Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information and Software Technology 46 (2004) 255–271

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting

Kyoung-Gu Woo*, Jeong-Hoon Lee, Myoung-Ho Kim, Yoon-Joon Lee

Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Kusong-Dong 373-1, Yusung-Gu Taejon 305-701, South Korea

Received 14 October 2002; revised 10 May 2003; accepted 24 July 2003

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 important 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 requirements for good clustering methods such as accuracy of results, robustness to the noise and the cluster density, and scalability to the dataset size and the dimensionality. Moreover, it is gracefully scalable to full dimension without any modification to algorithm.

Keywords: Data mining; Data clustering; Subspace clustering; High dimension

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 [3–9,12,15,16,22]. 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 performing 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. However, 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

* Corresponding author.
E-mail addresses: kgwoo@dbserver.kaist.ac.kr (K.-G. Woo), leejh@dbserver.kaist.ac.kr (J.-H. Lee), mhkim@dbserver.kaist.ac.kr (M.-H. Kim), yjlee@dbserver.kaist.ac.kr (Y.-J. Lee).

0950-5849/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2003.07.003

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 Ref. [10], several different subspace clustering methods have been presented and had some success [13,14,18–20]. Recent subspace clustering algorithms can be broadly classified into two categories: the grid-based approach [10,14,18] and the partitioning approach [13,19]. The grid-based approaches mainly focus 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 increase 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 also presents one measure named *entropy measure* that satisfies those three 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 the 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 α and β (note that the initial number of seeds is much bigger than the number of clusters k). The algorithm stops when the remaining number of seeds is 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 the 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 is very important because the noise (i.e. outlier) would probably increase according to the dimensionality of the dataset, and clustering in the high dimension is what subspace clustering is proposed for.

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 explanation of our algorithm is presented in Section 3. Section 4 explains about the datasets used in the experiments and the performance results. In Section 5, we discuss about the various properties of FINDIT. Section 6 contains the conclusion of our research work.

2. Preliminaries

2.1. Motivation

2.1.1. Dimension-oriented distance

Dimension-oriented distance (*dod*) is our unique distance measure which utilizes dimensional difference information

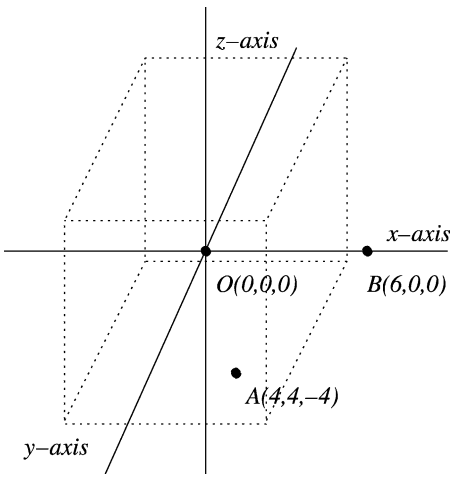


Fig. 1. Conventional distance vs. dimension-oriented distance when $\epsilon = 5$.

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 ϵ because we think they are 'near enough' on that dimension. That is, if the Manhattan distance between two points is less than ϵ 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,[1,2] and let $p(d)$ be a value of p in the dth-dimension. We define directed *dod* from a point p to a point q as

$$dod_\epsilon(p \rightarrow q) = |D_p| - |\{d | |p(d) - q(d)| \leq \epsilon, d \in D_p \cap D_q\}|,$$

and we define $dod(p,q)$ as the larger value out of two directed *dod* values between p and q. That is

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

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

[1] We use the notation D to denote the whole dimensional space of the given dataset.

[2] '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.

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

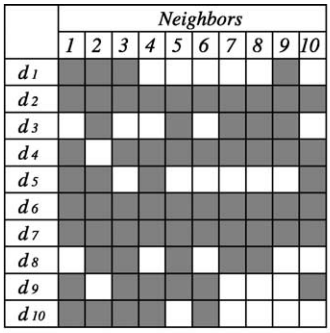
In this example, a dataset composed of 1000 points in a 10-dimensional space is presented in Fig. 2(a). There are five different clusters that have their own correlated dimensions, and for each correlated dimension of a cluster, all points in the cluster have values generated according to the normal distribution with a certain mean value and average variance of nine. For the non-correlated dimensions and outliers, the dimensional values are randomly selected from the whole value range. (The value ranges for all dimensions are set to [0,100].) Finally, 30% of dataset are generated as outliers (or noise).

Two matrices in Fig. 2(b) and (c) show the 10 nearest neighbors of points p and q, respectively, where ϵ is given as 15. In Fig. 2(b) and (c), a grey cell means that difference between the given point and the corresponding neighbor point is smaller than or equal to the given ($\epsilon = 15$) value for the dimension identified by the row number. For each dimension in Fig. 2(b), if we pick up the dimensions which have more than seven grey cells, the set of all selected dimension becomes {2,4,6,7} which is same as the correlated dimensions of Cluster E in Fig. 2(a). We can obtain the dimension set {1,3,5,6,8,9,10} for the case of Fig. 2(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).

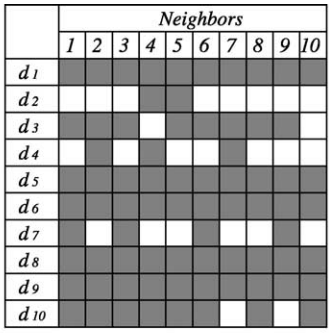
This example shows the possibility that we can estimate a original cluster's correlated dimensions by using neighbors' information of a point which belong to the original cluster, although some problems remain such as the decision threshold. To judge the correlatedness for each dimension, we developed 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 a certain dimension d is a correlated dimension of the original cluster where the voters belong. To use dimension voting, we should find out the appropriate number of voters, the threshold for judging a dimension's correlatedness, and the proper value of ϵ. The method to determine these values is written in Section 3.

Cluster	Size	Correlated Dimension
A	43	3, 5, 6, 8, 10
B	160	1, 3, 5, 6, 8, 9, 10
C	129	9, 10
D	31	2, 3, 6, 8, 9, 10
E	337	2, 4, 6, 7
Outlier	300	

(a) Sample DataSet



(b) Point p's 10 nearest neighbors



(c) Point q's 10 nearest neighbors

Fig. 2. Sample dataset and 10 nearest neighbors of points p and q.

2.2. Algorithm overview

2.2.1. User parameters and result

The inputs for FINDIT are 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 meanings here. The first parameter $C_{minsize}$ reflects the user's wish about the minimum size[3] of clusters, since the clusters containing relatively a small number of points compared to the dataset size do not interest the user in most cases. This minimum cluster size of interest usually increases according to the dataset size, and FINDIT does not report the clusters smaller than $C_{minsize}$. This size constraint, which is very natural to the user but has been neglected by previous methods, becomes a useful information for FINDIT. The second user parameter $D_{mindist}$ is the merge threshold between two resultant clusters. If the *dod* distance between two clusters is smaller than $D_{mindist}$ for the selected ϵ, 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. In the context of *dod* distance measure, $D_{mindist}$ can be thought to be the number of correlated dimensions that a cluster should neglect to be merged with another cluster.

2.2.2. Brief summary of the clustering process

FINDIT is composed of three phases: sampling phase, cluster forming phase and data assigning phase (Fig. 3). 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.

[3] 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.

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 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 ϵ, an evaluation criteria is used to choose the best ϵ and the corresponding medoid cluster set MC_ϵ. 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.

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 referring 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 Ref. [22]. According to *Chernoff bounds*, the minimum size of a sample S, which assures that every cluster in S should have more than ξ points in the sample by the probability of $1-\delta$, is computed by the following equation:

Chernoff-bounds(S)

$$= \xi k\rho + k\rho\log\left(\frac{1}{\delta}\right) + k\rho\sqrt{\left(\log\left(\frac{1}{\delta}\right)\right)^2 + 2\xi\log\left(\frac{1}{\delta}\right)}. \quad (2)$$

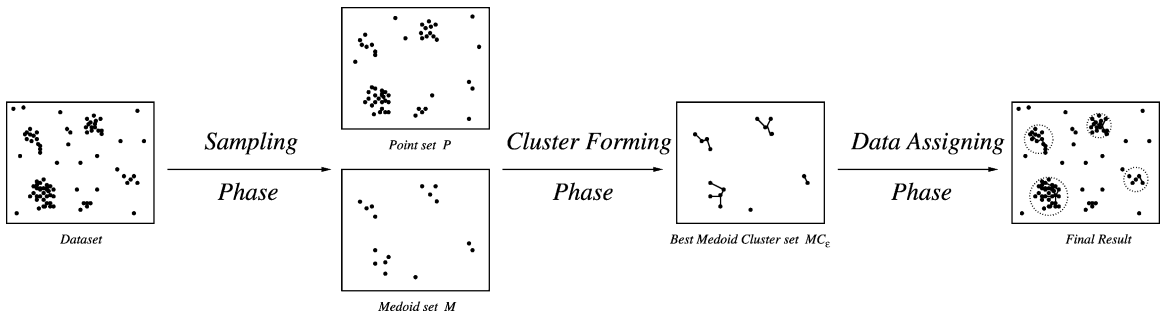


Fig. 3. Overall framework of FINDIT.

Constant k is the number of clusters, and ρ is a value satisfying $minsize = N/k\rho$ ($\rho \geq 1$), 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 a sample S, typically we use the setting $\xi = 30$, $\delta = 0.01$, $minsize = C_{minsize}$, $k = N/C_{minsize}$ and $\rho = 1$ (ρ is computed following the equation $\rho = N/k \times C_{minsize}$). Constant k is set to $N/C_{minsize}$ from the fact that there can be maximally $N/C_{minsize}$ 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 5000, sample size $|S|$ satisfying Eq. (2) is approximated to 1037. Especially, since constant ξ, 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 ξ. Hence $s_{minsize}$ is 30 in the above example. The *Chernoff bounds* for a medoid sample M is obtained within the same setting used to estimate $|S|$, except for ξ. In this case, ξ 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 samples 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 the dataset size is N and the required random sample size is $|S|$, the complexity of sampling is known to be $\Omega(|S| + |S| \log N/|S|)$ [1]. The sampling process needs just one sequential scan of the dataset because M and S are generated simultaneously. More detailed information about sampling can be found in Refs. [1,11,21,22].

3.2. Cluster forming phase

Cluster forming phase is iterated several times with an increasing ϵ value. Throughout the whole process of FINDIT, except sampling phase, ϵ is a key running parameter which determines *dod* distance. If ϵ 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 the appropriate ϵ, we try 25 different values in $[(1/100)valuerange, (25/100)valuerange]$.[4] In this paper, we assume that all dimensions' *valueranges* are already normalized for clustering. At each end of iteration, 25 medoid cluster sets generated by corresponding 25 different ϵ values are evaluated by our soundness criteria. Finally, we determine the best medoid cluster set and use it in the following data assignment phase (Fig. 4).

3.2.1. Step 1. Dimension voting

The purpose of this step is to determine the correlated dimensions for all points (= medoids[5]) 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

$$dod_\epsilon(m,p) = |D| - |\{d | |m(d) - p(d)| \leq \epsilon, d \in D\}|, \qquad (3)$$

where $p(d)$ is the dth-dimensional value of a point p. 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., i.e. no correlated dimensions are selected for both points yet.)

As mentioned in Section 2, we regard this V nearest neighbors as voters, and dimension selection as a voting process for D independent questions. For the question 'Is Dimension d a *correlated dimension* of the cluster to which m belongs?', if a neighbor's dth-dimensional value from m is less than or equal to the given ϵ, it is considered that the neighbor votes for 'Yes'. In this paradigm, the question is transformed into two new ones: 'What is the threshold

[4] *valuerange* means the width between the minimum value and the maximum value of the given dimension's value domain.

[5] From now on, we call a point in M a medoid to distinguish it from other points.

Step 1. Dimension Voting
The purpose of this step is to determine the correlated dimensions for all

procedure **ClusterForming**($s_{minsize}$, $D_{mindist}$)
// $s_{minsize}$: size threshold for the smallest cluster in S
// $D_{mindist}$: merge threshold for any two medoid clusters
S: set of sampled points
M: set of sampled medoids
E: set of 25 epsilons = $\{\frac{1}{100}valuerange, \frac{2}{100}valuerange, \cdots, \frac{25}{100}valuerange\}$
MC_ϵ: set of medoid clusters for the given ϵ
begin
  $bestepsilon := 0$; $MC_{bestepsilon} := 0$;
  for each ϵ 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_ϵ;
      //Any two medoid clusters with closer than $D_{mindist}$ are merged.
    Step 2.3. Refine medoid clsuters in MC_ϵ;
      //medoid clusters which are too small are removed from MC_ϵ.
      //A few medoid clusters are additionally merged.
    Step 3 if Soundness(MC_ϵ) $\geq$ Soundness($MC_{bestepsilon}$)
      then $bestepsilon := \epsilon$;
  end
  return $MC_{bestepsilon}$
end

Fig. 4. Cluster forming subroutine.

for a correct decision?' and 'What is the proper number of voters?' We explain them in Properties 1 and 2.

Property 1. *The number of 'Yes' votes on any non-correlated dimension follows binomial probability distribution* $B(n : V, p)$ *where* $p = 2\epsilon/valuerange$, *provided that all voters belong 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 $|m(d) - v(d)| \leq \epsilon$, which means the 'Yes' vote by chance, becomes $2\epsilon/valuerange$. Because there are V voters and each voter's dth-dimensional value is independent of each other, this voting process follows binomial probability distribution $B(n : V, 2\epsilon/valuerange)$ where n is the random variable between 0 and V. Based on the above model, the threshold making the probability $P(X \geq T)$ 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 which 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 ϵ values. Moreover, when two points are near enough from each other in many dimensions, we cannot say they are irrelevant. Therefore, the nearer a point is to a medoid, the higher the probability the two are from the same cluster goes. 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 ϵ 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 discriminating power of binomial probability distribution is weakened when V is less than 10. When a dimension d

$p_1 = (5, 6, 6, 5, 8)$

$p_2 = (3, 2, 9, 9, 7)$

$D = 5 \quad |M| = 5 \quad \varepsilon = 2$

Medoid	KD_m	m's position
m_1	{1,3,4}	(1, 7, 3, 2, 8)
m_2	{2,3}	(3, 5, 7, 1, 4)
m_3	{3,5}	(2, 6, 1, 4, 3)
m_4	{1,4,5}	(9, 9, 8, 1, 2)
m_5	{2,3,4,5}	(7, 8, 5, 6, 9)

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

Medoid	$dod_2(m \to p_1)$	$dod_2(p_1 \to m)$	$dod_2(m \to p_1)$
m_1	3	5	5
m_2	0	3	3
m_3	2	5	5
m_4	3	5	5
m_5	0	1	1

Medoid	$dod_2(m \to p_2)$	$dod_2(p_2 \to m)$	$dod_2(m, p_2)$
m_1	2	4	4
m_2	1	4	4
m_3	2	5	5
m_4	3	5	5
m_5	2	3	3

(b) *directed* dod_ε and dod_ε

Fig. 5. *directed* dod_ϵ as the *member* assignment condition.

has obtained more 'Yes' votes than the referenced threshold, we regard that a certain correlation exists in d.[6] 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.

3.2.2. 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. Therefore, to verify whether there is really a cluster in the subspace of KD_m, points in S are assigned to medoids in M. We refer to the point assigned to m as *member* of m. For every point p in S and every medoid m in M, p is assigned to the nearest m satisfying the member assignment condition in Eq. (4)

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

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

Fig. 5 is an example of member assignment. In this example, the dataset dimensionality D is 5, valueranges for all dimensions are normalized to [0,10], the sample medoid set M has five medoids and ϵ is 2. When we assign p_1 from S, there are two medoids m_2 and m_5 which satisfy Eq. (4) (i.e. $dod_2(m_2 \to p_1) = 0$ and $dod_2(m_5 \to p_1) = 0$.) 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 $dod_2(m_2, p)$. 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 $dod_\epsilon(m, p)$ is small.

The strict condition $dod_\epsilon(m \to p) = 0$ forces the medoids with irrelevant *key dimensions* to have few members and with the assignment policy which prioritizes medoids with more *key dimensions* we can effectively give stress to medoids which have more dimensional information.

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 these groups 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 original corresponding cluster's correlated dimensions and the location within 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 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_ϵ. 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_j}, respectively, *dod* measure for them is rewritten as follows

$$dod_\epsilon(m_i, m_j) = \max\{|D_{m_i}|, |D_{m_j}|\} - |\{d \mid |m_i(d) - m_j(d)| \le \epsilon, d \in D_{m_i} \cap D_{m_j}\}|, \tag{5}$$

[6] 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*.

where $m_i(d)$ is the dth-dimensional value of m_i and $m_j(d)$ is the dth-dimensional value of m_j.

To improve the effectiveness of *gac*, we make an exception to Eq. (5). That is, if $|\{d||m_i(d) - m_j(d)| \leq \epsilon, d \in D_{m_i} \cap D_{m_j}\}| \leq 2$, the distance $dod_\epsilon(m_i, m_j)$ is set to D as a penalty (note that D is the maximum distance in dod_ϵ 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

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

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 cluster 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 the 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 ($KD_m x$)s. The average selection ratio for a dimension d is obtained as follows

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

where m_i is a medoid in the given medoid cluster, $|m_i|$ is the number of *members* assigned to m_i, and δ_i is a binary function which is set to 1 when the dimension d is a *key dimension* of m_i, and otherwise set to 0. If $avg_d = 0.9$, it means that 90% out of all members are assigned to this medoid cluster regarding d as the *key dimension*. Therefore, we take a dimension d as the *key dimension* of the medoid cluster when avg_d is large enough, so to speak, over than 95%. In fact, any value between 90 and 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 for every resultant medoid cluster, 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 $s_{minsize}$ are removed because their corresponding original clusters would be smaller than $C_{minsize}$. To denote the resultant medoid cluster set generated based on a given ϵ, we use the notation MC_ϵ.

3.2.3. Step 3. Evaluation

Because the cluster forming phase is iterated 25 times variating ϵ from (1/100)*valuerange* to (25/100)*valuerange* by the degree of (1/100)*valuerange*, we should determine a medoid cluster set MC_ϵ which has best property, i.e. the one that would possibly extract cluster's information best from $|S|$ and $|M|$.

The soundness criteria MC_ϵ 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 a medoid cluster mc and $|KD_{mc}|$ is the number of *key dimensions* of mc.

Using this soundness criteria we represent the amount of information contained in MC_ϵ by medoid clusters' *key dimensions* and members. 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 chosen for the data assigning phase.

3.3. Data assigning phase

In this phase, all points in the original dataset are assigned either to a medoid cluster in the $MC_{bestepsilon}$ 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 as the point is assigned to mc. The only difference here is that the *key dimension* set KD_{mc} of a medoid cluster is used instead of the *key dimension* set of each medoid $m(m \in mc)$ because each m in mc became meaningful in KD_{mc} (Fig. 6).

Because the *bestepsilon* is sometimes selected as large as (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 three, the probability of the risk becomes very low even for the largest ϵ. In addition, the relatively high risk in low dimension such as two and three 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.

```
procedure DataAssigning(MC_bestepsilon)
// MC_bestepsilon: the best medoid cluster set generated in the cluster simulating
phase.
begin
    min_dod := D; nearest_mc := null;
    for each point p in the given dataset do begin
        for each medoid cluster mc in MC_bestepsilon do begin
            for each medoid m in mc do begin
                if dod_bestepsilon(m → p) = 0 and dod_bestepsilon(m, p) < min_dod then
                begin
                    min_dod := dod_bestepsilon(m, p); nearest_mc := mc;
                end
            end
        end
        if nearest_mc <> null then assign p into nearest_mc;
        else assign p into Outlier_Cluster;
    end
end
```

Fig. 6. Data assigning subroutine.

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 Ref. [13], variating the range of parameters. All of the experiments were run on a Pentium-3 500 MHz 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 Zhang's [7] method into subspace clustering. In Aggarwal's method, there are six different parameters which determine the properties of the generated datasets: the size of the dataset N, the number of clusters K, the dimensionality of the 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 then chooses other correlated dimensions randomly. In each dataset, $N(1 - PO)$ 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 dimension d, from the property of normal distribution, 95% of cluster points would be within $2sr$ distance from the cluster's center point in dimension d. Therefore, we approximate that the cluster's dth-dimensional diameter is 4sr.[7] Eventually, we refer to the value $4SR_{max}$ as the *maximum cluster diameter* of the dataset generated, utilizing SR_{max} as its maximum standard deviation. Note that SR_{max} corresponds to rs in Ref. [13]. A more detailed description about data generation method can be found in Ref. [13].

4.2. Generated datasets

To analyze FINDIT's performance with a wide range of dataset characteristics we have generated different datasets, variating a set of parameters summarized in Fig. 7.

A total of 360 different datasets were generated based on the parameter values in Fig. 7, and all datasets have K clusters with sizes larger than 5000 which is 1/20 of each dataset size. We primarily explain FINDIT's performance using the test results on datasets containing five clusters. (In what follows, the number of clusters is five, 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 $S_1D_{20}AD_7PO_{0.1}$ refers to the dataset generated when $[SR_{min}, SR_{max}] = [2, 4]$, $D = 20$, $AD = D/3$ and $PO =$

[7] In a strict sense, the diameter of a cluster is hard to tell. But we use this term for the explanation purpose in this paper.

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
PO	10%, 20%, 30%, 40%, 50%	Percentage of the Outliers in the Dataset

Fig. 7. Parameter settings for the generated datasets.

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 Ref. [13] corresponds to dataset $S_1D_{20}AD_4PO_{0.05}$ in our notation, which has small cluster diameter and a low percentage of outliers compared to our average datasets.

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 tendency 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 PROCLUS. For FINDIT, the user parameter $S_{minsize}$ is set to 5000, and $D_{mindist}$ is set to $D/10$ 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 $S_{minsize}$ 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, we set the number of voters V as 20 which is a value between 10 and $s_{minsize}$ (Performance results from different values for V are also presented in Section 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 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 resultant clusters compared to original clusters. Fig. 8 shows the shape of this matrix. The meaning of an element $c_{i,j}$ is the number of points which belong to the original cluster j and are assigned to result the cluster i.

Because Confusion Matrix is not suitable for comparing many test 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 cluster 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 F_1-value for cluster j are computed as the following

$$F_1\text{-}value_j = \frac{2precision_j \times recall_j}{precision_j + recall_j}, \tag{9}$$

where $precision_j$ is defined as $c_{i,j}/\sum_{j=1}^{K+1} c_{i,j}$, and $recall_j$ is defined as $c_{i,j}/\sum_{l=1}^{L+1} c_{l,j}$.

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 possession rate of the cluster j within the cluster i. The best F_1-$value_j$ is 1, and it can be obtained when both $recall_j$ and $precision_j$ are all 1. It is when the matching result cluster i has all points of the cluster j and the cluster i has no point from any other original cluster than j. For each test, we make the averaged F_1-value which is $(1/K)\sum_{j=1}^{K}(F_1\text{-}value_j)$. (In

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

(a) Confusion Matrix

K : the number of original clusters

$\Sigma_{j=1}^{L+1} c_{i,j}$: the size of original cluster j

Column$(K+1)$: Outliers in the original dataset

L : the number of result clusters

$\Sigma_{i=1}^{K+1} c_{i,j}$: the size of result cluster i

Row$(L+1)$: Outlier cluster in the result

(b) Rows and Columns of Confusion Matrix

Fig. 8. Confusion matrix and its meaning.

this macro averaging, 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 in view of original clusters. However, neither of them can tell how well the dimension information of original clusters is discovered. Therefore, we introduce a new measure named DF_1-*pair* which can have the meaning of (*precision*, *recall*) in the context of dimensional information. For every cluster j in the dataset, the third measure DF_1-*pair* is defined as follows

$$DF_1\text{-}pair_j = (dprecision_j, drecall_j)$$

$$= \left(1 - \frac{|KD_i - CD_j|}{|CD_j|}, 1 - \frac{|CD_j - KD_i|}{|CD_j|}\right), \qquad (10)$$

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

As in F_1-value, we mainly use the averaged DF_1-*pair* to compare many test results at once. The optimal DF_1-*pair* is (1,1), which means that all clusters' dimensional information is correctly discovered by result 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 these 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 (Fig.9, Fig.10) The F_1-value for Case 1 in Fig.9 was 99.3% and for Case 2 in Fig.10, it was 98.96%. In both cases the dimensionality D was set to 20, the average number of correlated dimensions AD was set to 7, and the percentage of outliers PO 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.

For Case 1 (Fig. 9), *bestepsilon* was chosen as 17, and six different clusters with an additional outlier cluster were generated as the clustering result on the dataset $S_1D_{20}AD_7PO_{0.3}$. Except for the sixth 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 point. The good result as this can be explained by the fact that the averaged DF_1-*pair* of this test is (1,1) which means that the *key dimensions* found are exactly same as original correlated dimensions.

In Confusion Matrix of Case 1, 113 points were assigned to the result cluster 2 from the original cluster B while most points consisting the cluster 2 are from the original cluster D. This seemingly wrong assignment was found out to be from a coincidental conjunction of two events: a medoid in the cluster 2 satisfied the member assignment condition for

Cluster	Dim. Size	Correlated Dimensions	Size
A	9	2 3 5 7 9 10 15 16 18	33626
B	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) Dataset Description

	A	B	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

(b) Confusion Matrix Result

Fig. 9. (Case 1). Accuracy result of dataset $S_1D_{20}AD_7PO_{0.3}$.

Cluster	Dim. Size	Correlated Dimensions	Size
A	6	3 10 11 13 17 19	6011
B	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) Dataset Description

	A	*B*	*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

(b) Confusion Matrix Result

Fig. 10. (Case 2). Accuracy result of dataset $S_3D_{20}AD_7PO_{0.3}$.

these 113 points, and *dod* distances to the medoid from these points were equal to the distance to the cluster 3. When there is more than one cluster, which satisfies the member assignment condition, a point is assigned to the cluster with more *key dimensions*. In this case, it is not necessarily a wrong assignment because the numbers of *key dimensions* are same for the clusters 2 and 3 and those 113 points practically have correlations with both clusters.

Cluster 6 which is not matched to any original cluster, was made of outlier points, and its *key dimension* set is {5,15}. A user can use this low dimensional cluster or can just discard it. Because this cluster has too little dimensional information, it is totally up to the user. Since clusters formed 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 as in Case 1, it generated almost as many as or slightly more clusters than the number of original clusters. Moreover, even when there was an additional cluster, from the reason described above the overall performance was not affected by the additional cluster made by chance.

For Case 2 in Fig. 10, *bestepsilon* was chosen as 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. However, compared to the whole cluster sizes, those exceptional points are very small and the total accuracy was still 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 A.

4.6. Robustness and scalability

Robustness to the dimensionality D and AD. As a robustness result with respect to the dimensionality, we compared the performance of FINDIT and PROCLUS in Fig. 11. Each F_1-value plotted in Fig. 11 is the result on the datasets $S_2D_*AD_*PO_{0.1}$ where $*$ means the entire parameter range.

As we can see from the graphs, FINDIT's high accuracy is always stable and is not much affected by the changes in *D* and *AD*, while both *D* and *AD* are important to the performance of PROCLUS. Especially for PROCLUS, the performance drops when the dimensionality of clusters

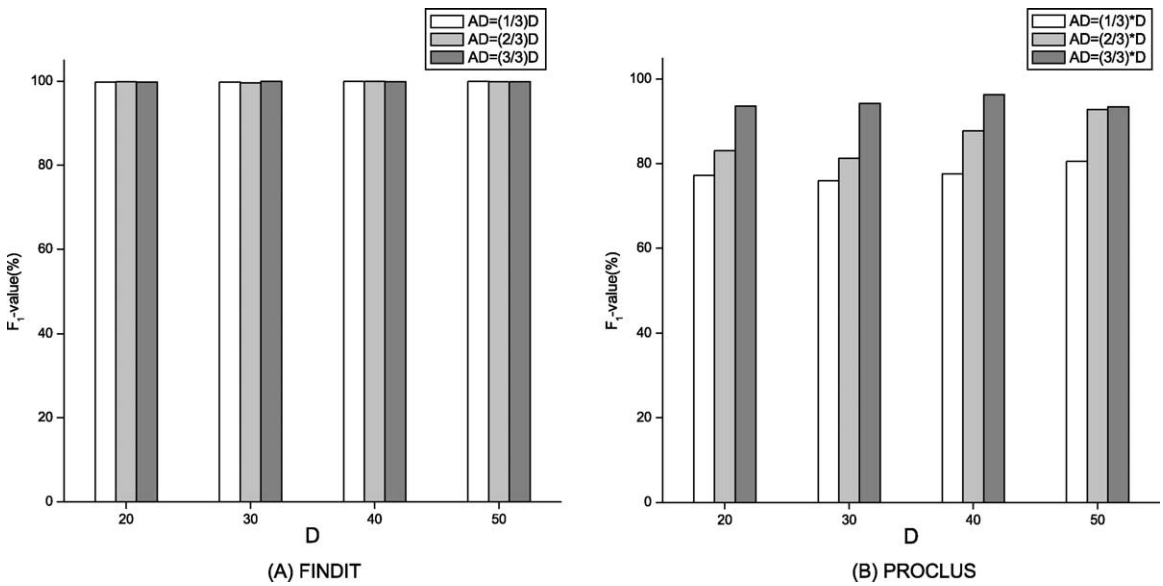


Fig. 11. Robustness to the dimensionality *D* and *AD* : F_1-value result.

decreases. In all cases, FINDIT has shown the better quality than PROCLUS. For FINDIT, 10 out of 12 test results 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. The *dprecision* value around 99% means that only one or two *key dimensions* are wrongly added to a cluster. So there is a possibility that some cluster points can be regarded as outliers based on the wrong dimensional information. However, since there are usually many medoids in a medoid cluster, points are naturally assigned to one of its medoids when *dprecision* is not significantly low.

Robustness to the percentage of outliers PO. Fig. 12 shows the F_1-values for the dataset $S_2D_{20}AD_*PO_*$, where $*$ means the entire parameter range. While FINDIT remains highly accurate in spite of increased outliers, the performance of PROCLUS drastically decreases when the percentage of outliers increases. It is because *k*-means style algorithms such as PROCLUS inevitably suffer from incorrect information of outliers due to their natural characteristics. *k*-means algorithm believes that the moving center is the natural way to find existing cluster centers so does PROCLUS. But because the center is moving based on newly assigned points, it is hard to determine whether a point is an outlier in iteration steps. Therefore, when the number of outliers becomes large, the moving center may 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. In addition, since an outlier point hardly becomes a nearest neighbor in *dod* measure, the correct dimension selection process is not affected by the noisy condition.

Robustness to the number of clusters and the cluster diameter. We describe here the performance of FINDIT to the number of clusters and the cluster diameter. In Fig. 13, each has the average performance value of 60 datasets of $D_*AD_*PO_*$ ($*$ means the entire parameter range) with corresponding row options and column options.

We observe that FINDIT is very robust to the number of clusters and also to the cluster diameter. 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 time scalability to the dimensionality, we plotted the execution time of FINDIT and PROCLUS for the datasets $S_1D_*AD_*PO_{0.1}$ in Fig. 14(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 execution time of PROCLUS is largely dependent on D and AD. Although the time efficiency of these two algorithms is similar with small D and AD, the difference between them increases up to several times when D or AD becomes large.

To test time scalability to the dataset size, we created a series of datasets with different dataset sizes using the dataset $S_1D_{30}AD10_PO_{0.1}$ as a base model (K is commonly set to 5). Each generated dataset has a size between 100 000 and 5 000 000. The results on the six generated datasets are plotted in Fig. 14(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 five times when the dataset size is as large as 5 000 000. For a fair comparison, we had chosen a small number for the iteration threshold of PROCLUS for

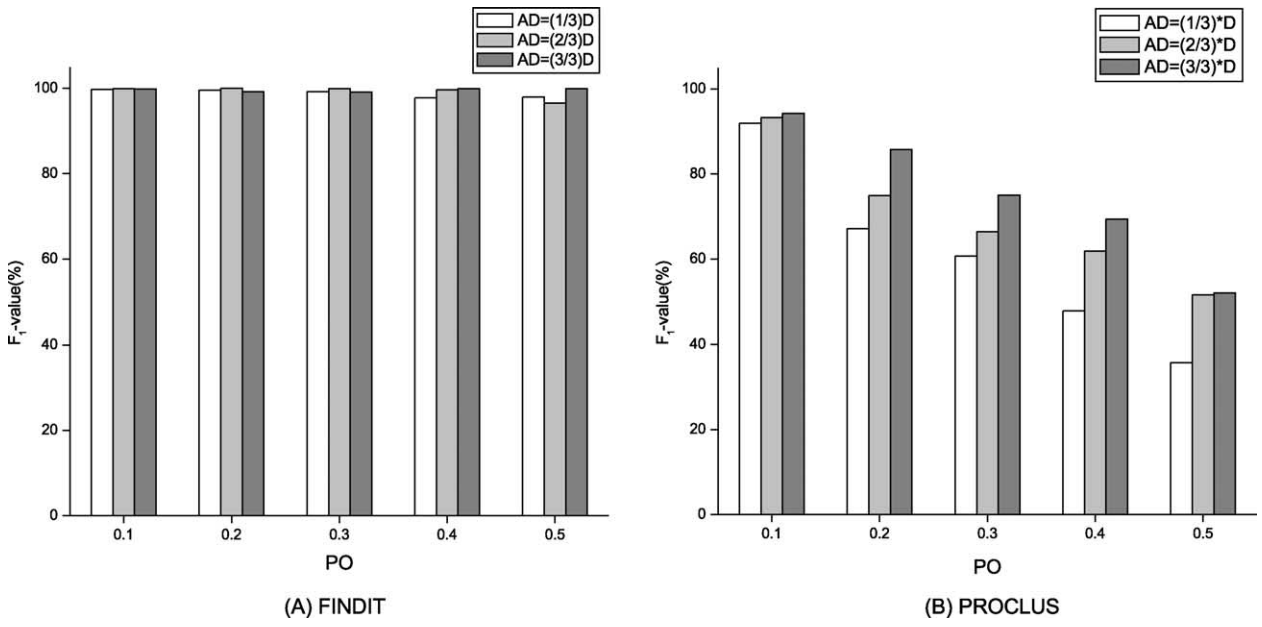


Fig. 12. Robustness to the percentage of outliers PO : F_1-value result.

	max cluster diameter					
	16($S1$)		24 ($S2$)		32 ($S3$)	
$No.of clusters$	$precision$	$recall$	$precision$	$recall$	$precision$	$recall$
$K = 5$	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 ($S3$)	
$No.of clusters$	$dprecision$	$drecall$	$dprecision$	$drecall$	$dprecision$	$drecall$
$K = 5$	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

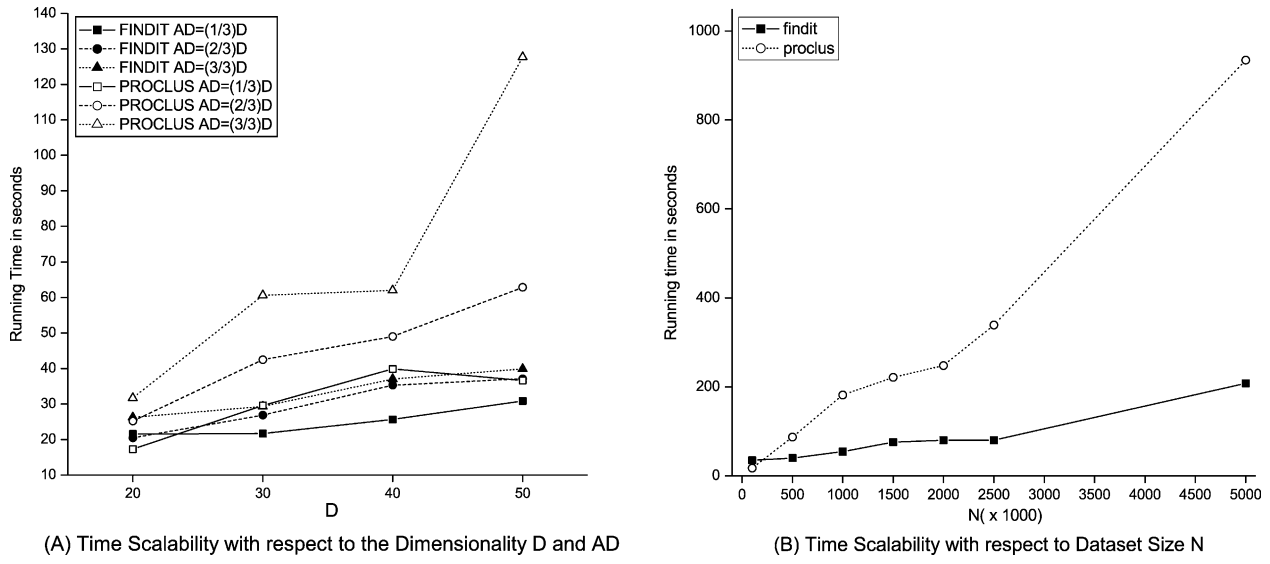
Fig. 13. (FINDIT) Quality results on the number of clusters and the cluster diameter.

the scalability tests (note that the $S_{minsize}$ of FINDIT was set to $N/20$ for all datasets).

With FINDIT, about 15 s were consumed in the cluster forming phase to find the best medoid cluster set $MC_{bestepsilon}$. The time consumed for cluster forming was almost same in all tests because the cluster forming phase used the fixed sizes of S and M for all datasets because we fixed the minimum cluster size to $N/20$ for all tests. In the cluster forming phase, the medoid clustering step which uses *gac* algorithm took most of the time. We expect that a more optimized implementation of *gac* clustering algorithm 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.) FINDIT's efficiency is due to fast dimension voting process based on sampling. Because FINDIT's dimension selection requires no iteration as in partitional approaches and no dimension partitioning as in grid approaches, it does not suffer significantly from the increased dimensionality or the increased dataset size.

5. Discussion

Epsilon and the soundness criteria. Selecting *bestepsilon* is very important in our method since the running parameter ϵ plays a key role in shaping clusters. To find out the property of ϵ, we conducted an experiment on two datasets $S_1D_{20}AD_7PO_{0.4}$ and $S_3D_{20}AD_7PO_{0.4}$. For these two datasets, we obtained 25 different medoid cluster sets variating ϵ



(A) Time Scalability with respect to the Dimensionality D and AD

(B) Time Scalability with respect to Dataset Size N

Fig. 14. Time scalability to the dimensionality and dataset size.

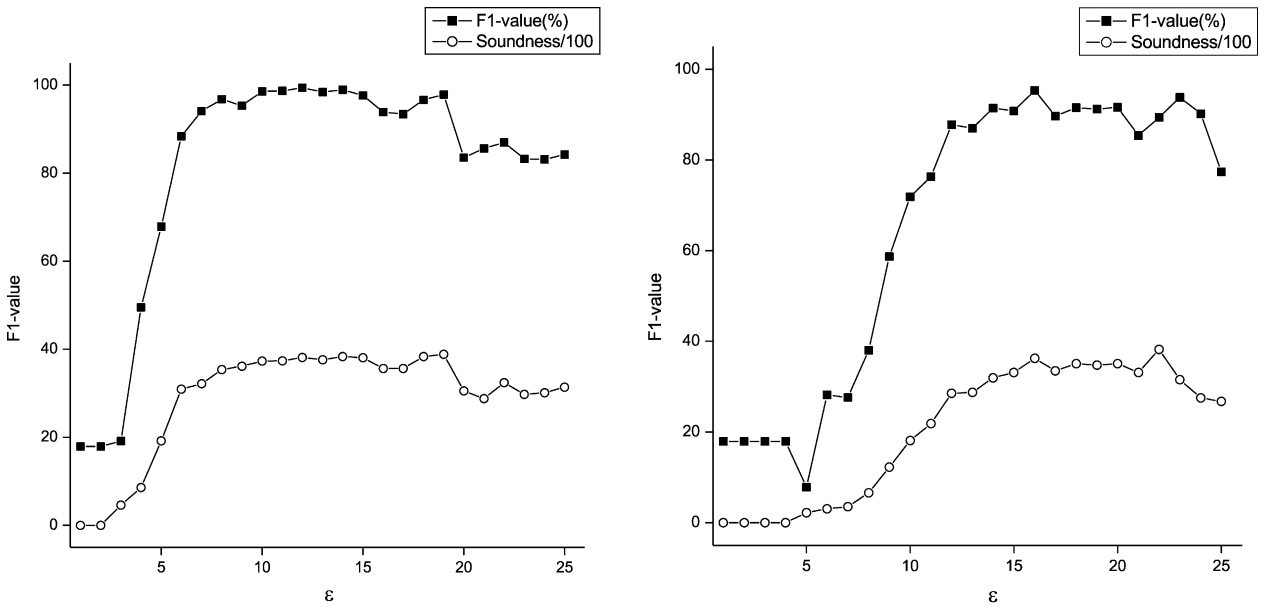


Fig. 15. The relationship between epsilon, soundness and quality of results.

from 1 to 25 (note that all dimensions have the normalized valuerange of [1,100]), and performed the data assigning phase against 25 different medoid cluster sets. Fig. 15(a) and (b) show the variation of F_1-value and *Soundness* according to ϵ values.

There are two things worth to be mentioned. For the first, we can see that the performance F_1-value is highly correlated to the soundness value in Fig. 15. In most cases, our soundness criteria selected the ϵ that gave the best performance between 25 different ϵ values. This simple but effective soundness criteria makes FINDIT intelligent enough to automatically select the ϵ values which guarantee the good performance. We think our soundness criteria can be useful also for other subspace clustering algorithms which try to generate disjoint clusters.

For the second, the F_1-value saturates near to optimal (100%) when 2ϵ becomes larger than the maximum cluster diameter of each dataset. In Fig. 15(a), the F_1-value and Soundness value start to saturate when ϵ becomes 8, and in the case of Fig. 15(b), they start to saturate when ϵ 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 ϵ 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 ϵ values.

The number of voters V. We ran 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 respect to the number of voters V; the results are shown in Fig. 16. When the maximum cluster diameter increased, the cases which used larger V values usually showed the better performance. However, the difference is not so significant. The quality of clustering is stable for various V values between 10 and $s_{minsize}$ (note that it is set to 30 in our settings).

Parameters. In the clustering problem, there is always a tradeoff between controllability and ease of application. If a clustering algorithm accepts more user parameters, the result becomes more likely what the user expected. On the other hand, if an algorithm requires less parameters, the algorithm becomes easier to apply.

	max cluster diameter								
	16($SR_{max} = 4$)			24 ($SR_{max} = 6$)			32 ($SR_{max} = 8$)		
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

Fig. 16. Quality results for different number of voters.

The user parameters define the meaning of the similarity regardless of their different forms. Therefore, good user parameters should not only be easy to understand but also be effective enough to deliver users' requirements to the algorithm. In this point of view, we can say that FINDIT's parameters are well balanced between controllability and ease of application because the two user parameters $C_{minsize}$ and $D_{mindist}$ describe the final shapes of clusters, which are very intuitive to users.

6. Conclusion

In this paper, we have suggested a new subspace clustering algorithm named FINDIT. We have experimentally shown that the proposed algorithm significantly improves quality of clustering, robustness to the increasing noise and cluster diameter, and time scalability to the dataset size and the dimensionality. It generates disjoint clusters accurately with their subspace information, and this high accuracy does not degrade at all even when 50% out of data points become 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 *dod* 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.

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

Fig. A1. Entire quality results on 60 datasets in Suite S_2 ($N = 100\,000, K = 5$).

Appendix A

See Fig. A1.

References

[1] J.S. Vitter, Random sampling with a reservoir, in: ACM Transactions on Mathematical Software, 1985.

[2] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice Hall, Englewood Cliff, NJ, 1988.

[3] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.

[4] D.R. Cutting, D.R. Karger, J.O. Pedersen, 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, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Portland, OR (1996).

[7] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for large databases, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Que., Canada (1996).

[8] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, in: Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997.

[9] G. Sheikholeslami, S. Chatterjee, A. Zhang, 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, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, WA (1998) 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, 1998.

[12] S. Guha, R. Rastogi, 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, J.S. Park, Fast algorithms for projected clustering, in: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadelphia, PA (1999).

[14] C.-h. Cheng, A. Fu, Y. 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, 1999.

[15] M. Ankerst, M.M. Breunig, H.-P. Kriegel, 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, V. Kumar, Chameleon: hierarchical clustering using dynamic modeling, IEEE Computer 32 (1999) 68–75.

[17] K. Beyer, When is nearest neighbor meaningful?, in: Proceedings of the Seventh International Conference on Database Theory, Jerusalem, Israel, 1999.

[18] S. Goil, H. Nagesh, A. Choudhary, MAFIA: efficient and scalable subspace clustering for very large data sets, Technical Report CPDC-TR-9906-010, Department of Electrical and Computer Engineering, Northwestern University, 1998.

[19] C.C. Aggarwal, P.S. Yu, Finding generalized projected clusters in high dimensional spaces, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, Texas, 2000.

[20] B. Liu, Y. Xia, P.S. Yu, Clustering through decision tree construction, in: Proceedings of the Ninth International Conference on Information Knowledge Management CIKM, McLean, VA, 2000.

[21] C.R. Palmer, C. Faloutsos, Density biased sampling: an improved method for data mining and clustering, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, Texas, 2000.

[22] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, Information Systems 26 (1) (2001).