω be a set of clusters. For any descendant C' of node $C = (G, S)$ in the set enumeration tree, a tight upper bound of quality$(C'|\Omega)$ is given by $[(|G|+|gTail|)(|S+|sTail|)-|overlap(((G\cup gTail), (S\cup sTail)), \Omega)|]\cdot cScore(C)$.*

Proof sketch. $[(|G| + |gTail|) * (|S + |sTail|) - overlap(((G \cup gTail), (S \cup sTail)), \Omega_{k-1})]$ is the upper bound of the non-overlapping size of the descendants of C. Given the anti-monotonicity of the coherence measure, $cScore(C)$ is the upper bound of the coherence score of the descendants of C. Therefore, Lemma 1 gives an upper bound of the quality of the descendants of C. The bound can be shown tight. Limited by space, we omit the details here. ∎

Based on Lemma 1, we have the following rule immediately.

Rule 3 (Pruning low quality submatrices). *The subtree of C can be pruned if the upper bound of the quality gain given by Lemma 1 is smaller than the best quality gain that has got so far in the current iteration.* ∎

In summary, Figure 2 shows algorithm *Q-Clustering* to mine the top-k quality clusters. Limited by space, we omit the details of pruning techniques using the $sTail$ list, which is basically symmetric to the case of $gTail$ list.

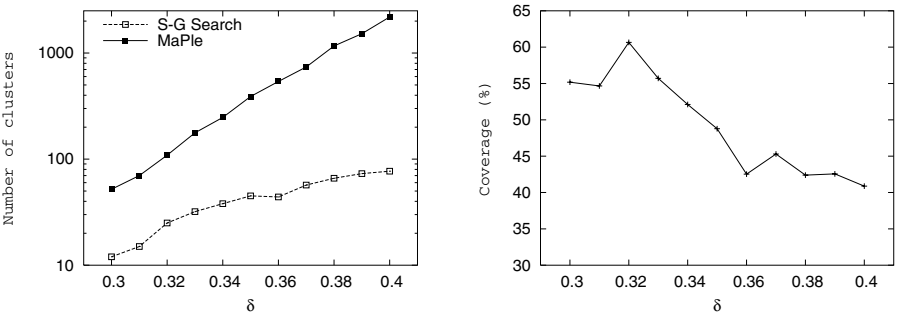
5 Experimental Results

We tested algorithm *Q-Clustering* on both synthetic data sets and real gene expression data sets. The system is implemented in Java. The tests are conducted on a Sun Ultra 10 work station with a 440MHz CPU and 256 MB main memory. The results are consistent. Limited by space, we only report the results on a real data set here.

Spellman et al. [8] reported the genome-wide 6,220 mRNA transcript levels during the cell cycle of the budding yeast *S. cerevisiae*. The complete data set consists of 3 independent time-series, namely, the $\alpha factor$ (18 time points), the *elutriation* (14 time points) and the *cdc*15 (24 time points). We choose the *cdc*15 data set since it contains the longest time-series. Out of the 6,220 monitored genes, only 800 genes are found cell-cycle-dependent. We call this subset of data *cdc*_800. To test the performance of our algorithm extensively, we sample the complete data set (6,220 genes and 56 time points) with various sizes.

To test the effectiveness of our general quality-driven approach, we chose a representative pattern-based clustering model, the δ-*pCluster* [9], and compare the mining results reported by our approach with those by a representative pattern-based clustering algorithm, MaPle [6]. Given the *cdc*_800 data set, both *Q-Clustering* and MaPle were invoked when $min_s = 5$ and $min_g = 5$, while the δ value ranges from 0.3 to 0.4. For *Q-Clustering*, we only return the clusters with a quality gain beyond $min_s * min_g$. According to Definition 3 and the semantic meaning of min_s and min_g, such clusters may carry interesting patterns.

Figure 3(a) shows the number of clusters reported by the two algorithms. Since MaPle finds the complete set of maximal δ-pClusters, we can see the number of clusters increases dramatically when the threshold value increases. However, *Q-Clustering* only returns the quality δ-pClusters, and the number of clusters is much more stable.



(a) Number of clusters (b) Coverage of clusters

Fig. 3. Clusters reported by MaPle and Gene-Sample Search

δ	*quality clusters*	*all maximal clusters*
3.0	0.0%	69.4%
3.2	1.2%	70.0%
3.4	1.5%	73.8%
3.6	1.6%	75.5%
3.8	2.6%	77.6%
4.0	5.8%	79.3%

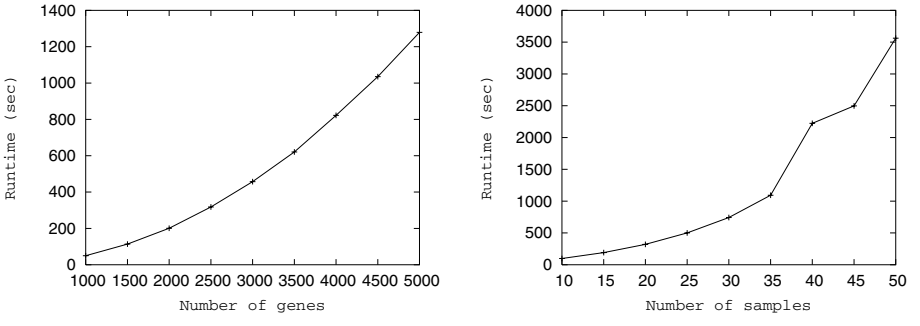
Fig. 4. Overlap between clusters

We then evaluate the correlation between the clusters reported by *Q-Clustering* and MaPle. That is, we want to measure to which extent the quality clusters cover the set of δ-pClusters. We represent a cluster C by $\{(g_i, s_j)\}$, where g_i and s_j are the gene and sample in C, respectively. Given two sets of clusters $\Omega = \{C_1, \ldots, C_m\}$ and $\Omega' = \{C'_1, \ldots, C'_n\}$, the coverage of Ω on Ω' is defined by $\frac{(C_1 \cup \ldots \cup C_m) \cap (C'_1 \cup \ldots \cup C'_n)}{C'_1 \cup \ldots \cup C'_n}$. Figure 3 (b) illustrates the coverage of the quality clusters on the complete set of clusters. We can see that when $\delta = 0.4$, although the number of quality clusters is only 3% of the total number of clusters, the coverage of quality clusters is over 40%. That is, our quality-driven approach focuses on finding a small set of clusters which can effectively represent the underlying patterns in the data set. Please note that, to increase the coverage, users can always ask more clusters from the system until no more interesting patterns are identified.

Why the number of clusters reported by our quality-driven approach is much smaller than that by MaPle?

The rationale is that, due to the high-connectivity of microarray data, the pattern-based clusters usually highly overlap with each other. Figure 4 demonstrates the average overlap among the clusters by MaPle and Sample-Gene Sample, respectively. Given a set of clusters Ω, the average overlap of Ω is

$\frac{\sum_{C_i \in \Omega} overlap(C_i)}{|\Omega|}$, where the overlap of a cluster C_i is measured by $overlap(C_i) = max\{\frac{C_i \cap C_j}{C_i} | C_j \in \Omega, i \neq j\}$. We can see that the average overlap among the complete set of clusters is usually higher than 70%, while the average overlap among the quality clusters is less than 6%. In practice, a gene may participate in multiple cellular processes or correlate to several phenotypes. Consequently, it may belong to more than one cluster. However, such situation is not common and a ratio of about 6% overlap is biologically plausible.



(a) Runtime vs. number of genes (b) Runtime vs. number of samples

Fig. 5. Scalability with respect to the sizes of the data sets

Finally, we test the scalability of our algorithm. We set $min_s = 6, min_g = 10$ and $\delta = 0.2$. We sample the *cdc*15 time-series (24 time points) when we test the scalability with respect to the number of genes. To test the scalability with respect the number of samples, we fix the number of genes to 3,000 and sample the time points from the complete data set. The results are shown in Figure 5(a) and (b). We can see that our algorithm scales well with respect to both the number of genes and the number of samples.

6 Discussion and Conclusions

In this paper, we proposed a general approach to mining top-k quality pattern-based clusters. The experimental results on gene expression data show that our method is general, effective and efficient. Several interesting and important problems still remain open, such as how to find multiple quality clusters during a single iteration, and how to handle non-anti-monotonic coherence functions.

Acknowledgements. We thank the reviewers for their comments and suggestions which help to improve the presentation of the paper.

References

1. Cheng, Y. and Church, G.M. Biclustering of expression data. *ISMB'00*.
2. Jain, A.K., Murty, M.N. and Flynn, P.J. Data clustering: a review. *ACM Computing Surveys*, 31:264–323, 1999.
3. Jiang, D., Pei, J. and Zhang, A. Interactive Exploration of Coherent Patterns in Time-Series Gene Expression Data. In *KDD'03*.
4. Jiang, D., Pei, J., Ramanathan, M., et al. Mining Coherent Gene Clusters from Gene-Sample-Time Microarray Data. In *KDD'04*.
5. Liu, J., Wang, W. OP-Cluster: Clustering by Tendency in High Dimensional Space. In *ICDM'03*.
6. Pei, J., Zhang, X., Cho, M., et al. MaPle: A Fast Algorithm for Maximal Pattern-based Clustering. *ICDM'03*.
7. Rymon, R. Search through systematic set enumeration. In *KR'92*.
8. Spellman, P.T., Sherlock, G., Zhang, M.Q., et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3272–3297, 1998.
9. Wang, H., Wang, W., Yang, J. et al. Clustering by Pattern Similarity in Large Data Sets. In *SIGMOD'02*.
10. Yang, J., Wang, W., Wang, H. et al. δ-cluster: Capturing Subspace Correlation in a Large Data Set. In *ICDE'02*.