# *MaPle*: A Fast Algorithm for Maximal Pattern-based Clustering[*]

Jian Pei  Xiaoling Zhang  Moonjung Cho  Haixun Wang  Philip S. Yu
State University of New York at Buffalo  IBM T.J. Watson Research Center
{jianpei, xzhang7, mcho}@cse.buffalo.edu  {haixun, psyu}@us.ibm.com

## Abstract

*Pattern-based clustering is important in many applications, such as DNA micro-array data analysis, automatic recommendation systems and target marketing systems. However, pattern-based clustering in large databases is challenging. On the one hand, there can be a huge number of clusters and many of them can be redundant and thus make the pattern-based clustering ineffective. On the other hand, the previous proposed methods may not be efficient or scalable in mining large databases.*

*In this paper, we study the problem of* maximal pattern-based clustering. *Redundant clusters are avoided completely by mining only the maximal pattern-based clusters.* MaPle, *an efficient and scalable mining algorithm is developed. It conducts a depth-first, divide-and-conquer search and prunes unnecessary branches smartly. Our extensive performance study on both synthetic data sets and real data sets shows that maximal pattern-based clustering is effective. It reduces the number of clusters substantially. Moreover,* MaPle *is more efficient and scalable than the previously proposed pattern-based clustering methods in mining large databases.*

## 1 Introduction

Clustering large databases is a challenging data mining task with many important applications. Most of the previously proposed methods are based on similarity measures defined globally on a (sub)set of attributes/dimensions. However, in some applications, it is hard or even infeasible to define a good similarity measure on a global subset of attributes to serve the clustering.

To appreciate the problem, let us consider clustering the $5$ objects in Figure 1(a). There are $5$ dimensions. No patterns among the $5$ objects are visibly explicit. However, as elaborated in Figure 1(b) and (c), respectively, objects $1$, $2$ and $3$ follow the same pattern in dimensions $a$, $c$ and $d$, while objects $1$, $4$ and $5$ share another similar pattern in dimensions $b$, $c$, $d$ and $e$. If we use the patterns as features, they form two *pattern-based clusters*.

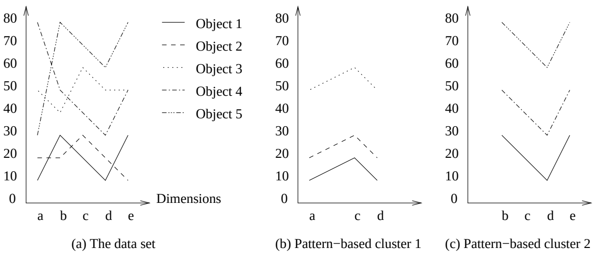


**Figure 1. A motivating example.**

Some recent researches (e.g., [12]) indicate that pattern-based clustering is useful in many applications. In general, given a set of data objects, a subset of objects form a pattern-based clusters if these objects follow a similar pattern in a subset of dimensions. Comparing to the conventional clustering, pattern-based clustering is a more general model and has two distinct features. On the one hand, *it does not require a globally defined similarity measure*. Different clusters can follow different patterns on different subsets of dimensions. On the other hand, *the clusters are not necessary exclusive*. That is, an object can appear in more than one cluster.

The generality and flexibility of pattern-based clustering may provide interesting and important insights in some applications where conventional clustering methods may meet difficulties. For example, in DNA micro-array data analysis, the gene expression data are organized as matrices, where rows represent genes and columns represent samples/conditions. The number in each cell records the expression level of the particular gene under the particular condition. The matrices are often large, containing thousands of genes and hundreds of conditions. It is important to identify subsets of genes whose expression levels change coherently under a subset of conditions. Such information is critical in revealing the significant connections in gene regulatory networks. As another example, in the applications of automatic recommendation and target marketing, it is essential to identify sets of customers/clients with similar behavior/interest. As a concrete example, suppose that the ranks of movies given by customers are collected. To identify customer groups, it is essential to find the subsets

[*]This research is partially supported by NSF Grant IIS-0308001. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

of customers who rank subsets of movies similarly. In the above two examples, pattern-based clustering is the major data mining task.

Although the pattern-based clustering problem is proposed and a mining algorithm is developed by Wang et al. [12], some important problems remain not thoroughly explored. In particular, we address the following two fundamental issues and make corresponding contributions in this paper.

First, *what is the effective representation of pattern-based clusters?* As can be imagined, there can exist many pattern-based clusters in a large database. Given a pattern-based cluster $C$, any non-empty subset of the objects in the cluster is trivially a pattern-based cluster on any non-empty subset of dimensions. Mining and analyzing a huge number of pattern-based clusters may become the bottleneck of effective analysis. *Can we devise a non-redundant representation of the pattern-based clusters?*

**Our contributions.** In this paper, we propose the mining of *maximal pattern-based clusters*. The idea is to report only those non-redundant pattern-based clusters, and skip their trivial sub-clusters. We show that, by mining maximal pattern-based clusters, the number of clusters can be reduced substantially. Moreover, many unnecessary searches for sub-clusters can be pruned and thus the mining efficiency can be improved dramatically as well.

Second, *how to mine the maximal pattern-based clusters efficiently?* Our experimental results indicate that the algorithm *p-Clustering* developed in [12] may not be satisfactorily efficient or scalable in large databases. The major bottleneck is that it has to search many possible combinations of objects and dimensions.

**Our contributions.** In this paper, we develop a novel mining algorithm, *MaPle* (for Maximal Pattern-based Clustering). It conducts a depth-first, progressively refining search to mine maximal pattern-based clusters. We propose techniques to guarantee the completeness of the search and also prune unpromising search branches. An extensive performance study on both synthetic data sets and real data sets is reported. The results show that *MaPle* is significantly more efficient and more scalable in mining large databases than method *p-Clustering* in [12].

The remainder of the paper is organized as follows. Section 2 defines the problem of mining maximal pattern-based clusters and reviews related work. In Section 3, we develop algorithm *MaPle*. An extensive performance study is reported in Section 4. Section 5 concludes the paper.

## 2 Problem Definition and Related Work

Given a set of objects, where each object is described by a set of attributes. A pattern-based cluster $(R, D)$ is a subset of objects $R$ that exhibit a coherent pattern on a subset of attributes $D$. To formulate the problem, it is essential to describe, given a subset of objects $R$ and a subset of attributes $D$, how coherent the objects are on the attributes. The measure $pScore$ serves this purpose.

**Definition 2.1 (pScore)** Let $DB = \{r_1, \ldots, r_n\}$ be a database with $n$ *objects*. Each object has $m$ *attributes* $A = \{a_1, \ldots, a_m\}$. We assume that each attribute is in the domain of real numbers. The value of object $r_j$ on attribute $a_i$ is denoted as $r_j.a_i$. For any objects $r_x, r_y \in DB$ and any attributes $a_u, a_v \in A$, the $pScore$ is defined as $pScore\left(\begin{bmatrix} r_x.a_u & r_x.a_v \\ r_y.a_u & r_y.a_v \end{bmatrix}\right) = \|(r_x.a_u - r_y.a_u) - (r_x.a_v - r_y.a_v)\|$.

Clearly, the $pScore$ describes the similarity between two objects on two attributes. The smaller the $pScore$ value, the more similar are the two objects on the two dimensions. Pattern-based clusters can be defined as follows.

**Definition 2.2 (Pattern-based cluster)** Let $R \subseteq DB$ be a subset of objects in the database and $D \subseteq A$ be a subset of attributes. $(R, D)$ is said a *$\delta$-pCluster* (for pattern-based cluster) if for any objects $r_x, r_y \in R$ and any attributes $a_u, a_v \in D$, $pScore\left(\begin{bmatrix} r_x.a_u & r_x.a_v \\ r_y.a_u & r_y.a_v \end{bmatrix}\right) \leq \delta$, where $\delta \geq 0$.

In a large database with many attributes, there can be many coincident, statistically insignificant pattern-based clusters. A cluster may be considered *statistically insignificant* if it contains a small number of objects, or a small number of attributes. Thus, a user may want to impose constraints on the minimum numbers of objects and attributes in a pattern-based cluster.

In general, given (1) a cluster threshold $\delta$, (2) an *attribute threshold* $min_a$ (i.e., the minimum number of attributes), and (3) an *object threshold* $min_o$ (i.e., the minimum number of objects), the task of *mining $\delta$-pClusters* is to find the complete set of $\delta$-pClusters $(R, D)$ such that $(\|R\| \geq min_o)$ and $(\|D\| \geq min_a)$. A $\delta$-pCluster satisfying the above requirement is called *significant*.

Although the attribute and object thresholds are used to filter out insignificant pClusters, there still can be some "*redundant*" significant pClusters. For example, consider the objects in Figure 1. Let $\delta = 5$, $min_a = 3$ and $min_o = 3$. Then, we have 6 significant pClusters: $C_1 = (\{1,2,3\}, \{a,c,d\})$, $C_2 = (\{1,4,5\}, \{b,c,d\})$, $C_3 = (\{1,4,5\}, \{b,c,e\})$, $C_4 = (\{1,4,5\}, \{b,d,e\})$, $C_5 = (\{1,4,5\}, \{c,d,e\})$, and $C_6 = (\{1,4,5\}, \{b,c,d,e\})$. Among them, $C_2$, $C_3$, $C_4$ and $C_5$ are subsumed by $C_6$, i.e., the objects and attributes in the four clusters, $C_2$-$C_5$, are subsets of the ones in $C_6$.

In general, a pCluster $C_1 = (R_1, D_1)$ is called a *sub-cluster* of $C_2 = (R_2, D_2)$ provided $(R_1 \subseteq R_2) \wedge (D_1 \subseteq D_2)$. Moreover, $C_1$ is called a *proper sub-cluster* of $C_2$ if either $R_1 \subset R_2$ or $D_1 \subset D_2$. Pattern-based clusters have the following property.

**Lemma 2.1 (Closure of sub-clusters)** *Let $C = (R, D)$ be a $\delta$-pCluster. Then, every sub-cluster $(R', D')$ is a $\delta$-pCluster.*

Mining the redundant sub-clusters is tedious and ineffective for analysis. Therefore, it is natural to mine only the "maximal clusters", i.e., the pClusters that are not sub-cluster of any other pClusters.

**Definition 2.3 (maximal pCluster)** A $\delta$-pCluster $C$ is said *maximal* (or called a $\delta$-MPC in short) if there exists no $\delta$-pCluster $C'$ such that $C$ is a proper sub-cluster of $C'$.

**Problem Statement (mining maximal $\delta$-pClusters).** Given (1) a cluster threshold $\delta$, (2) an attribute threshold $min_a$, and (3) an object threshold $min_o$, the task of *mining maximal $\delta$-pClusters* is to find the complete set of maximal $\delta$-pClusters with respect to $min_a$ and $min_o$.

### 2.1 Related Work

The problem of pattern-based clustering and an algorithm, *p-Clustering*[1], are proposed in [12]. According to the extensive performance study reported in the paper, *p-Clustering* outperforms all previous methods.

The study of pattern-based clustering is related to previous work on subspace clustering and frequent itemset mining.

The meaning of clustering in high dimensional data sets is often unreliable [6]. Some recent studies (e.g. [3, 1, 2, 7]) focus on mining clusters embedded in some subspaces. For example, CLIQUE [3] is a density and grid based method. It divides the data into hyper-rectangular cells and uses the dense cells to construct subspace clusters.

Subspace clustering can be used to semantically compress data. An interesting study in [10] employs a randomized algorithm to find fascicles, the subsets of data that share similar values in some attributes. While their method is effective for compression, it does not guarantee the completeness of mining the clusters.

In some applications, global similarity-based clustering may not be effective. Still, strong correlations may exist among a set of objects even if they are far away from each other as measured by distance functions (such as Euclidean) used frequently in traditional clustering algorithms. Many scientific projects collect data in the form of Figure 1(a), and it is essential to identify clusters of objects that manifest coherent patterns. A variety of applications, including DNA microarray analysis, collaborative filtering, will benefit from fast algorithms that can capture such patterns.

Cheng and Church propose the biclustering model [8], which captures the coherence of genes and conditions in a sub-matrix of a DNA micro-array. Yang et al. [13] develop a move-based algorithm to find biclusters more efficiently.

On the other hand, a transaction database can be modelled as a binary matrix, where columns and rows stand for items and transactions, respectively. A cell $r_{i,j}$ is set to 1 if item $j$ is contained in transaction $i$. Then, the problem of mining frequent itemsets [4] is to find subsets of rows and columns such that the sub-matrix is all 1's, and the number of rows is more than a given support threshold. If a minimum length constraint $min_a$ is imposed to find only frequent itemsets of no less than $min_a$ items, then it becomes a problem of mining $0$-pClusters on binary data. Although there are many efficient methods for frequent itemset mining, such as [5, 9], they cannot be extended to handle the general pattern-based clustering problem since they can only handle the binary data.

[1]Wang et al. did not give a specific name to their algorithm in [12]. We call it *p-Clustering* since the main function in the algorithm is *pCluster*() and we want to distinguish the algorithm from the pclusters.

## 3 Algorithm *MaPle*

### 3.1 Overview

Essentially, *MaPle* enumerates all the maximal pClusters systematically. It guarantees both the completeness and the non-redundancy of the search, i.e., every maximal pCluster will be found, and each combination of attributes and objects will be tested at most once.

*MaPle* enumerates every combination of attributes systematically in a dictionary order according to an order of attributes. For each subset of attributes $D$, *MaPle* finds the maximal subsets of objects $R$ such that $(R, D)$ is a $\delta$-pCluster. If $(R, D)$ is not a sub-cluster of another pCluster $(R', D)$ such that $R \subset R'$, then $(R, D)$ is a maximal $\delta$-pCluster. This "*attribute-first-object-later*" search is illustrated in Figure 2.

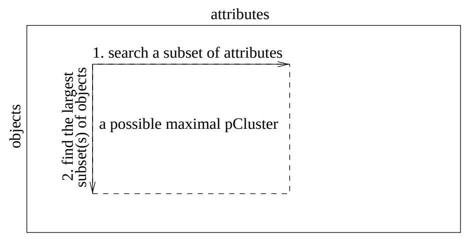

attributes
1. search a subset of attributes
objects
2. find the largest subset(s) of objects
a possible maximal pCluster

**Figure 2. Attribute-first-object-later search.**

There can be a huge number of combinations of attributes. *MaPle* prunes many combinations unpromising for $\delta$-pClusters. Following Lemma 2.1, for subset of attributes $D$, if there exists no subset of objects $R$ such that $(R, D)$ is a significant pCluster, then we do not need to search any superset of $D$. On the other hand, when searching under a subset of attributes $D$, *MaPle* only checks those subsets of objects $R$ such that $(R, D')$ is a pCluster for every $D' \subset D$. Clearly, only subsets $R' \subseteq R$ may achieve $\delta$-pCluster $(R', D)$. Such pruning techniques are applied recursively. Thus, *MaPle* progressively refines the search step by step.

Moreover, *MaPle* also prunes searches that are unpromising to find maximal pClusters. It detects the attributes and objects that can be used to assemble a larger pCluster from the current pCluster. If *MaPle* finds that the current subsets of attributes and objects as well as all possible attributes and objects together turn out to be a sub-cluster of a pCluster having been found before, then the recursive searches rooted at the current node are pruned, since it cannot lead to a maximal pCluster.

*Why does* MaPle *enumerate attributes first and then objects later, but not in the reverse way?* In real databases, the number of objects is often much larger than the number of attributes. In other words, the number of combinations of objects is often dramatically larger than the number of combinations of attributes. In the pruning using maximal pClusters discussed above, if the attribute-first-object-later approach is adopted, once a set of attributes and its descendants are pruned, all searches of related subsets of objects are pruned as well. Heuristically, the attribute-first-object-later search may bring a better chance to prune a more bushy search sub-tree.[2]

Essentially, we rely on MDSs, the maximal $\delta$-MPC containing only two objects or two attributes, to determine whether a subset of objects and a subset of attributes together form a pCluster. Therefore, as a preprocessing, we materialize all non-redundant MDSs.

Based on the above discussion, we have the framework of *MaPle* as shown in Figure 3.

```
Input: database $DB$, cluster threshold $\delta$, attribute threshold
       $min_a$ and object threshold $min_o$;
Output: the complete set of maximal $\delta$-pClusters;
Method:
(1)  compute and prune attribute-pair MDSs and object-pair
       MDSs; // Section 3.2
(2)  progressively refining, depth-first search for maximal
       $\delta$-pClusters; // Section 3.3
```

**Figure 3. Algorithm *MaPle*.**

Comparing to *p-Clustering*, *MaPle* has several advantages. First, in the third step of *p-Clustering*, for each node in the prefix tree, the combinations of the objects registered in the node will be explored to find pClusters. This can be expensive if there are many objects in a node. In *MaPle*, the information of pClusters is inherited from the "parent node" in the depth-first search and the possible combinations of objects can be reduced substantially. Moreover, once a subset of attributes $D$ is determined hopeless for pClusters, the searches of any superset of $D$ will be pruned. Second, *MaPle* prunes non-maximal pClusters. Many unpromising searches can be pruned in their early stages. Third, new pruning techniques are adopted in the computing and pruning MDSs. That also speeds up the mining.

In the remainder of this section, we will explain the two steps of *MaPle* in detail.

[2]However, there is no theoretical guarantee that the attribute-first-object-later search is optimal. There exist counter examples that object-first-attribute-later search wins. Limited by space, we omit the details here.

### 3.2 Computing and Pruning MDSs

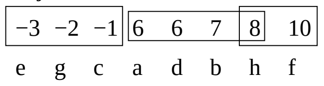
A pCluster must have at least two objects and two attributes. Intuitively, we can use those pClusters containing only two objects or two attributes to construct larger pClusters having more objects and attributes. Given a database $DB$ and a cluster threshold $\delta$. A $\delta$-pCluster $C_1 = (\{o_1, o_2\}, D)$ is called an *object-pair MDS* (for maximal dimension set) if there exists no $\delta$-pCluster $C_1' = (\{o_1, o_2\}, D')$ such that $D \subset D'$. On the other hand, a $\delta$-pCluster $C_2 = (R, \{a_1, a_2\})$ is called an *attribute-pair MDS* if there exists no $\delta$-pCluster $C_2' = (R', \{a_1, a_2\})$ such that $R \subset R'$.

*MaPle* computes all attribute-pair MDSs as *p-Clustering* does. The method is illustrated in Figure 4(b). Limited by space, we omit the detailed algorithm here and only show the following example.

**Example 1 (Finding attribute-pair MDSs)** Figure 4(a) shows the object values of two attributes, $x$ and $y$. The last row shows the differences of the object values.

| Attribute | Objects | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ |
| $x$ | 13 | 11 | 9 | 7 | 9 | 13 | 2 | 15 |
| $y$ | 7 | 4 | 10 | 1 | 12 | 3 | 4 | 7 |
| $x - y$ | 6 | 7 | $-1$ | 6 | $-3$ | 10 | $-2$ | 8 |

(a) The object values of two attributes $x$ and $y$.



(b) Finding MDS

**Figure 4. Finding MDS for two attributes.**

To compute the attribute-pair MDS, *p-Clustering* sorts the objects in the difference ascending order, as shown in Figure 4(b). Suppose $\delta = 2$. *P-Clustering* runs through the sorted list using a sliding window of variable width. The objects in the sliding window form a $\delta$-pCluster provided the difference between the rightmost element and the leftmost one is no more than $\delta$. For example, *p-Clustering* firstly sets the left edge of the sliding window at the left end of the sorted list, and moves rightward until it sees the first 6. The objects in between, $\{e, g, c\}$, is the set of objects of an attribute-pair MDS. Then, *p-Clustering* moves the left edge of the sliding window to object $g$, and repeats the process until the left end of the window runs through all elements in the list. In total, three MDSs can be found, i.e., $(\{x, y\}, \{e, g, c\})$, $(\{x, y\}, \{a, d, b, h\})$ and $(\{x, y\}, \{h, f\})$. A similar method can be used to find the object-pair MDSs.

As our running example, consider mining maximal pattern-based clusters in a database $DB$ as shown in Figure 5(a). Suppose $min_a = 3$, $min_o = 3$ and $\delta = 1$. For each pair of attributes, we calculate the attribute pair MDSs. The attribute-pair MDSs returned are shown in Figure 5(b).

We can also generate all the object-pair MDSs similarly. However, we can speed up the calculation of object-pair

| Object | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $o_1$ | 5 | 6 | 7 | 7 | 1 |
| $o_2$ | 4 | 4 | 5 | 6 | 10 |
| $o_3$ | 5 | 5 | 6 | 1 | 30 |
| $o_4$ | 7 | 7 | 15 | 2 | 60 |
| $o_5$ | 2 | 0 | 6 | 8 | 10 |
| $o_6$ | 3 | 4 | 5 | 5 | 1 |

(a) The database

| Objects | Attribute-pair |
|---|---|
| $\{o_1, o_2, o_3, o_4, o_6\}$ | $\{a_1, a_2\}$ |
| $\{o_1, o_2, o_3, o_6\}$ | $\{a_1, a_3\}$ |
| $\{o_1, o_2, o_6\}$ | $\{a_1, a_4\}$ |
| $\{o_1, o_2, o_3, o_6\}$ | $\{a_2, a_3\}$ |
| $\{o_1, o_2, o_6\}$ | $\{a_2, a_4\}$ |
| $\{o_1, o_2, o_6\}$ | $\{a_3, a_4\}$ |

(b) The attribute-pair MDSs

**Figure 5. The running example.**

| Object-pair | Attributes |
|---|---|
| $\{o_1, o_2\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_1, o_3\}$ | $\{a_1, a_2, a_3\}$ |
| $\{o_1, o_6\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_2, o_3\}$ | $\{a_1, a_2, a_3\}$ |
| $\{o_2, o_6\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_3, o_6\}$ | $\{a_1, a_2, a_3\}$ |

(a) Object-pair MDSs in *MaPle*.

| Object-pair | Attributes |
|---|---|
| $\{o_1, o_2\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_1, o_3\}$ | $\{a_1, a_2, a_3\}$ |
| $\{o_1, o_6\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_2, o_3\}$ | $\{a_1, a_2, a_3\}$ |
| $\{o_2, o_6\}$ | $\{a_1, a_2, a_3, a_4\}$ |
| $\{o_3, o_4\}$ | $\{a_1, a_2, a_4\}$ |
| $\{o_3, o_6\}$ | $\{a_1, a_2, a_3\}$ |

(b) Object-pair MDSs in method *p-Clustering*

**Figure 6. Pruning using Lemma 3.1.**

MDSs by utilizing the information on the number of occurrences of objects and attributes in the attribute-pair MDSs.

**Lemma 3.1 (Pruning MDSs)** *Given a database $DB$ and a cluster threshold $\delta$, object threshold $min_o$ and attribute threshold $min_a$. (1) An attribute $a$ cannot appear in any significant $\delta$-pCluster if $a$ appears in less than $\frac{min_o \cdot (min_o - 1)}{2}$ object-pair MDSs, or appears in less than $(min_a - 1)$ attribute-pair MDSs; (2) An object $o$ cannot appear in any significant $\delta$-pCluster if $o$ appears in less than $\frac{min_a \cdot (min_a - 1)}{2}$ attribute-pair MDSs, or appears in less than $(min_o - 1)$ object-pair MDSs.*

**Example 2 (Pruning using Lemma 3.1)** Let us check the attribute-pair MDSs in Figure 5(b). Object $o_5$ does not appear in any attribute-pair MDS, and object $o_4$ appears in only 1 attribute-pair MDS. According to Lemma 3.1, $o_4$ and $o_5$ cannot appear in any significant $\delta$-pCluster. Therefore, object-pairs containing $o_4$ or $o_5$ can be pruned.

There are 6 objects in the database. Without this pruning, we have to check $\frac{6 \times 5}{2} = 15$ pairs of objects. With this pruning, only four objects, $o_1$, $o_2$, $o_3$ and $o_6$ survive. Thus, we only need to check $\frac{4 \times 3}{2} = 6$ pairs of objects. $60\%$ of the original searches are pruned.

Moreover, since attribute $a_5$ does not appear in any attribute-pair MDS, it cannot appear in any significant $\delta$-pCluster. The attribute can be pruned, i.e., $a_5$ can be removed from any object-pair MDS.

In summary, after the pruning, only attributes $a_1$, $a_2$, $a_3$ and $a_4$, and objects $o_1$, $o_2$, $o_3$ and $o_6$ survive. We use these attributes and objects to generate object-pair MDSs. The result is shown in Figure 6(a). In method *p-Clustering*, it uses all attributes and objects to generate object-pair MDSs. The result is shown in Figure 6(b). As can be seen, not only the computation cost in *MaPle* is less, the number of object-pair MDSs in *MaPle* is also one less than that in method *p-Clustering*.

Once we get the initial object-pair MDSs and attribute-pair MDSs, we can conduct a mutual pruning between the object-pair MDSs and the attribute-pair MDSs, as method *p-Clustering* does. Furthermore, Lemma 3.1 can be applied in each round to get extra pruning. The pruning algorithm is shown in Figure 7.

```
(1) REPEAT
(2)     count the number of occurrences of objects and attributes
          in the attribute-pair MDSs;
(3)     apply Lemma 3.1 to prune objects and attributes;
(4)     remove object-pair MDSs containing less than $min_a$
          attributes;
(5)     count the number of occurrences of objects and attributes
          in the object-pair MDSs;
(6)     apply Lemma 3.1 to prune objects and attributes;
(7)     remove attribute-pair MDSs containing less than $min_o$
          objects;
(8) UNTIL no pruning takes place
```

**Figure 7. The algorithm of pruning MDSs.**

## 3.3 Progressively Refining, Depth-first Search of Maximal pClusters

The algorithm of the progressively refining, depth-first search of maximal pClusters is shown in Figure 8. We explain the algorithm step by step in this subsection.

### 3.3.1 Dividing Search Space

By a list of attributes, we can enumerate all combinations of attributes systematically. The idea is shown in the following example.

**Example 3 (Enumeration of combinations of attributes)** In our running example, there are four attributes surviving from the pruning: $a_1$, $a_2$, $a_3$ and $a_4$. We list any subset of attributes in the order of $a_1$-$a_2$-$a_3$-$a_4$. Suppose that $min_a = 3$, i.e., every maximal $\delta$-pCluster should have at

```
(1)  let $n$ be the number of attributes;
     make up an attribute list $AL = a_1 \cdots a_n$;
(2)  FOR $i = 1$ TO $n - min_a + 1$ DO
(3)    FOR $j = i + 1$ TO $n - min_a + 2$ DO
(4)      find row-maximal pClusters $(R, \{a_i, a_j\})$;
           //Section 3.3.2
(5)      FOR EACH row-maximal pCluster $(R, \{a_i, a_j\})$ DO
(6)        call $search(R, \{a_i, a_j\})$;
(7)      END FOR EACH
(8)    END FOR
(9)  END FOR
(10)
(11) FUNCTION $search(R, D)$;
       // $(R, D)$ is a row-maximal pCluster.
(12)   compute $PD$, the set of possible attributes;
         //Optimization 1 in Section 3.3.3
(13)   apply optimizations in Section 3.3.3 to prune, if possible;
(14)   FOR EACH attribute $a \in PD$ DO
(15)     find row-maximal pClusters $(R', D \cup \{a\})$;
           //Section 3.3.2
(16)     FOR EACH row-maximal pCluster $(R', D \cup \{a\})$ DO
(17)       call $search(R', D \cup \{a\})$;
(18)     END FOR EACH
(19)   END FOR EACH
(20)   IF $(R, D)$ is not a subcluster of some maximal pCluster
         having been found
(21)   THEN output $(R, D)$;
(22) END FUNCTION
```

**Figure 8. Projection-based search.**

least 3 attributes. We divide the complete set of maximal pClusters into 3 exclusive subsets according to the first two attributes in the pClusters: (1) the ones having attributes $a_1$ and $a_2$, (2) the ones having attributes $a_1$ and $a_3$ but no $a_2$, and (3) the ones having attributes $a_2$ and $a_3$ but no $a_1$.

In general, the set of maximal pClusters can be divided into exclusive subsets by a list of attributes. Heuristically, for an attribute $a$, if there are many distinct objects appearing in the attribute-pair MDSs containing $a$, then it is likely that $a$ may appear in a maximal pCluster of large size (i.e., a maximal pCluster containing many objects and attributes). Such attributes should be considered first in our search. Based on the heuristic, given a database $DB$, the *rank* of an attribute $a$ is the number of distinct objects in the attribute-pair MDSs containing $a$. That is, $rank(a) = \| \bigcup_{(R,D)|a \in D} R \|$, where $(R, D)$ is an attribute-pair MDS. The list of all attributes in the database in rank descending order is called the *attribute-list* of $DB$. Attributes having an identical rank can be sorted arbitrarily.

For example, from the attribute-pair MDSs in Figure 5(b), we can compute the ranks of the attributes. The ranks of $a_1$, $a_2$, $a_3$ and $a_4$ are 4, 4, 4 and 3, respectively. Thus, we make up the attribute-list as $a_1$-$a_2$-$a_3$-$a_4$. We will use this list to search for maximal $\delta$-pClusters in our running example.

Since a pCluster has at least 2 attributes, *MaPle* first partitions the complete set of maximal pClusters into exclusive subsets according to the first two attributes, then searches the subsets one by one in the depth-first manner. For each subset, *MaPle* further divides the pClusters in the subset into smaller exclusive sub-subsets according to the third attributes in the pClusters, and searches the sub-subsets. Such a process proceeds recursively until all the maximal pClusters are found. This is implemented by line (1)-(3) and (14) in Figure 8.

### 3.3.2 Finding Row-maximal pClusters

Now, the problem becomes how to find the maximal $\delta$-pClusters on the subsets of attributes. For each subset of attributes $D$, we will find the maximal subsets of objects $R$ such that $(R, D)$ is a pCluster. Such a pCluster is a maximal pCluster if it is not a sub-cluster of some others.

Given a set of attributes $D$ such that $(\|D\| \geq 2)$. A pCluster $(R, D)$ is called a *row-maximal $\delta$-pCluster* if there exists no any $\delta$-pCluster $(R', D)$ such that $R \subset R'$. In other words, a row-maximal pCluster is maximal in the sense that no more objects can be included so that the objects are still coherent on the same subset of attributes. For example, in the database shown in Figure 5(a), $(\{o_1, o_2, o_3, o_6\}, \{a_1, a_2\})$ is a row-maximal pCluster for subset of attributes $\{a_1, a_2\}$. Clearly, a maximal pCluster must be a row-maximal pCluster, but not vice versa.

*Given a subset of attributes $D$, how can we find all row-maximal pClusters efficiently?* There are two cases.

If $D$ has only two attributes, then the row-maximal pClusters are the attribute-pair MDSs for $D$. Since the MDSs are computed and stored before the search, they can be retrieved immediately.

Now, let us consider the case where $(\|D\| \geq 3)$. Suppose $D = \{a_{i_1}, \ldots, a_{i_k}\}$ where the attributes in $D$ are listed in the order of attribute-list $AL$. Intuitively, $(R, D)$ is a pCluster if $R$ is shared by attribute-pair MDSs from any two attributes from $D$. $(R, D)$ is a row-maximal pCluster if $R$ is a maximal set of objects.

One tricky thing here is that, in general, there can be more than one attribute-pair MDS for given attributes $a_u, a_v$. Thus, there can be more than one row-maximal pCluster on a subset of attributes $D$. Technically, $(R, D)$ is a row-maximal pCluster if for each pair of attributes $\{a_u, a_v\} \subset D$, there exists an attribute-pair MDS $(\{a_u, a_v\}, R_{uv})$, such that $R = \bigcap_{\{a_u, a_v\} \subset D} R_{uv}$.

Recall that *MaPle* searches the combinations of attributes in the depth-first manner, all row-maximal pClusters for subset of attributes $D - \{a_{ik}\}$ is found before we search for $D$. Therefore, we only need to find the subset of objects in a row-maximal pCluster of $D - \{a_{ik}\}$ that are shared by attribute-pair MDSs of $a_{i_j}, a_{i_k}$ $(j < k)$.

### 3.3.3 Pruning and Optimizations

Several optimizations can be used to prune the search so that the mining can be more efficient.

**Optimization 1: Only *possible attributes* should be considered to get larger pClusters.**

Suppose that $(R, D)$ is a row-maximal pCluster. *For every attribute $a$ such that $a$ is after all attributes in $D$ in the attribute-list, under what condition can we find a significant pCluster $(R', D \cup \{a\})$ such that $R' \subseteq R$?*

If $(R', D \cup \{a\})$ is significant, i.e., has at least $min_o$ objects, then $a$ must appear in at least $\frac{min_o(min_o-1)}{2}$ object-pair MDSs $(\{o_i, o_j\}, D_{ij})$ such that $\{o_i, o_j\} \subseteq R'$. In other words, for an attribute $a$ that appears in less than $\frac{min_o(min_o-1)}{2}$ object-pair MDSs of objects in $R$, there exists no row-maximal pCluster with respect to $D \cup \{a\}$.

Based on the above observation, an attribute $a$ is called a *possible attribute* with respect to row-maximal pCluster $(R, D)$ if $a$ appears in $\frac{min_o(min_o-1)}{2}$ object-pair MDSs $(\{o_i, o_j\}, D_{ij})$ such that $\{o_i, o_j\} \subseteq R$. In line (12) of Figure 8, we compute possible attributes and only those attributes are used to extend the set of attributes in pClusters.

**Optimization 2: Pruning local maxiaml pClusters having insufficient possible attributes.**

Suppose that $(R, D)$ is a row-maximal pCluster. Let $PD$ be the set of possible attributes with respect to $(R, D)$. Clearly, if $\|D \cup PD\| < min_a$, then it is impossible to find any maximal pCluster of a subset of $R$. Thus, such a row-maximal pCluster should be discarded and all the recursive search can be pruned.

**Optimization 3: Extracting common attributes from possible attribute set directly.**

Suppose that $(R, D)$ is a row-maximal pCluster with respect to $D$, and $D'$ is the corresponding set of possible attributes. If there exists an attribute $a \in D'$ such that for every pair of objects $\{o_i, o_j\}$, $\{a\} \cup D$ appears in an object pair MDS of $\{o_i, o_j\}$, then we immediately know that $(R, D \cup \{a\})$ must be a row-maximal pCluster with respect to $D \cup \{a\}$. Such an attribute is called a *common attribute* and should be extracted directly.

**Example 4 (Extracting common attributes)** In our running example, $(\{o_1, o_2, o_3, o_6\}, \{a_1, a_2\})$ is a row-maximal pCluster with respect to $\{a_1, a_2\}$. Interestingly, as shown in Figure 6(a), for every object pair $\{o_i, o_j\} \subset \{o_1, o_2, o_3, o_6\}$, the object-pair MDS contains attribute $a_3$. Therefore, we immediately know that $(\{o_1, o_2, o_3, o_6\}, \{a_1, a_2, a_3\})$ is a row-maximal pCluster.

**Optimization 4: Prune non-maximal pClusters.**

Our goal is to find maximal pClusters. Once we know that the recursive search on a row-maximal pCluster cannot lead to a maximal pCluster, the recursive search thus can be pruned. The earlier we detect the impossibility, the more search efforts can be saved. We can use the *dominant attributes* to detect the impossibility. We illustrate the idea in the following example.

**Example 5 (Detect non-maximal pClusters)** In our running example, let us try to find the maximal pClusters whose first two attributes are $a_1$ and $a_3$. Following the above discussion, we identify a row-maximal pCluster $(\{o_1, o_2, o_3, o_6\}, \{a_1, a_3\})$.

One interesting observation from the object-pair MDSs on objects in $\{o_1, o_2, o_3, o_6\}$ (Figure 6(a)): attribute $a_2$ appears in every object pair. We called $a_2$ a *dominant attribute*. That means $\{o_1, o_2, o_3, o_6\}$ also coherent on attribute $a_2$. In other words, we cannot have a maximal pCluster whose first two attributes are $a_1$ and $a_3$, since $a_2$ must also be in the same maximal pCluster. Thus, the search of maximal pClusters whose first two attributes are $a_1$ and $a_3$ can be pruned.

The idea in Example 5 can be generalized. Suppose $(R, D)$ is a row-maximal pCluster. If there exists an attribute $a$ such that $a$ is before the last attribute in $D$ according to the attribute-list, and $\{a\} \cup D$ appears in an object-pair MDS $(\{o_i, o_j\}, D_{ij})$ for every $(\{o_i, o_j\} \subseteq R)$, then the search from $(R, D)$ can be pruned, since there cannot be a maximal pCluster having attribute set $D$ but no $a$. Attribute $a$ is called a *dominant attribute* with respect to $(R, D)$.

## 4 Empirical Evaluation

We test both *MaPle* and *p-Clustering* extensively on both synthetic and real life data sets. In this section, we report the results.

*MaPle* is implemented using C/C++. We obtained the executable of *p-Clustering* from the authors of [12]. Please note that the authors of *p-Clustering* improved their algorithm dramatically after their publication in SIGMOD'02. All the experiments are conducted on a PC with a P4 1.2 GHz CPU and 384 M main memory running a Microsoft Windows XP operating system.

The algorithms are tested against both synthetic and real life data sets. Synthetic data sets are generated by a synthetic data generator reported in [12]. Limited by space, we only report the results on a real data set, the Yeast microarray data set [11]. This data set contains the expression levels of $2,884$ genes under $17$ conditions.

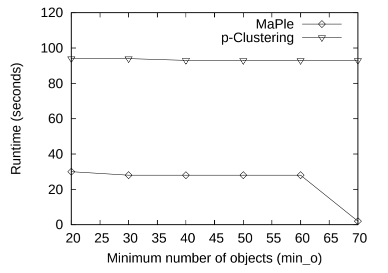
**Results on Yeast Data Set**

The results on Yeast data set are shown in Figure 9. We can obtain the following two interesting observations.

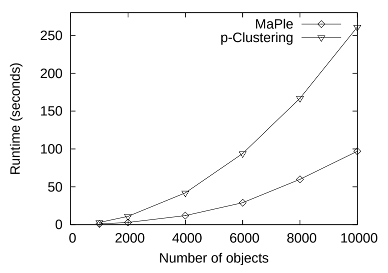
| $\delta$ | $min_a$ | $min_o$ | # of MPC | # of pClusters |
|---|---|---|---|---|
| 0 | 9 | 30 | 5 | 5520 |
| 0 | 8 | 40 | 5 | $2.05 \times 10^9$ |
| 0 | 7 | 50 | 11 | $3.37 \times 10^{15}$ |

**Figure 9. Test results on Yeast raw data set.**

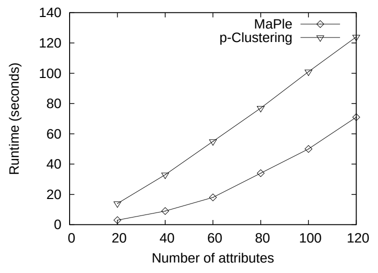
On the one hand, there are significant pClusters existing in real data. For example, we can find pure pCluster (i.e., $\delta = 0$) containing more than 30 genes and 9 attributes in Yeast data set. That shows the effectiveness and utilization of mining maximal pClusters in bioinformatics applications.



(a) Runtime vs. minimum number of objects in pClusters.



(b) Scalability with respect to the number of objects in the data sets.



(c) Scalability with respect to the number of attributes in the data sets.

**Figure 10. Results on synthetic data sets.**

On the other hand, while the number of maximal pClusters is often small, the number of all pClusters can be huge, since there are many different combinations of objects and attributes as sub-clusters to the maximal pClusters. This shows the effectiveness of the notation of maximal pClusters.

**Results on Synthetic Data Sets**

We test the scalability of the algorithms on the two parameters, the minimum number of objects $min_o$ and the minimum number of attributes $min_a$ in pClusters. In Figure 10(a), the runtime of the algorithms versus $min_o$ is shown. The data set has $6000$ objects and $30$ attributes.

Both algorithms are in general insensitive to parameter $min_o$, but *MaPle* is faster than *p-Clustering*. The major reason is that the number of pClusters in the synthetic data set does not change dramatically as $min_o$ decreases and thus the overhead of the search does not increase substantially. Please note that we do observe the slight increases of runtime in both algorithms as $min_o$ goes down. One interesting observation here is that, when $min_o > 60$, the runtime of *MaPle* is significantly shorter. That is because there is no pCluster in such a setting. *MaPle* can detect this in an early stage and thus can stop early.

We observe the similar trends on the runtime versus parameter $min_a$. The reasoning similar to that on $min_o$ holds here. Limited by space, we omit the details here.

We test the scalability of both algorithms on the number of objects in the data sets. The result is shown in Figure 10(b). The data set contains $30$ attributes, where there are $30$ embedded clusters. We fix $min_a = 5$ and set $min_o = n_{obj} \cdot 1\%$, where $n_{obj}$ is the number of objects in the data set. $\delta = 1$. The result clearly shows that both *MaPle* and *p-Clustering* are scalable with respect to the number of objects in the data sets. *MaPle* performs substantially better than *p-Clustering* in mining large data sets.

We also test the scalability of both algorithms on the number of attributes. The result is shown in Figure 10(c). The number of objects is fixed to $3,000$ and there are $30$ embedded pClusters. We set $min_o = 30$ and $min_a = n_{attr} \cdot 20\%$, where $n_{attr}$ is the number of attributes in the data set. Both *MaPle* and *p-Clustering* are approximately linearly scalable with respect to the number of attributes, and *MaPle* performs consistently better than *p-Clustering*.

In summary, from the tests on synthetic data sets, we can see that *MaPle* clearly outperforms *p-Clustering*. *MaPle* is efficient and scalable in mining large data sets.

## 5 Conclusions

In this paper, we propose *MaPle*, an efficient and scalable algorithm for mining maximal pattern-based clusters in large databases. We test the algorithm on both real life data sets and synthetic data sets. The results show that *MaPle* outperforms the best method previously proposed.

## References

[1] C.C. Aggarwal et al. Fast algorithms for projected clustering. In *SIGMOD'99*.

[2] C.C. Aggarwal and P.S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD'00*.

[3] R. Agrawal et al. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD'98*.

[4] R. Agrawal et al. Mining association rules between sets of items in large databases. In SIGMOD'93.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB'94*.

[6] K. S. Beyer et al. When is "nearest neighbor" meaningful? In *ICDT'99*.

[7] C. H. Cheng et al. Entropy-based subspace clustering for mining numerical data. In *KDD'99*.

[8] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. of the 8th Int'l Conf on Intelligent System for Molecular Biology*.

[9] J. Han et al. Mining frequent patterns without candidate generation. In *SIGMOD'00*.

[10] H. V. Jagadish et al. Semantic compression and pattern extraction with fascicles. In *VLDB'99*.

[11] S. Tavazoie et al. Yeast micro data set. In *http://arep.med.harvard.edu/biclustering/yeast.matrix*, 2000.

[12] H. Wang et al. Clustering by pattern similarity in large data sets. In *SIGMOD'02*.

[13] J. Yang et al. $\delta$-cluster: Capturing subspace correlation in a large data set. In *ICDE'02*.