OP-Cluster: Clustering by Tendency in High Dimensional Space

Jinze Liu and Wei Wang
Computer Science Department
University of North Carolina
Chapel Hill, NC, 27599
{liuj, weiwang }@cs.unc.edu

Abstract

Clustering is the process of grouping a set of objects into classes of similar objects. Because of unknownness of the hidden patterns in the data sets, the definition of similarity is very subtle. Until recently, similarity measures are typically based on distances, e.g Euclidean distance and cosine distance. In this paper, we propose a flexible yet powerful clustering model, namely OP-Cluster (Order Preserving Cluster). Under this new model, two objects are similar on a subset of dimensions if the values of these two objects induce the same relative order of those dimensions. Such a cluster might arise when the expression levels of (co-regulated) genes can rise or fall synchronously in response to a sequence of environment stimuli. Hence, discovery of OP-Cluster is essential in revealing significant gene regulatory networks. A deterministic algorithm is designed and implemented to discover all the significant OP-Clusters. A set of extensive experiments has been done on several real biological data sets to demonstrate its effectiveness and efficiency in detecting co-regulated patterns.

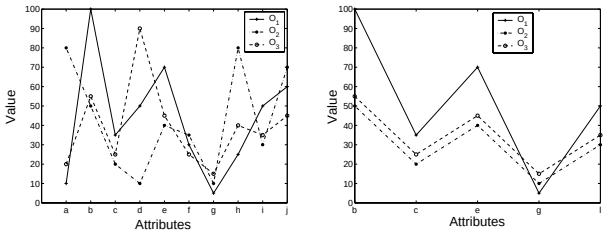
1 Introduction

As a fundamental tool to analyze large databases, clustering has been studied extensively in many areas including statistics, machine learning and pattern recognition. Most clustering models, including those proposed for subspace clustering, define similarities among objects via some distance functions. Some well-known distance functions include Euclidean distance, Manhattan distance, and cosine distance. However, distance functions are not always adequate in capturing correlations among objects. In fact, strong correlations may still exist among a set of objects even if they are far apart from each other in distance.

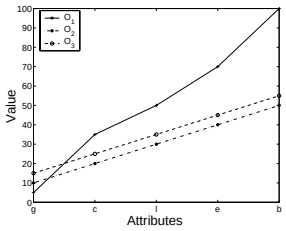
In light of this observation, the δ-pCluster model [17] was introduced to discover clusters by pattern similarity (rather than distance) from raw data sets. A major limitation of the δ-pCluster model is that it only considers either strict shifting patterns or strict scaling patterns[1], which is insufficient in many cases.

In this paper, we propose a flexible clustering model, order preserving cluster (OP-Cluster), which is able to capture the general tendency of objects across a subset of dimensions in a high dimensional space.

Figure 1 a) shows a set of 3 objects with 10 attributes. In this raw data, no obvious pattern is visible. However, if we pick the set of attributes $\{b, c, e, g, l\}$ as in Figure 1 b) for these 3 objects, we can observe the following fact: *The ranks for each of these attributes are the same for all the three objects*. If we rearrange the columns in the ascending order of their ranks: $< g, c, l, e, b >$, in Figure 1 c), the consistency of *escalation* along the ordered list of attributes can be seen much clearer.



a) Raw Data b) An OP-cluster



c) Rearranged by Rank

Figure 1. An Example of OP-cluster

1.1 Applications

- *DNA microarray analysis*. Microarray is one of the latest breakthroughs in experimental molecular biology.

[1]The scaling patterns can be transformed into shifting patterns by applying a logarithmic function on the raw data.

Investigators show that more often than not, if several genes contribute to a disease, it is possible to identify a subset of conditions, under which these genes show a coherent tendency. Since a gene's expression level may vary substantially due to its sensitivity to systematic settings, the direction of movement (up or down) in response to condition change is often considered more credible than its actual value. Discovering clusters of genes sharing coherent tendency is essential in revealing the significant connections in gene regulatory networks[9].

- *E-commerce*. In recommendation systems and target marketing, the tendency of people's affinities plays a more important role than the absolute value. Revealing sets of customers/clients with similar behavior can help the companies to predict customers' interest and make proper recommendation for future marketing.

1.2 Challenges and Our Contributions

To discover the general tendency in a cluster, the major challenge is the huge number of potential rankings. If we have n attributes, there are $N!$ different permutations of (subsets of) attributes. Each permutation corresponds to one unique ordering for this set of attributes, which is embedded in a subset of objects. Moreover, it is possible that similar ranking only occur in a subset of the N attributes. So totally, the number of potential candidates for OP-Clusters with at least nc attributes is

$$\sum_{nc \leq i \leq N} \frac{N!}{(N-i)!} \tag{1}$$

Data sets in DNA array analysis or collaborative filtering can have hundreds of attributes. This results in a huge number of candidates of various lengths. To tackle this problem, we introduce OPC-Tree to guide the enumeration of potential candidates. The following are our contributions.

- We propose a new clustering model, namely OP-Cluster, to capture general tendencies exhibited by the objects. The OP-Cluster model is a generalization of existing subspace clustering models. It has a wide variety of applications, including DNA array analysis and collaborative filtering, where tendency along a set of attributes carries significant meaning.

- We design a compact tree structure OPC-Tree to mine OP-Cluster effectively. Compared with one of fastest sequential pattern mining algorithms, prefixSpan(modified to serve our purpose), the OPC-Tree based algorithm delivers a shorter response time in most cases, especially when the data is pattern-rich.

- We apply the model of OP-Cluster to two real data sets and discover interesting patterns that tend to be overlooked by previous models.

1.3 Paper Layout

The remainder of the paper is organized as follows. Section 2 discusses some related work. Section 3 defines the model of OP-Cluster. Section 4 presents the algorithm to mine OP-Cluster in detail. An extensive performance study is reported in Section 5. Section 6 concludes the paper.

2 Related Work

2.1 Subspace Clustering

Clustering in high dimensional space is often problematic as theoretical results [8] questioned the meaning of closest matching in high dimensional spaces. Recent research work [18, 19, 3, 4, 6, 9, 12] has focused on discovering clusters embedded in the subspaces of a high dimensional data set. This problem is known as subspace clustering. Based on the measure of similarity, there are two categories of clustering model. The first category is distance based. In this category, one of the well known subspace clustering algorithms is CLIQUE [6]. CLIQUE is a density and grid based clustering method. The PROCLUS [3] and the ORCLUS [4] 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 second category is pattern-based. The first algorithm proposed in this category is the bicluster model [10] by Cheng et al. The algorithm tries to measure the coherence of the genes and the conditions in a sub-matrix of a DNA array. Recently, δ-pCluster is introduced by Wang et al [17] to cluster objects exhibiting shifting or scaling patterns in a data set in a very efficient way. In many applications, only allowing shifting or scaling patterns is too restrictive. To include more generic pattern in a cluster, the threshold has to be relaxed. This, in turn, can result in unavoidable noise inside a cluster.

The concept of OPSM(order preserving submatrix) was first introduced in [7] to represent a subset of genes identically ordered among a subset of the conditions in microarray analysis. The OPSM problem is proven to be NP-hard. A model-based statistical algorithm was also given in [7] to discover OPSMs. There are some drawbacks in this pioneering work. First, only one cluster can be found at a time. And the quality of the resulted cluster is very sensitive to some given parameters and the initial selection of partial models. Secondly, OPSM algorithm favors clusters with a large row support, which as a result, can obstruct the discovery of the small but significant ones.

In our work, we generalize the OPSM model by allowing grouping. Based on the new model, we propose a deterministic subspace clustering algorithm, namely OPC-Tree, to capture all the general tendencies exhibited by a subset of objects along a subset of dimensions in one run.

2.2 Sequential Pattern Mining

Since it was first introduced in [5], sequential pattern mining has been studied extensively. Conventional sequential pattern mining finds frequent subsequences in the database based on exact match. There are two classes of algorithms. On one hand, the breadth-first search methods (e.g., GSP [15] and SPADE [21]) are based on the Apriori principle [5] and conduct level-by-level candidate-generation-and-tests. On the other hand, the depth-first search methods (e.g., PrefixSpan [14] and SPAM [1]) grow long patterns from short ones by constructing projected databases.

In our paper, we are facing a similar but more complicated problem than sequential pattern mining. Rows in matrix will be treated as a sequence to find sequential patterns. However, in order to finally determine OP-Cluster, the ID associated with each sequence has to be kept during the mining process. A depth-first traversal of the tree is carried out to generate frequent subsequences by recursively concatenating legible suffixes with the existing frequent prefixes.

3 Model

In this section, we define the OP-Cluster model for mining objects that exhibit tendency on a set of attributes.

3.1 Notations

$\mathcal{D}$	A set of objects
$\mathcal{A}$	A set of attributes of the objects in $\mathcal{D}$
$(\mathcal{O}, \mathcal{T})$	A sub-matrix of the data set, where $\mathcal{O} \subseteq \mathcal{D}$, $\mathcal{T} \subseteq \mathcal{A}$
$x, y, ...$	Objects in $\mathcal{D}$
$a, b, ...$	Attributes in $\mathcal{A}$
d_{xa}	Value of object x on attribute a
δ	User-specified grouping threshold
δ^p	User-specified shifting threshold
nc, nr	User-specified minimum # of columns and minimum # of rows of a model

3.2 Definitions and Problem Statement

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

Definition 3.1 *Let o be an object in the database, $\langle d_{o1}, d_{o2}, ..., d_{on}\rangle$ be the attribute values in a non-decreasing order, n be the number of attributes and δ be the user specified threshold. We say that o is* **similar** *on attributes i, $i+1$, ..., $i+j$, ($0 < i \leq n$, $0 < j \leq n$), if*

$$(d_{o(i+j)} - d_{oi}) < \mathcal{G}(\delta, d_{oi}) \quad (2)$$

where $\mathcal{G}(\delta, d_{oi})$ is a grouping function that defines the equivalent class. We call the set of attributes $\langle i, i+1, ..., i+j\rangle$ a **group** *for object o. Attribute d_{oi} is called a* **pivot point** *of this group.*

The intuition behind this definition is that, if the difference between the values of two attributes is not significant, we regard them to be "equivalent" and do not order them. For example, in gene expression data, each tissue(condition) might belong to a class of tissues corresponding to a stage or time point in the progression of a disease, or a type of genetic abnormality. Hence, within the same class, no restrict order would be placed on the expression levels.

There are multiple ways to define the grouping function $\mathcal{G}(\delta, d_{oi})$. One way is to define it as the average difference between every pair of attributes whose values are closest.

$$\mathcal{G}(\delta, d_{oi}) = \mathcal{G}(\delta) = \delta \times \sum_{0<i<n} (d_{o(i+1)} - d_{oi}) \quad (3)$$

This definition is independent of d_{oi} and is usually used when each attribute has a finite domain and its value is evenly distributed within its domain. The previous example on movie rating belongs to this case. When the value of each attribute may follow a skew distribution as the gene expression data, Equation 4 is a better choice. For the sake of simplicity in explanation, we use Equation 4 in the remainder of this paper, unless otherwise specified.

$$\mathcal{G}(\delta, d_{oi}) = \delta \times d_{oi} \quad (4)$$

For example, a viewer rates five movies (A, B, C, D, E) as (1, 4, 4.5, 8, 10). If $\delta = 0.2$, 4 and 4.5 will be grouped together and the corresponding attributes B and C will be considered equivalent. The rating is divided into four groups $\{\{A\}, \{B, C\}, \{D\}, \{E\}\}$.

Definition 3.2 *Let o be an object in the database, and $(g_{o1})\ (g_{o2})...(g_{ok})$ be a sequence of similar groups of o by Equation 2 and in non-descending order of their values. o shows an '***UP pattern** *' on an ordered list of attributes $a_1, a_2, ..., a_j$ if $a_1, a_2, ..., a_j$ is a subsequence of $(g_{o1})(g_{o2})...(g_{ok})$.*

In the above example, $(1, 4, 4.5, 8, 10)$ is the rating for movies (A, B, C, D, E). After we apply the similar group, we are able to transform the original rating to the sequence $A(BC)DE$. The subsequence $ABDE$, AE, and $(BC)E$, for example, show 'UP' patterns.

Definition 3.3 *Let $\mathcal{O}$ be a subset of objects in the database, $\mathcal{O} \subseteq \mathcal{D}$. Let $\mathcal{T}$ be a subset of attributes $\mathcal{A}$. ($\mathcal{O}$, $\mathcal{T}$) forms an* **OP-Cluster(Order Preserving Cluster)** *if there exists a permutation of attributes in $\mathcal{T}$, on which every object in $\mathcal{O}$ shows the "UP" pattern.*

Suppose that we have two movie ratings o_1 and o_2 for movies (A, B, C, D, E). The ratings are (1, 4, 4.5, 8, 10) and (2, 5, 7, 4.5, 9), respectively. According to Definition 3.3, the corresponding sequence of groups for o_1 is $A(BC)DE$, and for o_2 is $A(DB)CE$. Since $ABCE$ is a

common subsequence of them, we say that o_1 and o_2 form an OP-Cluster on the attribute set of $ABCE$.

Essentially, OP-Cluster captures the consistent tendency exhibited by a subset of objects in a subspace. Compared with δ-pCluster, which is restricted to either shifting or scaling pattern, OP-Cluster is more flexible. It includes δ-pCluster as a special case. The following lemmas address the relationship between these two models.

Lemma 3.1 *Let $(\mathcal{O}^p, \mathcal{T}^p)$ be a δ-pCluster, and δ^p is maximum skew factor allowed by this δ-pCluster. $(\mathcal{O}^p, \mathcal{T}^p)$ can be identified as an OP-Cluster if the absolute difference between any two attributes a, b, $a, b \subseteq \mathcal{T}^p$ of any object o, $o \in \mathcal{O}^p$ is at least $\frac{\delta^p}{2}$.*

Please refer to [13] for the detailed proof.

When the condition in Lemma 3.1 cannot be met in real data sets, the grouping threshold δ in OP-Cluster can be set higher to accommodate any δ-pCluster with a maximum skew factor δ^p and to include it as a special case of an OP-Cluster. For example, for any δ-pCluster with threshold δ^p, we can set the grouping threshold to be δ^p. By this means, the order between two attributes with the difference less than δ^p will be put in alphabetical order. If another object has the same set of attributes grouped together, these two objects and the two attributes will form an OP-Cluster. This will be summarized in Lemma 3.2.

Lemma 3.2 *Let $(\mathcal{O}^p, \mathcal{T}^p)$ be a δ-pCluster, and δ^p is maximum skew factor allowed by this δ-pCluster. $(\mathcal{O}^p, \mathcal{T}^p)$ can be identified as an OP-Cluster, if the grouping threshold $\delta \geq \delta^p$.*

Please refer to [13] for the detailed proof.

In the following sections, since the input data is a matrix, we refer to objects as rows and attributes as columns.

Problem Statement Given a grouping threshold δ, a minimal number of columns nc, and a minimal number of rows nr, the goal is to find all (maximum) submatrices $(\mathcal{O}, \mathcal{T})$ such that $(\mathcal{O}, \mathcal{T})$ is an OP-Cluster according to Definition 3.3, and $|\mathcal{O}| \geq nr$, $|\mathcal{T}| \geq nc$.

4 Algorithm

In this section, we present the algorithm to generate OP-Clusters. It consists of two steps: (1) preprocess each row in the matrix into a sequence of groups by Defintion 3.1; (2) mine the sets of rows containing frequent subsequences in the sequences generated in step(1).

In this paper, we propose a novel compact structure OPC-Tree to organize the sequences and to guide the pattern generation. Compared with prefixSpan, OPC-Tree can group sequences sharing the same prefixes together to eliminate multiple projections in the future. Meanwhile, single path subtree can be identified to avoid any usefulless projections at all.

4.1 Preprocessing

To preprocess the data, first, we sort all the entry values in non-decreasing order for each row. Secondly, each sorted row will be organized into a sequence of groups based on their similarity. The resulted sequences of column labels will be taken as the input to the second step— mining OP-Cluster. This process is illustrated in the following example. In the raw data present in Table 1, if the grouping threshold δ is set to be 0.1, for the row 1, the sorted order of attributes for row 1 is $[228 : d, 284 : b, 4108 : c, 4392 : a]$. a and c can be grouped together since $4392 - 4108 < 4108 \times 0.1$. By processing the rest of rows in the same way, the sequences are generated as shown in the last column of row 1. Attributes in each "()"are in the same group. Without loss of generality, they are put in the alphabetical order.

rID	*a*	*b*	*c*	*d*	*seq*
1	4392	284	4108	228	$db(ac)$
2	401	281	120	298	$c(bd)a$
3	401	292	109	238	$cdba$
4	280	318	37	215	$cdab$
5	2857	285	2576	226	$dbca$
6	48	290	224	228	$a(cd)b$

Table 1. Example Data Set

4.2 OPC-Tree

In the above subsection, each row in the matrix has been converted into a sequence of column labels. The goal in the next step is to discover all the frequent subsequences hidden in the generated sequences. This problem seems to be a sequential pattern mining problem. However, it is different from a conventional sequential pattern mining problem in the following two aspects. First, the identities of the rows associated with each frequent subsequence have to be recorded in order to determine the rows involved in an OP-Cluster. Conventional sequential mining algorithms only keep the number of appearance of frequent subsequences but not their identities. To discover the set of rows associated with them, one possible approach is to scan database to collect the related rows during postprocessing. However, this method is very time consuming and is not scalable to the size of the database. Secondly, our data sets are special in the sense that the number of appearance of each item(column) is the same since each item appears once and exactly once in each sequence. As a result, no pruning can actually happen in the first round of operation by using either apriori-based or projection-based algorithm. Based on the above observation, we develop the following algorithm.

Our algorithm uses a compact tree structure to store the crucial information used in mining OP-Clusters. The discovery of frequent subsequences and the association of rows with frequent subsequences occur simultaneously. Sequences sharing the same prefixes will be gathered and recorded in the same location. Hence, further operations

along the shared prefixes will be performed only once for all the rows sharing them. Pruning techniques can also be applied easily in the OPC-Tree structure.

Before we define the OPC-Tree formally, we first give the following example.

Example 4.1 *For the sequences in Table 1, with* $nc = 3, nr = 3$*, the OPC-Tree algorithm makes a pre-order depth-first traversal of the tree and works in the following steps.*

Step 1: *Create root -1(NULL) and insert all the sequences into the tree.* This is shown in Figure 2 (A). Notice that all rows sharing the same prefix fall on the same prefix of the tree. The sequence ID is stored in the leaves. This is the *initial* OPC-Tree on which a recursive procedure depicted in Step 2-5 is performed to fully develop the tree.

Step 2: *For each child of the root, insert suffixes in its subtree to the root's child that has a matching label.* In Figure 2 (B), c is a child of the root -1. In this subtree, the suffix subtree starting at d (for the sequences $3, 4$) is inserted into the root -1's child d. If the same subtree exists in the destination node, the sequence IDs associated with the suffixes are combined with existing IDs in the destination node. Otherwise a new subtree will be created in the destination node. In the case where a suffix is too short to satisfy *current depth + length of the suffix* $> nc$, the suffix will not be inserted. For example, ba in sequence 3 is also a suffix, it is not to be inserted because the *depth 0 + length of* $ba < nc$.

Step3: *Prune current root's children.* If the number of rows that fall in a subtree is smaller than nr, the subtree will be deleted because no further development can generate a cluster with more than nr rows. For example, the subtree leading from $-1b$ in Figure 2 (B) is deleted in Figure 2 (C) since there are only two sequences falling in this subtree.

Step4: *Repeat Step2-Step5 on the root's first child and its subtree recursively*. For example, c is the first child of root -1. Therefore, the same procedure in Step2 is applied to c first. The suffixes of c's subtree d, such as ba and ab are inserted into c's subtree b and a respectively. Since there are less than three sequences falling on c's subtrees a and b, the branches $-1ca-$ and $-1cb-$ are deleted. Following the same procedure, we develop c's only subtree $-1cd-$, which is shown in Figure 2(D).

Step5: *Follow the sibling link from the first child and repeat Step2-Step5 on each sibling node recursively.* For example, after finishing $-1c-$'s subtree development, the next subtree to develop is $-1c-$'s sibling $-1d-$.

Definition 4.1 OPC-tree (Order Preserving Clustering tree). *An OPC-Tree is a tree structure defined below.*
1. It consists of one root labeled as "-1", a set of subtrees as the children of the root;
2. Each node in the subtrees consists of four entries: entry value, a link to its first children node, a link to its next sibling node, and a list of all the rows that share the same path leading from root to this node but do not have longer common subsequences passing this node. In another word,

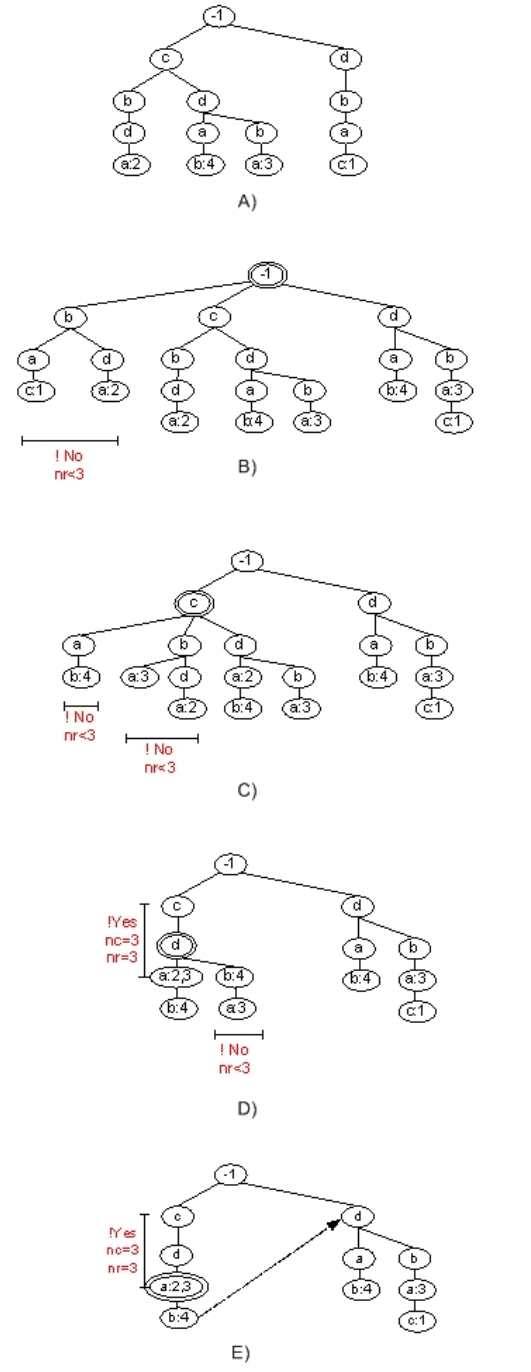


Figure 2. OPC-Tree for Table 1. The label in the oval shape represents the column name. The number following ':' represents the row ID. The node with double oval means the active node in the depth first traversal. '!No' means that the subtree must be pruned. 'Yes' means that subtree is an OP-Cluster. A). Initiate the tree with all the rows B). Active node -1, Insert the suffix of -1's subtrees to node -1. C). Active node -1c-, Insert and Prune the subtree ($nr < 3$). D). Active node -1cd-. Identify the first OP-Cluster E). Finish growing the -1's first subtree-1c, the next subtree is -1d.

the sequence IDs are only recorded at the nodes that mark the end of a commom subsequence.

Algorithm *growTree*(T,nc,nr,depth)
Input: T: the root of the initiated tree, nc and nr
Output: OP-Cluster existed in T
($*$ Grow patterns based on original T $*$)
1. **if** $T =$ **nil**
2. **return**;
3. $T_{child} \leftarrow T$'s first child;
4. **for** any sub-tree $subT$ of T
5. **do** insertSubTree($subT$, T);
6. pruneTreeNode(T);
7. growTree(T_{child}, nc, nr, $depth + 1$);
8. growTree(T's next sibling, nc, nr, $depth$);
9. **return**.

Analysis of OPC-Tree construction Only one scan of the entire data matrix is needed during the construction of the OPC-Tree. Each row is converted into a sequence of column labels. The sequences are then inserted into the OPC-Tree. With OPC-Tree structure, sequences that have the same prefix naturally fall onto the same path from root to the node corresponding to the end of prefix. To save memory, the row IDs associated with each path are only recorded at the node marking the end of the longest common prefix shared by these rows. To find the OP-Cluster using the OPC-Tree, the common subsequences are developed by adding suffixes of each sub-tree as the tree's children, via a pre-order traversal of the OPC-Tree.

Lemma 4.1 *Given a matrix M, a grouping threshold, the initial OPC-Tree contains all the information of matrix M.*

Rationale: Based on the OPC-Tree construction process, each row in the matrix is mapped onto one path in the OPC-Tree. The row IDs and the order of the columns are completely stored in the initial OPC-Tree.

4.2.1 Mining OP-Cluster Using OPC-Tree

Lemma 4.2 *The developed OPC-Tree on a set of sequences contains all subsequences hidden in the initial OPC-Tree.*

Rationale: Given any sequence $S = x_1x_2x_3x_4 \dots x_n$, we want to show that all of the subsequences of S can be found in a path starting from root. Through the initiation of OPC-Tree, we know that S will exist in the OPC-Tree. Then given any subsequence $SS = x_ix_j \dots x_s, (1 \leq i, s \leq n)$, we can obtain SS by the following steps. First, at node x_i, insert suffix $x_ix_{i+1} \dots x_n$. Now in the subtree of x_i, node x_j can be found because it should be along the path $x_ix_{i+1} \dots x_n$ that is inserted in the first step. Similarly, we insert the suffix $x_j \dots x_n$. As a result, we get the path $x_ix_jx_{j+1} \dots x_n$. By repeating the same procedure until we insert the suffix starting with x_s, we get the path $x_ix_j \dots x_s$. Because all the suffixes are inserted in the OPC-Tree, the OPC-Tree contains all the subsequences presented in the original OPC-Tree.

Rows in an OP-Cluster share the same rank of a set of columns, which corresponds to the same path in the OPC-Tree. We can conclude that the OPC-Tree contains all the clusters. This leads to the following lemma.

Lemma 4.3 *The developed OPC-Tree on a set of sequences contains all potential OP-Clusters. The columns in these clusters are on the paths leading from the root to any tree node with depth no less than nc and row support count in its subtree no less than nr .*

4.2.2 Pruning OPC-Tree

Without any pruning, the whole OPC-Tree fits well into memory when we have a small to medium sized matrix (15 columns by 3000 rows). However, for large matrices, some pruning strategies have to be employed to minimize the size of the OPC-Tree. There are two pruning techniques used in our implementation. One strategy is to prune the suffixes to be inserted that are shorter than nc; the other is to prune the subtrees where the row support count is below nr.

Lemma 4.4 *For a node N in the OPC-Tree with depth d, and for a suffix S with length l in its sub-tree, if $d + l < nc$ (the minimum columns required for a cluster), this suffix S will not be useful in forming any OP-Cluster cluster.*

Rationale: The length of the path L we can get by combining the path from root to N and S is $d + l$. Based on Lemma 4.3, L will not form any cluster. Therefore, suffix S need not to be inserted. In our implementation, we check depth of the node at which the end of the suffix is inserted. If the depth is smaller than nc, the row IDs recorded in this node will be deleted.

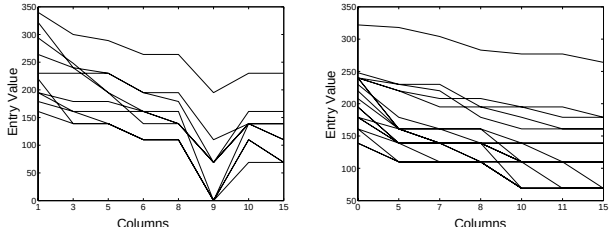
The major cost of OPC-Tree development is suffix concatenation. To minimize the storage cost of the suffixes, the single-path subtree can be collapsed into one node. The detailed data structure and algorithm can be found in technical report [13]

5 Experiments

We experimented our OP-Cluster algorithm, OPC-Tree by collapsing nodes on two real data sets. The program was implemented in C and executed on a Linux machine with a 700 MHz CPU and 2G main memory. We also implemented an optimized version of prefixSpan algorithm for comparison. The following tests are organized into three categories. First, we show promising patterns found in real data sets, which failed to be discovered by other models. Secondly, we study the sensitivity of OP-Cluster to various parameters. At last, we evaluate the performance of OPC-Tree and compare it with the optimized prefixSpan algorithm.

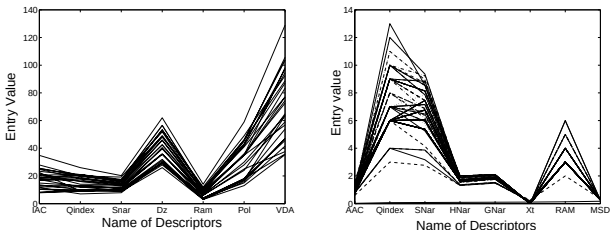
5.1 Data Sets

The following two real data sets are used in our experiments.



(a) Cluster #1 (b) Cluster #2

Figure 3. Cluster Analysis: Two examples of OP-Clusters in yeast data



(a) Cluster #1 (b) Cluster #2

Figure 4. Cluster Analysis: Two examples of OP-Clusters in drug activity data

Gene Expression Data

Gene expression data are generated by DNA chips and other microarray techniques. The yeast microarray contains expression levels of 2,884 genes under 17 conditions [16]. 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 or down-regulation under a subset of conditions [10].

Drug Activity Data

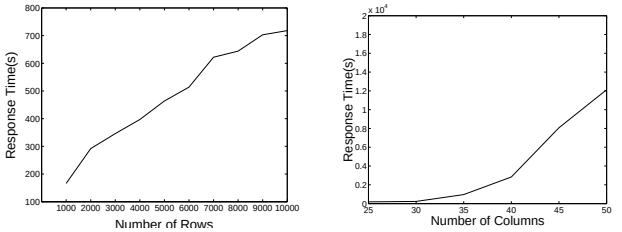
Drug activity data is also a matrix with 10000 rows and 30 columns. Each row corresponds to a chemical compound and each column represents a descriptor/feature of the compound. The value of each entry varies from 0 to 1000.

5.2 Results from Real Data

We apply the OP-Cluster algorithm to the two data sets. With parameter $\delta = 0.1$, Some interesting clusters are reported in both of the data sets. As showed in Figure 3, the two patterns generated from yeast dataset [10] present the coherent tendency along the columns. In Figure 3(a), Figure 3 (b) shows another interesting cluster which presents with a descending tendency itself. In both of the figures, we can observe that each of the four example OP-Clusters contains the curves with sharp slopes and the curves with potentially long distances to the rest.

5.3 Scalability

We evaluate the performance of the OP-Cluster algorithm with the drug activity data which has a larger size. Figure 5 shows the performance data. As we know, the columns and the rows of the matrix carry the same significance in the OP-Cluster model, which is symmetrically defined in Formula 2. Although the algorithm is not entirely



(a) Scalability with respect to number of rows (b) Scalability with respect to number of columns

Figure 5. Performance Study: Response time V.S. number of columns and number of rows

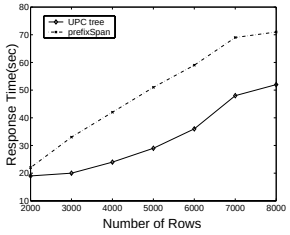
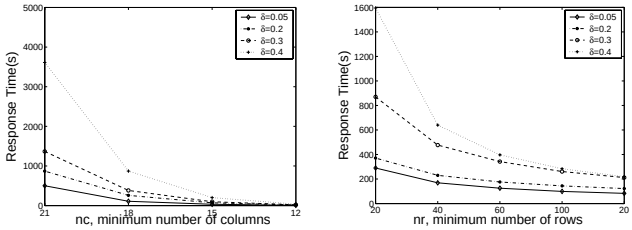


Figure 6. Performance comparison of prefixSpan and UPC-tree



(a) Response time varying similarity threshold and nc (b) Response time varying similarity threshold and nr

Figure 7. Performance Study: Response time V.S. similarity threshold , nc and nr

symmetric in the sense that it chooses to project column-pairs first, the curves in Figure 5 demonstrate similar trends.

Data sets used in Figure 5 are taken from the drug activity data. For experiments in Figure 5(a), the total number of columns is fixed at 30. The mining program is invoked with $\delta = 0.2$, $nc = 9$, and $nr = 0.01N$, where N is the total number of rows. For the experiemnts in Figure 5(b), the total number of rows is fixed as 1000. The mining algorithm is invoked with $\delta = 0.2$, $nc = 0.66C$, and $nr = 30$, where C is the total number of columns. The response time of the OPC-Tree is mostly determined by the size of the tree. As the number of rows and number of columns increase, the size of developed OPC-Tree will get deeper and broader. Hence, the response time will unavoidably become longer.

Figure 6 presents the performance comparison between the prefixSpan algorithm and the OPC-Tree algorithm. The parameter setting for this set of experiment is the following: $nc = 9$, $nr = 0.01N$, $\delta = 0.2$. The number of columns is 20. We can observe that the OPC-Tree algorithm can constantly outperform the prefixSpan algorithm and the advantage becomes more substantial with larger data set.

Next, we study the impact of the parameters (δ, nc, and nr) towards the response time. The results are shown in Figure 7. The data set used in this experiment is the yeast data. When nc and nr are fixed, the response time gets shorter as the group threshold δ relaxes. The reason is that as more columns are grouped together, the number of rows sharing the same path in the OPC-Tree is increasing. Hence, the OPC-Tree is shorter, which results in less overall response time. As nc or nr decreases, the response time prolonged. This is showed in Figure 7. According to the pruning techniques discussed in Lemma 4.4, fewer number of subsequences can be eliminated when nc is smaller. As a result, a larger tree is constructed, which consumes more time. A similar effect can be observed with respect to nr from Figure 7(b).

6 Conclusions

In many applications including collaborative filtering and DNA array analysis, although the distance (e.g., measured by Euclidean distance or cosine distance) among the objects may not be close, they can still manifest consistent patterns on a subset of dimensions. In this paper, we proposed a new model called OP-Cluster to capture the consistent tendency exhibited by a subset of objects in a subset of dimensions in high dimensional space. We proposed a compact tree structure, namely OPC-Tree, and devised a depth-first algorithm that can efficiently and effectively discover all the closed OP-Clusters with a user-specified threshold.

References

[1] J. Ayres, J. E. Gehrke, T. Yiu, and J. Flannick. Sequential PAttern Mining Using Bitmaps. In *SIGKDD*, July 2002.

[2] R. C. Agarwal, C. C. Aggarwal, and V. Parsad. Depth first generation of long patterns. In *SIGKDD*, 2000.

[3] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[4] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, pages 70-81, 2000.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, 3-14, Mar. 1995.

[6] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Authomatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[7] A. Ben-Dor, B. Chor, R.Karp, and Z.Yakhini. Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem. In *RECOMB* 2002.

[8] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful. In *Proc. of the Int. Conf. Database Theories*, pages 217-235, 1999.

[9] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, pages 84-93, 1999.

[10] Y. Cheng and G. Church. Biclustering of expression data. In *Proc. of 8th International Conference on Intelligent System for Molecular Biology*, 2000.

[11] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-bsed algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226-231, 1996.

[12] H.V.Jagadish, J.Madar, and R. Ng. Semantic compression and pttern extraction with fasicicles. In *VLDB*, pages 186-196, 1999.

[13] J.Liu and W.Wang. Flexible clustering by tendency in high dimensional spaces. Technical Report TR03-009, Computer Science Department, UNC-CH, 2003.

[14] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen U. Dayal, and M.-C. Hsu. PrefixSpan mining sequential patterns e.ciently by prefix projected pattern growth. In *ICDE* 2001, pages 215-226, Apr. 2001.

[15] R.Srikant and R.Agrawal. Mining sequential patterns: Generalizations and performance improvements. In $EDBT'96$, pages 3-17, Mar. 1996.

[16] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church. Yeast micro data set. In *http://arep.med.harvard.edu/biclustering/yeast.matrix*, 2000.

[17] H. Wang, W. Wang, J. Yang, and P. Yu. Clustering by pattern similarity in large data sets, in *SIGMOD*, pp. 394-405, 2002.

[18] J. Yang, W. Wang, H. Wang, and P. S. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *ICDE*, pages 517-528, 2002.

[19] J. Yang, W. Wang, P. Yu, and J. Han. Mining long sequential patterns in a noisy environment. In *SIGMOD*,pp.406-417, 2002.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD*, pages 103114, 1996.

[21] M. J. Zaki, S. Parthasarathy, M.Orihara, and W. Li. Parallel algorithm for discovery of association rules. In *DMKD*, 343-374, 1997.