Wang H, Pei J. Clustering by pattern similarity. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(4): 481–496 July 2008

Clustering by Pattern Similarity

Haixun Wang[1] (王海勋) and Jian Pei[2] (裴 健)

[1] *IBM T. J. Watson Research Center, Hawthorne, NY 10533, U.S.A.*

[2] *Simon Fraser University, British Columbia, Canada*

E-mail: haixun@us.ibm.com; jpei@cs.sfu.ca

Received December 4, 2007; revised May 28, 2008.

Abstract The task of clustering is to identify classes of *similar* objects among a set of objects. The definition of similarity varies from one clustering model to another. However, in most of these models the concept of similarity is often based on such metrics as Manhattan distance, Euclidean distance or other L_p distances. In other words, similar objects must have *close values* in at least a set of dimensions. In this paper, we explore a more general type of similarity. Under the *pCluster* model we proposed, two objects are similar if they exhibit a *coherent pattern* on a subset of dimensions. The new similarity concept models a wide range of applications. For instance, in DNA microarray analysis, the expression levels of two genes may rise and fall synchronously in response to a set of environmental stimuli. Although the magnitude of their expression levels may not be close, the patterns they exhibit can be very much alike. Discovery of such clusters of genes is essential in revealing significant connections in gene regulatory networks. E-commerce applications, such as collaborative filtering, can also benefit from the new model, because it is able to capture not only the closeness of values of certain leading indicators but also the closeness of (purchasing, browsing, etc.) patterns exhibited by the customers. In addition to the novel similarity model, this paper also introduces an effective and efficient algorithm to detect such clusters, and we perform tests on several real and synthetic data sets to show its performance.

Keywords data mining, clustering, pattern similarity

1 Introduction

Cluster analysis, which identifies classes of similar objects among a set of objects, is an important data mining task[1−3] with broad applications. Clustering methods have been extensively studied in many areas, including statistics[4], machine learning[5,6], pattern recognition[7], and image processing. Much active research has been devoted to various issues in clustering, such as scalability, the curse of high-dimensionality, etc.

However, clustering in high dimensional spaces is often problematic. Theoretical results[8] have questioned the meaning of closest matching in high dimensional spaces. Recent research work[9−13] has focused on discovering clusters embedded in subspaces of a high dimensional data set. This problem is known as subspace clustering. In this paper, we explore a more general type of subspace clustering which uses pattern similarity to measure the distance between two objects.

1.1 Goal

Most clustering models, including those used in subspace clustering, define the similarity among different objects by distances over either all or only a subset of the dimensions. Some well-known distance functions include Euclidean distance, Manhattan distance, and cosine distance. However, distance functions are not always adequate for capturing correlations among the objects. In fact, strong correlations may still exist among a set of objects even if they are far apart from each

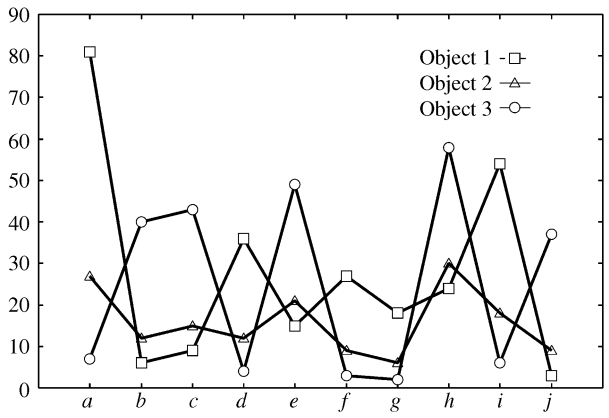

Fig.1. Small data set of 3 objects and 10 attributes.

Regular Paper

other as measured by the distance functions. As an example, let us consider the data set plotted in Fig.1.

In Fig.1, which shows a data set of 3 objects and 10 attributes (columns), no patterns among the 3 objects are visibly explicit. However, if we pick the subset of the attributes $\{b, c, h, j, e\}$, and plot the values of the 3 objects on these attributes as shown in Fig.2(a), it is easy to see that they manifest similar patterns. However, these objects may not be considered to be in a cluster by any traditional (subspace) clustering model because the distance between any two of them is not close to each other.

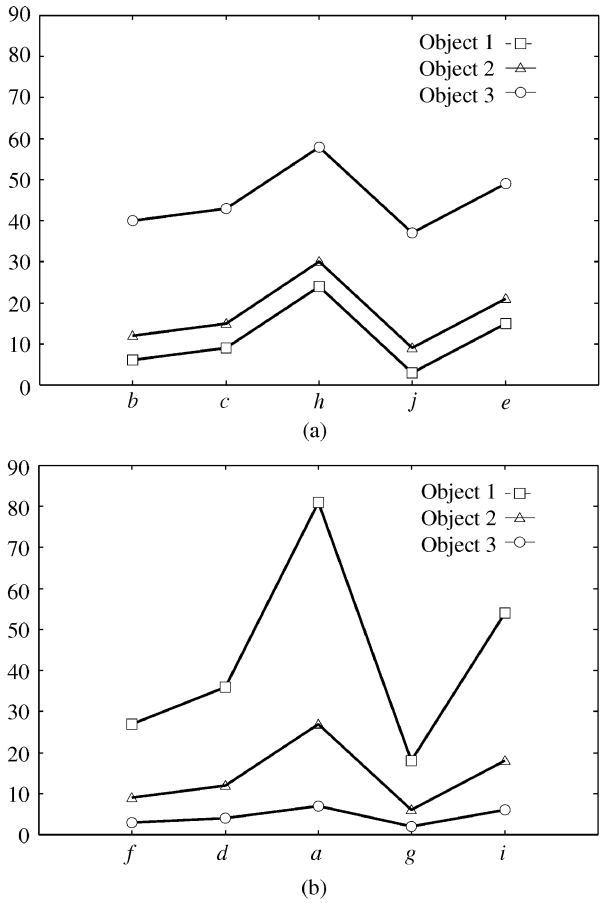

Fig.2. Objects form patterns on a set of columns. (a) Objects in Fig.1 form a *Shifting Pattern* in subspace $\{b, c, h, j, e\}$. (b) Objects in Fig.1 form a *Scaling Pattern* in subspace $\{f, d, a, g, i\}$.

The same set of objects can form different patterns on different sets of attributes. In Fig.2(b), we show another pattern in subspace $\{f, d, a, g, i\}$. This time, the three curves do not have a shifting relationship. Instead, values of object 2 are roughly three times larger than those of object 3, and values of object 1 are roughly three times larger than those of object 2. If we think of columns f, d, a, g, i as different environmental stimuli or conditions, the pattern shows that the 3 objects respond to these conditions coherently, although object 1 is more responsive or more sensitive to the stimuli than the other two.

We use *pattern similarity* to denote the shifting and scaling correlations exhibited by objects in a subspace (Fig.2). While most traditional clustering algorithms focus on *value similarity*, that is, they consider two objects similar if at least some of their coordinate values are close, our goal is to model and discover clusters based on shifting or scaling correlations from raw data sets such as the one shown in Fig.1.

1.2 Applications

Discovery of clusters in data sets based on pattern similarity is of great importance because of its potential for actionable insights. Here, let us mention two elaborate applications as follows.

Application 1: *DNA Micro-Array Analysis.* Microarray is one of the latest breakthroughs in experimental molecular biology. It provides a powerful tool by which the expression patterns of thousands of genes can be monitored simultaneously and is already producing huge amounts of valuable data. Analysis of such data is becoming one of the major bottlenecks in the utilization of the technology. The gene expression data are organized as matrices — tables where rows represent genes, columns represent various samples such as tissues or experimental conditions, and numbers in each cell characterize the expression level of the particular gene in the particular sample. Investigations show that more often than not, several genes contribute to a disease, which motivates researchers to identify a subset of genes whose expression levels rise and fall coherently under a subset of conditions, that is, they exhibit fluctuation of a similar shape when conditions change. Discovery of such clusters of genes is essential for revealing the significant connections in gene regulatory networks[14].

Application 2: *E-Commerce.* Recommendation systems and target marketing are important applications in the E-commerce area. In these applications, sets of customers/clients with similar behavior need to be identified so that we can predict customers' interest and make proper recommendations. Let us consider the following example. Three viewers rate four movies of a particular type (action, romance, etc.) as (1, 2, 3, 6), (2, 3, 4, 7), and (4, 5, 6, 9), respectively, where 1 is the lowest and 10 is the highest score. Although the rates given by each individual are not close, these three viewers have coherent opinions on the four movies. In

the future, if the first viewer and the third viewer rate a new movie of that category as 7 and 9 respectively, then we have certain confidence that the 2nd viewer will probably like the movie too, since they have similar tastes in that type of movies.

1.3 Our Contributions

Our objective is to cluster objects that exhibit similar patterns on a subset of dimensions. Traditional subspace clustering is a special case in our task, so in the sense that objects in a subspace cluster have exactly the same behavior, there is no coherence need to be related by shifting or scaling. In other words, these objects are physically close — their similarity can be measured by functions such as the Euclidean distance, the Cosine distance, and etc.

Our contributions include:

• We propose a new clustering model, namely the *pCluster*①, to capture not only the closeness of objects but also the similarity of the patterns exhibited by the objects.

• The *pCluster* model is a generalization of subspace clustering. However, it finds a much broader range of applications, including DNA array analysis and collaborative filtering, where pattern similarities among a set of objects carry significant meanings.

• We propose an efficient depth-first algorithm to mine *pCluster*s. Compared with the bicluster approach[15,16], our method mines multiple clusters simultaneously, detects overlapping clusters, and is resilient to outliers. Our method is deterministic in that it discovers all qualified clusters, while the bicluster approach is a random algorithm that provides only an approximate answer.

1.4 Paper Layout

The rest of the paper is structured as follows. In Section 2, we study the background of this work and review some related work, including the bicluster model. We present the *pCluster* model in Section 3. In Section 4, we present the process of finding base clusters. Section 5 studies different pruning approaches. The experimental results are shown in Section 6 and we conclude the paper in Section 7.

2 Background and Related Work

As clustering is always based on a similarity model, in this section, we discuss traditional similarity models used for clustering, as well as some new models that focus on correlations of objects in subspaces.

2.1 Traditional Similarity Models

Clustering in high dimensional spaces is often problematic as theoretical results[8] questioned the meaning of closest matching in high dimensional spaces. Recent research work[9−13,17] has focused on discovering clusters embedded in the subspaces of high dimensional data sets. This problem is known as *subspace clustering*.

A well known clustering algorithm capable of finding clusters in subspaces is CLIQUE[11]. CLIQUE is a density-and grid-based clustering method. It discretizes the data space into non-overlapping rectangular cells by partitioning each dimension to a fixed number of bins of equal length. A bin is dense if the fraction of total data points contained in the bin is greater than a threshold. The algorithm finds dense cells in lower dimensional spaces and merges them to form clusters in higher dimensional spaces. Aggarwal *et al.*[9,10] used an effective technique for the creation of clusters for very high dimensional data. The PROCLUS[9] and the ORCLUS[10] algorithms find projected clusters based on representative cluster centers in a set of cluster dimensions. Another interesting approach, Fascicles[12], finds subsets of data that share similar values in a subset of dimensions.

The above algorithms discover *value similarity*, that is, the objects in the cluster share similar values in a subset of dimensions. The similarity among the objects is measured by distance functions, such as Euclidean. However, this model captures neither the shifting pattern in Fig.2(a) nor the scaling pattern in Fig.2(b), since objects therein do not share similar values in the subspace where they manifest the patterns. Rather, we are interested in *pattern similarity*, that is, whether objects exhibit a certain type of correlation in subspace.

The task of capturing the similarity exhibited by objects in Fig.2 is not to be confused with pattern discovery in time series data, such as trending analysis in stock closing prices. In time series analysis, patterns occur during a continuous time period. Here, mining is not restricted by any fixed ordering among the columns of the data set. Patterns on an arbitrary subset of the columns are usually deeply buried in the data when the entire set of the attributes are present, as exemplified in Figs.1 and 2.

The similar reasoning reveals why the models treating the entire set of attributes as a whole do not work in

① *pCluster* stands for pattern-based cluster.

mining pattern-based clusters. For example, the *Pearson R* model[18] studies the coherence among a set of objects, and *Pearson R* defines the correlation between two objects X and Y as:

$$\frac{\sum_i (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_i (X_i - \overline{X})^2 \times \sum_i (Y_i - \overline{Y})^2}}$$

where X_i and Y_i are the i-th attribute value of X and Y, and $\overline{X}$ and $\overline{Y}$ are the means of all attribute values in X and Y, respectively. From this formula, we can see that the *Pearson R* correlation measures the correlation between two objects with respect to all attribute values. A large positive value indicates a strong positive correlation while a large negative value indicates a strong negative correlation. However, some strong coherence may only exist on a subset of dimensions. For example, in collaborative filtering, six movies are ranked by viewers. The first three are action movies and the next three are family movies. Two viewers rank the movies as $(8, 7, 9, 2, 2, 3)$ and $(2, 1, 3, 8, 8, 9)$. The viewers' ranking can be grouped into two clusters, the first three movies in one cluster and the rest in another. It is clear that the two viewers have consistent bias within each cluster. However, the *Pearson R* correlation of the two viewers is small because globally no explicit pattern exists.

2.2 Correlations in Subspaces

One way to discover the shifting pattern in Fig.2(a) using traditional subspace clustering algorithms (such as CLIQUE) is through data transformation. Given N attributes, $a_1, \ldots, a_N$, we define a derived attribute, $A_{ij} = a_i - a_j$, for every pair of attributes a_i and a_j. Thus, our problem is equivalent to mining subspace clusters on the objects with the derived set of attributes. However, the converted data set will have $N(N-1)/2$ dimensions and it becomes intractable even for a small N because of the curse of dimensionality.

Cheng *et al.* introduced the bicluster concept[15] as a measure of the coherence of the genes and conditions in a sub matrix of a DNA array. Let X be the set of genes and Y the set of conditions. Let $I \subset X$ and $J \subset Y$ be subsets of genes and conditions, respectively. The pair (I, J) specifies a sub matrix $\boldsymbol{A}_{IJ}$ with the following *mean squared residue score*:

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

where

$$d_{iJ} = \frac{1}{|J|} \sum_{j \in J} d_{ij}, \quad d_{Ij} = \frac{1}{|I|} \sum_{i \in I} d_{ij},$$
$$d_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} d_{ij}$$

are the row and column means and the means in the submatrix $\boldsymbol{A}_{IJ}$, respectively. A submatrix $\boldsymbol{A}_{IJ}$ is called a δ-bicluster if $H(I, J) \leqslant \delta$ for some $\delta > 0$. A random algorithm is designed for finding such clusters in a DNA array.

Yang *et al.*[16] proposed a move-based algorithm to find biclusters more efficiently. It starts from a random set of seeds (initial clusters) and iteratively improves the clustering quality. It avoids the cluster overlapping problem as multiple clusters are found simultaneously. However, it still has the outlier problem, and it requires the number of clusters as an input parameter.

We noticed several limitations of this pioneering work as follows.

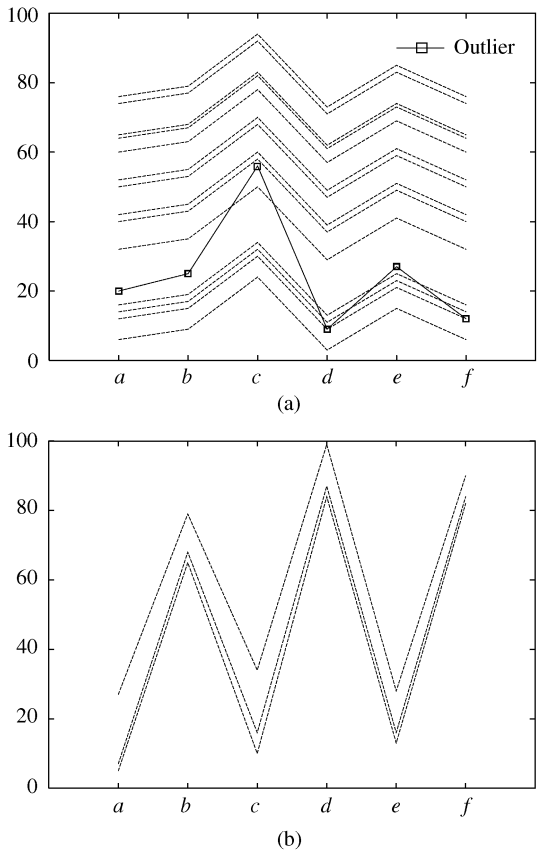

Fig.3. Mean squared residue cannot exclude outliers in a bicluster. (a) Dataset A: Residue 4.238 (without the outlier residue is 0). (b) Dataset B: Residue 5.722.

1) The mean squared residue used in [15, 16] is an averaged measurement of the coherence for a set of objects. A much undesirable property of (1) is that *a submatrix of a δ-bicluster is not necessarily a δ-bicluster*. This creates difficulties in designing efficient algorithms. Furthermore, many δ-biclusters found in a given data set may differ only in one or two outliers they contain. For instance, the bicluster shown in Fig.3(a) contains an obvious outlier but it still has a fairly small mean squared residue (4.238). The only way to get rid of such outliers is to reduce the δ threshold, but that will exclude many biclusters which do exhibit coherent patterns, e.g., the one shown in Fig.3(b) with residue 5.722.

2) The algorithm presented in [15] detects a bicluster in a greedy manner. To find other biclusters after the first one is identified, it mines on a new matrix derived by replacing entries in the discovered bicluster by random data. However, clusters are not necessarily disjoint, as shown in Fig.4. The random data will obstruct the discovery of the second cluster.

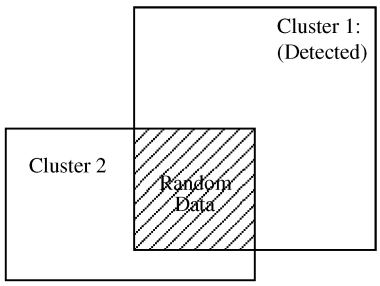

Fig.4. Replacing entries in the shaded area by random values may obstruct the discovery of the second cluster.

3 *pCluster* Model

This section describes the *pCluster* model for mining clusters of objects that exhibit coherent patterns on a set of attributes. The notations used in this paper are summarized in Table 1.

Table 1. Notations

$\mathcal{D}$	A set of objects
$\mathcal{A}$	Attributes of objects in $\mathcal{D}$
$(\mathcal{O}, \mathcal{T})$	A submatrix of the data set, where $\mathcal{O} \subseteq \mathcal{D}$, $\mathcal{T} \subseteq \mathcal{A}$
$x, y, \ldots$	Objects in $\mathcal{D}$
$a, b, \ldots$	Attributes of $\mathcal{A}$
d_{xa}	Value of object x on attribute a
δ	User-specified clustering threshold
nc	User-specified minimum # of columns of a *pCluster*
nr	User-specified minimum # of rows of a *pCluster*
$\mathcal{T}_{xy}$	A maximal dimension set for objects x and y
$\mathcal{O}_{ab}$	A maximal dimension set for columns a and b

Let $\mathcal{D}$ be a set of objects, where each object is defined by a set of attributes $\mathcal{A}$. We are interested in objects that exhibit a coherent pattern on a subset of attributes of $\mathcal{A}$.

Definition 1 (*pScore* and *pCluster*). *Let $\mathcal{O}$ be a subset of objects in the database ($\mathcal{O} \subseteq \mathcal{D}$), and $\mathcal{T}$ be a subset of attributes ($\mathcal{T} \subseteq \mathcal{A}$). Pair $(\mathcal{O}, \mathcal{T})$ specifies a submatrix. Given $x, y \in \mathcal{O}$, and $a, b \in \mathcal{T}$, we define the pScore of the 2×2 matrix as*:

$$pScore\left(\begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}\right) = |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})|. \tag{2}$$

For a user-specified parameter $\delta \geqslant 0$, pair $(\mathcal{O}, \mathcal{T})$ forms a δ-pCluster if for any 2×2 submatrix $\boldsymbol{X}$ in $(\mathcal{O}, \mathcal{T})$, we have $pScore(\boldsymbol{X}) \leqslant \delta$.

Intuitively, $pScore(\boldsymbol{X}) \leqslant \delta$ means that the change of values on the two attributes between the two objects in $\boldsymbol{X}$ is confined by δ, a user-specified threshold. If such a constraint applies to every pair of objects in $\mathcal{O}$ and every pair of attributes in $\mathcal{T}$, then we have found a δ-pCluster.

In the bicluster model, a submatrix of a δ-bicluster is not necessarily a δ-bicluster. However, based on the definition of *pScore*, the *pCluster* model has the following property.

Property 1 (Anti-Monotonicity). *Let $(\mathcal{O}, \mathcal{T})$ be a δ-pCluster. Any of its submatrix, $(\mathcal{O}', \mathcal{T}')$, where $\mathcal{O}' \subseteq \mathcal{O}$, $\mathcal{T}' \subseteq \mathcal{T}$, is also a δ-pCluster.*

Note that the definition of *pCluster* is symmetric: as shown in Fig.5(a), the difference can be measured horizontally or vertically, as the right hand side of (2) can be rewritten as

$$\begin{aligned} |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})| &= |(d_{xa} - d_{ya}) - (d_{xb} - d_{yb})| \\ &= pScore\left(\begin{bmatrix} d_{xa} & d_{ya} \\ d_{xb} & d_{yb} \end{bmatrix}\right). \end{aligned} \tag{3}$$

When only 2 objects and 2 attributes are considered, the definition of *pCluster* conforms with that of the bicluster model[15]. According to (1), and assuming $I = \{x, y\}$, $J = \{a, b\}$, the mean squared residue of a 2×2 matrix $\boldsymbol{X} = \begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}$ is:

$$\begin{aligned} H(I, J) &= \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (d_{ij} - d_{Ij} - d_{iJ} + d_{IJ})^2 \\ &= \frac{((d_{xa} - d_{xb}) - (d_{ya} - d_{yb}))^2}{4} \\ &= (pScore(\boldsymbol{X})/2)^2. \end{aligned} \tag{4}$$

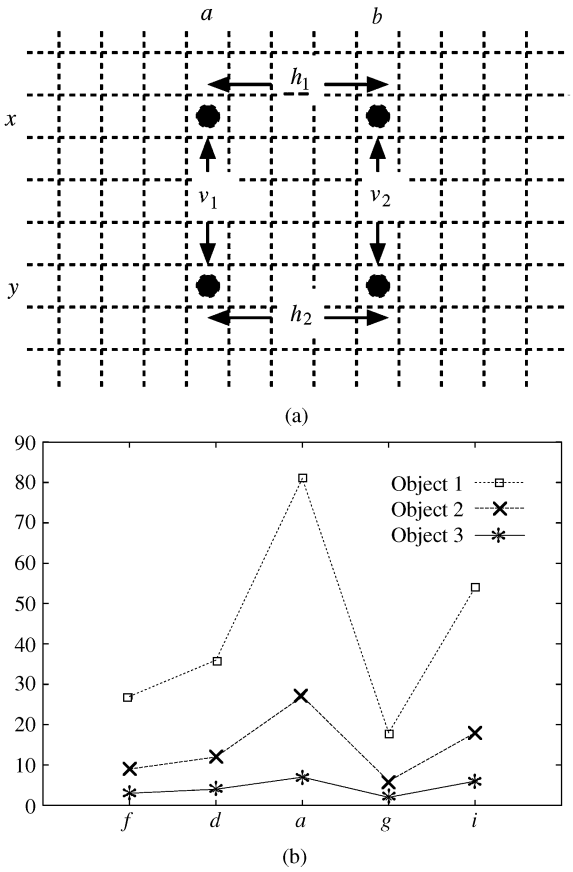

Fig.5. *pCluster* Definition. (a) Definition is symmetric: $|h_1 - h_2| \leqslant \delta$ is equivalent to $|v_1 - v_2| \leqslant \delta$. (b) Objects 1, 2, 3 form a *pCluster* after we take the logarithm of the data.

Thus, for a 2-object/2-attribute matrix, a δ-bicluster is a $(\frac{\delta}{2})^2$-*pCluster*. However, since a *pCluster* requires that every 2 objects and every 2 attributes conform with the inequality, it models clusters that are more homogeneous. Let us review the problem of bicluster in Fig.3. The mean squared residue of data set A is 4.238, less than that of data set B, 5.722. Under the *pCluster* model, the maximum *pScore* between the outlier and another object in A is 26, while the maximum *pScore* found in data set B is only 14. Thus, any δ between 14 and 26 will eliminate the outlier in A without obstructing the discovery of the *pCluster* in B.

In order to model the cluster in Fig.5(b), where there is a scaling factor among the objects, it seems we need to introduce a new inequality:

$$\frac{d_{xa}/d_{ya}}{d_{xb}/d_{yb}} \leqslant \delta'. \tag{5}$$

However, this is unnecessary because (2) can be regarded as a logarithmic form of (5). The same *pCluster* model can be applied to the dataset after we convert the values therein to the logarithmic form. As a matter of fact, in DNA micro-array, each array entry d_{ij}, representing the expression level of gene i in sample j, is derived in the following manner:

$$d_{ij} = \log\Big(\frac{Red\ Intensity}{Green\ Intensity}\Big), \tag{6}$$

where *Red Intensity* is the intensity of gene i, the gene of interest, and *Green Intensity* is the intensity of a reference (control) gene. Thus, the *pCluster* model can be used to monitor the changes in gene expression and to cluster genes that respond to certain environmental changes in a coherent manner.

Conditions

Genes	CH1I	CH1B	CH1D	CH2I	CH2B
CTFC3	4392	284	4108	280	228
VPS8	401	281	120	275	298
EFB1	318	280	37	277	215
SSA1	401	292	109	580	238
FUN14	2857	285	2576	271	226
SP07	228	290	48	285	224
MDM10	538	272	266	277	236
CYS3	322	288	41	278	219
DEP1	312	272	40	273	232
NTG1	329	296	33	274	228

(a)

Conditions

Genes	CH1I	CH1B	CH1D	CH2I	CH2B
CTFC3					
VPS8	401		120		298
EFB1	318		37		215
SSA1					
FUN14					
SP07					
MDM10					
CYS3	322		41		219
DEP1					
NTG1					

(b)

Fig.6. *pCluster* of yeast genes. (a) Gene expression data. (b) *pCluster*.

Fig.6(a) shows a micro-array matrix with ten genes (one for each row) under five experiment conditions (one for each column). This example is a portion of the micro-array data that can be found in [19]. A *pCluster*

({VPS8, EFB1, CYS3}, {CH1I, CH1D, CH2B}) is embedded in the micro-array. Apparently, their similarity cannot be revealed by Euclidean distance or Cosine distance.

Objects form a cluster when a certain level of density is reached. In other words, a cluster often becomes interesting if it is of reasonable volume. Too small clusters may not be interesting or scientifically significant. The volume of a *pCluster* is defined by the size of $\mathcal{O}$ and the size of $\mathcal{T}$. The task is thus to find all those *pCluster*s beyond a user-specified volume.

Problem Statement. Given: i) δ, a cluster threshold, ii) nc, a minimal number of columns, and iii) nr, a minimal number of rows, the task of *mining pClusters* or *pattern-based clustering* is to find all pairs $(\mathcal{O}, \mathcal{T})$ such that $(\mathcal{O}, \mathcal{T})$ is a δ-*pCluster* according to Definition 1, and $|\mathcal{O}| \geqslant nr$, $|\mathcal{T}| \geqslant nc$.

4 *pCluster* Algorithm

In this section, we describe the *pCluster* algorithm. We aim at achieving efficiency in mining high quality *pCluster*s.

4.1 Overview

More specifically, the *pCluster* algorithm focuses on achieving the following goals.

• Our first goal is to mine clusters simultaneously. The bicluster algorithm[15], on the other hand, finds clusters one by one, and the discovery of one cluster might obstruct the discovery of other clusters. This is not only time consuming but also leads to the second issue we want to address.

• Our second goal is to find each and every qualifying *pCluster*. This means our algorithm must be deterministic. More often than not, random algorithms based on the bicluster approach[15,16] provide only an incomplete approximation to the answer, and the clusters they find depend on the order of their search.

• Our third goal is to address the issue of pruning search spaces. Objects can form clusters in any subset of the data columns, and the number of data columns in real life applications, such as DNA array analysis and collaborative filtering, is usually in hundreds or even thousands. Many subspace clustering algorithms[9,11] find clusters in lower dimensions first and then merge them to derive clusters in higher dimensions. This is a time consuming approach. The *pCluster* model gives us many opportunities of pruning, that is, it enables us to remove many objects and columns in a candidate cluster before it is merged with other clusters to form clusters in higher dimensions. Our approach explores several different ways to find the effective pruning methods.

For a better understanding of how the *pCluster* algorithm achieves these goals, we will present the algorithm in three steps.

1) *Pair-Wise Clustering.* Based on the *maximal dimension set Principle* to be introduced in Subsection 4.2, we find the largest (column) clusters for every two objects, and the largest (object) clusters for every two columns. Apparently, clusters that span a larger number of columns (objects) are usually of more interest, and finding larger clusters first also enables us to avoid generating clusters which are part of other clusters.

2) *Pruning Unfruitful Pair-Wise Clusters.* Apparently, not every column (object) cluster found in pair-wise clustering will occur in the final *pClusters*. To reduce the combinatorial cost in clustering, we remove as many pair-wise clusters as early as possible by using the *Pruning Principle* to be introduced in Subsection 4.3.

3) *Forming δ-pCluster.* In this step, we combine pruned pair-wise clusters to form *pClusters*.

The following subsections present the *pCluster* algorithm in these three steps.

4.2 Pairwise Clustering

Our first step is to generate pairwise clusters in the largest dimension set. Note that if a set of objects cluster on a dimension set $\mathcal{T}$, then they also cluster on any subset of $\mathcal{T}$ (Property 1). Clustering will be much more efficient if we can find *pClusters* on the largest dimension set directly. To facilitate further discussion, we define the concept of *Maximal Dimension Set* (*MDS*).

Definition 2 (Maximal Dimension Set). *Assuming $c = (\mathcal{O}, \mathcal{T})$ is a δ-pCluster. Column set $\mathcal{T}$ is a Maximal Dimension Set (MDS) of c if there does not exist $\mathcal{T}' \supset \mathcal{T}$ such that $(\mathcal{O}, \mathcal{T}')$ is also a δ-pCluster.*

In our approach, we are interested in objects clustered on column set $\mathcal{T}$ only if there does not exist $\mathcal{T}' \supset \mathcal{T}$, such that the objects also cluster on $\mathcal{T}'$. We are only interested in *pCluster*s that cluster on MDSs, because all other *pClusters* can be derived from these maximum *pClusters* using Property 1. Note that from the definition, it is clear that an attribute can appear in more than one MDS. Furthermore, for a set of objects $\mathcal{O}$, there may exist more than one MDS.

Given a set of objects $\mathcal{O}$ and a set of columns $\mathcal{A}$, it is not trivial to find all the *maximal dimension sets* for $\mathcal{O}$, since $\mathcal{O}$ may cluster on any subset of $\mathcal{A}$. Below, we study a special case where $\mathcal{O}$ contains only two objects. Given objects x and y, and a column set $\mathcal{T}$, we define

$S(x, y, \mathcal{T})$ as:

$$S(x, y, \mathcal{T}) = \{d_{xa} - d_{ya} | a \in \mathcal{T}\}.$$

Based on the definition of δ-cluster, we can make the following observation.

Property 2 (Pairwise Clustering). *Given objects x and y, and a dimension set $\mathcal{T}$, x and y form a δ-pCluster on $\mathcal{T}$ iff the difference between the largest and the smallest values in $S(x, y, \mathcal{T})$ is no more than δ.*

Proof. Given objects x and y, we define function $f(a, b)$ on any two dimensions a,$b \in \mathcal{T}$ as:

$$f(a, b) = |(d_{xa} - d_{ya}) - (d_{xb} - d_{yb})|.$$

According to the definition of *δ-pCluster*, objects x and y cluster on $\mathcal{T}$ if $\forall a, b \in \mathcal{T}, f(a, b) \leqslant \delta$. In other words, $(\{x, y\}, \mathcal{T})$ is a *pCluster* if the following is true: $\max_{a,b \in \mathcal{T}} f(a, b) \leqslant \delta$. It is easy to see that $\max_{a,b \in \mathcal{T}} f(a, b) = \max S(x, y, \mathcal{T}) - \min S(x, y, \mathcal{T})$. □

According to the above property, we do not have to compute $f(a, b)$ for every two dimensions a, b in $\mathcal{T}$. Instead, we only need to know the largest and smallest values in $S(x, y, \mathcal{T})$.

We use $\boldsymbol{S}(x, y, \mathcal{T})$ to denote a sorted sequence of values in $S(x, y, \mathcal{T})$. That is,

$$\boldsymbol{S}(x, y, \mathcal{T}) = s_1, \ldots, s_k,$$
$$s_i \in S(x, y, \mathcal{T}) \quad \text{and} \quad s_i \leqslant s_j \text{ where } i < j.$$

Thus, x and y form a *δ-pCluster* on $\mathcal{T}$ if $s_k - s_1 \leqslant \delta$. Given a set of attributes $\mathcal{A}$, it is also not difficult to find the maximal dimension sets for object x and y.

Proposition 3 (Maximal Dimension Set (MDS) Principle). *Given a set of dimensions $\mathcal{A}$, $\mathcal{T}_s \subseteq \mathcal{A}$ is a maximal dimension set of x and y iff:*

i) $\boldsymbol{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$ is a (contiguous) subsequence of $\boldsymbol{S}(x, y, \mathcal{T}) = s_1 \cdots s_i \cdots s_j \cdots s_k$, and

ii) $s_j - s_i \leqslant \delta$, whereas $s_{j+1} - s_i > \delta$ and $s_j - s_{i-1} > \delta$.

Proof. Given $\boldsymbol{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$ and $s_j - s_i \leqslant \delta$, according to the pairwise clustering principle, $\mathcal{T}_s$ is a *δ-pCluster*. Furthermore, $\forall a \in \mathcal{T} - \mathcal{T}_s$, we have $d_{xa} - d_{ya} \geqslant s_{j+1}$ or $d_{xa} - d_{ya} \leqslant s_{i-1}$, otherwise $a \in \mathcal{T}_s$ since $\boldsymbol{S}(x, y, \mathcal{T}_s) = s_i \cdots s_j$. If $d_{xa} - d_{ya} \geqslant s_{j+1}$, from $s_{j+1} - s_i > \delta$ we get $(d_{xa} - d_{ya}) - s_i > \delta$, thus $\{a\} \cup \mathcal{T}_s$ is not a *δ-pCluster*. On the other hand, if $d_{xa} - d_{ya} \leqslant s_{i-1}$, from $s_j - s_{i-1} > \delta$ we get $s_j - (d_{xa} - d_{ya}) > \delta$, thus $\{a\} \cup \mathcal{T}_s$ is not a *δ-pCluster* either. Since $\mathcal{T}_s$ cannot be enlarged, $\mathcal{T}_s$ is an MDS. □

According to the MDS principle, we can find the MDSs for objects x and y in the following manner: we start with both the left-end and the right-end placed on the first element of the sorted sequence, and we move the right-end rightward one position at a time. For every move, we compute the difference of the values at the two ends, until the difference is greater than δ. At that time, the elements between the two ends form a maximal dimension set. To find the next maximal dimension set, we move the left-end rightward one position, and repeat the above process. It stops when the right-end reaches the last element of the sorted sequence.

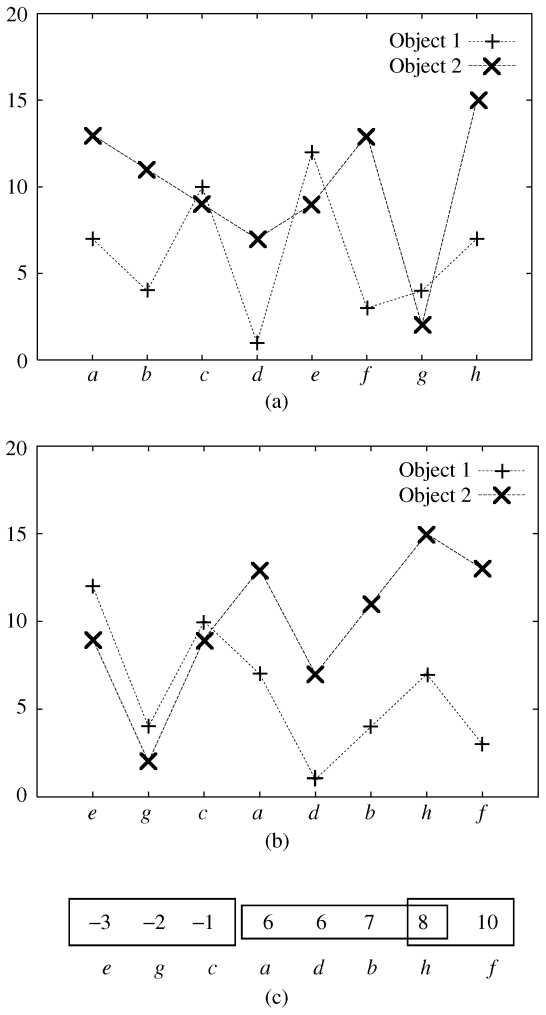

Fig.7. Finding MDS for two objects. (a) Raw data. (b) Sort by dimension discrepancy. (c) Cluster on sorted differences ($\delta = 2$).

Fig.7 gives an example of the above process. We want to find the maximal dimension sets for two objects, whose values on 8 dimensions are shown in Fig.7(a). The patterns are hidden until we sort the dimensions by the difference of x and y on each dimension. The sorted sequence $S = -3, -2, -1, 6, 6, 7, 8, 10$ is shown in Fig.7(c). Assuming $\delta = 2$, we start from the

left end of S. We move rightward until we stop at the first 6, since $6-(-3) > 2$. The columns between the left end and 6, $\{e, g, c\}$, is an MDS. We move the left end to -2 and repeat the process until we find all 3 maximal dimension sets for x and y: $\{e, g, c\}$, $\{a, d, b, h\}$, and $\{h, f\}$. Note that maximal dimension sets might overlap.

The pseudocode of the above process is given in Algorithm 1. Thus, to find the MDSs for objects x and y, we invoke the following procedure:

$$pairCluster(x, y, \mathcal{A}, nc)$$

where nc is the (user-specified) minimal number of columns in a *pCluster*.

Algorithm 1. Find Two-Object *pCluster*s: pairCluster(x, $y, \mathcal{T}, nc$)
Input: x, y: two objects, $\mathcal{T}$: set of columns, nc: minimal number of columns, δ: cluster threshold
Output: All δ-*pClusters* with more than nc columns
$s \leftarrow d_x - d_y$; /* i.e., $s_i \leftarrow d_{xi} - d_{yi}$ for each i in $\mathcal{T}$ */
sort array s;
$start \leftarrow 0$; $end \leftarrow 1$;
$new \leftarrow$ TRUE; /* new=TRUE indicates there is an untested column in $[start, end]$ */
repeat
  $v \leftarrow s_{end} - s_{start}$;
  if $|v| \leqslant \delta$ **then**
    /* expands δ-*pCluster* to include one more columns */
    $end \leftarrow end + 1$;
    $new \leftarrow$ TRUE;
  else
    Return δ-*pCluster* if $end - start \geqslant nc$ and $new =$ TRUE;
    $start \leftarrow start + 1$;
    $new \leftarrow$ FALSE;
until $end \geqslant |\mathcal{T}|$;
Return δ-pCluster if $end - start \geqslant nc$ and $new =$ TRUE;

According to the definition of the *pCluster* model, the columns and the rows of the data matrix carry the same significance. Thus, the same method can be used to find MDSs for each column pair, a and b:

$$pairCluster(a, b, \mathcal{O}, nr).$$

The above procedure returns a set of MDSs for column a and b, except that here the maximal dimension set is made up of objects instead of columns.

As an example, we study the data set shown in Fig.8(a). We find 2 object-pair MDSs and 4 column-pair MDSs.

	c_0	c_1	c_2
o_0	1	4	2
o_1	2	5	5
o_2	3	6	5
o_3	4	200	7
o_4	300	7	6

(a)

$(o_0, o_2) \rightarrow \{c_0, c_1, c_2\}$
$(o_1, o_2) \rightarrow \{c_0, c_1, c_2\}$

(b)

$(c_0, c_1) \rightarrow \{o_0, o_1, o_2\}$
$(c_0, c_2) \rightarrow \{o_1, o_2, o_3\}$
$(c_1, c_2) \rightarrow \{o_1, o_2, o_4\}$
$(c_1, c_2) \rightarrow \{o_0, o_2, o_4\}$

(c)

Fig.8. *Maximal dimension sets* for Column- and Object-pairs ($\delta = 1$, $nc = 3$, and $nr = 3$). (a) 5×3 data matrix. (b) MDS for object pairs. (c) MDS for column pairs.

4.3 Pruning

For a given pair of objects, the number of its MDSs depends on the clustering threshold δ and the user-specified minimum number of columns, nc. However, if $nr > 2$, then only some of these MDSs are *valid*, i.e., they actually occur in δ-*pClusters* whose size is equal to or larger than $nr \times nc$. In this section, we introduce a pruning principle, based on which invalid pairwise clusters can be eliminated.

Given a clustering threshold δ, and a minimum cluster size $nr \times nc$, we use $\mathcal{T}_{xy}$ to denote an MDS for objects x and y, and $\mathcal{O}_{ab}$ to denote an MDS for columns a and b. We have the following result.

Proposition 4 (MDS Pruning Principle). *Let $\mathcal{T}_{xy}$ be an MDS for objects x, y, and $a \in \mathcal{T}_{xy}$. For any $\mathcal{O}$ and $\mathcal{T}$, a necessary condition of $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ being a δ-pCluster is $\forall b \in \mathcal{T}$, $b \neq a$, $\exists \mathcal{O}_{ab} \supseteq \{x, y\}$.*

Proof. Assume $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ is a δ-*pCluster*. Since a submatrix of a δ-*pCluster* is also a δ-*pCluster*, we know $\forall b \in \mathcal{T}$, $(\{x, y\} \cup \mathcal{O}, \{a, b\})$ is a δ-*pCluster*. According to the definition of MDS, there exists at least one MDS $\mathcal{O}_{ab} \supseteq \{x, y\} \cup \mathcal{O} \supseteq \{x, y\}$. Thus, there are at least $|\mathcal{T}|$ such MDSs. □

We are only interested in δ-*pClusters* $(\{x, y\} \cup \mathcal{O}, \{a\} \cup \mathcal{T})$ with size $\geqslant nr \times nc$. In other words, we require $|\mathcal{T}| \geqslant nc - 1$, that is, we must be able to find at least $n - 1$ column pair MDSs that contain $\{x, y\}$.

Symmetric MDS Pruning

Based on Proposition 4, the pruning criterion can be stated as follows. *For any dimension a in an MDS $\mathcal{T}_{xy}$, we count the number of $\mathcal{O}_{ab}$ that contains $\{x, y\}$. If the number of such $\mathcal{O}_{ab}$ is less than $nc - 1$, we remove a from $\mathcal{T}_{xy}$. Furthermore, if the removal of a makes $|\mathcal{T}_{xy}| < nc$, we remove $\mathcal{T}_{xy}$ as well.*

Because of the symmetry of the model (Definition 1), the pruning principle can be applied to object-pair MDSs as well as column-pair MDSs. That is, *for any object x in an MDS $\mathcal{O}_{ab}$, we count the number of $\mathcal{T}_{xy}$ that contains $\{a, b\}$. If the number of such $\mathcal{T}_{xy}$ is less*

than $nr-1$, we remove x from $\mathcal{O}_{ab}$. Furthermore, if the removal of x makes $|\mathcal{O}_{ab}| < nr$, we remove $\mathcal{O}_{ab}$ as well.

This means we can prune the column-pair MDSs and object-pair MDSs by turns. Without loss of generality, we first generate column-pair MDSs from the data set. Next, when we generate object-pair MDSs, we use column-pair MDSs for pruning. Then, we prune column-pair MDSs using the pruned object-pair MDSs. This procedure can go on until no more MDSs can be eliminated.

We continue with our example using the dataset shown in Fig.8(a). To prune the MDSs, we first generate column-pair MDSs, and they are shown in Fig.9(a). Second, we generate object-pair MDSs. MDS $(o_0, o_2) \rightarrow \{c_0, c_1, c_2\}$ is to be eliminated because the column-pair MDS of (c_0, c_2) does not contain o_0. Third, we review the column-pair MDSs based on the remaining object-pair MDSs, and we find that each of them is to be eliminated. Thus, the original data set in Fig.8(a) does not contain any 3×3 *pCluster*.

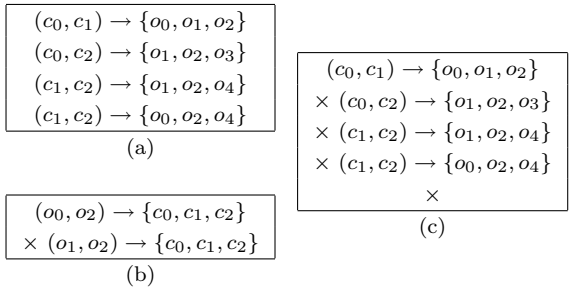

Fig.9. Generating and Pruning MDS iteratively ($\delta = 1$, $nc = 3$, and $nr = 3$). (a) Generating MDS_c from data. (b) Generating MDS_o from data, using MDS_c in (a) for pruning. (c) Pruning MDS_c in (a) using MDS_o in (b).

Algorithm 2 gives a high level description of the symmetric MDS pruning process. It can be summarized as two steps. In the first step, we scan the dataset to find column-pair MDSs for every column-pair, and object-pair MDSs for every object-pair. This step is realized by calling for procedure *pairCluster*() in Algorithm 1. In the second step, we iteratively prune column-pair MDSs and object-pair MDSs until no changes can be made.

MDS Pruning by Object Block

Symmetric MDS pruning iteratively eliminates column-pair MDSs and object-pair MDSs as the definition of pCluster is symmetric for rows and columns. However, in reality, large datasets are usually not symmetric in the sense that they often have much more rows (objects) than columns (attributes). For instance, the yeast microarray contains expression levels of 2884 genes under 17 conditions[19].

Algorithm 2. Symmetric MDS Pruning: symmetricPrune()

Input: $\mathcal{D}$: data set, δ: *pCluster* threshold, nc: minimal number of columns, nr: minimal number of rows

Output: all *pCluster*s with size $\geqslant nr \times nc$

for each $a, b \in \mathcal{A}, a \neq b$ **do**
  find column-pair MDSs: $pairCluster(a, b, \mathcal{D}, nr)$;
for each $x, y \in \mathcal{D}, x \neq y$ **do**
  find object-pair MDSs: $pairCluster(x, y, \mathcal{A}, nc)$;
repeat
  for each object-pair *pCluster* $(\{x, y\}, T)$ **do**
    use column-pair MDSs to prune columns in T;
    eliminate MDS $(\{x, y\}, T)$ if $|T| < nc$;
  for each column-pair *pCluster* $(\{a, b\}, O)$ **do**
    use object-pair MDSs to prune objects in O;
    eliminate MDS $(\{a, b\}, O)$ if $|O| < nr$;
until no pruning takes place;

In symmetric MDS pruning, for any dimension a in an MDS $\mathcal{T}_{xy}$, we count the number of $\mathcal{O}_{ab}$ that contains $\{x, y\}$. When the size of the dataset increases, the size of each column-pair MDS also increases. This brings some negative impacts on efficiency. First, generating a column-pair MDS takes more time, as the process has a complexity of $O(n \log n)$. Second, it also makes the set-containment query time consuming during pruning. Third, it makes symmetric pruning less effective because we cannot remove any column-pair $\mathcal{O}_{ab}$ before we reduce it to contain less than nr objects, which means we need to eliminate more than $|\mathcal{O}_{ab}| - nr$ objects.

To solve this problem, we group object-pair MDSs into blocks. Let $B_x = \{\mathcal{T}_{xy} | \forall y \in \mathcal{D}\}$ represent block x. Apparently, any *pCluster* that contains object x must reside in B_x. Thus, mining *pClusters* over dataset $\mathcal{D}$ is equivalent to finding *pClusters* in each B_x. Pruning will take place within each block as well, yet removing entries in one block may trigger the removing of entries in other blocks, which improves pruning efficiency.

Algorithm 3 gives a detailed description of the process of pruning MDS based on blocks. The algorithm can be summarized as two steps.

In the first step, we compute object-pair MDSs. We represent an object-pair MDS by a bitmap: the i-th bit is set if column i is in the MDS. However, unlike Algorithm 2, we do not compute column-pair MDSs.

In the second step, we prune object-pair MDSs. To do this, we collect column information for objects within each block. This is more efficient than computing column-pair MDSs for the entire dataset (the com-

putation has a complexity of $O(n \log n)$ for each pair), and still, we are able to support the pruning across the blocks using column information maintained in each block. Indeed, cross-pruning occurs on three levels and pruning on lower levels will trigger pruning on higher levels: i) clearing a bit in a bitmap for pair $\{x, y\}$ in B_x will cause the corresponding bit to be cleared in B_y; ii) removing a bitmap (when it has less than nc bits set) for pair $\{x, y\}$ in B_x will cause the corresponding bitmap to be removed in B_y; and iii) removing B_x (when it contains less than $nr - 1$ $\{x, y\}$ pairs) will recursively invoke ii) on every bitmap it has.

Algorithm 3. MDS Pruning by Blocks: blockPrune()

Input: $\mathcal{D}$: data set, δ: *pCluster* threshold, nc, nr: minimal number of columns and rows

Output: pruned object-pair MDSs

```
for each x, y ∈ D, x ≠ y do
    invoke pairCluster(x, y, A, nc) to find MDSs for {x, y};
    represent each MDS by a bitmap (of columns) and
    add it into block B_x and block B_y;
repeat
    for each block B_x do
        for each column i do
            cc[i] ← number of unique {x, y} pairs whose
            MDS bitmap has the i-th bit set
            if cc[i] < nr − 1 then
                for each entry {x, y} in block x do
                    if {x, y}'s MDS bitmap has less than
                      nc − 1 bits set
                    then
                        remove the bitmap (if it is the only
                        MDS bitmap for {x, y}, then remove
                        entry {x, y} in B_x and B_y);
                    else
                        clear bit i in the bitmaps of {x, y} in
                        B_x and B_y;
        eliminate B_x if it contains less than nr − 1 entries;
until no changes take place;
```

4.4 Clustering Algorithm

In this subsection, we focus on the final step of finding *pClusters*. We mine *pClusters* from the pruned object-pair MDSs. A direct approach is to combine smaller clusters to form larger clusters based on the anti-monotonicity property[20]. In this paper, we propose a new approach, which views the pruned object-pair MDSs as a graph, and mines *pClusters* by finding cliques in the graph. Our experimental results show that the new approach is much more efficient.

After MDS pruning in the second step, the remaining objects can be viewed as a graph $G = (V, E)$. In graph G, each node $v \in V$ is an object, and an edge $e \in E$ that connects two nodes v_1 and v_2 means that v_1 and v_2 cluster on an MDS $\{c_1, \ldots, c_k\}$. We use $\{c_1, \ldots, c_k\}$ to label edge e.

Property 5. *A pCluster of size $nr \times nc$ is a clique $G' = (V', E')$ that satisfies $|V'| = nr$ and $|\bigcap_{e \in E'} label(e)| = nc$ where $label(e)$ is the MDS of an object-pair connected by edge e in G'.*

Proof. Let $G' = (V', E')$ be a clique. Any two nodes $\{v_1, v_2\} \subset V'$ is connected by an edge $e \in E'$. Since e's label, which represents the MDS of $\{v_1, v_2\}$, contains at least nc same columns, it means $\{v_1, v_2\}$ form a *pCluster* with the column set. Thus, according to the definition of *pCluster*,

$$(V', \bigcap_{e \in E'} \text{label}(e))$$

is a *pCluster* of size at least $nr \times nc$. □

Furthermore, there is no need to find cliques in the graph composed of the entire set of object-pair MDSs. Instead, we can localize the search within each pruned block $B_x = \{\mathcal{T}_{xy} | \forall y \in \mathcal{D}\}$. This is because B_x contains all objects that are connected to object x. Thus, if object x indeed appears in a *pCluster*, the objects in that *pCluster* must reside entirely in B_x. This means we do not need to search cliques or *pClusters* across blocks.

Algorithm 4 illustrates the process of finding pClusters block by block. First, we collect all available MDSs that appear in each block. For MDSs that associate with $\geqslant nr$ objects, we invoke the Cliquer procedure[21] to find cliques of size $\geqslant nr$. The procedure will check edges between objects using information of other blocks. It also allows one to set the maximum search time for finding a clique. Next, we generate new MDSs by joining the current MDSs and repeat the process on the new MDSs which contain $\geqslant nc$ columns, provided that the potential cliques are not subsets of found pClusters.

4.5 Algorithm Complexity

The step of generating MDSs for symmetric pruning has time complexity $O(M^2 N \log N + N^2 M \log M)$, where M is the number of columns and N is the number of objects. For block pruning, this is reduced to $O(N^2 M \log M)$ since only object-pair MDSs are generated. The worst case for symmetric pruning and block pruning is $O(M^2 N^2)$, although on average it is much less, since the average size of a column-pair MDS (num-

ber of objects in an MDS) is usually much smaller than M. In the worst case, the final clustering process (Algorithm 4) has exponential complexity with regard to the number of columns. However, since most invalid MDSs are eliminated in the pruning phase, the actual time it takes is usually much less than that of generating MDSs and pruning MDSs.

Algorithm 4. Main Algorithm for Mining *pCluster*s: pCluster()

Input: $\mathcal{D}$: data set, δ: *pCluster* threshold, nc, nr: minimal number of columns and rows

Output: all *pCluster*s with size $\geqslant nr \times nc$

```
for each block B_x do
    S ← all MDSs that appear in B_x;
    (each s ∈ S is associated with no less than nr
    objects in B_x);
    repeat
        for each MDS s ∈ S do
            if s and the objects it associates is not a
            subset of a found pCluster then
                invoke Cliquer on s and the objects
                it associates with;
                if a clique is found then
                    output a pCluster;
                    prune entries in related blocks;
        S' ← {};
        for every s1, s2 ∈ S do
            s' ← s1 ∩ s2;
            if |s'| ⩾ nc then
                S' = S' ∪ {s'};
        S ← S';
    until no clique can be found;
```

5 Experimental Results

We experimented our *pCluster* algorithm with both synthetic and real life data sets. The algorithm is implemented on a Linux machine with a 1.0GHz CPU and 256MB main memory.

The *pCluster* algorithm is the first algorithm that studies clustering based on subspace pattern similarity. Traditional subspace clustering algorithms cannot find clusters based on pattern similarity. For the purpose of comparison, we implemented an alternative algorithm that first transforms the matrix by creating a new column A_{ij} for every two columns a_i and a_j, provided $i > j$. The value of the new column A_{ij} is set to $a_i - a_j$. Thus, the new data set will have $N(N-1)/2$ columns, where N is the number of columns in the original data set. Then, we apply a subspace clustering algorithm on the transformed matrix, and discover subspace clusters from the data. There are several subspace clustering algorithms to choose from and we used CLIQUE[11] in our experiments.

5.1 Data Sets

We experiment our *pCluster* algorithm with synthetic data and two real life data sets: one is the MovieLens data set and the other is a DNA microarray of gene expression of a certain type of yeast under various conditions.

Synthetic Data

We generate synthetic data sets in matrix forms. Initially, the matrix is filled with random values ranging from 0∼500, and then we embed a fixed number of *pCluster*s in the raw data. Besides the size of the matrix, the data generator takes several other parameters: *nr*, the average number of rows of the embedded *pCluster*s; *nc*, the average number of columns; and k, the number of *pCluster*s embedded in the matrix. To make the generator algorithm easy to implement, and without loss of generality, we embed *perfect pCluster*s in the matrix, i.e., each embedded *pCluster* satisfies a cluster threshold $\delta = 0$. We investigate both the correctness and the performance of our *pCluster* algorithm using the synthetic data.

Gene Expression Data

Gene expression data are being generated by DNA chips and other microarray techniques. The yeast microarray contains expression levels of 2884 genes under 17 conditions[19]. The data set is presented as a matrix. Each row corresponds to a gene and each column represents a condition under which the gene is developed. Each entry represents the relative abundance of the mRNA of a gene under a specific condition. The entry value, derived by scaling and logarithm from the original relative abundance, is in the range of 0 and 600. Biologists are interested in finding a subset of genes showing strikingly similar up-regulation and down-regulation under a subset of conditions[15].

MovieLens Data Set

The MovieLens data set[22] was made available by the GroupLens Research Project at the University of Minnesota. The data set contains 100 000 ratings, 943 users and 1682 movies. Each user has rated at lease 20 movies. A user is considered as an object while a movie is regarded as an attribute. In the data set, many entries are empty since a user only rated less than 10%

movies on average.

5.2 Performance Analysis Using Synthetic Datasets

We evaluate the performance of the *pCluster* algorithm as we increase the number of rows and columns in the dataset. The results presented in Fig.10 are average response time obtained from a set of 10 synthetic datasets.

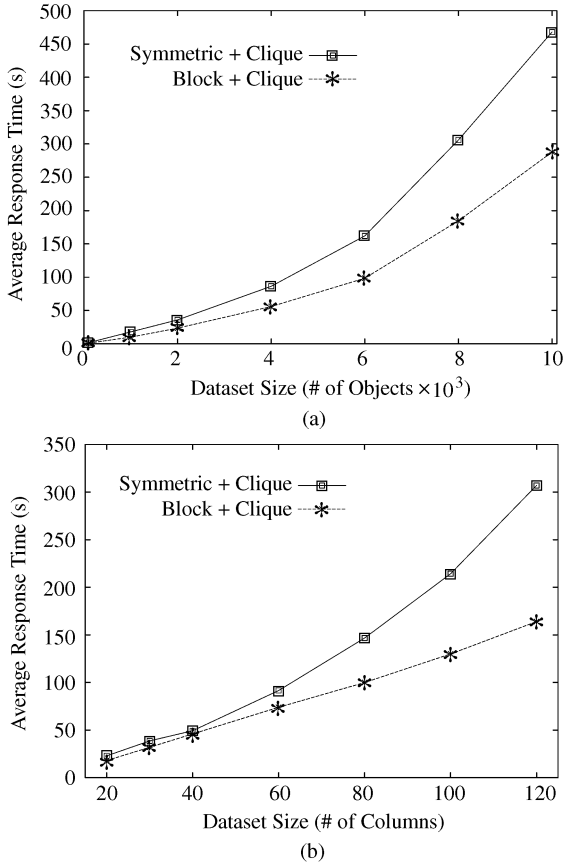

Fig.10. Performance study: pruning (2nd step). (a) Varying # of rows in data sets. (b) Varying # of columns in data sets.

As we know, the columns and the rows of the matrix carry the same significance in the *pCluster* model, which is symmetrically defined in (2). The performance of the algorithm, however, is not symmetric in terms of the number of columns and rows. Apparently, the algorithm based on block pruning is not symmetric, because it only generates object-pair MDSs. Although the algorithm based on symmetric pruning generates both types of MDSs using the same algorithm, one type of the MDSs (column-pair MDSs in our algorithm) has to be generated first, which breaks the symmetry in performance.

The synthetic data sets used for Fig.10(a) are generated with the number of columns fixed at 30. There is a total of 30 embedded *pCluster*s in the data. The mining algorithm is invoked with $\delta = 1$, $nc = 5$, and $nr = 0.01N$, where N is the number of rows of the synthetic data. Data sets used in Fig.10(b) are generated in the same manner, except that the number of rows is fixed at 3000. The mining algorithm is invoked with $\delta = 3$, $nr = 30$, and $nc = 0.02C$, where C is the number of columns of the data set.

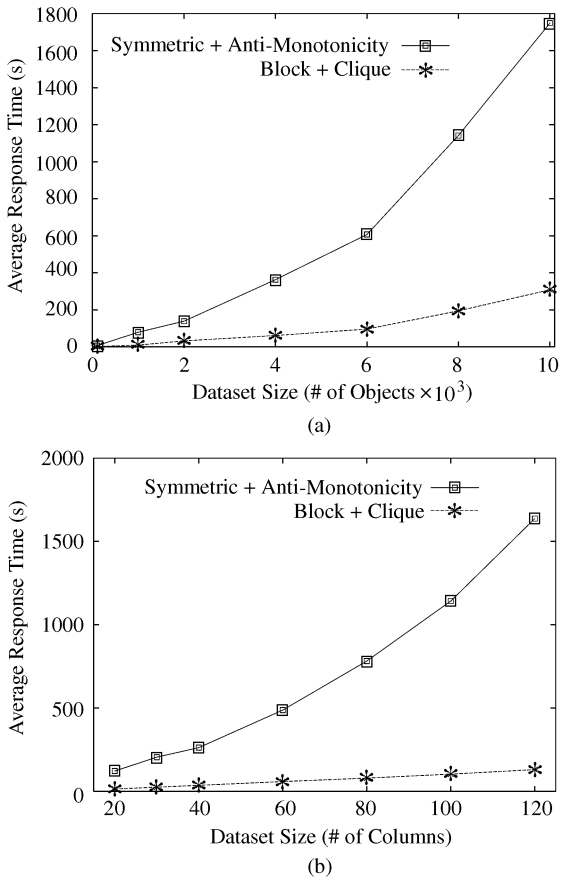

Fig.11. Performance study: pruning and clustering (2nd and 3rd step). (a) Varying # of columns in data sets. (b) Varying # of columns in data sets.

We first compare our approach with the approach in [20]. The two approaches differ in the 2nd and 3rd steps of the algorithm. We used block-based pruning in the 2nd step and clique-based clustering in the 3rd step, while the approach in [20] used symmetric-based pruning and direct clustering based on the anti-monotonicity property only. The pruning effectiveness and the advantage of the clique-based clustering method are demonstrated in Fig.11.

Second, we specifically focus on the pruning meth-

ods. The two approaches we compare in Fig.10 use the same clique-based clustering method but different pruning method. We find that block pruning outperforms symmetric pruning. Their differences become more significant when the dataset becomes larger. Particularly, in Fig.10(b), we find that the block pruning almost has linear performance, while symmetric pruning is clearly super linear with regard to the number of columns. However, it is clear that the performance difference is not as large as those shown in Fig.11. The above results demonstrate that, i) clique-based clustering is more efficient than the direct clustering; ii) block-based pruning is not only more efficient but also more effective — it prunes more invalid object-column pairs than the symmetric pruning method, which further improves the performance of clique-based clustering.

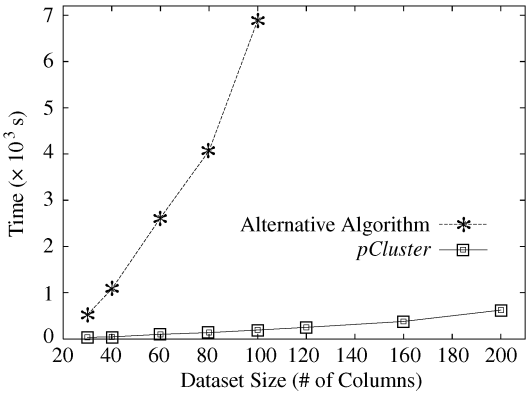

Fig.12. Subspace clustering vs. *pCluster*.

Finally, in Fig.12, we compare the *pCluster* algorithm with an alternative approach based on the subspace clustering algorithm CLIQUE[11]. The data set has 3000 objects and the subspace algorithm does not scale when the number of columns goes beyond 100.

5.3 Experimental Results on Real Life Datasets

We apply the *pCluster* algorithm on the yeast gene microarray[19]. First, we show that *pClusters* do exist in DNA microarrays.

Table 2. *pClusters* in Yeast DNA Microarray Data

δ	*nc*	*nr*	# of Maximum *pClusters*	# of *pClusters*
0	9	30	5	5520
0	7	50	11	–
0	5	30	9370	–

In Table 2, with different parameters of *nc* and *nr*, we find 5, 11, and 9370 pure *pClusters* ($\delta = 0$) in the Yeast DNA microarray data. Note that the entire set of *pClusters* is often huge (every subset of a *pCluster* is also a *pCluster*), and the *pCluster* algorithm only outputs maximum *pClusters*.

Next, we show the quality of the found *pClusters* and we compare them with those found by the bicluster approach[15] and the δ-cluster approach[16]. The results are shown in Table 3. We use each of the three approaches to find the top 100 clusters. Because it is unfair to compare their quality by the *pScore* measure used in our *pCluster* model, we use the residue measure, which is adopted by both the bicluster and the δ-cluster approaches. We found that the *pCluster* approach is able to find larger clusters with small residue, which means genes in the *pClusters* are more homogeneous.

Table 3. Quality of Clusters Mined from Yeast DNA Microarray Data

	Avg Residue	Avg Volume	Avg # of Genes	Avg # of Conditions
bicluster[15]	204.3	1577.0	167	12.0
δ-cluster[16]	187.5	1825.8	195	12.8
pCluster	164.4	1930.3	199	13.5

Third, we show some pattern similarities in the Yeast DNA microarray data and a *pCluster* that is based on such similarity. In Fig.13(a), we show a pairwise cluster, where the two genes, YGL106W and YAL046C, exhibit a clear shifting pattern under 14 out of 17 conditions. Fig.13(b) shows a *pCluster* where-gene YAL046C is a member. Clearly, these genes demonstrate pattern-based similarity under a subspace formed by set of the conditions. However, because of the limitation of the bicluster model and the random nature of the bicluster algorithm[15], the strong similarity between YGL106W and YAL046C that span 14 conditions is not revealed in any of the top 100 biclusters they discover. It is well known that YGL106W plays an important role② in the essential light chain for myosin Myo2p. Although YAL046C has no known function, it is reported[23] that gene X1.22067 in African clawed frog and gene Str.15194 in tropical clawed frog exhibit similarity to hypothetical protein YAL046C, and the transcribed sequence has 72.3% and 77.97% similarity to that of human. We cannot speculate whether YGL106W and YAL046C are related as they exhibit such high correlation, nevertheless, our goal is to pro-

②It may stabilize Myo2p by binding to the neck region; may interact with Myo1p, Iqg1p, and Myo2p to coordinate formation and contraction of the actomyosin ring with targeted membrane deposition.

pose better models and faster algorithms so that we can better serve the needs of biologists in discovering correlations among genes and proteins.

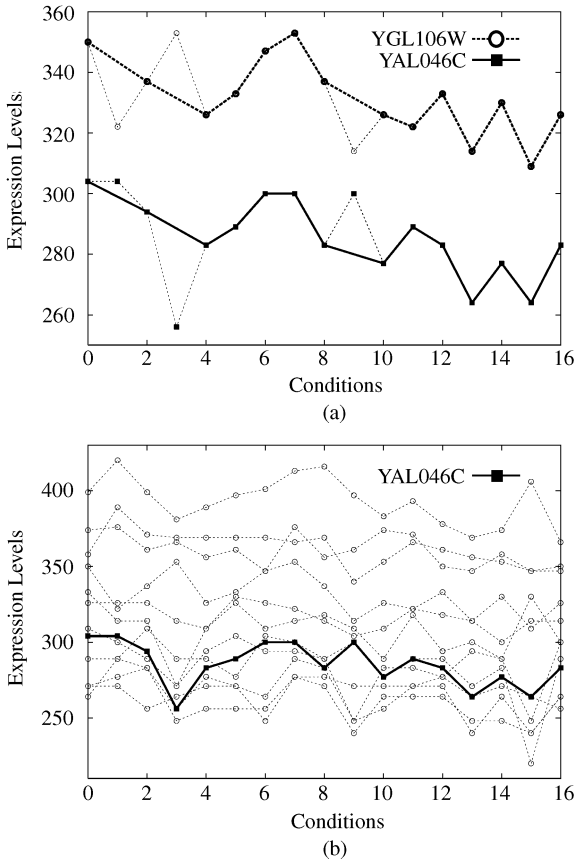

Fig.13. Pattern similarity and pCluster of genes.

In terms of response time, the majority of maximal dimension sets are eliminated during pruning. For Yeast DNA microarray data, the overall response time is around 80 seconds, depending on the user parameters. Our algorithm has performance advantage over the bicluster algorithm[15], as it takes roughly 300∼400 seconds for the bicluster algorithm to find a single cluster.

We also discovered some interesting *pCluster*s in the MovieLens dataset. For example, there is a cluster whose attributes consists of two types of movies, family movies (e.g., First Wives Club, Adam Family Values, etc.) and the action movies (e.g., Golden Eye, Rumble in the Bronx, etc.). Also the rating given by the viewers in this cluster is quite different, however, they share a common phenomenon: the rating of the action movies is about 2 points higher than that of the family movies. This cluster can be discovered in the *pCluster* model. For example, two viewers rate four movies as (3, 3, 4, 5) and (1, 1, 2, 3). Although the absolute distance between the two rankings are large, i.e., 4, but the *pCluster* model groups them together because they are coherent.

6 Conclusions

Recently, there has been considerable amount of research in subspace clustering. Most of the approaches define similarity among objects based on their distances (measured by distance functions, e.g., Euclidean) in some subspace. In this paper, we proposed a new model called *pCluster* to capture the similarity of the patterns exhibited by a cluster of objects in a subset of dimensions. Traditional subspace clustering, which focuses on value similarity instead of pattern similarity, is a special case of our generalized model. We devised a depth-first algorithm that can efficiently and effectively discover all the *pCluster*s with a size larger than a user-specified threshold.

The *pCluster* model finds a wide range of applications including management of scientific data, such as the DNA microarray, and e-commerce applications, such as collaborative filtering. In these datasets, although the distance among the objects may not be close in any subspace, they can still manifest shifting or scaling patterns, which are not captured by tradition (subspace) clustering algorithms. We have demonstrated that these patterns are often of great interest in DNA microarray analysis, collaborative filtering, and other applications.

As for future work, we believe the concept of similarity in pattern distance spaces has opened the door to quite a few research topics. For instance, currently, the similarity model used in data retrieval and nearest neighbor search is based on value similarity. By extending the model to reflect pattern similarity will benefit a lot of applications.

References

[1] Ester M, Kriegel H, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. SIGKDD*, 1996, pp.226–231.

[2] Ng R T, Han J. Efficient and effective clustering methods for spatial data mining. In *Proc. Santiago de Chile, VLDB*, 1994, pp.144–155.

[3] Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method for very large databases. In *Proc. SIGMOD*, 1996, pp.103–114.

[4] Murtagh F. A survey of recent hierarchical clustering algorithms. *The Computer Journal*, 1983, 26: 354–359.

[5] Michalski R S, Stepp R E. Learning from observation: Conceptual clustering. Machine Learning: An Artificial Intelligence Approach, Springer, 1983, pp.331–363.

[6] Fisher D H. Knowledge acquisition via incremental conceptual clustering. In *Proc. Machine Learning*, 1987.
[7] Fukunaga K. Introduction to Statistical Pattern Recognition. Academic Press, 1990.
[8] Beyer K, Goldstein J, Ramakrishnan R, Shaft U. When is nearest neighbors meaningful. In *Proc. the Int. Conf. Database Theories*, 1999, pp.217–235.
[9] Aggarwal C C, Procopiuc C, Wolf J, Yu P S, Park J S. Fast algorithms for projected clustering. In *Proc. SIGMOD*, Philadephia, USA, 1999, pp.61–72.
[10] Aggarwal C C, Yu P S. Finding generalized projected clusters in high dimensional spaces. In *Proc. SIGMOD*, Dallas, USA, 2000, pp.70–81.
[11] Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Authomatic subspace clustering of high dimensional data for data mining applications. In *Proc. SIGMOD*, 1998.
[12] Jagadish H V, Madar J, Ng R. Semantic compression and pattern extraction with fascicles. In *Proc. VLDB*, 1999, pp.186–196.
[13] Cheng C H, Fu A W, Zhang Y. Entropy-based subspace clustering for mining numerical data. In *Proc. SIGKDD*, San Diego, USA, 1999, pp.84–93.
[14] D'haeseleer P, Liang S, Somogyi R. Gene expression analysis and genetic network modeling. In *Proc. Pacific Symposium on Biocomputing*, Hawaii, 1999.
[15] Cheng Y, Church G. Biclustering of expression data. In *Proc. of 8th International Conference on Intelligent System for Molecular Biology*, 2000, pp.93–103.
[16] Yang J, Wang W, Wang H, Yu P S. δ-clusters: Capturing subspace correlation in a large data set. In *Proc. ICDE*, San Jose, USA, 2002, pp.517–528.
[17] Nagesh H, Goil H, Choudhary A. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University, 1999.
[18] Shardanand U, Maes P. Social information filtering: Algorithms for automating "word of mouth". In *Proc. ACM CHI*, Denver, USA, 1995, pp.210–217.
[19] Tavazoie S, Hughes J, Campbell M, Cho R, Church G. Yeast micro data set. http://arep.med.harvard.edu/biclustering/yeast.matrix, 2000.
[20] Wang H, Wang W, Yang J, Yu P S. Clustering by pattern similarity in large data sets. In *Proc. SIGMOD*, Madison, USA, 2002, pp.394–405.
[21] Niskanen S, Ostergard P R J. Cliquer user's guide, version 1.0. Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, 2003. http://www.hut.fi/p̃at/cliquer.html.
[22] Riedl J, Konstan J. Movielens dataset. In http://www.cs.umn.edu/Research/GroupLens.
[23] Clifton S, Johnson S, Blumberg B *et al.* Washington university Xenopus EST project. Technical Report, Washington University School of Medicine, 1999.

Haixun Wang is currently a research staff member at IBM T. J. Watson Research Center. He has been a technical assistant to Stuart Feldman, vice president of computer science of IBM Research, since 2006. He received the B.S. and M.S. degrees, both in computer science, from Shanghai Jiao Tong University in 1994 and 1996. He received the Ph.D. degree in computer science from the University of California, Los Angeles in 2000. His main research interest is database language and systems, data mining, and information retrieval. He has published more than 100 research papers in refereed international journals and conference proceedings. He has served regularly in the organization committees and program committees of many international conferences, and has been a reviewer for many leading academic journals in the database and data mining field.

Jian Pei received his Ph.D. degree in computing science from Simon Fraser University, Canada, in 2002. He is currently an assistant professor of computing science at Simon Fraser University, Canada. His research interests can be summarized as developing effective and efficient data analysis techniques for novel data intensive applications. Currently, he is interested in various techniques of data mining, data warehousing, online analytical processing, and database systems, as well as their applications in web search, sensor networks, bioinformatics, privacy preservation, software engineering, and education. His research has been supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the National Science Foundation (NSF) of the United States, Microsoft, IBM, Hewlett-Packard Company (HP), the Canadian Imperial Bank of Commerce (CIBC), and the SFU Community Trust Endowment Fund. He has published prolifically in refereed journals, conferences, and workshops. He is an associate editor of IEEE Transactions on Knowledge and Data Engineering. He has served regularly in the organization committees and the program committees of many international conferences and workshops, and has also been a reviewer for the leading academic journals in his fields. He is a senior member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). He is the recipient of the British Columbia Innovation Council 2005 Young Innovator Award, an IBM Faculty Award (2006), and an IEEE Outstanding Paper Award (2007).