# FINDIT : a Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting

Kyoung-Gu Woo Jeong-Hoon Lee Department of Electrical Engineering and Computer Science Korea Advanced Institute of Science and Technology Taejon, Korea

{kgwoo, leejh }@dbserver.kaist.ac.kr

# Abstract

The aim of this paper is to present a novel subspace clustering method named FINDIT. Clustering is the process of finding interesting patterns residing in the dataset by grouping similar data objects from dissimilar ones based on their dimensional values. Subspace clustering is a new area of clustering which achieves the clustering goal in high dimension by allowing clusters to be formed with their own correlated dimensions.

In subspace clustering, selecting correct dimensions is very important because the distance between points is easily changed according to the selected dimensions. However, to select dimensions correctly is difficult because data grouping and dimension selecting should be performed simultaneously. FINDIT determines the correlated dimensions for each cluster based on two key ideas: *dimension-oriented distance* measure which fully utilizes dimensional difference information, and *dimension voting* policy which determines dimensions in a probabilistic way based on *V* nearest neighbors' information. Through various experiments on synthetic data, FINDIT is shown to be very successful in the high dimensional clustering problem. FINDIT satisfies most of the requirements for good clustering methods such as the accuracy of results, the robustness to noise and cluster density, and the scalability to dataset size and dimensionality. Moreover, it is gracefully scalable to full dimension without any modification to algorithm.

# **1** Introduction

Data Mining is the process of extracting unknown and potentially useful information from database. It can be used in many application areas like insurance, health-care, database marketing, stock management, and scientific knowledge discovery. Clustering is one of the frequently used tools in Data Mining, which refers to the process of partitioning data so that intra-group similarities are maximized and inter-group similarities are minimized at the same time. It is especially useful in the condition when there is little knowledge about the given dataset. Therefore, many data clustering techniques have been proposed[3, 5, 6, 7, 8, 9, 15, 16, 22]. However conventional clustering methods do not scale well to high dimension in terms of effectiveness and efficiency. There are some problems that prevent them from perfoming well in high dimensional datasets. Firstly, it is difficult to distinguish similar points from dissimilar ones, since the distance between any two points becomes almost the same value[17]. Secondly, clusters tend to exist in different subspaces[10].

One possible extension of conventional clustering is the application of dimension reduction techniques. These approaches lower the dimensionality first, either by removing less important dimensions or by transforming the original space to a low dimensional space. Then conventional clustering techniques are applied to the dataset in reduced dimensions. But because clusters can be formed in different subspaces, such kinds of dimension reduction can get rid of useful dimensional information from some clusters. As a result of the lost information, it can generate clusters that may not fully reflect the original clusters' properties. Moreover, the generated result is usually not suitable for further analysis, which is required by many data mining applications.

Subspace clustering is the answer to this challenge. It achieves the clustering goal by allowing clusters to be formed with their own correlated dimensions. Since it was first proposed by [10], several different subspace clustering methods have been presented and had some success[13, 14, 18, 19, 20]. Recently suggested subspace clustering algorithms can be broadly classified into two categories: gridbased approach[10, 14, 18] and partitioning approach[13, 19]. The grid-based approaches mainly focuses on the detailed space segmentation to report the dense regions, and the partitioning approaches focus on the disjoint cluster generation.

CLIQUE[10], the first subspace clustering algorithm, partitions the whole data space into non-overlapping rectangular units and then searches for dense units based on the assumption: "If a k-dimensional region is dense, the (k-1) dimensional region containing it should also be dense". After dense units are found,

several sets of connected dense units are reported as clusters. CLIQUE performs a heuristic pruning step to reduce the number of candidate spaces which increases exponentially according to the dimensionality. However, there is a tradeoff between accuracy and time. ENCLUS[14] uses the similar approach suggested by CLIQUE. It suggests three requirements for good clusters and one measure, named *entropy measure*, that satisfies those requirements. In the clustering process, the entropy measure is used to prune away uninteresting subspaces efficiently. MAFIA[18] is another extension of CLIQUE. It proposes so-called adaptive grid to enhance the effectiveness and efficiency of CLIQUE and adopts parallel processing to deal with large datasets. These three subspace clustering methods are effective at finding arbitrarily-shaped clusters. However, the scalability to high dimension is the common problem of grid-based approaches.

PROCLUS[13] is a variation of k-medoid algorithm in subspace clustering. It starts by choosing a random set of k medoids and iterates to improve the quality of result by exchanging bad medoids with new ones. In each iteration, all data points are assigned to their nearest medoids, and each cluster's dimensions are selected based on assigned points. When the quality of result does not change within a certain number of medoid changes, the algorithm stops and reports the generated clusters as the result. ORCLUS[19] is the extended version of PROCLUS, that deals with the correlation of the arbitrary axis. As in PROCLUS, in each iteration, points are assigned to the nearest seeds to form clusters, but the distance is measured in arbitrary dimensions of each cluster. To find out the hidden dimensions of clusters, ORCLUS uses singular value decomposition which is a famous dimension reduction technique. At the end of each iteration, the number of seeds and their dimension sizes are reduced according to the given factor  $\alpha$  and  $\beta$ .(Note that the initial number of seeds are much larger than the number of clusters k.) The algorithm stops when the remaining number of seeds are reduced to k. In general, both PROCLUS and ORCLUS generate highly disjoint clusters compared to CLIQUE.

CLTree[20] is another interesting method which does not belong to any of the above two classes. It modifies *decision tree*, which is originally a classification method, to make a subspace clustering algorithm. Because decision tree algorithm requires every point to have a binary class label, CLTree regards all points in the given region as labeled Y, and then scatters virtual points labeled N into the given region uniformly. For Y class points and N class points in the given region, the best cut distinguishing two classes is selected. Sequentially two partitioned regions become the child nodes of the original region, and this process is repeated until the decision tree is completely constructed. After decision tree construction, since too many regions are usually generated, pruning and merging steps are subsequently performed to make appropriately-sized clusters.

Although these previous subspace clustering methods have been successful in some points, they have problems relating the scalability to the dimensionality[10, 14, 18] and dataset size[13, 19]. Moreover, none of them has reported the robustness result to the noise ratio and the volume of clusters. The robustness against them becomes important, because when the dimensionality of the dataset increases, the noise(i.e. outliers) would probably increase, and a cluster can have some dimensions in which its points are widely distributed.

# 1.1 Contributions and Layout of the paper

We present an algorithm, named FINDIT(a Fast and INtelligent subspace clustering algorithm using DImendion voTing), which is highly accurate in various conditions and very fast in spite of the increasing dataset size and dimensionality. We also suggest a new distance measure devised for subspace clustering which could be usefully adopted by other high dimensional clustering methods.

The remainder of the paper is organized as follows. Our motivation and the algorithm overview are written in Section 2. The detailed explanations of the algorithm are presented in Section 3. Section 4 explains about the datasets used in the experiments and the performance results. In Section 5, we discuss the various properties of FINDIT. Section 6 contains conclusions of our research work.

# 2 Preliminaries

# 2.1 Motivation

### **A. Dimension-Oriented Distance**

Dimension oriented distance(dod) is our unique distance measure which utilizes dimensional difference information and value difference information together, while the conventional distance measures use only value difference information. We measure the similarity between two points by counting the number of dimensions in which the two points' value difference is smaller than the given  $\epsilon$ , because we think they are "near enough" on that dimension. That is, if the Manhattan distance between two points is less than  $\epsilon$  on that dimension, we think the two points are "equal" on that dimension. This distance measure is based on the philosophy that in high dimension it is more meaningful to be close to each other in many dimensions than to be very close to each other only in a few dimensions. For a point p in a dataset, let  $D_p$  be a subspace composed of the dimensions in which the dimensional values of p are meaningful,<sup>12</sup> and let p(d) be a value of p in d-dimension. We define directed dod from a point p to a point q as :

$$dod_{\epsilon}(p \to q) = |D_p| - |\{ d \mid |p(d) - q(d)| \le \epsilon, d \in D_p \cap D_q \}|,$$

and we define dod(p,q) as the maximum value of two directed dod values between p and q. That is, ;

$$dod_{\epsilon}(p,q) = max\{dod_{\epsilon}(p \to q), dod_{\epsilon}(q \to p)\}.$$
(1)



Figure 1: Conventional Distance vs. Dimension-Oriented Distance when  $\epsilon$ =5

To explain the meaning of *dimension-oriented distance*, we present an example in Figure 1. For a given  $\epsilon = 5$ , point *B* is closer to *O* than point *A* if we use conventional value difference distance measures such as Euclidean distance measure and Manhattan distance measure. On the other hand by our distance measure(i.e. by *dimension-oriented distance measure*), *A* is closer to *O* than *B* is because *A* is within  $\epsilon$ -range of *O* on every dimension. Note that, if not specified differently, the distance in this paper always means *dod* distance for the given  $\epsilon$ .

#### **B.** Estimating Correlated Dimensions based on Nearest Neighbors

In subspace clustering, since the similarity between two points is easily changed according to the selected dimensions, finding the real correlated dimensions is the key to generate good clusters. However, this dimension selection problem is very difficult because data grouping and dimension selecting should

<sup>&</sup>lt;sup>1</sup>We use the notation D to denote the whole space of the given dataset.

<sup>&</sup>lt;sup>2</sup>"Meaningful" means that the corresponding dimensional value is used to measure the distance to other points. For each point p in the dataset, the initial  $D_p$  is D because we do not know the subspace in which the point p belongs at the beginning of clustering.

be performed at the same time. FINDIT estimates the original correlated dimensions of a point p, utilizing its nearest neighbors' information. Figure 2 explains the usefulness of using nearest neighbors' information.

Cluster	Size	Correlated Dimension
Α	43	3, 5, 6, 8, 10
В	160	1, 3, 5, 6, 8, 9, 10
С	129	9, 10
D	31	2, 3, 6, 8, 9, 10
Ε	337	2, 4, 6, 7
Outlier	300	





(a) Sample DataSet

(b) Point p 's 10 nearest neighbors



Figure 2: Sample dataset and 10 nearest neighbors of point p and q

In this example, a dataset composed of 1,000 points in a 10-dimensional space is presented in (a). There are five different clusters that have their own correlated dimensions, and for each correlated dimension of a cluster C, all points in C have values generated according to the normal distribution with a certain mean value and average variance of 9. For the non-correlated dimensions and outliers, the dimensional values are uniformly selected from the whole value range.(All dimensions' value ranges are set to [0,100].) Finally, 30% of dataset are generated as noise(or outliers).

Two matrices (b) and (c) show the 10 nearest neighbors of points p and q respectively, where  $\epsilon$  is given as 15. In (b) and (c), a grey cell means that difference between the given point and the corresponding neighbor is smaller than or equal to the given  $\epsilon$ (=15) value for the dimension identified by the row number. For each dimension in (b), if we pick up the dimensions which has more than 7 grey cells, its selected dimension set becomes {2,4,6,7} which is same as the correlated dimensions of Cluster E in (a). We can obtain the dimension set {1,3,5,6,8,9,10} for the case of (b) by this method.(Note that, the outlier points usually have empty or tiny dimension sets, it is not shown here due to the lack of space)

This example shows the possibility that we can estimate an original cluster's correlated dimensions by using neighbors' information of a point which belong to the original cluster, although some problems remain such as the decision threshold. We developed our unique dimension selection method, named Dimension Voting, based on this possibility. When V nearest neighbors are chosen for the dimension selection process, we call them voters because the following process is similar to voting on D different questions related to whether or not a certain dimension d is a correlated dimension of the original cluster where the given point belongs. To use dimension voting, we should find out the proper values for the number of voters, the threshold for the decision, and the appropriate size of  $\epsilon$ . The method to determine these values is written in Sec 3.

## 2.2 Algorithm Overview

#### A. User Parameters and Result

The input for FINDIT is the dataset and two user parameters  $C_{minsize}$  and  $D_{mindist}$ . Several disjoint clusters in their own subspaces and one outlier cluster are generated as a result of clustering. Because our user parameters  $C_{minsize}$  and  $D_{mindist}$  are not usual parameters used in previous clustering methods, we explain their meaning here. The first parameter  $C_{minsize}$  reflects the user's wish about the minimum size<sup>3</sup> of clusters, since the clusters containing relatively a small number of points compared to dataset size do not interest the user in most cases. This minimum cluster size of interst usually increases according to the dataset size. Hence, FINDIT does not report the clusters smaller than  $C_{ninsize}$ , and this size constraint which is natural but has been neglected by many previous methods, becomes a useful information for FINDIT. The second user parameter  $D_{mindist}$  is the requirement for the minimum difference between two resultant clusters. If the *dod* distance between two clusters is smaller than  $D_{nindist}$  for the selected  $\epsilon$ , they are regarded as one cluster by FINDIT and are subsequently merged. FINDIT uses  $D_{mindist}$  to make a decision on whether two clusters should be merged or not, especially when they are similarly distributed for some dimensions and differently for other dimensions. From a different point of view,  $D_{mindist}$  can be thought of as the number of correlated dimensions that a cluster should neglect to be merged with another.

#### **B.** Brief Summary of the Clustering Process

FINDIT is composed of three phases: sampling phase, cluster forming phase, and data assigning phase. In sampling phase, two different samples are generated by a random sampling method. The first set S is constructed from the dataset as a distribution sample representing the given dataset, and the second set M is also made from the dataset but is smaller than S. The points in M are used as representatives(=medoids) of original clusters.

The goal of the second phase, i.e. cluster forming phase, is to obtain the original clusters' correlated dimensions and locations using two samples S and M. To achieve the goal, firstly, we determine the

<sup>&</sup>lt;sup>3</sup>Whenever we refer to the *size* of a cluster or a medoid, *size* means the number of points assigned to it. For the range of value distribution of a cluster's points in a dimension, we use the term *diameter* or *radius* instead.

correlated dimensions of all medoids in M by our dimension voting method. Then the medoids which are near from each other in *dod* measure are grouped together. This group of medoids, named medoid cluster, is a sort of wireframe which simulates the original cluster's size and correlated dimensions. Because different medoid cluster sets are generated for each iteration on  $\epsilon$ , an evaluation criteria is used to choose the best  $\epsilon$  and the corresponding medoid cluster set  $MC_{\epsilon}$ . After cluster forming phase, the selected medoid cluster set is given to the following phase as the best summary of the original clusters. In data assignment phase, all points are assigned to their nearest medoid clusters, and the points not assigned to any medoid cluster are regarded as outliers.



Figure 3: Overall Framework of FINDIT

# **3** FINDIT(a Fast and INtelligent subspace clustering algorithm using DImension voting)

# 3.1 Sampling Phase

In sampling phase, two different samples S and M are made from the dataset considering the dataset size N and the user parameter  $C_{minsize}$ . In order for the subsequent process to work properly, any original cluster larger than  $C_{minsize}$  should have more than a certain number of points in S and have at least one point in M. For S and M to satisfy the above property, we should answer to the question of "How many points should be selected for S and M respectively?". As a solution of this question, *Chernoff bounds* is proposed in [22]. According to *Chernoff bounds*, the minimum size of sample S, which assures that every cluster in sample S will have more than  $\xi$  points in the sample by the probability of  $1-\delta$ , is computed by the following equation:

$$Chernof f-bounds(S) = \xi k\rho + k\rho log(\frac{1}{\delta}) + k\rho \sqrt{(log(\frac{1}{\delta}))^2 + 2\xi log(\frac{1}{\delta})}.$$
 (2)

Constant k is the number of clusters, and  $\rho$  is a value satisfying  $minsize = \frac{N}{k \cdot \rho}$  ( $\rho \ge 1$ ), where N is the size of the dataset and minsize is the size of the smallest cluster in the dataset.

To obtain *Chernoff bounds* for sample *S*, typically we use the setting  $\xi = 30$ ,  $\delta = 0.01$ , minsize =  $C_{minsize}$ ,  $k = \frac{N}{C_{minsize}}$ , and  $\rho = 1.(\rho$  is computed following the equation  $\rho = \frac{N}{k \times C_{minsize}}$ .) Constant *k* is set to  $\frac{N}{C_{minsize}}$  from the fact that there can be maximally  $\frac{N}{C_{minsize}}$  clusters because the smallest cluster size is given as  $C_{minsize}$  by the user. This setting means that when |S| is greater than *Chernoff-bounds(S)*, even the smallest cluster in the dataset has a 99% probability to contain more than 30 points in sample *S*. When *N* is 100,000 and  $C_{minsize}$  are given as 5,000, sample size |S| satisfying Eq. 2 is approximated to 1037. Especially since Constant  $\xi$ , which defines the minimum cluster size in *S*, corresponds to the role of  $C_{minsize}$  in the dataset, we use the notation  $s_{minsize}$  to indicate the value used for  $\xi$ . Hence  $s_{minsize}$  is 30 in the above example. The *Chernoff bounds* for medoid sample *M* is obtained within the same setting used to estimate |S|, except for  $\xi$ . In this case,  $\xi$  is set to 1 because at least one medoid is required for the smallest cluster. As a result, the *Chernoff bounds* for *M* is approximated to 224 for the same dataset.

To get sample S and M, we randomly select points from the dataset, because previous experiments show that a random sample is at least better than the sample obtained by BIRCH clustering method[7, 11]. Note that any sampling method can be used instead of random sampling as long as it guarantees good summary of the dataset. From the point of the time complexity, the time required in sampling phase is negligible compared to the whole clustering process. When dataset size is N and the required random sample size is |S|, the complexity of sampling is known to be  $\Omega(|S| + |S|log_{\overline{|S|}}^N)$ [1]. And the sample process needs just one sequential scan of the dataset because M and S are generated simulaneously. More detailed information about sampling can be found in [1, 11, 21, 22].

# 3.2 Cluster Forming Phase

Cluster forming phase is iterated several times with increasing  $\epsilon$  value. Throughout the whole phases of FINDIT, except sampling phase,  $\epsilon$  is a key running parameter which determines *dod* distance. If  $\epsilon$  is too small, the distance between any two points becomes *D*, which is the maximum distance, and as a result the clustering cannot be performed. Hence to find appropriate  $\epsilon$ , we try 25 different values in  $\left[\frac{1}{100}valuerange, \frac{25}{100}valuerange}\right]$ . At each end of iteration, 25 medoid cluster sets generated by corresponding 25 different  $\epsilon$  values are evaluated by our soundness criteria. Finally, we determine the best medoid cluster set and use it in the following data assignment phase. procedure **ClusterForming**( $s_{minsize}$ ,  $D_{mindist}$ ) //  $s_{minsize}$ : Size Threshold for the smallest cluster in S//  $D_{mindist}$ : Distance Threshold for any two medoid cluster pair S: Set of sampled points M: Set of sampled medoids E: Set of 25 candidate epsilons =  $\{\frac{1}{100}valuerange, \frac{2}{100}valuerange, \cdots, \frac{25}{100}valuerange}\}$  $MC_{\epsilon}$ : Set of medoid clusters for given  $\epsilon$ 

#### begin

 $bestepsilon := 0; MC_{bestepsilon} := 0;$ for each  $\epsilon$  in E do begin
Step 1. Determine key dimensions for every medoid m in M;
Step 2.1. Assign every point p in S to the nearest medoid in M;
//Each assigned medoid should satisfy Eq. 4
Step 2.2. Merge similar medoids in M to make medoid cluster set  $MC_{\epsilon}$ ;
//Any two medoid clusters with less than  $D_{mindist}$  distance are merged together.
Step 2.3. Refine medoid clusters in  $MC_{\epsilon}$ ;
//Medoid clusters which are too small are removed from  $MC_{\epsilon}$ .
//After dimension tuning, a few medoid clusters are additionally merged together.
Step 3 if Soundness( $MC_{\epsilon}$ )  $\geq$  Soundness( $MC_{bestepsilon}$ ) then bestepsilon :=  $\epsilon$ ;
end
return  $MC_{bestepsilon}$ 

end

Figure 4: Cluster Forming Subroutine

#### **Step 1. Dimension Voting**

The purpose of this step is to determine the correlated dimensions for all points(=medoid<sup>§</sup>.) in M and we will use the notation *key dimensions* to indicate these correlated dimensions. To get the information required to decide *key dimensions*, V nearest neighbors are sequentially searched from S for each medoid in M. (note that sequential search is fast enough because  $|S| \cdot |M|$  is small enough) The distance between a point p in S and a medoid m in M is measured as follows:

$$dod_{\epsilon}(m,p) = |D| - |\{ d \mid |m(d) - p(d)| \le \epsilon, d \in D\}|.^{5}$$
(3)

This is the *dod* measure defined in Eq. 1 because  $D_m = D$  and  $D_p = D$ . (Until now, p and m are all meaningful in the whole data space D. That is, no correlated dimensions are selected for both points yet.)

As mentioned in Sec. 2, we regard this V nearest neighbors as voters, and the dimension selection as the voting process for D questions. For the question "Is Dimension d a *correlated dimension* of the

 $<sup>^4</sup>$ From now on, we refer to a point in M as a medoid to distinguish it from the point in S

 $<sup>{}^{5}</sup>p(d)$  is the *d*-dimensional value of a point *p*.

cluster to which m belongs?", if a neighbor's d-dimensional value from m is less than or equal to the given  $\epsilon$ , it is considered that the neighbor votes on "Yes". Then the question is transformed into two different questiones: "What is the threshold for a correct decision?" and "What is the proper number of voters?" We explain them in Property 1 and Property 2.

**Property 1** The number of "Yes" votes on any non-correlated dimension follows binomial probability distribution B(n : V, p) where  $p = \frac{2\epsilon}{valuerange}$ , provided that all voters belongs to the same original cluster as the given medoid m.

*Proof.* When we assume that all V voters belong to the same original cluster as m, the remaining question is to categorize the dimensions polled "Yes" votes into two classes: the dimensions supported by a certain correlation and the dimensions supported by chance. Based on the meaning of *correlation* and the assumption of V voters, we can say that the values of voters for a non-correlated dimension d are uniformly distributed throughout all value ranges of the dimension d. Then the random probability p of  $|m(d) - v(d)| \le \epsilon$ , which means the "Yes" vote by chance, becomes  $\frac{2 \cdot \epsilon}{valuerange}$ . Because there are V voters and each voter's d-dimensional value is independent from each other, this voting process follows *binomial probability distribution*  $B(n : V, \frac{2 \cdot \epsilon}{valuerange})$  where n is the random variable between 0 and V. Based on the above model, the threshold that makes the probability  $P(X \ge T)$  to 0% can be easily obtained by referencing cumulative binomial distribution table.

# **Property 2** The maximum number of reliable voters is $s_{minsize}$ .

*Proof.* The assumption for Property 1 is that all V voters belong to the same original cluster as that of the given medoid m belongs to. Because the distance measure used to find V nearest neighbors is our suggested dod measure, the probability for an irrelevant point to be close to m without any correlation is very low, even with large  $\epsilon$  value. Moreover, when two points are near enough from each other in many dimensions, we cannot say they are irrelevant. Then how many nearest neighbors can participate in the voting? The reliable bound for this question is obtained from the setting for sample S. All original clusters larger than  $C_{minsize}$  have more than  $s_{minsize}$  points of their own in S. When  $\epsilon$  value increases, the points which belong to the same original cluster eventually become nearest neighbors of m, and their number amounts to  $s_{minsize}$  even for the smallest cluster. Hence, the maximum number of reliable voters is  $s_{minsize}$ .

Any number less than  $s_{minsize}$  is good for V. However, we suggest a value between 10 and  $s_{minsize}$  because the disciminating power of binomial probability distribution is weakened when V is less than

10. When a dimension d has polled more "Yes" votes than the referenced threshold, we regard that a certain correlation exists in d.<sup>6</sup> For example, when V = 20,  $\epsilon = 10$ , and the normalized valuerange is [0,100], the probability that a certain dimension d polls more than 11 votes without any correlation is referenced as 0. So we use 12 as the decision threshold in this case.

### Step 2. Cluster Simulating

Step 2.1. Member Assignment : Since the set of key dimensions  $KD_m$  is generated for each medoid m in Step 1, each m is meaningful in its subspace composed of  $KD_m$ . To verify whether there is really a cluster in the subspace of  $KD_m$ , points in S are assigned medoids. For every point p in S and every medoid m in M, p is assigned to the nearest m satisfying the member condition in Eq. 4.

member assignment condition : 
$$dod_{\epsilon}(m \to p) = 0.$$
 (4)

This means that p can be assigned to (m)s such that p is within  $\epsilon$  from m for all dimensions in  $KD_m$  and is assigned to the one of them which has the largest key dimensions.

	Medoid	$KD_m$	m's position
$p_1 = (5, 6, 6, 5, 8)$	<i>m</i> <sub>1</sub>	{1,3,4}	(1, 7, 3, 2, 8)
$n = (3 \ 2 \ 0 \ 0 \ 7)$	<i>m</i> <sub>2</sub>	{2,3}	(3, 5, 7, 1, 4)
$p_2 = (3, 2, 9, 9, 7)$	$m_{_3}$	{3,5}	(2, 6, 1, 4, 3)
$D=5$ $ M =5$ $\varepsilon = 2$	$m_{_4}$	{1,4,5}	(9, 9, 8, 1, 2)
	$m_{s}$	{2,3,4,5}	(7, 8, 5, 6, 9)

Medoid	$dod_2(m > p_1)$	$dod_2(p_1 \rightarrow m)$	$dod_2(m, p_1)$	Medoid	$dod_2(m \rightarrow p_2)$	$dod_2(p_2 \rightarrow m)$	$dod_2(m, p_2)$
$m_{_{I}}$	3	5	5	<i>m</i> <sub>1</sub>	2	4	4
$m_{2}$	0	3	3	<i>m</i> <sub>2</sub>	1	4	4
$m_{_3}$	2	5	5	<i>m</i> <sub>3</sub>	2	5	5
$m_{_4}$	3	5	5	$m_{_{4}}$	3	5	5
$m_{_5}$	0	1	1	$m_{_{5}}$	2	3	3

(a) Points  $p_1$ ,  $p_2$  and All Medoids in M

(b) directed dod<sub>e</sub> and dod<sub>e</sub>

Figure 5: *directed*  $dod_{\epsilon}$  as the *member* assignment condition

Figure. 5 is an example of member assignment. In this example, the dataset dimensionality D is 5, the value range for all dataset is [0,10], the sample medoid set M has five medoids, and  $\epsilon$  is 2. When we assign  $p_1$  from S, there are two medoids  $m_2$  and  $m_5$  which satisfy Eq. 4 which means *directed dod*<sub> $\epsilon$ </sub> is 0. Then the point  $p_1$  is finally assigned to  $m_5$  between  $m_2$  and  $m_5$ , because  $dod_2(m_5, p)$  is smaller than

<sup>&</sup>lt;sup>6</sup>We refer to the selected correlated dimension as key dimension and use the notation  $KD_m$  to indicate the set of m's key dimensions.

 $dod_2(m_2, p)$ . In the case of  $p_2$ , because there are no medoid satisfying member assignment condition, the point  $p_2$  is regarded as an outlier. Note that, if the member assignment condition is not satisfied between a medoid m and a point p, p is not assigned to m no matter how small  $dod_{\epsilon}(m, p)$  is.

We refer to the point assigned to m as *member* of m. By the above assignment policy, medoids with many *key dimensions* have more chances to obtain many members. Moreover, the strict condition  $dod_{\epsilon}(m \rightarrow p) = 0$  forces the medoids with irrelevant key dimensions, which is chosen as false positive, to have few points.

Step 2.2. Medoid Clustering : After Step 2.1, there are many medoids which have their own key dimensions and members. We group similar medoids together until their dod distances are smaller than the threshold  $D_{mindist}$ , which is given as the user parameter. We refer to this group of medoids as medoid clusters, to distinguish them from the clusters that would be presented to the user as the final result.(A medoid cluster can be thought to be a wireframe of the original cluster, which reflects the corresponding original cluster's correlated dimensions and the location in the subspace made by those dimensions.)

To cluster medoids, we use the groupwise average clustering method(gac) which is a well-known hierarchical clustering algorithm[2]. It starts clustering by regarding all medoids as different medoid clusters, and repeatedly merges the closest pair into one until all medoid clusters are different from each other more than  $D_{mindist}$ . To use gac, we first make a distance matrix for every medoids, and the distance is measured by  $dod_{\epsilon}$ . For the distance between two medoids  $m_i$  and  $m_j$ , which are meaningful in the subspace made of  $KD_{m_i}$  and  $KD_{m_i}$  respectively, dod measure for them is rewritten as follows:

$$dod_{\epsilon}(m_{i}, m_{j}) = max\{|D_{m_{i}}|, |D_{m_{j}}|\} - |\{ d \mid |m_{i}(d) - m_{j}(d)| \le \epsilon, d \in D_{m_{i}} \cap D_{m_{j}}\}|,$$
(5)

where  $m_i(d)$  is the *d*-dimensional value of  $m_i$  and  $m_j(d)$  is the *d*-dimensional value of  $m_j$ . To improve the effectiveness of *gac*, we make an exception to Eq. 5. That is, if  $|\{ d \mid |m_i(d) - m_j(d)| \le \epsilon, d \in D_{m_i} \cap D_{m_j} \}| \le 2$ , the distance  $dod_{\epsilon}(m_i, m_j)$  is set to *D* as a penalty. Note that *D* is the maximum distance in  $dod_{\epsilon}$  measure. This penalty is imposed on the pair of medoids which are far from each other so as to be overlapped in just one or no dimension. It improves hierarchical clustering's effectiveness by preventing the union of a pair of medoids which are initially very far from each other. Eventually, the distance between any two medoid clusters are computed as the weighted average between the medoids belonging to them.

$$dod_{\epsilon}(mc_A, mc_B) = \frac{\sum_{m_i \in mc_A, m_j \in mc_B} (|m_i| \cdot |m_j| \cdot dod(m_i, m_j))}{(\sum_{m_i \in mc_A} |m_i|) \cdot (\sum_{m_j \in mc_B} |m_j|)}$$
(6)

where  $|m_x|$  means the number of  $m_x$ 's members and  $mc_X$  means a medoid cluster X.

Gac clustering stops when the distance for every pair of medoid clusters is greater than  $D_{mindist}$ . The meaning of  $D_{mindist}$  is the highest limit of the amount of key dimension information that should be discarded for a medoid cluster to be merged. In Eq. 6, the weight of a medoid is the number of points assigned to it. Therefore, a medoid containing many members also has more impact in clustering process. Step 2.3. Medoid Cluster Tuning : Each resulting medoid clusters obtained in Step 2.2 has information inherited from the medoids in it. That is, the size of a medoid cluster |mc| is defined as the sum of its |m|s, and its key dimension set  $KD_{mc}$  is made from the key dimension sets  $(KD_m)$ s (for all  $m \in mc$ ). Since each medoid m in mc can have slightly different  $KD_m$  from each other, we make  $KD_{mc}$  by averaging the information in  $(KD_m)$ s. The average selection ratio for a dimension d is obtained as follows:

$$avg_d = \frac{\sum_{m_i} \delta_i \cdot |m_i|}{\sum_{m_i} |m_i|},\tag{7}$$

where  $m_i$  is a medoid in the given medoid cluster,  $|m_i|$  is the number of *members* assigned to  $m_i$ , and  $\delta_i$  is a binary function which is set to 1 when dimension d is a key dimension of  $m_i$  and is otherwise set to 0. If  $avg_d = 0.9$ , it means that 90% out of all *members* are assigned to mc regarding d as the key dimension. Therefore we take a dimension d as the key dimension of the medoid cluster when  $avg_l$  is large enough, so to speak, over 95%. In fact, any value between 90% - 100% brings almost the same result. Once  $KD_{mc}$  is obtained, all medoids in mc are meaningful only in the subspace made by  $KD_{mc}$ .

When  $KD_{mc}$  is determined to every resultant medoid clusters, a few medoid cluster become closer to each other than  $D_{mindist}$ . Because we want the final resultant clusters to be disjoint enough, those similar medoid clusters are merged( $KD_{mc}$  for each merged medoid cluster is regenerated also.) In addition, before going to the evaluation phase, some of the medoid clusters smaller than  $s_{ninsize}$  are removed because their corresponding original clusters would be smaller than  $G_{minsize}$ . For the given  $\epsilon$ , we use the notation  $MC_{\epsilon}$  to denote the resultant medoid cluster set generated as the result of cluster simulating step.

#### Step 3. Evaluation

Because the cluster forming phase is iterated 25 times variating epsilon from  $\frac{1}{100}$  valuerange to  $\frac{25}{100}$  valuerange by the degree of  $\frac{1}{100}$  valuerange, we should determine a medoid cluster set  $MC_{\epsilon}$  which has best property, i.e. the one that would possibly extract clusters' information best from |S| and |M|.

The soundness criteria  $MC_{\epsilon}$  is measured by the following equation:

$$Soundness(MC_{\epsilon}) = \sum_{mc \in MC_{\epsilon}} (|mc| \times |KD_{mc}|), \tag{8}$$

where |mc| is the size of medoid cluster mc and  $|KD_{mc}|$  is the number of key dimensions of mc.

Using this soundness criteria we represent the amount of the information contained in  $MC_{\epsilon}$  by medoid clusters' key dimensions and members(i.e.points assigned to them). That is, when there are two medoid clusters  $mc_a$  and  $mc_b$  representing an identical original cluster ( $mc_a \in MC_{\epsilon_1}, mc_b \in MC_{\epsilon_2}, \epsilon_1 \neq \epsilon_2$ ), we prefer the one which has more dimensional information if their sizes are equal. As a result of 25 evaluations, the best medoid cluster set  $MC_{bestepsilon}$  with the greatest soundness value is given to the data assigning phase.

```
procedure DataAssigning(MC_{bestepsilon})
// MC_{bestepsilon}: the medoid cluster set generated when \epsilon = bestepsilon
```

#### begin

```
\begin{array}{l} min\_dod := D; \ nearest\_mc := \ null; \\ \textbf{for each point $p$ in the given dataset do begin \\ \textbf{for each medoid cluster $mc$ in $MC_{bestepsilon}$ do begin \\ \textbf{for each medoid $m$ in $mc$ do begin \\ \textbf{if } dod_{bestepsilon}(m \rightarrow p) = 0$ and $dod_{bestepsilon}(m, p) < min\_dod$ then begin \\ min_dod := dod\_bestepsilon(m, p); \ nearest\_mc := \ mc; \\ \textbf{end} \\ \textbf{end} \\ \textbf{if } nearest\_mc <> null$ then assign $p$ into $nearest\_mc; \\ \textbf{else}$ assign $p$ into $Outlier\_Cluster; \\ \textbf{end} \\ \textbf{end} \\ \textbf{end} \end{array}
```

Figure 6: Data Assigning Subroutine

### **3.3 Data Assigning phase**

In this phase, all points in original dataset are assigned either to a medoid cluster in the  $MG_{bestepsilon}$  or to an outlier cluster. The principle of point assignment in this phase is exactly the same as that used in the member assignment step in the cluster forming phase. If a point is assigned to a medoid m, which belongs to a medoid cluster mc, it has the same meaning that the point is assigned to mc. The only difference here is that the key dimension set  $KD_{mc}$  of a medoid cluster is used as the key dimension set of each medoid  $m(m \in mc)$  because each m in mc is meaningful in  $KD_{mc}$ . Because the *bestepsilon* is sometimes selected as large as  $\frac{25}{100}$  valuerange, we should think about the risk for a cluster to have outliers. However, when the correlated dimensions of a cluster is more than 3, the probability of the risk is very low even for the largest  $\epsilon$ . In addition, the relatively high risk in low dimension such as 2 and 3 cannot degrade the value of our approach because subspace clustering is for high dimensional clustering where the data space is composed of decades of dimensions.

# 4 **Experimentations**

We conducted a series of experiments designed to measure the performance of FINDIT in terms of accuracy, robustness, and scalability. For this purpose we generated various synthetic datasets based on the method described in [13], variating the range of parameters. All of the experiments were run on a Pentium-3 500MHz Linux machine with 1 GB Memory and 15 GB SCSI type disk drive.

# 4.1 Dataset Generation Method

We implemented the data generation method suggested by Aggarwal[13], which is an extension of the Zhang[7]'s method to subspace clustering. In Aggarwal's method, there are six different parameters which determine the properties of the generated datasets: the size of dataset N, the number of clusters K, the dimensionality of dataset D, the number of average correlated dimensions AD, the percentage of outliers PO, and the standard deviation value range  $[SR_{min},SR_{max}]$  which is related to the cluster point distribution in each cluster. The number of correlated dimensions for each cluster is obtained from Poisson distribution using AD as its mean value. The first cluster's correlated dimensions are selected randomly and the successive cluster takes half of its predecessor's correlated dimensions as its correlated dimensions and randomly choose other correlated dimensions. In each dataset,  $N \cdot (1 - PO)$  points are generated as cluster points and the remaining points are uniformly distributed in the whole data space. The cluster size of each cluster is obtained based exponential distribution. The assigned cluster points are distributed around the cluster's center point in correlated dimensions and uniformly distributed in non-correlated dimensions. For every correlated dimension, cluster points are distributed according to a normal distribution with the cluster's center point value as its mean and with a certain standard variation value from  $[SR_{min}, SR_{max}]$ .

If a standard deviation sr is selected for a certain dimension d, from the property of normal distribution, 95% of cluster points would be within  $2 \cdot sr$  distance from the cluster's center point in dimension d. Therefore, we approximate the cluster's d-dimensional diameter is  $4 \cdot sr$ .<sup>7</sup> Eventually, we refer to the value  $4 \cdot SR_{max}$  as the maximum cluster diameter of the dataset generated, utilizing  $SR_{max}$  as its maximum standard deviation. Note that  $SR_{max}$  corresponds to  $r \cdot s$  in [13]. A more detailed description about data generation method can be found in [13].

# 4.2 Generated Datasets

To study FINDIT's performance with a wide range of dataset characteristics we have generated different datasets, variating a set of parameters summarized in Table 1.

Parameter	Value(s)	Interpretation
N	100,000	Dataset Size
K	5, 10	Number of Generated Clusters
D	20, 30, 40, 50	Dimensionality of the Dataset
AD	$\frac{1}{3}D, \frac{2}{3}D, \frac{3}{3}D$	Average Number of Correlated Dimensions for Each Cluster
$[SR_{min}, SR_{max}]$	[2,4], [3,6], [4,8]	Standard Deviation Range for Cluster Point Distribution
$\overline{PO}$	10%, 20%, 30%, 40%, 50%	Percentage of the Outliers in the Dataset

Table 1: Prameter Settings for the Generated Datasets

A total of 360 different datasets were generated based on the parameter values in Table 1, and all datasets have K clusters with size larger than 5,000 which is  $\frac{1}{20}$  of each dataset size. We primarily explain FINDIT's performance using the results of datasets containing 5 clusters.(In what follows, the number of clusters is 5, if not specified differently.) We named all datasets after the parameter values used in dataset generation. First of all, we use  $S_1$ ,  $S_2$ , and  $S_3$  to indicate the cases when  $[SR_{min}, SR_{max}]$  are [2,4], [3,6], [4,8] respectively. For example, the dataset  $S_1 D_{20} A D_7 P O_{0.1}$  refers to the dataset generated when  $[SR_{min}, SR_{max}]=[2,4]$ , D = 20,  $AD = \frac{D}{3}$ , and PO = 10%. We grouped 60 datasets having the same standard deviation ranges and refer to them as a test suite. Therefore, suite  $S_1$  means the collection of 60 datasets which have names starting with  $S_1$ . In terms of cluster diameter, each dataset in  $S_1$ ,  $S_2$ , and  $S_3$  have maximum cluster diameters of 16, 24, 32 respectively. Note that the dataset represented as Case (2) in [13] corresponds to dataset  $S_1 D_{20} A D_4 P O_{0.05}$  in our notaion, which has low cluster diameter and a low percentage of outliers compared to our average datasets.

<sup>&</sup>lt;sup>7</sup>In a strict sense, we cannot say the diameter of a cluster but we use it for the explanation purpose.

# 4.3 Parameters and Default Setting for Algorithms

For the purpose of comparison, we implemented the code of PROCLUS[13]. We compared the performance of FINDIT to that of PROCLUS for robustness tests and scalability tests. Although the version of PROCLUS we used might not be an optimized version of PROCLUS, we can see the tendancy of its performance. We selected PROCLUS between previous subspace clustering algorithms from several reasons. First of all, grid-based methods such as CLIQUE are not suitable to compare the accuracy of disjoint clusters which is one of the virtues of FINDIT(it is experimentally shown that CLIQUE is not as good as PROCLUS for disjoint cluster generation[13]). ORCLUS, which is an extended version of PROCLUS focuses on arbitrary axis subspace clustering, which we do not consider in this paper. Finally, CLTree is also not suitable for the comparison study because it needs the user's intervention in its pruning steps so that the clustering results are dependent on the user.

Now we describe the settings for sampling parameters and user parameters for FINDIT and PRO-CLUS. For FINDIT, the user parameter  $S_{minsize}$  is set to 5,000, and  $D_{mindist}$  is set to  $\frac{D}{10}$  in all tests. Note that the user does not have to input correct minimum cluster size in the dataset; the users can use any number for  $S_{minsize}$  as they want for the final clusters' minimum size. There is certainly a tradeoff between clustering speed and the size of clusters reported when  $S_{minsize}$  is too small compared to the dataset size. However, without loss of generality, we can expect that  $S_{ninsize}$  increases according to dataset size N. We used sample S with size of 1037, and sample M with size of 224 for all cases based on the user parameter  $S_{minsize}$  and  $N(\delta = 0.01, s_{minsize} = 30)$ . Because  $s_{minsize}$  is selected as 30, the number of voters V is set to 20, which is a value between 10 and  $s_{minsize}$  (Performance results from different values for V are also presented in Sec. 5.)

For PROCLUS, it is well known that the performance of PROCLUS depends on user parameters. So for PROCLUS, we set the user parameters k and l as the optimal values for each dataset. That is, the user parameter k, which is the number of clusters, is set to K that used to generate the dataset, the other user parameter l, which is the number of average correlated dimensions of clusters, is set to the the number of average correlated dimensions of clusters, is set to the the number of average correlated dimensions of all generated clusters in the given dataset.

# 4.4 Performance Measures

We measured the performance of FINDIT using three different measures: Confusion Matrix,  $F_1$ -value, and  $DF_1$ -pair. The first measure Confusion matrix[13], is used to reveal the composition of results compared to original clusters, and Figure 7 shows the shape of this matrix. The meaning of element  $q_{,j}$ is the number of points which belong to original cluster j and are assigned to result cluster i.

	1	 j	 K	K+1
1				
i		$c_{i,j}$		
Ĺ				
L+1				

K: the number of original clusters.  $\Sigma_{j=1}^{L+1}c_{i,j}$ : original cluster *j*'s size. Column(K + 1): Outliers in the original dataset.

L : the number of result clusters.  $\Sigma_{i=1}^{K+1} c_{i,j} : \text{result cluster } i \text{'s size.}$ Row(L+1) : Outlier cluster in the result.

### (a) Confusion Matrix

trix (b) Rows and Columns of Confusion Matrix Figure 7: Confusion Matrix and its meaning

Because Confusion Matrix is not suitable to compare many results at once, we use  $F_1$ -value as the second measure.  $F_1$ -value is the harmonic mean of *precision* and *recall* which are two famous performance measures in Information Retrieval. For every K original clusters in the dataset, the best result cluster is determined so that  $c_{i,j}$  is the maximum value between  $c_{1,j}$  and  $c_{L,j}$ . Then *recall*, *precision* and  $F_1$ -value for cluster j are computed as the following:

$$precision_j = \frac{c_{i,j}}{\sum_{j=1}^{K+1} c_{i,j}} \qquad recall_j = \frac{c_{i,j}}{\sum_{l=1}^{L+1} c_{l,j}} \qquad F_1 \text{-}value_j = \frac{2 \cdot precision_j \cdot recall_j}{precision_j + recall_j}.$$
 (9)

The meaning of  $recall_j$  is the ratio of cluster j represented by its best matching cluster i, and the meaning of  $precision_j$  is the posession rate of cluster j within cluster i. The best  $F_1$ -value<sub>j</sub> is 1, and it can be obtained when both  $recall_j$  and  $precision_j$  are all 1. That is, when the matching result cluster i has all points of cluster j and has no point from any other original cluster. We use the averaged form of  $F_1$ -value in the form of  $\frac{1}{K} \sum_{j=1}^{K} (F_1$ -value<sub>j</sub>). In this averaged  $F_1$ -value, all original clusters have identical importance irrespective of their sizes.

We use Confusion Matrix and  $F_1$ -value to show how well the points are clustered compared to original clusters. However, since Confusion matrix and  $F_1$ -value have no dimension information, we cannot see how well the original clusters' correlations are discovered. Therefore, we introduce a new measure, named  $DF_1$ -pair, which can be thought to be a form of (precision, recall) translated according to the dimensional information's view. For every cluster j in the dataset, the third measure  $DF_1$ -pair is defined as follows:

$$DF_{1}\text{-}pair_{j} = (dprecision_{j}, drecall_{j}) = (1 - \frac{|KD_{i} - CD_{j}|}{|CD_{j}|}, 1 - \frac{|CD_{j} - KD_{i}|}{|CD_{j}|}),$$
 (10)

where *i* is the best matching result cluster for *j*,  $CD_j$  is the set of real correlated dimensions for original cluster *j*, and  $KD_i$  is the set of *key dimensions* of result cluster *i*.

As in  $F_1$ -value, we mainly use the averaged  $DF_1$ -pair to compare many results at once. The optimal  $DF_1$ -pair is (1,1), which means that all clusters' dimensioal information is correctly discovered by results clusters.

### 4.5 Quality of Result

For three test suites  $S_1$ ,  $S_2$ , and  $S_3$ , containing 60 datasets respectively, the overall performances measured by average  $F_1$ -value are 99.49%, 99.59% and 98.59%. To illustrate the accuracy of those results in detail, we picked up two clustering results on  $S_1D_{20}AD_7PO_{0.3}$  and  $S_3D_{20}AD_7PO_{0.3}$  because their quality was similar to the average. The  $F_1$ -value for Case 1 was 99.3% and for Case 2, it was 98.96%. In both cases the dimensionality D was set to 20, the average correlated dimensions of the dataset was set to 7, and the percentage of outliers was set to 30%, as we can guess from the dataset names. The only difference between the two cases is the maximal cluster diameter, which was set to 16( $SR_{max} = 4$ ) for Case 1 and 32( $SR_{max} = 8$ ) for Case 2.

Cluster	Dim. Size	n. Size Correlated Dimensions			
A	9	2 3 5 7 9 10 15 16 18	33626		
В	8	0 4 9 10 14 15 16 18	8558		
C	6	0 3 7 12 14 18	7293		
D	8	0 6 7 12 13 15 17 18	12807		
E	7	0 2 6 12 15 16 17	7714		
Out.			30002		

	A	В	C	D	E	Out.
1	33626	0	0	0	0	3
2	0	113	0	12807	0	12
3	0	8445	0		0	15
4	0	0	0	0	7714	100
5	0	0	7293	0	0	237
6	0	0	0	0	0	5822
Out.	0	0	0	0	0	23813

#### (a) Dataset Description

(b) Confusion Matrix Result

Table 2: (Case 1). Accuracy Result of dataset  $S_1 D_{20} A D_7 P O_{0.3}$ 

For the Case 1, the selected *bestepsilon* was 17, and 6 different clusters with additional outlier cluster were generated as a result of the dataset  $S_1D_{20}AD_7PO_{0.3}$ . Except for the 6th cluster, all five clusters were well matched to original clusters. Although FINDIT adopts a strict assignment policy(Eq. 4), no point from original clusters is regarded as an outlier. This result can be explained by the fact that

Cluster	Dim. Size	Correlated Dimensions	Size
A	6	3 10 11 13 17 19	6011
В	7	2 8 11 13 15 16 19	18132
C	9	0 2 9 11 13 14 15 16 19	5551
D	9	0 1 6 13 14 16 17 18 19	13656
E	5	0 1 6 14 18	26648
Out.			30002

	A	В	C	D	E	Out.
1	0	0	0	13576	0	5
2	0	0	5455	1	0	16
3	77	18015	0		36	36
4	5888	2	0	0	65	94
5	0	4	0	0	26523	290
Out.	46	111	96	24	0	29561

#### (a) Dataset Description

### (b) Confusion Matrix Result

Table 3: (Case 2). Accuracy Result of dataset  $S_3 D_{20} A D_7 P O_{0.3}$ 

 $DF_1$ -pair was (1,1) which means the effectiveness of dimension voting policy in finding correlated dimensions. In Confusion Matrix of Case 1, 113 points were assigned to Result Cluster 2 from Original Cluster *B*. This kind of assignment occurrs from a coincidental conjunction of two events : a certain medoid in Cluster 2 satisfied the member assignment condition, and *dod* distance between those medoids and 114 points was equal to the distance of Cluster 3. When there is more than one cluster which satisfy member assignment condition, a point is assigned to the cluster with more key dimensions. In this case, it is not necessarily thought to be wrong assignment because the correlated dimension sets of Cluster 2 and Cluster 3, which have the same sizes, were correctly found, and those 114 points really had correlation to Cluster 2 as well as Cluster 3.

Cluster 6, which is not matched to any original cluster, was made from outliers, and its key dimension set was {5, 15}. The user can use the information of low dimensional clusters or can just throw them from correlation which is too low. It is totally up to the user's decision. Since the kind of clusters made by chance have few key dimensions, they usually have a large *dod* distance to points in the dataset. Therefore, even if they can meet the member assignment condition easily, they do not prevent points from being assigned to the correct result clusters. FINDIT does not require the number of clusters as a parameter, however it generated almost as many as or slightly more clusters than the number of original clusters. Moreover, even when there were additional clusters, from the reason described above, the overall performance was not affected by those additional clusters.

For Case 2 in Table 3, the best epsilon selected was 24. Compared to Case 1, some points belonging to original clusters were regarded as outliers because a few exceptional points can be generated from the increased cluster diameter in Case 2: a point can have a value which is more than *bestepsilon* distance from the any medoid in the corresponding result cluster for one or two dimensions from the increased cluster diameter. However, compared to the whole cluster sizes, those exceptional points are

very small, and the total accuracy was also very high: as much as 98.96% of  $F_1$ -value and (100%, 100%) of  $DF_1$ -pair. The whole accuracy results on 60 datasets in Suite  $S_2(N=100,000, K=5)$  can be found in Appendix.

# 4.6 Robustness and Scalability

**Robustness to the Dimensionality** D and AD: As a robustness result with respect to dimensionality, we compared the performance of FINDIT and PROCLUS in Fig. 8. Each  $F_1$ -value plotted in Fig 8 is the result on the datasets  $S_2D_*AD_*PO_{0.1}$  where \* means the entire parameter range.



Figure 8: Robustness to the Dimensionality D and AD:  $F_1$ -value result

As we can see from the results, FINDIT's high accuracy is always stable and is almost not affected by the changes in D and AD. Both D and AD have changed the performance of PROCLUS. Especially for PROCLUS, the performance improves when the cluster dimensionality increases. In all cases, FINDIT has shown higher quality than PROCLUS. Moreover, 10 cases out of 12 had the best  $DF_1$ -pair of (100%,100%), and the remaining two had a  $DF_1$ -pair of (99.50%,100%) and (99.13%,100%) respectively. When a cluster's *dprecision* is less than 100%, i.e. one or two correlated dimensions are wrongly added, there is a risk of wrong decisions such that some cluster points are regarded as outliers. However, since there are usually many medoids in a medoid cluster, points are usually assigned to one of them when *dprecision* is not significantly low.

**Robustness to the Percentage of Outliers** PO: Fig. 9 shows the  $F_1$ -values for dataset  $S_2D_{20}AD_*PO_*$ , where \* means the entire parameter range.

While FINDIT maintains very high accuracy in spite of the increase of outliers, the performance of PROCLUS drastically decreases when the percentage of outliers increases. It is from the fact that in



Figure 9: Robustness to the Percentage of Outliers  $PO: F_1$ -value result

*k-means* style algorithms such as PROCLUS, incorrect information from outliers is inevitable from its generic properties. *k-means* algorithms believe that the moving center is the natural way to find existing cluster centers, because the center is moving based on newly assigned points, it is hard to determine whether a point is an outlier in its iteration steps. As a result, when the number of outliers becomes large, the moving center would not converge to a real cluster center. However, FINDIT does not use all points' information to determine the correlated dimensions; it uses only the guaranteed nearest neighbors' information, and since an outlier point hardly becomes a nearest neighbor in *dod* measure, the correct dimension selection process is not affected by noisy condition.

**Robustness to the Number of Clusters and the Maximum Cluster Diameters :** Now we descibe the performance of FINDIT on different numbers of clusters and different cluster diameters. In Table 4, each has the average performance value of 60 datasets of  $D_*AD_*PO_*$  (the \* means the entire parameter range.) with corresponding row options and column options.

We observe that the quality of results are very robust to the number of clusters or to the size of cluster diameters. When the maximum cluster diameter is 32(note that in this case, the corresponding minimum cluster diameter is as large as 16.), the performance of quality slightly decreases. But considering the large cluster diameter of 32, this is a rather impressive result.

Time Scalability to the dimensionality and dataset size: To test the time scalability to the dimensionality, we plotted the execution time of FINDIT and PROCLUS for the datasets  $S_1 D_* A D_* P O_{0.1}$  in Figure 10(a). In the graph, we observe that FINDIT's execution time increases linearly in a low rate according to the increase of the dataset dimensionality D, and the cluster dimensionality AD, while the

	max cluster diameter									
	16(S	1)	24 (S	2)	32 ( <i>S</i> 3)					
No. of clusters	precision	recall	precision	recall	precision	recall				
K = 5	99.88	99.26	99.76	99.44	99.23	98.15				
K = 10	99.83	99.66	99.38	99.22	97.88	97.81				

(a) Avg(precision) and Avg(recall) of each 60 datasets

	max cluster diameter								
	16(S1) 24 (S2)			32 (S3)					
No. of clusters	d precision	drecall	d precision	drecall	d precision	drecall			
K = 5	99.61	99.71	99.74	99.63	99.76	99.39			
K = 10	99.63	99.21	99.34	99.72	99.23	99.19			

(b) Avg(dprecision) and Avg(drecall) of each 60 datasets

Table 4: (FINDIT) Quality Results on different number of clusters and different maximum cluster diameters

execution time of PROCLUS is largely dependent on D and AD. Although the efficiency of the two methods are similar with small D and AD, the difference between them increases up to several times according to D and AD.



Figure 10: Time Scalability to Dimensionality and Dataset Size

To test the time scalability to dataset size, we created a series of datasets with different dataset sizes based on  $S_1D_{30}AD10_PO_{0.1}$  (K is set to 5.). Each generated dataset has a size between 100,000 -5,000,000. The results of the generated 6 datasets with different sizes are plotted in Figure 10(b). When the dataset size is as small as 100,000, the performance of PROCLUS is better than FINDIT. However, FINDIT outperforms PROCLUS by more than 5 times when dataset size is as large as 5,000,000, even though we had chosen a small number for the iteration threshold of PROCLUS for the scalability tests.(note that the  $S_{minsize}$  of FINDIT was set to  $\frac{N}{20}$  for all datasets.) With FINDIT, an average of 15 seconds were consumed in cluster forming phase to find the best medoid cluster set  $MC_{bestepsilon}$ . The time consumed for cluster forming phase is almost same in all results because the cluster forming phase worked with the same size of S and M for all datasets (note that minimum cluster size was fixed to  $\frac{N}{20}$  for all datasets). In one iteration of cluster forming phase, the medoid clustering step, which uses *gac* clustering algorithm took most of time. We expect the adoption of a more optimized implementation of *gac* clustering can make FINDIT even faster, although the current version of FINDIT is fast enough compared to previous methods(Grid-based clustering algorithms and CLTree are even slower than PROCLUS.) The time consumed for the nearest neighbor search in dimension voting and the nearest medoid search in data assigning was not so much because |M| is so small that using sequential search is enough. FINDIT's efficiency is due to fast dimension voting process based on sampling. Because FINDIT's dimension selection requires no iteration and no dimension partitioning, it does not suffer significantly from the increased dimensionality or dataset size.

# **5** Discussion



Figure 11: The relationship between Epsilon, Soundness, and Quality of Results

**Epsilon and the Soundness Criteria :** To select the *bestepsilon* is very important in our method since the running parameter  $\epsilon$  plays a key role in shaping clusters. For each dataset  $S_1 D_{20} A D_7 P O_{0.4}$  and  $S_3 D_{20} A D_7 P O_{0.4}$ , we obtained 25 different medoid cluster sets variating  $\epsilon$  from 1 to 25(note that all dimension have the normalized value range of [1,100]), and assigned data to each medoid cluster set. Figure 11(a) and (b) show the variation of  $F_1$ -value and Soundness according to  $\epsilon$  values.

There are two things that are worth to be mentioned. For the first, we can see that the performance  $F_1$ -value is highly correlated with the soundness value in Figure 11(a) and (b). In most cases, our

soundness criteria selected the  $\epsilon$  that gives the best performance between 25 different  $\epsilon$  values. This simple but effective soundness criteria makes FINDIT intelligent enough to automatically select the  $\epsilon$  values which guarantee a good performance. We expect this soundness criteria can be useful to other subspace clustering which try to generate disjoint clusters.

For the second, the  $F_1$ -value saturates near to optimal(100%) when  $2 \times \epsilon$  becomes larger than the maximum cluster diameter of each dataset. In Figure 11(a), the  $F_1$ -value and Soundness value start to saturate when  $\epsilon$  value becomes 8, and in the case of Figure 11(b), they start to saturate when  $\epsilon$  becomes 16.(note that the maximum cluster diameter is 16 for  $S_1D_{20}AD_7PO_{0.4}$ , and 32 for  $S_3D_{20}AD_7PO_{0.4}$ ) It is from the fact that when  $2 \times \epsilon$  is relatively small compared to the maximum cluster diameter, some points can have a few dimensional values which do not lie within  $\epsilon$  distance from the medoids, and eventually are not assigned to the appropriate medoid cluster owing to the strict member assignment policy. But after the saturation point, the accuracy remains stable even for relatively large  $\epsilon$  values.

The number of voters V: We run experiments on test suites  $S_1$ ,  $S_2$ , and  $S_3$  with various V as 10, 15, and 20 to see the differences in performance with varying number of voters V; the results are shown in Table 5. When the maximum cluster diameter increased, the cases which used larger V values usually showed higher performance. However, the difference is not so significant, and the quality of clustering is stable for various V between 10 and  $s_{minsize}$  (note that it is set to 30 in our settings).

		max cluster diameter										
	16(.	$16(SR_{max} = 4)    24 (SR_{max} = 6)    32 (SR_{max} = 8))$			$24 (SR_{max} = 6)$			8))				
voters	precision	recall	$F_1$ -value	precision	recall	$F_1$ -value	precision	recall	$F_1$ -value			
V = 10	99.83	99.39	99.54	96.28	97.59	96.70	97.10	97.45	97.10			
V = 15	99.57	99.30	99.38	98.90	98.25	98.49	96.90	96.25	96.35			
V = 20	99.88	99.26	99.49	99.76	99.44	99.59	99.23	98.15	98.58			

Table 5: Quality results for different number of voters

**Parameters :** In the clustering problem, there is always a tradeoff between controllability and ease of application. If a clustering algorithm accepts more parameters from the user, the result could be more accurate, i.e., more similar to what the user expected. On the other hand, if the algorithm requires few parameters, it is easy to apply, but the result can be somewhat different from the expectation. The main reason for this is that the user parameters define the meaning of the similarity regardless of their different forms. Therefore, a good clustering algorithm should not only find out the parameters that need little expert knowledge but also are effective enough to deliver the user's definition about similarity simultaneouly. In this point of view, we can say that FINDIT's parameters are well balanced between

controllability and ease of application, because the two control parameters,  $C_{minsize}$  and  $D_{mindist}$ , describes the final shape of clusters which are familiar to the user.

# 6 Conclusions

In this paper, we have suggested a new subspace clustering algorithm named FINDIT. We have experimentally shown that the proposed algorithm significantly improves the quality of clustering, the robustness to a large scale of noise and cluster diameters, and the time scalability to dataset size and dimensionality. It generates disjoint clusters accurately with their subspace information, and this high accuracy does not degrade at all, even when 50% of the dataset are outliers. The accuracy and the robustness achieved by FINDIT can be explained as the result of its novel correlated dimension finding approach which is based on *dimension-oriented distance* measure and *dimension voting* policy. Moreover, FINDIT can be used in full-space clustering as well as subspace clustering. This scalability, in conjunction with its robustness and easy parameters to control, makes FINDIT a viable solution to the general high dimensional clustering problem which is a new challenge in data mining area.

# References

- [1] J. S. Vitter. Random sampling with a reservoir. *In ACM Transactions on Mathematical Software*, March 1985.
- [2] A.K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice Hall, 1988
- [3] L. Kaufman, and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley & Sons, 1990.
- [4] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. *In the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Denmark, 1992.
- [5] R. Ng, J. Han. Efficient and effective clustering methods for spatial data mining. *In Proceedings of the 20th VLDB Conference*, Santiago de Chile, Chile, 1994.
- [6] M. Ester, H.-P. Kriegel, J. Sander and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. *In Proceedings of the 1996 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Portland, OR, USA, 1996
- [7] T. Zhang, R. Ramakrishnan, and Livny M. BIRCH: an efficient data clustering method for large databases. In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Qubec Canada, 1996.

- [8] Wei Wang, Jiong Yang, Richard Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. *In Proceedings of the 23th VLDB Conference*, Athens, Greece, 1997.
- [9] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster. A Multi-Resolution Clustering Approach for Very Large Spatial Databases. *In Proceedings of the 24th VLDB conference*, New York City, 1998.
- [10] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA USA, 1998.
- [11] N. Joze-Khajavi, K. Salem. Two-phase clustering of large datasets. *Technical Report CS-98-27*, Department of Computer Science, University of Waterloo, November 1998.
- [12] S. Guha, R. Rastogi, and K. Shim. ROCK: a robust clustering algorithm for categorical attributes. *In Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia, 1999.
- [13] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, and J. S. Park. Fast algorithms for Projected Clustering. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadelphia, PA USA, 1999.
- [14] Chun-hung Cheng, Ada Wai-chee Fu, and Yi Zhang. Entropy-based Subspace Clustering for Mining Numerical Data. In Proceedings of the 1999 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, USA, 1999.
- [15] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In Proceedings of ACM SIGMOD International Conference on Management of Data, Philadelphia, PA, 1999.
- [16] G. Karypis, E.H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer 32, pp.* 68-75, August 1999.
- [17] K. Beyer. When is nearest neighbor meaningful?. In Proceedings of the 7th International Conference on Database Theory. Jerusalem, Israel, 1999
- [18] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets *Technical Report CPDC-TR-9906-010*, *Dept. of Electrical and Computer Engineering, Northwestern University*, November 1998.
- [19] Charu C. Aggarwal, Philip S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. In proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas USA, 2000
- [20] Bing Liu, Yiyuan Xia, Philip S. Yu. Clustering Through Decision Tree Construction. In proceedings of the Ninth International Conference on Information Knowledge Management CIKM, McLean, VA USA, 2000

- [21] Christopher R. Palmer, Christos Faloutsos. Density Biased Sampling: An Improved Method for Data Mining and Clustering. In proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas USA, 2000.
- [22] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. *In Information Systems Vol. 26, No. 1*, 2001.

# 7 Appendix

Dataset Name	Precision	Recall	$F_1$ -value	Dataset Name	Precision	Recall	$F_1$ -value
$D_{20}AD_7O_{0.1}$	99.62	99.92	99.77	$D_{40}AD_{13}O_{0.1}$	99.98	99.95	99.97
$D_{20}AD_7O_{0.2}$	99.92	99.29	99.6	$D_{40}AD_{13}O_{0.2}$	99.75	99.96	99.86
$D_{20}AD_7O_{0.3}$	99.34	99.17	99.25	$D_{40}AD_{13}O_{0.3}$	99.98	99.96	99.97
$D_{20}AD_7O_{0.4}$	96.67	99.03	97.77	$D_{40}AD_{13}O_{0.4}$	99.78	94.24	96.9
$D_{20}AD_7O_{0.5}$	99.23	96.73	97.94	$D_{40}AD_{13}O_{0.5}$	99.98	99.6	99.79
$D_{20}AD_{13}O_{0.1}$	99.98	99.84	99.91	$D_{40}AD_{27}O_{0.1}$	100	99.95	99.98
$D_{20}AD_{13}O_{0.2}$	100	99.98	99.99	$D_{40}AD_{27}O_{0.2}$	100	99.98	99.99
$D_{20}AD_{13}O_{0.3}$	99.92	99.99	99.96	$D_{40}AD_{27}O_{0.3}$	100	99.92	99.96
$D_{20}AD_{13}O_{0.4}$	100	99.3	99.64	$D_{40}AD_{27}O_{0.4}$	100	100	100
$D_{20}AD_{13}O_{0.5}$	95.51	97.92	96.57	$D_{40}AD_{27}O_{0.5}$	100	99.97	99.99
$D_{20}AD_{20}O_{0.1}$	99.99	99.68	99.83	$D_{40}AD_{40}O_{0.1}$	100	99.84	99.92
$D_{20}AD_{20}O_{0.2}$	100	98.55	99.26	$D_{40}AD_{40}O_{0.2}$	100	98.95	99.47
$D_{20}AD_{20}O_{0.3}$	100	98.31	99.12	$D_{40}AD_{40}O_{0.3}$	100	99.2	99.6
$D_{20}AD_{20}O_{0.4}$	99.99	99.81	99.9	$D_{40}AD_{40}O_{0.4}$	100	100	100
$D_{20}AD_{20}O_{0.5}$	100	99.89	99.95	$D_{40}AD_{40}O_{0.5}$	100	100	100
$D_{30}AD_{10}O_{0.1}$	100	99.65	99.82	$D_{50}AD_{17}O_{0.1}$	100	99.94	99.97
$D_{30}AD_{10}O_{0.2}$	99.95	99.84	99.89	$D_{50}AD_{17}O_{0.2}$	100	99.82	99.91
$D_{30}AD_{10}O_{0.3}$	99.14	95.69	97.34	$D_{50}AD_{17}O_{0.3}$	99.97	99.94	99.96
$D_{30}AD_{10}O_{0.4}$	99.63	99.91	99.77	$D_{50}AD_{17}O_{0.4}$	100	100	100
$D_{30}AD_{10}O_{0.5}$	99.83	99.73	99.78	$D_{50}AD_{17}O_{0.5}$	99.92	100	99.96
$D_{30}AD_{20}O_{0.1}$	100	99.22	99.6	$D_{50}AD_{33}O_{0.1}$	100	99.77	99.88
$D_{30}AD_{20}O_{0.2}$	100	98.57	99.28	$D_{50}AD_{33}O_{0.2}$	100	99.63	99.81
$D_{30}AD_{20}O_{0.3}$	100	99.93	99.97	$D_{50}AD_{33}O_{0.3}$	100	99.99	100
$D_{30}AD_{20}O_{0.4}$	100	100	100	$D_{50}AD_{33}O_{0.4}$	100	100	100
$D_{30}AD_{20}O_{0.5}$	100	100	100	$D_{50}AD_{33}O_{0.5}$	100	99.86	99.93
$D_{30}AD_{30}O_{0.1}$	99.98	99.9	99.94	$D_{50}AD_{50}O_{0.1}$	100	99.7	99.85
$D_{30}AD_{30}O_{0.2}$	100	97.93	98.92	$D_{50}AD_{50}O_{0.2}$	100	99.74	99.87
$D_{30}AD_{30}O_{0.3}$	97.56	99.22	98.3	$D_{50}AD_{50}O_{0.3}$	100	99.95	99.97
$D_{30}AD_{30}O_{0.4}$	100	99.98	99.99	$D_{50}AD_{50}O_{0.4}$	100	100	100
$D_{30}AD_{30}O_{0.5}$	100	99.59	99.79	$D_{50}AD_{50}O_{0.5}$	100	99.89	99.94

Table 6: Entire Quality Results on 60 datasets in Suite  $S_2$  (N = 100,000, K = 5)