# $\delta$-Clusters: Capturing Subspace Correlation in a Large Data Set

Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu
IBM T. J. Watson Research Center
{jiyang, ww1, haixun, psyu}@us.ibm.com

## Abstract

*Clustering has been an active research area of great practical importance for recent years. Most previous clustering models have focused on grouping objects with similar values on a (sub)set of dimensions (e.g., subspace cluster) and assumed that every object has an associated value on every dimension (e.g., bicluster). These existing cluster models may not always be adequate in capturing coherence exhibited among objects. Strong coherence may still exist among a set of objects (on a subset of attributes) even if they take quite different values on each attribute and the attribute values are not fully specified. This is very common in many applications including bio-informatics analysis as well as collaborative filtering analysis, where the data may be incomplete and subject to biases. In bio-informatics, a bicluster model has recently been proposed to capture coherence among a subset of the attributes. Here, we introduce a more general model, referred to as the $\delta$-cluster model, to capture coherence exhibited by a subset of objects on a subset of attributes, while allowing absent attribute values. A move-based algorithm (FLOC) is devised to efficiently produce a near-optimal clustering results. The $\delta$-cluster model takes the bicluster model as a special case, where the FLOC algorithm performs far superior to the bicluster algorithm. We demonstrate the correctness and efficiency of the $\delta$-cluster model and the FLOC algorithm on a number of real and synthetic data sets.*

## 1 Introduction

Clustering has become an active research area in recent years. Many clustering algorithms [4] have been proposed to efficiently cluster data in multi-dimensional space. An important advance in this area is the introduction of the *subspace clustering*. This is particularly useful in clustering high dimensional data since not every dimension may be relevant to a cluster. A subspace cluster consists of a (sub)set of dimensions and a (sub)set of points/vectors such that these points/vectors are close to each other in the subspace defined by these dimensions. However, the model of subspace cluster may not be adequate either to capture correlation among vectors/points/objects in the sense that it still only takes into account the physical distance between points/vectors. In fact, strong correlation may exist among a set of points/vectors/objects even if they are far apart from each other. Let's consider three data points/vectors, $d_1 = (1, 5, 23, 12, 20)$, $d_2 = (11, 15, 33, 22, 30)$, and $d_3 = (111, 115, 133, 122, 130)$. They may not be considered in the same cluster by any traditional (subspace) cluster model because the distance between any two of them is large. Let's take a closer look on these vectors. Each data point/vector is shown as a curve in Figure 1. The attribute values from each vector are connected. The difference between $d_1$ and $d_2$ is $(10, 10, 10, 10, 10)$ while the difference between $d_2$ and $d_3$ is $(100, 100, 100, 100, 100)$. We can see that these vectors have strong *coherence* on these five attributes because, given any one vector, the other two can be perfectly derived by shifting each entry (of the given vector) by a certain offset. In other words, the corresponding vectors show the same (or similar) tendency, but with some offsets. The offset can be viewed as *bias* of each object/vector. Note that the order of attributes is inconsequential. If we change the order of these five attributes, the vectors (represented by curves) would also show strong coherence. Although in the example, the three vectors are coherent on all five attributes, the coherent attributes may be buried in a large set of attributes in real applications. Identifying these coherent attributes can be a very challenging process. This type of coherence is very common in many applications where each object may naturally bear a certain degree of bias.

- 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 customer interest and make proper recommendations. Let's consider the following example. Three viewers rank four movies as (1, 2, 3, 5), (2, 3, 4, 6), and (3, 4, 5, 7), where 1 is the lowest and 10 is the highest score. Although the individual ranks are different, these three viewers have coherent opinions on these four movies. In the future, if the first two viewers

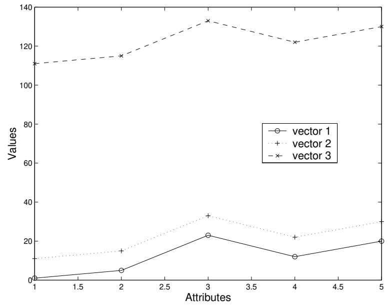



**Figure 1. Example of coherent objects**

ranked a new movie as 2 and 3, respectively, then we may think that the ranks of the new movie also follow the same coherence[1] and we can project that the third viewer may rank this movies as 4.

- DNA microarray analysis: Microarrays are one of the latest breakthroughs in experimental molecular biology, which provide a powerful tool by which the expression patterns of thousands of genes can be monitored simultaneously and are already producing huge amount 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 in revealing the significant connections in gene regulatory networks [13]. The model of *bicluster* [3] was recently proposed for this purpose under the assumption that the microarray matrix is fully specified, which can be regarded as a special case of our model.

From the above examples, we can see that users are interested in finding cluster of points/objects that have coherent behaviors rather than points/objects that are physically close to each other. To address this issue, we introduce a new model — **$\delta$-clusters**. The main objective of $\delta$-clusters is to capture a set of objects and a set of dimensions/attributes such that the objects exhibit strong coherence on the set of dimensions/attributes despite the fact that each object may bear a nonzero bias/offset. In this sense, the traditional subspace cluster can be viewed as a cluster of objects with zero bias. Thus, the subspace cluster is a degenerated case of the $\delta$-cluster model.

How to measure the coherence among objects (i.e., vectors) while allowing the existence of individual bias is essential to our model. One choice is the $PearsonR$ correlation [9]. The $PearsonR$ correlation of two objects/vectors/points $o_1$ and $o_2$ is defined as $\frac{\sum(o_1-\bar{o_1})(o_2-\bar{o_2})}{\sqrt{\sum(o_1-\bar{o_1})^2\times\sum(o_2-\bar{o_2})^2}}$ where $\bar{o_1}$ and $\bar{o_2}$ are the mean of all attribute values in $o_1$ and $o_2$, 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, there are six movies, the first three are the action movies while the next three are the 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 can be in one cluster while the remainder movies can be grouped into another cluster. It is clear that the two viewers have consistent bias within each cluster. However, the $Pearson\ R$ value is small because there is not much global bias held by the ranks of the two viewers. Thus, we propose a new concept, *base*, to represent the potential bias of each object or dimension in a $\delta$-cluster. While a $\delta$ cluster essentially corresponds to a data submatrix defined by the involved objects and attributes, a new measure called *residue* is introduced to capture the difference between the actual value of each entry (in this submatrix) and the expected value based on the the object and attribute bias within the cluster. The residue is a measurement of the degradation to the coherence of the $\delta$-cluster that an entry brings. The average residue across every entry in the submatrix can be used to measure the residue of the $\delta$-cluster. The smaller the residue, the stronger the coherence. Our objective is then to find $\delta$-clusters that minimize the residue value. In addition, in the application domain that we are interested in, missing attributes are a common place, e.g., a viewer does not give a rating on a movie. The $\delta$-cluster model can also handle the null values seamlessly.

The generality of the $\delta$-cluster model creates great challenges to the mining process. One possible solution is to map the $\delta$-cluster problem to the subspace cluster problem by considering the set of derived attributes that capture the differences between every pair of (original) attributes. This, however, increases the dimensionality quadratically and yields a very inefficient algorithm. The problem is in fact NP-hard. Thus, we devise a move-based algorithm FLOC to discover the $\delta$ clusters. Starting from a set of seeds (initial clusters), an iterative procedure is invoked to successively improve the cluster quality. During each iteration, each object and attribute is examined sequentially. The best action of each ob-

[1]It is true for the first two viewers at least.

ject or attribute towards improving the clustering quality is decided and performed. The best clustering obtained during each iteration is used as the stating point of the next round. This procedure terminates when no further improvement can be made in an iteration. In many applications, users may be interested in finding clusters with various constraints, e.g., overlapping or non-overlapping clusters. Our basic $\delta$-cluster algorithm can easily be modified to accommodate these constraints. In summary, this paper has the following contributions.

- We propose a new clustering model, namely $\delta$-clusters, to capture the points/objects that have similar trend (i.e., strong coherence) rather than close to each other on a subset of dimensions/attributes. The $\delta$-cluster can be viewed as a generalization of the subspace cluster model. The $\delta$-cluster model can seamlessly handle missing attribute values.
- The metric of residue is introduced to measure the coherency of among points/objects in a given cluster.
- Due to the NP-hard nature of this problem, we device a new *move-based* algorithm (FLOC) that can efficiently approximate the clusters with high accuracy. This algorithm can be easily modified to discover clusters with different constraints, e.g. overlapping or non-overlapping clusters, etc..

The remainder of this paper is organized as follows. Some related work is discussed in Section 2. We present the $\delta$-cluster model in Section 3. Section 4 and 5 present our basic move-based algorithm and some improvements, respectively. Experimental results are shown in Section 6 and we draw the conclusion in Section 7.

## 2 Related Work

Recent efforts in data mining have studied methods for efficient and effective cluster analysis in large databases [4]. Much active research focuses on the scalability of clustering methods and high dimensional clustering techniques. Clustering in high dimensional spaces is problematic as theoretical results [2] questioned the meaning of closest matching in high dimensional spaces. Recent research work has focused on discovering clusters embedded in the subspaces of a high dimensional data set. This problem is known as subspace clustering. CLIQUE [1] is the first to address the subspace clustering problem. It 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 is similar to the Apriori algorithm for association rule mining in that it finds dense cells in lower dimensional spaces and merge them to form clusters in higher dimensional spaces. The computation complexity and the quality of clustering is heavily dependent on the size of the grid and the density threshold of clusters.

In [3], a so-called *bicluster* model is proposed. The bicluster model is proposed in the bioinformatics field and is to capture a set of genes showing striking similarity under a set of conditions. Mean square residue score is used to qualify a bicluster. However, the bicluster model does not address the problem of missing attribute values. Thus, it can be viewed as a special case of our proposed $\delta$-cluster model. The algorithms presented in [3] identify biclusters in a data matrix in a greedy manner. After identifying the first cluster, each successive bicluster is mined on an altered matrix by replacing entries in the discovered biclusters with random data. We will show later that this approach no only produces less accurate result but also bears an inefficient performance.

Also in the information retrieval area, researchers are studying document clustering based on word appearances. In [10], authors proposed a two-stage method that first clusters words for each document and then cluster documents based on the word clusters of each document. Despite other differences in the problem formation, this approach significantly differs from our model on the following aspect: our model clusters the attributes (words) and objects (documents) simultaneously while [10] clusters the attributes (words) and objects (documents) sequentially. As a result, the cluster in our model consists of a subset of objects and a subset of attributes while the cluster in [10] is only a subset of objects.

## 3 The Model of $\delta$-cluster

To address the potential limitation inherit to the traditional models of cluster, we propose a so-called *$\delta$-cluster* model to provide a more powerful means to capture potential coherence among (a subset of) objects and attributes. There are two forms of coherent in many applications, shifting (or addition) and amplification (or production). In the case of amplification coherence, the value in one object can be twice as much as that in the first object, and we still consider they are coherent. The problem of finding amplification coherent $\delta$-cluster can be easily transformed to the problem of finding shifting coherent $\delta$-cluster by applying logarithm function to each entry in the data matrix. Thus, we only focus on finding shifting coherent $\delta$-clusters in this paper. In contrast to a cluster (or subspace cluster) of the traditional meaning, a $\delta$-cluster does not require the involved objects share similar values on each (relevant) attribute. As a more general concept, a set of objects may belong to a $\delta$-cluster if these objects exhibit similar trends on a (sub)set of attributes. This generality is very important in many applications, such as collaborative filtering, analysis of gene expression data, in the sense that potential object/attribute bias can be perfectly accommodated.

Let $\Im = \{A_1, A_2, \ldots, A_M\}$ be the set of attributes[2] and $\Re = \{O_1, O_2, \ldots, O_N\}$ be the set of objects. The data can be viewed as an $M \times N$ matrix $D$. Each entry $d_{ij}$ in this matrix corresponds to the value of object $O_i$ on attribute $A_j$. Figure 2 illustrates the general format of the matrix. Depending on the application domain, this matrix can be either sparse or dense.

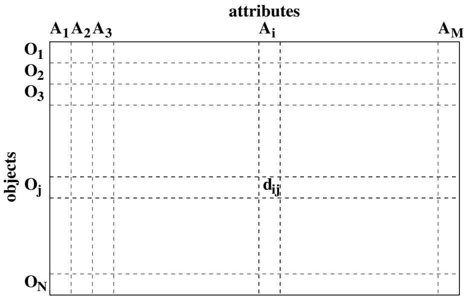



**Figure 2. The Data Matrix**

A *δ-cluster* essentially corresponds to a submatrix that exhibits some coherent tendency. Formally, each $\delta$-cluster can be uniquely identified by the set of relevant objects and attributes. Note that a $\delta$-cluster is similar to a subspace cluster in this respect, with the exception that the $\delta$-cluster also allows missing values. Even though allowing missing values brings great flexibility to the $\delta$-cluster model, the amount of missing entries in a $\delta$-cluster should be limited to some extent to avoid trivial cases. The rule of the thumb is that, despite the missing values, there should still be sufficient evidence to demonstrate the coherency. In Figure 3 (a), many values are missing, which prevents any potential coherence from being observed, even though there is no sign that contradicts the existence of coherence either. To exclude this kind of situation from being considered as a meaningful $\delta$-cluster, we introduce a parameter $\alpha$ (which is a positive number less than or equal to 1) to limit the amount of missing values for each object and attribute in a cluster.

| | attr 1 | attr 2 | attr3 | attr4 |
|---|---|---|---|---|
| object 1 | 1 | | 3 | |
| object 2 | | 4 | | 5 |
| object 3 | | 3 | 4 | |

(a) not a $\delta$ -cluster

| | attr 1 | attr 2 | attr3 | attr4 |
|---|---|---|---|---|
| object 1 | 1 | | 3 | 3 |
| object 2 | 3 | 4 | | 5 |
| object 3 | | 3 | 4 | 4 |

(b) a $\delta$ -cluster

**Figure 3. Missing Values in $\delta$-clusters**

[2] In general, the attributes can take either numerical or categorical values. In this paper, we assume numerical attributes unless specified otherwise. The scenario of having categorical attributes or even hybrid attribute types is left to the full version of this paper.

**Definition 3.1** *For a given matrix $\Im \times \Re$ and an occupancy threshold $\alpha$, a $\delta$-**cluster** (of $\alpha$ occupancy) can be represented by a pair $(I, J)$ where $I \subseteq \{1, \ldots, M\}$ is a subset of object IDs and $J \subseteq \{1, \ldots, N\}$ is a subset of attribute IDs. For each object $i \in I$, $\frac{|J'_i|}{|J|} > \alpha$ where $|J'_i|$ and $|J|$ are the number of specified attributes for object $i$ in the $\delta$-cluster and the number of attributes in the $\delta$-cluster, respectively. Similarly, for each attribute $j \in J$, $\frac{|I'_j|}{|I|} > \alpha$ where $|I'_j|$ and $|I|$ are the number of specified objects in attribute $j$ in the $\delta$-cluster and the number of objects in the $\delta$-cluster, respectively.*

Let $\alpha = 0.6$, the submatrix in Figure 3 (a) is not a $\delta$-cluster while the submatrix in Figure 3 (b) is a $\delta$-cluster. The number of specified (non-missing) entries in the corresponding submatrix is referred to as the *volume* of the $\delta$-cluster.

**Definition 3.2** *The **volume** of a $\delta$-cluster $(I, J)$ ($v_{IJ}$) is defined as the number of specified entries $d_{ij}$ such that $i \in I$ and $j \in J$.*

In the case that all entries are specified, $v_{IJ} = |I| \times |J|$ where $|I|$ and $|J|$ are the number of attributes and the number of objects participating in the $\delta$-cluster, respectively. Figure 4(a) shows a micro-array matrix with ten genes (one for each rows) under five experiment conditions (one for each column). This example is a portion of microarry data that can be found in [13]. In this example, each object is a gene and each attribute corresponds to a condition. Each entry in the data matrix measures the strength of the gene under the corresponding experiment condition. The $\delta$-cluster defined by picking $I = \{VPS8, EFB1, CYS3\}$ and $J = \{CH1I, CH1D, CH2B\}$ is shown in Figure 4(b). The volume of this $\delta$-cluster is $3 \times 3 = 9$.

| 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) a data matrix

| 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) a $\delta$-cluster

**Figure 4. An Example of $\delta$-cluster**

The fundamental difference between a $\delta$-cluster and a subspace cluster is the measure of cluster quality. In order to properly accommodate the object/attribute biases within a $\delta$-cluster, we introduce a concept — *base* to represent the bias of an object or attribute within a $\delta$-cluster.

**Definition 3.3** *For a given $\delta$-cluster $(I, J)$, the* **base** *of an object $O_i$ is defined as the average value of $O_i$ for all specified attributes in J, $d_{iJ} = \frac{\sum_{j \in J'_i} d_{ij}}{|J'_i|}$ where $J'_i \subseteq J$ is the set of specified attributes in $J$ for object $O_i$. Similarly, the* **base** *of an attribute $A_j$ is the average specified value of $A_j$ taken by all objects in I, i.e., $d_{Ij} = \frac{\sum_{i \in I'_j} d_{ij}}{|I'_j|}$ where $I'_j \subseteq I$ is the set of objects whose value is specified in attribute $A_j$. The* **base** *of the $\delta$-cluster is the average value of all specified entries of the submatrix defined by $(I, J)$, i.e., $d_{IJ} = \frac{\sum_{i \in I, j \in J} d_{ij}}{v_{IJ}}$ where $v_{IJ}$ is the volume of the $\delta$-cluster.*

For example, we have $d_{VPS8,J} = 273$, $d_{EFB1,J} = 190$, $d_{CYS3,J} = 194$, and $d_{I,CH1I} = 347$, $d_{I,CH1D} = 66$, $d_{I,CH2B} = 244$, and $d_{IJ} = 219$ in Figure 4(b). While $d_{iJ}$ and $d_{Ij}$ take care of the potential bias that may associate with each individual object or attribute, the value of $d_{IJ}$ set the base point of the entire $\delta$-cluster. In a perfect $\delta$-cluster where each object and attribute exhibits an absolutely consistent bias[3], the value of each entry $d_{ij}$ can be uniquely determined by its object base $d_{iJ}$, its attribute base $d_{Ij}$, and the cluster base $d_{IJ}$. The difference $d_{Ij} - d_{IJ}$ is essentially the relative bias held by attribute $A_j$ in contrast to other attributes in the $\delta$-cluster. This bias should hold exactly on the entry $d_{ij}$ as well in a perfect $\delta$ cluster. That is, $d_{ij} - d_{iJ} = d_{Ij} - d_{IJ}$. Consequently, we have $d_{ij} = d_{iJ} + d_{Ij} - d_{IJ}$. Figure 4(b) is a perfect $\delta$-cluster even though the values are quite far apart (which may produce poor quality cluster(s) of the traditional meaning). For example, the entry $d_{VPS8,CH1I} = d_{VPS8,J} - d_{I,CH1I} + d_{IJ} = 273 - 347 + 219 = 401$ and this property holds for every entry in Figure 4(b). This demonstrates that the model of $\delta$-cluster can successfully handle potential object (and/or attribute) bias in the form of shifting. Note that the $\delta$-cluster is different from many previous models (e.g., the *Pearson R* correlation [9]) that applies global normalization to each object or each attribute of the entire data matrix. Normalizing the value of an entire row or column (by subtracting its mean value) would not help much to accommodate the bias because such bias typically localizes to each $\delta$-cluster and do not hold true across all objects and/or attributes. In Figure 4(a), we may observe that the gene $VPS8$ has a positive bias compared to $EFB1$ within the $\delta$-cluster in Figure 4(b) but such bias does not preserve on the remaining two conditions $CH1B$ and $CH2I$.

Unfortunately, the $\delta$-cluster may not always be perfect. The concept of *residue* is thus introduced to quantify the difference between the actual value of an entry and the expected value of an entry predicted from the corresponding object base, attribute base, and the cluster base.

**Definition 3.4** *The* **residue** *of an entry $d_{ij}$ in a $\delta$-cluster is $r_{ij} = d_{ij} - d_{iJ} - d_{Ij} + d_{IJ}$ if $d_{ij}$ is specified. Otherwise, $r_{ij} = 0$.*

[3]The entries of each object (or attribute) can be exactly generated by shifting the entries of other objects (or attributes) by a common offset.

It is obvious that every entry in Figure 4(b) has a zero residue. The residue indeed serves as an indicator of the degree of coherence of an entry with the remaining entries in the $\delta$-cluster given the biases of the relevant object and the relevant attribute. The lower the residue, the stronger the coherence. To assess the overall quality of a $\delta$-cluster, the **residue** of the $\delta$-cluster can be defined as the mean residue of all specified entries. The mean can be in the form of either arithmetic, geometric, or square mean as in [3]. In this paper, we use the arithmetic mean in the assessment of the cluster residue, but the model is applicable to other types of means.

**Definition 3.5** *The* **residue** *of a $\delta$-cluster $(I, J)$ is $r_{IJ} = \frac{\sum_{i \in I, j \in J} |r_{ij}|}{v_{IJ}}$ where $r_{ij}$ is the residue of the entry $d_{ij}$ and $v_{IJ}$ is the volume of the $\delta$-cluster.*

In the above example, the residue of the $\delta$-cluster in Figure 4(b) is 0. The lower the residue, the stronger the coherence exhibited by the $\delta$-cluster, and the better the quality of the $\delta$-cluster. In the remainder of this paper, we also refer to a $\delta$-cluster $(I, J)$ as a $r$**-residue** $\delta$**-cluster** if its residue $r_{IJ} \le r$ where $r$ is a constant number.

The basic $\delta$-cluster model can be easily extended to support some additional constraints that may be very useful in many applications.

- The amount of overlap allowed between a pair of clusters $Cons_o$. Some application may require non-overlapping among clusters while others may allow some degree of overlap. The user can control the amount of overlap by specifying some threshold.
- The coverage of the clusters $Cons_c$: the number of objects/attributes should be covered by any of the clusters. In some application (e.g., collaborative filtering), the user may want every object (e.g., customer) to be covered by some cluster.
- The volume of the final clusters $Cons_v$. The user can also control the volume of the final cluster. This can be useful in the application where certain statistical significance needs to be warranted.

We shall see later in this paper that our proposed algorithm can be applied with minor modification to suit the above purposes and to produce promising results.

# 4 Basic FLOC Algorithm

## 4.1 Algorithm Description

In general, the $\delta$-cluster problem is NP-hard [3]. Thus, finding an exactly solution could be time consuming. In this section, we present a new randomized *move-based* algorithm, FLOC[4], which can efficiently and accurately approximate the $k$ $\delta$-clusters with the lowest average residue. The data is represented in the form of a matrix as shown in Figure 4(a) where the rows correspond to the objects and the columns correspond to the attributes. The FLOC algorithm starts from a set of seeds (initial clusters) and carries out an iterative process to improve the overall quality of the clustering. At each iteration, each row and column is moved among clusters to produce a better clustering in terms of a lower average residue[5]. The best clustering obtained during each iteration will serve as the initial clustering for the next iteration. The algorithm terminates when the current iteration fails to improve the overall clustering quality.

The FLOC algorithm has two phases (Figure 5). In the first phase, $k$ initial clusters are constructed. As we presented in the previous section, a $\delta$-cluster contains a set of objects (rows) and a set of attributes (columns). A parameter $\rho$ is introduced to control the size of a $\delta$-cluster. For each initial cluster, a random switch is employed to determine whether a row or column should be included. Each row and column is included in the cluster with probability $\rho$. Consequently, each initial cluster is expected to contain $M \times \rho$ rows and $N \times \rho$ columns. (We will discuss how to choose $\rho$ in the next section.)

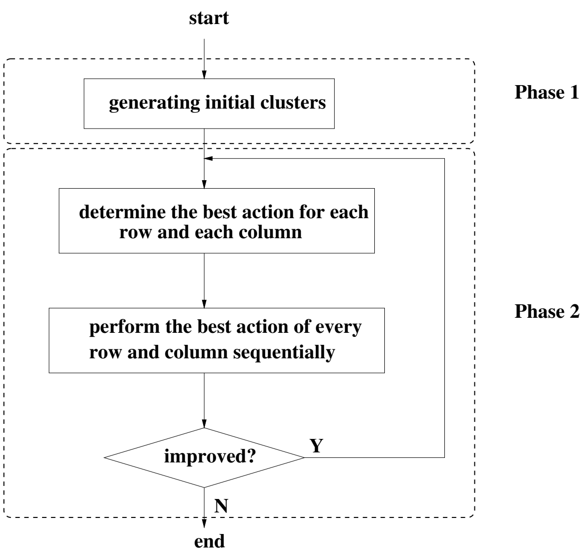



**Figure 5. The Flowchart of the FLOC Algorithm**

The second phase is an iterative process to improve the quality of the clusters continuously. During each iteration in the second phase, each row and each column is examined to determine its best action towards reducing the average residue. These actions are then performed successively to improve the clustering. An **action** is defined with respect to a row (or column) and a cluster. There are $k$ actions associated with each row (or column), one for each cluster. For a given row (or column) $x$ and a cluster $c$, the **action** $Action(x, c)$ is defined as the change of membership of $x$ with respect to $c$. Note that this action is uniquely defined at any stage. If $x$ is already included in $c$, then $Action(x, c)$ represents the removal $x$ from the cluster $c$. Otherwise, $Action(x, c)$ denotes the addition of $x$ to the cluster $c$. Figure 6 shows a data matrix with 3 rows and 4 columns. Assume that we want to find two clusters and their current statuses are indicated by the dashed lines. Cluster 1 contains row 1, 2 and column 1, 2; whereas cluster 2 contains row 2, 3 and column 1, 2, 3. Each row (or column) in Figure 6 is then associated with two actions, one for each cluster. For example, the actions associated with column 3 are (1) inserting into cluster 1, and (2) deleting from cluster 2. The better action among these two need to be identified and performed. In general, if the data matrix contains $N$ rows and $M$ columns, then $N + M$ actions will be performed during each iteration, one for each row (or column). We will discuss shortly that sometimes an action may be blocked temporarily during an iteration due to the violation of some constraints (by the action).

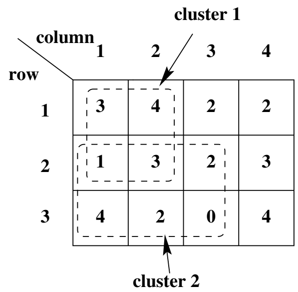



**Figure 6. An example of the Actions**

Since there are $k$ clusters, the number of potential actions associated with the row (or column) $x$ is $k$. Among these $k$ actions, the action that brings most improvement needs to be identified. To assess the amount of improvement that can be brought by an action, we introduce a new concept called **gain**. Since our objective is to find clusters with the lowest average residue, the **gain** of an action $Action(x, c)$ is defined as the *reduction* of $c$'s residue incurred by performing $Action(x, c)$. A positive gain indicates that performing $Action(x, c)$ can produce a better cluster while a negative gain suggests that such an action would degrade the cluster quality. The goal is that, for each row (or column) $x$, we want to find the action with the highest gain. In the above example, the residue of cluster 1 and cluster 2 are $\frac{1}{4}$ and $\frac{2}{3}$, respectively. Consider the two actions associated with column 3: (1) inserting into cluster 1 and (2) deleting from cluster

[4]FLOC stands for FLexible Overlapped Clustering.

[5]There are many ways to assess whether one clustering is better than another. In this paper, we adopt the average residue as the measurement and aim at finding $k$ clusters with the smallest average residue.

2. The gain of the first action is the difference between the current residue of cluster 1 and the residue of the new cluster obtained by inserting column 3 into cluster 1. In this example, the resulting cluster after inserting column 3 into cluster 1 would contain row 1,2 and column 1,2,3, and its residue is $\frac{1}{3}$. Therefore, the gain of inserting column 3 into cluster 1 is $\frac{1}{4} - \frac{1}{3} = -\frac{1}{12}$. Via similar computation, the gain of deleting column 3 from cluster 2 is $-\frac{1}{3}$. Consequently, the first action is chosen as the best action for column 3. Note that the best action for this particular column is negative, which means that the average residue would increase if this action is taken. Such negative gain action(s) will still be performed. The rationale is that the (temporary) degradation of the cluster quality may lead to an ultimate (bigger) improvement. We will explain shortly that such action will not take into effect if the cluster quality fails to improve by the end of the iteration. Nevertheless, the highest gain of any action associated with a given row (or column) is positive in many cases and will directly contribute to the improvement of the cluster quality. For example, the highest gain of any action associated with column 1 is $\frac{5}{12}$.

To compute the gain of a particular action, the resulting residue of the associated cluster (if the action was taken) needs to be computed. The straightforward way to compute the residue after each action is to recompute from scratch. This involves the computation of each object base, each attribute base, and cluster base, and finally the cluster residue. The complexity of computing the new reside is $O(n \times m)$ where $n$ and $m$ are the number of rows and columns in a cluster.

After the best action is identified for every row (or column), these $N + M$ actions are then performed sequentially. The best clustering obtained during the last iteration, denoted by $best\_clustering$, is used as the initial clustering of the current iteration. Let $Clustering_i$ be the set of clusters after applying the first $i$ actions. After applying all actions, we would obtain $M + N$ sets of clusterings. Among them, if the clustering with the minimum average residue has a lower average residue than that of $best\_clustering$, then there is an improvement in the current iteration. The clustering with the minimum average residue is stored in $best\_clustering$ and the process continues to the next iteration. Otherwise, there is no improvement in the current iteration and the process terminates. The clustering stored in $best\_clustering$ is then returned as the final result. It is obvious that the order to perform these actions plays an important role in this process. The simplest way to decide the order is to assume a fixed order among all actions, e.g., row 1 to row $N$ followed by column 1 to column $M$. We will explain in the next section that the fixed ordering suffers from some inherit drawback that may be overcome by employing a dynamic ordering.

### 4.2 Complexity Analysis

In the first phase, a set of $k$ clusters (seeds) are generated. Thus, the time complexity of the first phase is $O(k \times (N+M))$ where $N$ and $M$ are the number of rows and columns of the matrix $D$ while $k$ is the number of the clusters. The second phase is a series of iterations. During each iteration, each possible action of each row (or column) needs to be considered. There are $k$ possible actions for a given row (or column). Thus, $(N + M) \times k$ actions have to be considered. In turn, the overall time complexity to evaluate all of these actions is $O((N+M) \times N \times M \times k)$. The time complexity to perform an action is the same as to compute the gain of that action which is $O(N \times M)$. There are $(N + M)$ actions to perform. Thus, the overall time complexity for an iteration is $O((N+M) \times N \times M \times k)$, which implies that the complexity of the FLOC algorithm is $O((N + M) \times N \times M \times k \times p)$ where $p$ is the number of iterations till termination. In the experimental result section, we can see that $p$ is in the order of 10 with various conditions. If the number of columns (attributes) and the number of rows (objects) are of the same order of magnitude, then the complexity of the FLOC algorithm is $O(|D|^{\frac{3}{2}} \times k \times p)$.

### 4.3 Additional Feature

As mentioned previously, the $\delta$-cluster model can support many optional constraints specified by the user. In order to enforce these constraints, the basic FLOC algorithm needs to be modified. In the first phase where the set of initial clusters are generated, the produced clusters have to comply with the specified constraints [6]. During the second phase where the iterative improvement is carried out, some action may be "blocked" temporarily (e.g., the gain is assigned to $-\infty$) during an iteration if it will result in the violation of some constraint. Only those actions that fully comply with the constraints will be performed. It is obvious that the result produced by this modified version of the FLOC algorithm is guaranteed to satisfy the specified constraints.

### 4.4 Alternative Algorithm

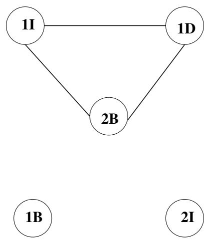
Another way to find the $\delta$-clusters is to map it to the traditional (subspace) clustering problem and apply an existing clustering algorithm (e.g., CLIQUE [1]). This can be achieved in three steps. (1) For each pair of attributes $A_{j1}$ and $A_{j2}$, a new attribute $A_{j1j2}$ is derived to store the difference $A_{j1} - A_{j2}$. If there are $M$ attributes, then $\frac{M \times (M-1)}{2}$ derived attributes will be introduced. Given the data matrix shown in Figure 4(a), Figure 7 shows derived attributes. (2) Given the set of derived attributes, some existing subspace

[6] This is easy to achieve since an initial cluster is not required have low residue.

clustering algorithm can be applied to find all subspace clusters on the derived data matrix. (3) Finally, for each discovered subspace cluster $C^D$ from the derived data, the $\delta$-clusters (from the original data) can be determined with the following method. Let $C^D = O_C \times A_C^D$ where $O_C$ and $A_C^D$ are the set of involved objects and the set of involved derived attributes, respectively. (The shaded entries in Figure 7(a) form a subspace cluster in the derived matrix.) A graph can be constructed from $A_C^D$, where each vertex corresponds to an original attribute and there is an edge between two vertices $A_{j1}$ and $A_{j2}$ if the derived attributes $A_{j1j2}$ is in $A_C^D$. (For the subspace cluster formed by the shaded entries, the corresponding graph is shown in Figure 7(b)). Any clique in this graph indicates the existence of a $\delta$-cluster on the corresponding attributes and the set of objects $O_C$. (Since condition 2I, 2D, and 2B form a clique in Figure 7(b), there exists a $\delta$-cluster involving these three conditions.) As a result, in Figure 7(a), the gene VPS8, EFB1, and CYS3 form a perfect subspace cluster on the dimension 1I1D (difference between condition CH1I and CH1D), 1I2B, and 1D2B. From this subspace cluster, we can find the $\delta$-cluster (shown in Figure 4(b)) at the original matrix.

| genes \ difference | 1I1B | 1B1D | 1D2I | 2I2B | 1I1D | 1B2I | 1D2B | 1I2I | 1B2B | 1I2B |
|---|---|---|---|---|---|---|---|---|---|---|
| CTFC3 | 4108 | -3824 | 3828 | 52 | 284 | 4 | 3880 | 4112 | 56 | 4164 |
| VPS8 | 120 | 161 | -155 | 410 | 281 | 6 | -178 | 126 | -17 | 103 |
| EFB1 | 38 | 243 | -240 | 62 | 281 | 3 | -178 | 41 | 65 | 103 |
| SSA1 | 109 | 183 | -471 | 342 | 292 | -288 | -129 | -179 | 54 | 163 |
| FUN14 | 2572 | -2291 | 2305 | 45 | 281 | 14 | 2350 | 2586 | 59 | 2631 |
| SP07 | -62 | 242 | -237 | 61 | 180 | 5 | -176 | -57 | 66 | 4 |
| MDM10 | 266 | 6 | -11 | 41 | 272 | -5 | 30 | 261 | 36 | 302 |
| CYS3 | 34 | 247 | -237 | 59 | 281 | 10 | -178 | 64 | 59 | 103 |
| DEP1 | 40 | 232 | -233 | 41 | 272 | -1 | -192 | 39 | 40 | 80 |
| NTG1 | 33 | 263 | -241 | 46 | 296 | 22 | -195 | 55 | 68 | 101 |

(a)




(b)

**Figure 7. Subspace Cluster on Derived Attributes**

The major disadvantage of this approach is that the second step (i.e., subspace clustering on the derived data) could be very costly. The dimensionality of the derived data is $\frac{M\times(M-1)}{2}$. In addition, in order to find a $\delta$-cluster with $m$ attributes, the subspace cluster on the derived data has to consist of at least $\frac{m\times(m-1)}{2}$ dimensions. For instance, if the original data has 100 attributes (dimensions) and a $\delta$-cluster consists of 10 attributes (dimensions), then the subspace clustering algorithm is applied to determine a 45-dimension subspace cluster from a data set of 4950 dimensions. This could be a very expensive process. As a result, a more efficient algorithm is needed to discover the $\delta$-clusters.

# 5 Improvement of Basic FLOC Algorithm

In the previous section, we presented the basic algorithm that still has room for improvement. In this section, we present the further optimization on two aspects of the FLOC algorithm.

## 5.1 Initial Size of a Cluster

As mentioned before, the parameter $\rho$ is used to control the size of a cluster in the initial assignment phase. The average number of rows and columns in a cluster is $\rho \times N$ and $\rho \times M$, respectively. We experiment with different value of $\rho$. If the optimal cluster size is very different from the initial (seed) cluster size, then it would take more iterations (iterations) to reach the optimal cluster and the response time could be prolonged. In order to expedite the response time, it is beneficial to make the initial seed as close to the optimal cluster size as possible.

The optimal cluster size is usually unknown in advance and may be significant disparate for different clusters. Therefore, we generate initial clusters of different sizes, i.e., different $\rho$ values for different clusters. In our experiments, we explore the effects of mixed $\rho$ values and find that the FLOC algorithm can well adjust the cluster size and discover both large and small clusters. Table 5 in the experimental results section shows the recall and precision of the FLOC algorithm with the mixed initial clustering technique.

## 5.2 Order of Action

In the previous section, we assume the same order to perform actions at all iterations. This implies that for the set of rows and columns which are at the beginning of the order, their membership always has a higher priority to be changed than the rows and columns at the end of the order. This approach has one drawback. If a large number of actions with negative gain proceed a small number of actions with positive gain at the end of the performance list, then the set of actions with positive gain may never be given a full play. To solve the problem, we need to deploy a dynamic ordering among all actions. In this section, we discuss two possible ways to determine the action order.

#### 5.2.1 Random Order of Actions

In this approach, the order of actions are randomly determined at the beginning of each iteration. This means that the membership of all columns and rows has the same priority to be changed. There are many algorithms to produce a random ordered sequence. Here we present one algorithm that can produce such a sequence. Assume that these actions are stored in an array. A series of action swapping is then performed, where each time two actions are randomly chosen and their positions are swapped. This random swapping procedure repeats $g$ times. We experiment with various value of $g$. We found that the randomness of the list is satisfactory where $g \geq 2 \times (M + N)$. Thus, we chose $g = 2 \times (M + N)$ to generate a random sequence in this paper.

An action may be performed first at one iteration, and performed very late at another iteration. With this technique, the problem mentioned above can be avoided since the positive actions at the end of one iteration may be at a very early position of the next iteration. Table 4 in the experimental results section shows the benefit of this improvement. On average, the quality of the final clustering (i.e., recall and precision) can be improved by a wide margin, e.g., $10\%$.

#### 5.2.2 Weighted Random Order of Actions

In the above random order scheme, each action has the same probability to be assigned as the first action, the second action, and so on, regardless its gain. It means that a positive action has the same chance to be scheduled as the first action as a negative action. Intuitively, it is more desirable to perform actions with greater positive gain early so that its effect can be brought into play early. However, if we sort the actions in descending order of their gains and perform them accordingly, then we may only find the local optimal clustering, but not the global optimal clustering. As a result, we propose a weighted random order scheme to provide better ordering. Informally, this algorithm is very similar to the random order algorithm. The only difference is whether a swap would occur for two randomly picked actions. The rule of thumb is that if the action in the front has a larger gain than the one in the back, then the swap is less likely to occur. Let's consider two actions $a_i$ and $a_j$ and $a_i$ is in front of $a_j$. The gain of the two actions are $g_i$ and $g_j$, respectively. The probability $p(i, j)$ of swapping of $a_i$ and $a_j$ is proportional to $g_j - g_i$. Let $R$ be the difference between the maximum gain and minimum gain of all actions. Then $p(i, j) = 0.5 + \frac{g_j - g_i}{2 \times R}$. When $a_j$ has the maximum gain and $a_i$ has the minimum gain, then the probability of swapping $a_i$ and $a_j$ is 1. On the other hand, if $g_j$ is the minimum gain and $g_i$ is the maximum gain, then $p(i, j) = 0$. In the case that $g_i = g_j$, $p(i, j) = 0.5$. Table 4 shows the improvement of the weighted random order algorithm. The weighted approach produces about $5\%$ improvement in the quality of the final clustering over the random ordering due to the fact that more positive actions are performed. In other words, the priority of membership migration favors the row or column whose gain is largest. As a result, the weighted ordering can provide the best final clustering.

**Table 1. Statistics of Discovered Clusters**

| | | | |
|---|---|---|---|
| cluster volume | 2111 | 1998 | 2755 |
| number of movies | 72 | 36 | 63 |
| number of viewers | 48 | 88 | 65 |
| residue | 0.51 | 0.47 | 0.56 |
| diameter | 10.37 | 18.22 | 15.35 |

## 6 Experimental Results

We experimented the FLOC algorithm with both synthetically generated and real data sets. The algorithm is implemented with C programming language and is executed on an IBM AIX machine with a 333 MHz CPU and 128 MB main memory. In addition, we also implement an alternative algorithm that first transforms the matrix, then applies the subspace clustering algorithm on the transformed matrix, and finally discover $\delta$-cluster from the subspace clusters. There are many subspace clustering algorithms, CLIQUE [1] is chosen for the implementation.

### 6.1 Real Data Sets

We experiment the FLOC algorithm with two real data sets, one is the MovieLens data set and the other is a microarray of gene expression of a certain type of yeast under various conditions.

#### 6.1.1 MovieLens Data Set

The MovieLens data set [8] were collected by the GroupLens Research Project at the University of Minnesota. The data set consists of 100,000 rates, 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. We choose $\alpha = 0.6$ for this experiment. In this test, we choose the initial number of clusters as 5, 10, and 20. It takes less than one minute (6 iterations) to complete the clustering process in all three cases. Table 1 shows the statistics of some discovered clusters. A viewer's rating can be regarded as a point in high dimension space. A $\delta$-cluster is a set of such points. The diameter of a cluster is defined as the diameter of the minimum bounding box for the cluster.

From this experiment, we found some interesting $\delta$-clusters. Let's take a closer look on some discovered clusters. For example, there exists a cluster whose attributes consists of two types of movie, 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 $\delta$-cluster model. For example, two viewer's rating for the same 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 $\delta$-cluster model groups them together because they are coherent. However, this type of coherence can not be discovered by the traditional clustering algorithm because the ratings are too disparate.

#### 6.1.2 Micro Array

The yeast micro array contains 2884 genes under 17 conditions [13]. This data set is in the form of a matrix. Each row corresponds to a gene while a column represents a condition under which the gene is developed. Each entry in the matrix is the logarithm of the ratio of the actual strength of the gene under the test condition and the expected strength of the gene. Biologists want to find clusters of genes and conditions such that the set genes whose expression levels rise and fall coherently under the set of conditions [3]. Fortunately, the $\delta$-cluster model can serve for this purpose. In [3], 100 clusters are presented based on the data set in [13]. The average residue of each cluster is 12.54. We also experiment the FLOC algorithm on the same data set to find the 100 clusters. The average residue of clusters discovered by the FLOC algorithm is 10.34. This means that the FLOC algorithm can find better $\delta$-clusters than that presented in [3]. Also the aggregated volume of all clusters discovered by the FLOC algorithm is almost 20% greater than that in [3] which means that our $\delta$-clusters cover more genes and conditions. Moreover, the response time of the FLOC algorithm is an order of magnitude less than that of the algorithm described in [3].

### 6.2 Synthetic Data

To better understand the FLOC algorithm, several large data sets are synthetically generated. We investigate in both the performance and the quality of the FLOC algorithm.

#### 6.2.1 Performance

**Data Matrix size**

In the first experiment, we study the effect of the data matrix size on the performance of the FLOC and the alternative algorithms. In this test, we choose the number of clusters that we are interested in $(K)$ to 10, 20, 50, and 100, and the average initial volume of each cluster is $0.05 \times N$ and $0.2 \times M$ where $N$ and $M$ are the number of objects and number of attributes, respectively. Table 2 shows the number of iterations with various matrix size and number of clusters. The matrix is generated in the following matter. Fifty $\delta$-clusters are embedded in each data set. The average volume of a cluster is $(0.04 \times N) \times (0.1 \times M)$. It is clear that the number of iterations depends on the size of the input matrix and the number of clusters. However, the increase of the number of iterations is at a very slow pace. For the matrix with 300,000 entries, it only takes 11 iterations to determine 100 $\delta$-clusters. This leads to a very efficient algorithm. Table 3 illustrates the response time to discover $\delta$-clusters with various data sets. In the most expensive case, i.e., 100 clusters in a matrix with $3000 \times 100$ entries, the response time is about 33 minutes. From Table 3, the response time of the FLOC algorithm is roughly linearly proportional to the volume of the matrix and the number of clusters. This coincides with the complexity analysis of the FLOC algorithm in the previous section.

**Table 2. Number of iterations with respect to the matrix size and number of clusters**

| Num of cluster | Matrix volume | | | |
|---|---|---|---|---|
| | $100 \times 20$ | $500 \times 50$ | $1000 \times 50$ | $3000 \times 100$ |
| 10 | 5 | 7 | 7 | 9 |
| 20 | 5 | 8 | 9 | 10 |
| 50 | 6 | 8 | 9 | 10 |
| 100 | 7 | 9 | 10 | 11 |

**Table 3. Response time (sec.) with respect to the matrix size and number of clusters**

| Num of cluster | Matrix volume | | | |
|---|---|---|---|---|
| | $100 \times 20$ | $500 \times 50$ | $1000 \times 50$ | $3000 \times 100$ |
| 10 | 12 | 58 | 134 | 335 |
| 20 | 20 | 98 | 209 | 602 |
| 50 | 40 | 192 | 399 | 1170 |
| 100 | 78 | 376 | 721 | 1950 |

**Initial Cluster Volume**

Now we are studying the effects of initial cluster volume on the number of iterations in the FLOC algorithm. In this test, we embed 100 clusters with volume 100 in a matrix of $3000 \times 100$. The expected initial cluster volume is set to $(c \times 3000) \times (c \times 100)$. When $c = 0.0033$, the initial clusters and the embedded clusters have similar volume. Figure 8(a) and (b) shows the number of iterations and the response time of the FLOC algorithm, respectively. The x axis is $\frac{V_{int} - V_{emb}}{V_{emb}}$ where $V_{int}$ and $V_{emb}$ are the average volume of the initial cluster and the embedded cluster, respectively. From this figure, we can see that the number of iterations is minimized if the initial cluster volume is similar to that of the final cluster volume (i.e., the ratio is 0) because less number of moves is required, and in turn leads to the efficient response time. Notice that the two curves in Figure 8(a) and (b) have a very similar shape because the response time is highly correlated to the number of iterations.

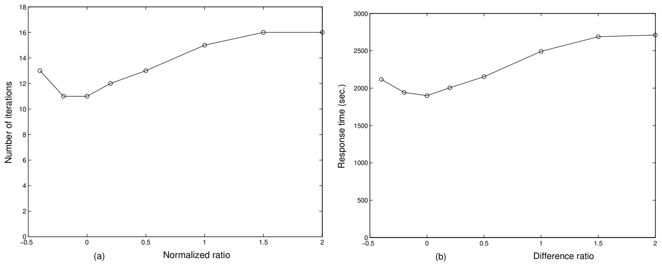



**Figure 8. Effects of the initial cluster volume**

In the previous experiment, the embedded clusters have the similar volume. However, in a real data set, the clusters usually have different volumes, e.g., one cluster may have 20 entries while another may have 200 entries. We now study the effects of different cluster volumes on the FLOC algorithm. First, clusters with various volumes are embedded in a Matrix of $3000 \times 100$ entries. The volume of embedded clusters follows an *Erlang distribution*[7]. When the variance of the embedded clusters increases, the volume of the embedded clusters becomes more different. On the other hand, the volume becomes more homogeneous with smaller variance. All clusters have the same volume if the variance is 0. In addition, four sets of initial clusters are also generated. The volume of a set of initial clusters also follows an Erlang distribution. The average volume of the embedded clusters and the initial clusters is the same, 300. Figure 9(a) and (b) show the number of iterations and the response time of the FLOC algorithm with various embedded cluster volume distribution. Each curve in the figure indicates a distinct set of initial clusters whose volume has the given variance. The algorithm yields the best performance when the embedded cluster volume is the same as the initial cluster volume. This could happen if we can correctly guess the volume of the targeted clusters. However, in many situations, users may not have a clear idea of the embedded clusters volume. In the case that the initial cluster volume is different from that of the embedded clusters, the algorithm that employs the most divergent initial cluster volumes can tolerate the most disparity among the embedded cluster volume.

**Comparison with Alternative Algorithm**

We also compare the FLOC algorithm with the alternative algorithm (subspace clustering). Figure 10 illustrates the performance of the alternative algorithm and the FLOC algorithm. (Note that due to the scale, we are only able to plot part of the curve for the subspace clustering algorithm, up to 100 attributes in the figure.) We fix the number of clusters to 100 and the number of objects to 3000. When the number of attributes increases, the response time of the alternative algorithm increases at a much faster pace than ours.

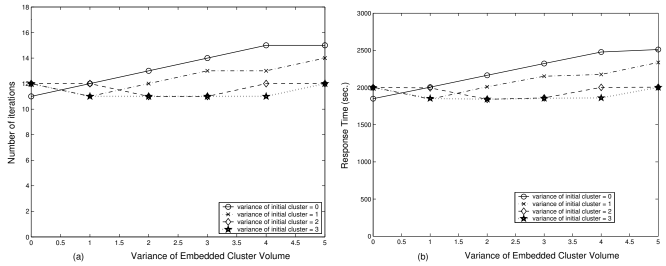



**Figure 9. Effects of the various initial cluster volume**

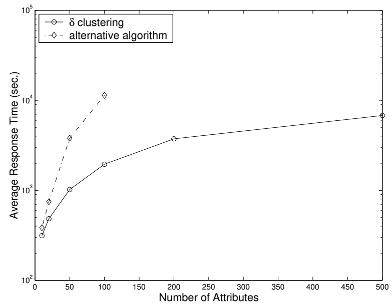



**Figure 10. Comparison with the alternative algorithm**

### 6.2.2 Quality of the FLOC Algorithm

In this section, we analyze the quality of our FLOC algorithm with respect to the order of actions and the initial volume of the clusters.

**Order of Actions**

One of the improvements that we made to the FLOC algorithm is the action reordering at the beginning of each iteration. Table 4 shows the effects of action reordering. There are three ordering methods: fixed, random, and weighted. The quality of results are measured by three measurements: residue, precision, and recall. Let $U$ be the set of entries in the embedded clusters and $V$ be the set of entries in the clusters discovered by our algorithm. The recall of our algorithm is defined as $\frac{U \cap V}{U}$ and the precision is $\frac{U \cap V}{V}$. We test the algorithm with various matrix sizes and embedded cluster volumes, and the average results are reported. In this test,

**Table 4. Quality of the FLOC algorithm with respect to action orders**

| | fixed order | random order | weighted order |
|---|---|---|---|
| residue | 12.5 | 11.5 | 11 |
| recall | 0.75 | 0.82 | 0.86 |
| precision | 0.77 | 0.84 | 0.88 |

**Table 5. Quality of the FLOC algorithm with respect to various embedded cluster volume**

| variance | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| residue | 10.9 | 11.1 | 11.0 | 10.9 | 11.0 | 11.1 |
| recall | 0.87 | 0.87 | 0.86 | 0.87 | 0.87 | 0.86 |
| precision | 0.87 | 0.88 | 0.90 | 0.89 | 0.89 | 0.88 |

the initial volume cluster follows an Erlang distribution with variance 3. It is clear that the random ordering outperforms the fixed ordering due to the fact that positive gain actions will not always be blocked by the preceding negative gain actions. In addition, the weighted order outperforms the random order because the weighted order favors actions with large gains while still allow enough room for the algorithm to surpass local optimum.

**Initial Cluster Volume**

In this experiment, we embedded 100 clusters with various volume in a matrix with $3000 \times 100$ entries. The average residue and average volume of these clusters are 5 and 300, respectively. The volume of the embedded clusters follows an Erlang distribution. Table 5 shows the quality of the our FLOC algorithm with respect to the variance of the Erlang distribution. In this test, the weighted order technique is employed. In the algorithm, the volume of initial clusters also follows an Erlang distribution with variance 3. The quality of the clustering (in terms of average residue, recall, and precision), is similar with respect to the variance of the distribution of embedded cluster volume. We, thus, can conclude that the various cluster volume has more impact on the efficiency of our algorithm rather than the quality of the results because, even if the initial clusters are far from the optimal ones, our FLOC algorithm can still “reshape” the clusters into good quality.

## 7 Conclusion

In this paper, we proposed a new model called $\delta$-cluster to capture the coherent objects. In many applications, e.g., collaborative filtering and bioinformatics, the objects may have a certain degree of bias which can conceal the true coherence among objects. To address this issue, we introduce the concept of *base* to represent potential bias an object or attribute might hold in a cluster and employ a new measurement — *residue* to assess the degradation of the coherency among a set of objects on a set of attributes. Our model also can accommodate many additional features, e.g., minimum coverage, maximum overlap, etc.. In addition, due to the complexity of the problem, we devised a move-based algorithm (FLOC) that can efficiently and effectively discover near optimal $\delta$ clusters. This algorithm can also be easily modified to support the additional feature of our model. Last but not least, several real and synthetic data sets are employed to show that the performance of the FLOC algorithm is far superior to alternative algorithms.

## References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, Authomatic subspace clustering of high dimensional data for data mining applications, *Proc. ACM SIGMOD*, pp. 94-105, 1998.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, When is nearest neighbors meaningful, *Proc. ICDT*, pp. 217-235, 1999.

[3] Y. Cheng and G. Church, Biclustering of expression data, *Proc. 8th ISMB*, 2000.

[4] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[5] H. V. Jagadish, J. Madar, and R. T. Ng. Semantic compression and pattern extraction with fascicles. *Proc. 25th VLDB*, pp. 186-198, 1999.

[6] L. Kaufmann and P. Rousseuw, Finding groups in data — an introduction to cluster analysis, *Wiley series in Probability and Mathematical Statistics*, 1990.

[7] L. Kleinrock, *Queueing Systems*, John Wiley & Sons, 1975.

[8] MovieLens Data Set, available at *http://www.cs.umn.edu/Research/GroupLens*.

[9] U. Shardanand and P. Maes, Social information filtering: algorithms for authomating “word of mouth”, *Proc. ACM SIGCHI*, pp. 210-217, 1995.

[10] N. Slonim and N. Tishby, Document clustering using word clusters via the information bottleneck, *Proc. ACM SIGIR*, pp. 208-215, 2000.

[11] W. Wang, J. Yang, and R. Muntz, STING: a statistical information grid approach to spatial data mining, *Proc. VLDB*, pp. 186-195, 1997.

[12] J. Yang, W. Wang, and R. Muntz, Collaborative web caching based on proxy affinity, *Proc. ACM SIGMETRICS*, pp. 78-89, 2000.

[13] Yeast Micro Data Set, available at http://arep.med.harvard.edu/network_discovery, 2000.