Density Conscious Subspace Clustering for High Dimensional Data using Genetic Algorithms

G.Sangeetha
MSc (Information Science & Management)
Lecture – Kg College of Arts & Science, Coimbatore – India

Sornamaheswari
MCA,(PH.D)
Asst.Professor – Dept of Computer Science
LRG Government Arts College for Women - Tirupur

ABSTRACT
Clustering has been recognized as an important and valuable capability in the data mining field. Instead of finding clusters in the full feature space, subspace clustering is an emergent task which aims at detecting clusters embedded in subspaces. Most of previous works in the literature are density-based approaches, where a cluster is regarded as a high-density region in a subspace. However, the identification of dense regions in previous works lacks of considering a critical problem, called "the density divergence problem" in this thesis, which refers to the phenomenon that the region densities vary in different subspace cardinalities. Without considering this problem, previous works utilize a density threshold to discover the dense regions in all subspaces, which incurs the serious loss of clustering accuracy (either recall or precision of the resulting clusters) in different subspace cardinalities. To tackle the density divergence problem, in this thesis, we devise a novel subspace clustering model to discover the clusters based on the relative region densities in the subspaces, where the clusters are regarded as regions whose densities are relatively high as compared to the region densities in a subspace. Based on this idea, different density thresholds are adaptively determined to discover the clusters in different subspace cardinalities. Due to the infeasibility of applying previous techniques in this novel clustering model, we also devise an innovative algorithm, referred to as DENCOS (DENsity Conscious Subspace clustering), to adopt a divide-and-conquer scheme to efficiently discover clusters satisfying different density thresholds in different subspace cardinalities. Another approach for subspace clustering in high dimensional data is proposed using Genetic Approach. The GAs work with a population of *individuals* representing abstract representations of feasible solutions. Each individual is assigned a *fitness* that is a measure of how good solution it represents. The better the solution is, the higher the fitness value it gets. The population evolves towards better solutions. The evolution starts from a population of completely random individuals and iterates in generations. In each generation, the fitness of each individual is evaluated. Individuals are stochastically selected from a current population (based on their fitness), and modified by means of operators *mutation* and *crossover* to form a new population. It is capable of optimizing the number of clusters for tasks with well formed and separated clusters. As validated by our extensive experiments on retail data set, GENETIC can discover the clusters in all subspaces with high quality, and the efficiency of GENETIC outperforms previous works using DENCOS.

Keywords
Density Conscious Subspace Clustering For High Dimensional Data Using Genetic Algorithms

1. INTRODUCTION
CLUSTERING has been recognized as an important and valuable capability in the data mining field. For high-dimensional data, recent research have reported that traditional clustering techniques may suffer from the problem of discovering meaningful clusters due to the curse of dimensionality. Specifically, the curse of dimensionality refers to the phenomenon that as the increase of the dimension cardinality, the distance of a given point x to its nearest point will be close to the distance of x to its farthest point. Due to the loss of the distance discrimination in high dimensions, discovering meaningful, separable clusters will be very challenging, if not impossible. A common approach to cope with the curse of dimensionality problem for mining tasks is to reduce the data dimensionality by using the techniques of feature transformation and feature selection. The feature transformation techniques, such as principal component analysis (PCA) and singular value decomposition (SVD), summarize the data in a fewer set of dimensions derived from the combinations of the original data attributes. However, the transformed features/dimensions have no intuitive meaning anymore and thus the resulting clusters are hard to interpret and analyze. On the other hand, the feature selection methods reduce the data dimensionality by trying to select the most relevant attributes from the original data attributes. In such way, only a particular subspace1 is selected to discover the clusters. However, in many real data sets, clusters may be embedded in varying subspaces, and thus in the feature selection approaches the information of data points clustered differently in varying subspaces is lost.

Motivated by the fact that different groups of points may be clustered in different subspaces, a significant amount of research has been elaborated upon subspace clustering, which aims at discovering clusters embedded in any subspace of the original feature space. The applicability of subspace clustering has been demonstrated in various applications, including gene expression data analysis, E-commerce, DNA microarray analysis, and so forth.

We deal with one particular task of data analysis – *clustering*. Its goal is partitioning of a given data set into subsets (called *clusters*), so that data in each subset share some common trait (based on some defined distance or similarity/dissimilarity measure). The need for clustering techniques arises in many fields, including machine learning, data mining, pattern

recognition, image analysis, and bioinformatics. Various clustering algorithms are available in current statistical or machine learning software. The most commonly used algorithms are represented by the *k-means* algorithm and several variants of Kohonen's Self Organising Maps. Comparison of available statistical techniques and neural network based algorithms can be found in our previous work. Main drawbacks of the mentioned algorithms are their sensitivity to an initial setting (typically random initialisation) and necessity for the user to determine the number of clusters.

SUBSPACE CLUSTERING

We adopt the grid-based approach to discover subspace clusters, where the data space is partitioned into a number of non-overlapping rectangular units by dividing each attribute into δ equal-length intervals. Consider the projection of the data set in a k-dimensional subspace. A "k-dimensional unit" u is defined as the intersection of one interval from each of the k attributes. Let count(u) denote the number of data points contained in unit u. Note that the units in the same subspace cardinalities have the same size, and therefore we can use the count values to approximate the densities of the units in the same subspace cardinalities. Thus, the k-dimensional clusters can be discovered by first identifying the k-dimensional dense units, and then grouping the connected ones into clusters. Two k-dimensional units u1, u2 are connected if they have a common face or if there exists another k-dimensional unit u3 such that both u1 and u2 are connected to u3.

For identifying dense units, we propose to use different density thresholds for different subspace cardinalities. Let τ k denote the density threshold for the subspace cardinality k, and let N be the total number of data points. In addition, an user input parameter α, called the unit strength factor, is introduced for specifying how dense a unit would be identified as a dense one. Then, we define the density threshold τ k as:

$$\tau_k = \alpha \frac{N}{\delta^k}.$$

When the data are uniformly distributed in a k-dimensional subspace, the number of data points in each of the δk kdimensional units in this subspace will be N/δk, i.e. the average unit density. In this scenario, there are no clusters discovered because everywhere in this space is almost of the same density. As the data are more compacted into clusters, the units within clusters will be much denser and would have a larger count value than the average density. Thus, the input parameter α is introduced such that a k-dimensional unit will be identified as a dense one if its count value exceeds α times of the average unit density, i.e., N/δk .

In addition, a user parameter, kmax, is introduced for specifying the maximal subspace cardinality in such a way that clusters in cardinality up to kmax are discovered.

CLUSTERING USING GENETIC ALGORITHM

Genetic algorithms **(GAs),** as a universal optimization technique, represent another tool that can be applied to clustering. Our goal is to study an applicability of such approach and compare it to standard techniques. In this thesis we proposes the Clustering Genetic Algorithm, based both on techniques used in previous attempts at genetic clustering and our ideas. We focus on comparison of different genetic operators and investigate the possibility of optimising number of clusters.

In this Paper, we devise a novel subspace clustering model, which is based on the relative region densities to discover the clusters. In our subspace clustering model, we regard the clusters in a subspace as the regions which have relatively high densities as compared to the average region density in the subspace. To discover such clusters, we introduce a novel density parameter for users to specify their expected relative rate of the densities of the dense regions and the average region density in a subspace. In earlier studies an innovative algorithm, referred to as "DENsity COnscious Subspace clustering"(abbreviated as DENCOS)has been devised, to efficiently discover the clusters satisfying different density thresholds in different subspace cardinalities. In DENCOS, the mechanism of computing the upper bounds of region densities to constrain the search of dense regions is devised, where the regions whose density upper bounds are lower than the density thresholds will be pruned away in identifying the dense regions. We compute the region density upper bounds by utilizing a novel data structure, DFP-tree (Density FP-tree), where we store the summarized information of the dense regions. Further we present a clustering technique based on **genetic algorithms** – Clustering Genetic Algorithm. It is capable of optimizing the number of clusters for tasks with well formed and separated clusters.

1. Create randomly an initial population P_0 of M individuals.
2. $i \leftarrow 0$
3. Evaluate fitness for individuals from P_i.
4. If the stop criterion is satisfied, stop and return the best individual.
5. $P_{i+1} \leftarrow$ empty set
6. $I_1 \leftarrow selection(P_i); I_2 \leftarrow selection(P_i)$
7. With probability p_c: $(I_1, I_2) \leftarrow crossover(I_1, I_2)$
8. With probability p_m: $I_k \leftarrow mutate(I_k), k = 1, 2$
9. Insert I_1, I_2 into P_{i+1}.
10. If P_{i+1} has less then M individuals goto 6.
11. $i \leftarrow i + 1$ Goto 3.

Algorithm. Genetic algorithm.

The Problem Definition

The identification of dense regions in previous works lacks of considering a critical problem, called "the density divergence problem", which refers to the phenomenon that the region densities vary in different subspace cardinalities. Without considering this problem, previous works utilize a density threshold to discover the dense regions in all subspaces, which incurs the serious loss of clustering accuracy (either recall or precision of the resulting clusters) in different subspace cardinalities.

A common approach to cope with the curse of dimensionality problem for mining tasks is to reduce the data dimensionality by using the techniques of feature transformation and feature selection. The feature transformation techniques, such as principal component analysis (PCA) and singular value decomposition

(SVD), summarize the data in a fewer set of dimensions derived from the combinations of the original data attributes. However, the transformed features/dimensions have no intuitive meaning anymore and thus the resulting clusters are hard to interpret and analyze. On the other hand, the feature selection methods reduce the data dimensionality by trying to select the most relevant attributes from the original data attributes. In such way, only a particular subspace1 is selected to discover the clusters. However, in many real data sets, clusters may be embedded in varying subspaces, and thus in the feature selection approaches the information of data points clustered differently in varying subspaces is lost.

Steps for Clustering the Data using Genetic Algorithms

- **Preprocessing Dataset**

We first transform the data set by transforming each d-dimensional data point into a set of d one-dimensional units, corresponding to the intervals within the d dimensions it resides in. the DFP-tree is constructed to condense the transformed data set. In this thesis, we devise the DFP-tree by adding the extra feature in the FP-tree for discovering the dense units with different density thresholds. In this thesis, we propose to compute the upper bounds of unit counts for constraining the searching of dense units such that we add extra features into the DFP-tree for the computation. The DFP-tree is constructed by inserting each transformed data as a path in the DFP-tree with the nodes storing the one-dimensional units of the data. The paths with common prefix nodes will be merged and their node counts are accumulated.

- **Generate and discover Inherent Dense Units**

In this discovering stage, we consider to utilize the nodes satisfying the thresholds to discover the dense units. For the nodes with node counts satisfying the thresholds for some set of subspace cardinalities, we will take their prefix paths to generate the dense units of their satisfied subspace cardinalities. However, a naive method to discover these dense units would require each node to traverse its prefix path several times to generate the dense units for the set of satisfied subspace cardinalities. In this thesis, we have explored that the set of dense units a node requires to discover from its prefix path can be directly generated by utilizing the dense units discovered by its prefix nodes, thus avoiding the repeated scans of the prefix paths of the nodes. Therefore, by a traversal of the DFP-tree, we can efficiently discover the dense units for all nodes satisfying the thresholds.

- **Generate and discover Acquired Dense Units**

In this discovering stage, for the nodes whose node counts do not exceed nk, we take the nodes carrying the same one dimensional unit together into consideration in discovering the k-dimensional dense units. Note that the surplus count SC_u^k is the maximal possible unit count of the units that can be generated from the prefix paths of the nodes in N_u^k, which is the case when a unit can be derived from all these paths so that this unit has the unit count equal to the summation of the node counts of the nodes in N_u^k, i.e., SC_u^k. Clearly, if $SC_u^k < \tau_k$, there will be no k-dimensional dense units that can be discovered from the prefix paths of the nodes in Nk u such that we need not apply the discovery process on Nk u to explore the k-dimensional dense units.

- **Implement path removal techniques**

In path removal technique, the two steps of the path reconstruction process, i.e., path exclusion and path reorganization, can correctly prepare the paths for performing the path removal. In the step 1, path exclusion, the paths in PathList cannot exist in current $\hat{u}_h$ conditional pattern base if they do not contain the set of one-dimensional units in I, i.e., $\hat{u}_h$ [B. Because $\hat{u}_h$'s conditional pattern base is constructed by extracting the prefix paths of the $\hat{u}_h$'s nodes in the DFP-tree TreeB, the paths in PathList which do not contain $\hat{u}_h$ cannot exist in $\hat{u}_h$'s conditional pattern base. The step 2, "path reorganization," in the reconstruction process is to reorganize the remaining paths in PathList to the form they should appear in $\hat{u}_h$'s conditional pattern base.

- **Implement Genetic Algorithm**

In this module the Genetic Algorithm is used to a particular optimization problem, one has to specify a representation of solutions in individuals (coding), fitness function, and typically also operators crossover and mutation.

Methodology

The methodology of DENCOS algorithm focuses on discovering the dense units because after the dense units are mined, the algorithm follows the procedure to group the connected dense units into clusters. In DENCOS, the dense unit discovery is performed by utilizing a novel data structure DFP-tree (Density FP-tree), which is constructed on the data set to store the complete information of the dense units. From the DFPtree, we compute the lower bounds and upper bounds of the unit counts for accelerating the dense unit discovery, and these informations are utilized in a divide-and-conquer scheme to mine the dense units. Therefore, DENCOS is devised as a two-phase algorithm comprised of the preprocessing phase and the discovering phase. The preprocessing phase is to construct the DFP-tree on the transformed data set, where the data set is transformed with the purpose of transforming the density conscious subspace clustering problem into a similar frequent itemset mining problem. Then, in the discovering phase, the DFP-tree is employed to discover the dense units by using a divide-and-conquer scheme.

The Genetic Algorithm is used to a particular optimization problem, one has to specify a representation of solutions in

individuals (coding), fitness function, and typically also operators crossover and mutation.

Crossover

We have designed two types of crossover operators –*one-point crossover* and *combining crossover*. The one-point crossover works in the same way as the binary one-point crossover, except the crosspoint is always generated on the boundary of the blocks. This ensures that always the whole centres are exchanged between individuals. The combining crossover combines the two solutions. It builds the new offsprings centre by centre. For each centre from the parent individual it finds the nearest centres from the second parent and generates two new centres randomly on the line joining the two parent centres.

Mutation

The CGA allows five kinds of mutation operators. The first two of them – *one-point mutation* and *biased one-point mutation* change the value of a centre randomly picked. In the former, the selected centre is replaced by point chosen at random. The latter moves slightly the centre in random direction. The third operator – *K-means mutation* – performs several (i.e. typically 2 to 5) steps of k-means algorithm. The other two operators – *cluster addition* and *cluster removal* – modify the number of clusters. The cluster addition adds one centre chosen randomly from the data set S, the cluster removal deletes one centre chosen at random.

The former in fact realises adding one cluster, while the latter removes one cluster. When these operators are used, the algorithm works with individuals of variable length and so not only the positions of cluster centres, but also their number (i.e. the number of clusters) is optimised. However, the fitness function has to reflect it and penalise the individuals with higher number of centres.

Fitness

The fitness function reflects a quality of individuals, so it should correspond to an objective function of an underlying optimisation problem. The goal of the CGA is to optimize the clusters quality, so the fitness of the individual I = {c1, . . . ,cK} is evaluated as

$$f(I) = -E_{VQ}(\vec{c}_1, \ldots, \vec{c}_K).$$

The fitness function is intended for applications, where the number of clusters is given in advance. Otherwise, one had to add a term penalising long individuals. Such penalisation is always problem dependent and requires user's assistance. Other cluster quality measures may be used as well. In the fitness function based on the *silhouette* is used. To define it, let us consider a vector x belonging to a cluster A. Then we define a(x) as the average distance between x and other vectors from A. Formally

$$a(\vec{x}) = \frac{1}{|A|} \sum_{\vec{y} \in A} ||\vec{x} - \vec{y}||^2$$

In addition, the average distance between x and vectors from the cluster C (C 6= A) we denote d(x,C). Now, we can define

$$b(\vec{x}) = \min_{C \neq A} d(\vec{x}, C).$$

The silhouette of vector ~x is then given by

$$s(\vec{x}) = \frac{b(\vec{x}) - a(\vec{x})}{max\{b(\vec{x}), a(\vec{x})\}}.$$

It easy to see that s(x) ∈ h−1, 1i and the higher the silhouette the better the assignment of x to the cluster A. The fitness function based on the silhouette is given by

$$f_s(I) = \sum_{i=1}^{N} s(\vec{x}_i).$$

It is shown that such fitness function may be capable also of determining the right number of clusters. However, the evaluation of the silhouette fitness function is quite time expensive and requires to store the matrix of distances between all data points. To reduce the time and space complexity the mean distances between the given point and all points from the particular clusters are replaces by the distances between the given point and centres of clusters.

Normalisation

In the CGA, the algorithm is enhanced by a normalisation step. The fitness evaluation of individual I = {c1, . . . , cK} consists of two sub steps:

1. Individual normalisation:
   (a) create empty clusters $C_1, \ldots, C_K$
   (b) for all $\vec{x}_i \in S$: add $\vec{x}_i$ to cluster C_j, where $j = \arg\min_k ||\vec{x}_i - \vec{c}_k||^2$
   (c) modify the individual so that the centres lay in the gravity centres of the clusters $C_1, \ldots, C_K$:

$$\vec{c}_i = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}, \qquad i = 1, \ldots, K \qquad (12)$$

2. Fitness evaluation

The normalisation step improves the convergence of the algorithm. Note that it is in fact one iteration of the k-means algorithm.

Fig. 1 Procedures to discover the inherent dense units.

Procedure GenerateInherentDenseUnits:
Input: a DFP-tree node: n, **Output:** the inherent dense units
1 Calculate the cardinality $k(n)$ for node n such that $\tau_{k(n)} \le n.count < \tau_{k(n)-1}$
2 **if** ($k(n) > k_{max}$) return;
3 Call the procedure GenerateOneNodeDenseUnit(n, $k(n)$); //Generate dense units for n
4 **for each** child node n_c of node n
5 Call the procedure GenerateInherentDenseUnits(n_c);

Procedure GenerateOneNodeDenseUnit:
Input: a DFP-tree node n_i, the cardinality $k(n_i)$, **Output:** the dense units related to node n_i
1 Let u_{n_i} be the 1-dimensional unit carried by n_i
2 //Generate dense units of cardinalities in $[k(n_i), k_{max}]$
3 **for each** prefix node n_j of n_i {
4 **if** ($k(n_j) < k(n_i)$) {
5 **for each** k with $k(n_i) \le k \le k_{max}$
6 Output the k-dimensional dense units generated by concatenating u_{n_i} with each $(k-1)$-dimensional dense unit in $I_{n_j}^{k-1}$;
7 }**else**{
8 **for each** k with $k(n_i)+1 \le k \le k_{max}$
9 Output the k-dimensional dense units generated by concatenating u_{n_i} with each $(k-1)$-dimensional dense unit in $I_{n_j}^{k-1}$;
10 Output the $k(n_j)$-dimensional dense units by first generating $I_{n_j}^{k(n_i)-1}$, and then concatenating u_{n_i} with the units in $I_{n_j}^{k(n_i)-1}$;
11 }
12 }

Fig. 2. Procedures to discover the acquired dense units.

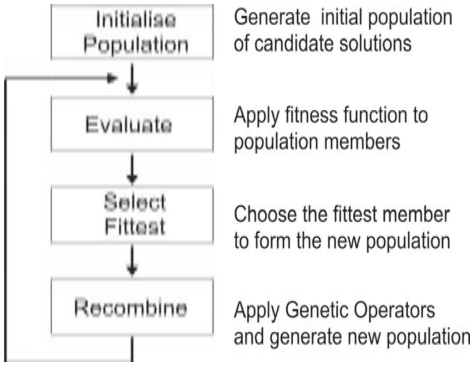
Procedure GenerateAcquiredDenseUnits:
Input: DFP-tree: *Tree*, **Output:** dense units
1 **for each** entry h in the header table of *Tree* {
2 Let u_h be the 1-dimensional unit carried by entry h
3 Let *cardList* be the buffer to store the subspace cardinalities with surplus counts not satisfying thresholds
4 Let $card_S$ and $card_L$ be the smallest and largest cardinalities in *cardList*
5 //Setup *cardList*, $card_S$, $card_L$
6 $card_S = k_{max}$; $card_L = 2$;
7 **for each** subspace cardinality k from 2 to k_{max} {
8 if($SC_{u_h}^k \ge \tau_k$){
9 insert k into *cardList*
10 if ($k < card_S$) $card_S = k$;
11 if ($k > card_L$) $card_L = k$;
12 }
13 }
14 **if** ($cardList = \phi$)
15 continue; //go to next header entry
16 Setup $P_{u_h}^{card_S}$ by following the node links of u_h;
17 Construct the DFP-tree $Tree_{u_h}^{card_S}$ on $P_{u_h}^{card_S}$
18 Call procedure MineDenseUnits(u_h, $Tree_{u_h}^{card_S}$, *cardList*, $card_L$, $P_{u_h}^{card_S}$, u_h)
19 }

Fig.3. Procedures to Mine dense units.

Procedure MineDenseUnits:
Input: (BaseUnits: B, DFP-tree: *Tree, cardList, card$_L$,* PathSet: *PSet,* Units: *u)* **Output:** dense units

```
1  for each entry ĥ in the header table of Tree {
2      Let û_h be the 1-dimensional unit carried by entry ĥ
3      Let I denote the unit by joining û_h with B (i.e., I = û_h ∪ B ), and let |I| be the cardinality of I
4      if ( |I| is not in cardList ){
5          if ( |I|==card_L )
6              Return;
7          Extract û_h's conditional pattern base from Tree by following û_h's node links
8          Construct a DFP-tree, Tree_û_h , on û_h's conditional pattern base, where the 1-d units
           with total counts below τ_card_L are removed
9          MineDenseUnits(I, Tree_û_h , cardList, card_L, PSet, u ); //Recursive mining
10     } else {
11         Set the count value of I, denoted as count(I), to be the unit count stored in entry ĥ
12         if ( count(I) ≥ τ_|I| )
13             Output I as a dense unit;
14         if ( |I|==card_L )
15             Return;
16         Extract û_h's conditional pattern base from Tree by following û_h's node links
17         if ( û_h's conditional pattern base is empty )
18             Return;
19         //Do path removal
20         Let card_next denote the next cardinality to discover dense units, which is the smallest
           cardinality larger than |I| in cardList
21         Let PathList be the buffer for storing paths to be removed from û_h's conditional pattern base
22         Identify the paths in PSet, whose count values are smaller than τ_|I| but are larger than or
           equal to τ_card_next , and insert them into PathList
23         Let I_u^- be the unit by removing the unit u from I;
24         Delete the paths in PathList, which do not contain I_u^- ; //path exclusion
25         for each path p in PathList{
26             //Path reorganization, to re-organize the path p to the form it appears in û_h's
               // conditional pattern base
27             Delete the nodes in p which are positioned below û_h in the header table of Tree
28             if ( p is not empty ){
29                 Sort the nodes in p according to the unit order in the header table of Tree
30                 Remove p from û_h's conditional pattern base; //perform path removal
31             }
32         }
33         Construct a DFP-tree, Tree_û_h , on û_h's conditional pattern base, where 1-d. units with
           total counts below τ_card_L are removed
34         if (Tree_û_h is not empty )
35             MineDenseUnits(I, Tree_û_h , cardList, card_L, PSet, u); //Recursive mining
36     }
37 }
```

1. Genetic Algorithm implementation pseudocode



1. Create randomly an initial population P_0 of M individuals.
2. $i \leftarrow 0$
3. Evaluate fitness for individuals from P_i.
4. If the stop criterion is satisfied, stop and return the best individual.
5. $P_{i+1} \leftarrow$ empty set
6. $I_1 \leftarrow selection(P_i); I_2 \leftarrow selection(P_i)$
7. With probability p_c: $(I_1, I_2) \leftarrow crossover(I_1, I_2)$
8. With probability p_m: $I_k \leftarrow mutate(I_k), k = 1, 2$
9. Insert I_1, I_2 into P_{i+1}.
10. If P_{i+1} has less then M individuals goto 6.
11. $i \leftarrow i + 1$ Goto 3.

Performance Comparison

Here we compare the DENCOS with respect to the GENETIC approach for the Density Conscious Subspace Clustering for High-Dimensional Data. We compare the two algorithms with two constraints (i.e) Execution Time and Clustering Accuracy.

The results are tabulated below:

(i) Table for execution time of DENCOS and GENETIC Algorithm

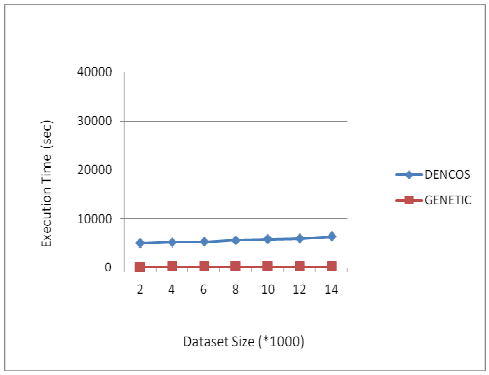
DATASET SIZE (*1000)	DENCOS (sec)	GENETIC (sec)
2	500	150
4	520	175
6	530	185
8	560	195
10	580	200
12	600	205
14	640	210

(ii) Table for Clustering Accuracy of DENCOS and GENETIC Algorithm

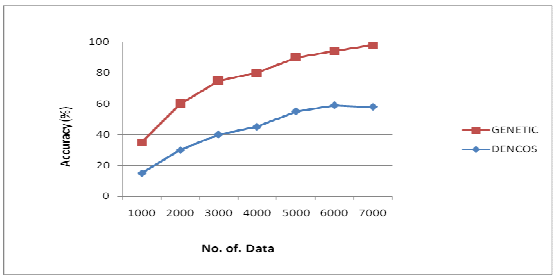
No.of. Data	DENCOS (%)	Genetic (%)
1000	15	20
2000	30	30
3000	40	35
4000	45	35
5000	55	35
6000	59	35
7000	58	40

As validated by our extensive experiments on retail data set, GENETIC can discover the clusters in all subspaces with high quality, and the efficiency of GENETIC outperforms previous works using DENCOS.

Time Graph



Accuracy Graph



REFERENCES

[1] C.C. Aggarwal, A. Hinneburg, and D. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," Proc. Eighth Int'l Conf. Database Theory (ICDT), 2001.

[2] I. Assent, R. Krieger, E. Muller, and T. Seidl, "DUSC: Dimensionality Unbiased Subspace Clustering," Proc. IEEE Int'l Conf. Data Mining (ICDM), 2007.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD), 1996.

[4] H. Fang, C. Zhai, L. Liu, and J. Yang, "Subspace Clustering for Microarray Data Analysis: Multiple Criteria and Significance," Proc. IEEE Computational Systems Bioinformatics Conf., 2004.

[5] G. Moise, J. Sander, and M. Ester, "P3C: A Robust Projected Clustering Algorithm," Proc. Sixth IEEE Int'l Conf. Data Mining (ICDM), 2006.

[6] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," ACM SIGKDD Explorations Newsletter, vol. 6, pp. 90-105, 2004.

[7] H.S. Nagesh, S. Goil, and A. Choudhary, "Adaptive Grids for Clustering Massive Data Sets," Proc. First SIAM Int'l Conf. Data Mining (SDM), 2001.

[8] H.-P. Kriegel, P. Kroger, M. Renz, and S. Wurst, "A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data," Proc. Fifth IEEE Int'l Conf. Data Mining (ICDM), 2005.

[9] Y.B. Kim, J.H. Oh, and J. Gao, "Emerging Pattern Based Subspace Clustering of Microarray Gene Expression DataUsing Mixture Models," Proc. Int'l Conf. Bioinformatics and Its Applications (ICBA), 2004