

Research Article

Subspace Clustering of High-Dimensional Data: An Evolutionary Approach

Singh Vijendra¹ and Sahoo Laxman²

¹ Department of Computer Science and Engineering, Faculty of Engineering and Technology, Mody Institute of Technology and Science, Lakshmanagarh, Rajasthan 332311, India

² School of Computer Engineering, KIIT University, Bhubaneswar 751024, India

Correspondence should be addressed to Singh Vijendra; vsingh.fet@gmail.com

Received 21 August 2013; Revised 20 October 2013; Accepted 11 November 2013

Academic Editor: Sebastian Ventura

Copyright © 2013 S. Vijendra and S. Laxman. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Clustering high-dimensional data has been a major challenge due to the inherent sparsity of the points. Most existing clustering algorithms become substantially inefficient if the required similarity measure is computed between data points in the full-dimensional space. In this paper, we have presented a robust multi objective subspace clustering (MOSCL) algorithm for the challenging problem of high-dimensional clustering. The first phase of MOSCL performs subspace relevance analysis by detecting dense and sparse regions with their locations in data set. After detection of dense regions it eliminates outliers. MOSCL discovers subspaces in dense regions of data set and produces subspace clusters. In thorough experiments on synthetic and real-world data sets, we demonstrate that MOSCL for subspace clustering is superior to PROCLUS clustering algorithm. Additionally we investigate the effects of first phase for detecting dense regions on the results of subspace clustering. Our results indicate that removing outliers improves the accuracy of subspace clustering. The clustering results are validated by clustering error (CE) distance on various data sets. MOSCL can discover the clusters in all subspaces with high quality, and the efficiency of MOSCL outperforms PROCLUS.

1. Introduction

Clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. The task of clustering has been studied in statistics [1], machine learning [2–4], bioinformatics [3, 5–7], and more recently in databases [8–10]. Clustering algorithms find a partition of the points such that points within a cluster are more similar to each other than to points in different clusters [11]. In traditional clustering each dimension is equally weighted when computing the distance between points. Most of these algorithms perform well in clustering low-dimensional data sets [12–15]. However, in higher dimensional feature spaces, their performance and efficiency deteriorate to a greater extent due to the high dimensionality [16]. Another difficulty we have to face when dealing with clustering is the dimensionality of data. In the clustering task, the overwhelming problem of high dimensionality presents a dual aspect. First, the presence of irrelevant attributes eliminates any hope on clustering

tendency, because such features cause the algorithm to search for clusters where there is no existence of clusters. This also happens with low-dimensional data, but the likelihood of presence of irrelevant features and their number grow with dimension. The second problem is the so-called “curse of dimensionality.” For clustering this means that clusters do not show across all attributes as they are hidden by irrelevant attributes or blurred by noise. Clustering methods are typically either based on distances (like partitioning and hierarchical clustering) or on densities (like density-based methods). In [17] the authors study the effects of high dimensions on the nearest neighbor $d_{\min}(o)$ and the farthest neighbor $d_{\max}(o)$ of an object o in detail. They have proven the following equation for different distributions:

$$\forall \varepsilon \geq 0: \lim_{\dim \rightarrow \infty} P(d_{\max}(o) < (1 + \varepsilon)d_{\min}(o)) = 1. \quad (1)$$

This statement formalizes that, with growing dimensionalities (\dim), the distance to the nearest neighbor is nearly

equal to the distance to the farthest neighbor (distances become more and more similar). Consequently, clustering methods based on distance functions have problems to extract meaningful patterns in high dimensional spaces as they either cluster only one object (the nearest neighbor) or nearly the complete data set (the farthest neighbor). Figure 1 shows that clusters are embedded in different subspaces of high-dimensional data sets.

Densities also suffer from the curse of dimensionality. In [18] the authors describe an effect of higher dimensions on density distributions: 99% of the mass of a ten-dimensional normal distribution is at points whose distance from the origin is greater than 1.6. This effect is directly opposite in lower dimensional spaces: 90% of the objects have a distance of less than 1.6 SD from the origin regarding a one-dimensional distribution. Density-based clustering methods [19, 20] hence have problem to determine the density of a region as the objects are scattered over the data space. Grid-based methods [10, 21, 22] are capable of discovering cluster of any shape and are also reasonably fast. However, none of these methods address how to efficiently cluster very large data sets that do not fit in memory. Furthermore, these methods also only work well with input spaces with low to moderate numbers of dimensions. As the dimensionality of the space increases, grid-based methods face some serious problems. The number of cells grows exponentially and finding adjacent high-density cells to form clusters becomes prohibitively expensive. Often, especially in high dimensional spaces, not all dimensions are relevant—the data are bound along such dimensions, to a given cluster. It is vital for a clustering method to be able to detect clusters being embedded in subspaces possibly formed by different combinations of dimensions in different data localities.

These observations motivate our effort to propose a novel subspace clustering algorithm called multiobjective subspace clustering (MOSCL) that efficiently clusters high dimensional numerical data sets. The first phase of MOSCL performs subspace relevance analysis by detecting dense and sparse regions and their locations in data set. After detection of dense regions it eliminates outliers. The discussion details key aspects of the proposed MOSCL algorithm including representation scheme, maximization fitness functions, and novel genetic operators. In thorough experiments on synthetic and real world data sets, we demonstrate that MOSCL for subspace clustering is superior to method such as PROCLUS [23]. Additionally we investigate the effects of first phase for detecting dense regions on the results of subspace clustering. The performance measure of MOSCL is evaluated by the clustering error (CE) distance [24]. It is an intuitive way to compare clustering because it uses the proportion of points which are clustered differently in generated subspace clusters and real subspace clusters after optimal matching of clusters.

The remainder of this paper is structured as follows. In Section 2, we review some related work. Section 3 describes multiobjective subspace clustering. Section 3.1 presents preprocessing phase that detects dense regions and sparse regions and removes outliers. Section 3.2 describes the design concepts of Multi Objective Subspace CLustering

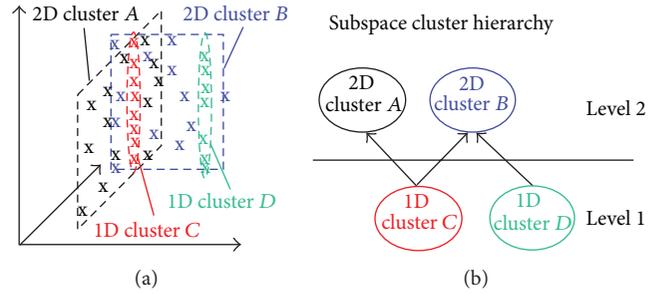


FIGURE 1: An example of subspace clusters.

(MOSCL) algorithm. Section 4 presents experimental evaluation. Finally, we conclude in Section 5.

2. Related Work

The subspace algorithms can be divided in to two categories: partition-based subspace clustering algorithms and grid-based subspace algorithms. Partition-based algorithms partition the set of objects into mutually exclusive groups. Each group along with the subset of dimensions shows the greatest similarity known as a subspace cluster. Similar to the K -means method, most algorithms in this category define an objective function to be minimized during the search. The major difference between these methods and the K -means algorithm is that here the objective functions are related to the subspaces where each cluster resides in. CLIQUE (clustering in quest) [25] is one of the first subspace algorithms that attempt to find clusters within subspaces of the data set. This algorithm combines density and grid-based clustering and uses an APRIORI style technique to find clusterable subspaces. It is difficult for CLIQUE to discover high-quality clusters in all subspaces. This may be because the unit densities vary in different subspace cardinalities such as the identification of the dense units in all subspaces by utilizing an absolute unit density threshold. It may suffer from the trade-off between precision and recall. SUBCLU (density-connected subspace clustering) [26] overcomes the limitations of grid-based approaches like the dependence on the positioning of the grids. Instead of using grids it uses density-connected sets of DBSCAN [19] cluster model. SUBCLU is based on a bottom-up, greedy algorithm to detect the density-connected clusters in all subspaces of high-dimensional data. However SUBCLU also suffers from the density divergence problem. PROCLUS (projected clustering) [23], a typical partition-based subspace clustering algorithm, searches for a partition of the dataset into clusters together with the set of dimensions on which each cluster is correlated. PROCLUS is a variation of the k -medoid algorithm, in which the number of clusters k and the average number of dimensions of clusters l need to be specified before the running of the algorithm. This algorithm also assumes that one data point can be assigned to at most one subspace cluster or classified as an outlier, while a dimension can belong to multiple clusters. ORCLUS (arbitrarily oriented projected CLUster generation) [27] is a generalization from PROCLUS

[23], which finds clusters in arbitrarily oriented subspaces. ORCLUS finds projected clusters as a set of data points C together with a set of orthogonal vectors such that these data points are closely clustered in the defined subspace. A limitation of these two approaches is that the process of forming the locality is based on the full dimensionality of the space. However, it is not useful to look for neighbors in data sets with very low-dimensional projected clusters. In addition, PROCLUS and ORCLUS require the user to provide the average dimensionality of the subspace, which also is very difficult to do in real-life applications. FIRES (Filter refinement subspace clustering) [28] is a general framework for efficient subspace clustering. It is generic in such a way that it works with all kinds of clustering notions. CLICK (subspace cLusterIng of categorical data via maximal k -partite cliques) [29] uses a novel formulation of categorical subspace clusters, based on the notion of mining cliques in a k -partite graph. It implements an efficient algorithm to mine k -partite maximal cliques, which correspond to the clusters. COSA (clustering objects on subsets of attribute) [30] formalizes the subspace clustering problem as an optimization problem. The algorithm returns with mutually exclusive subspace cluster with each data point assigned to exactly one subspace cluster. One dimension can belong to more than one subspace cluster. However, the subspace in which each cluster is embedded is not explicitly known from the algorithm. The FINDIT (fast and intelligent subspace clustering algorithm using dimension voting) [31] algorithm, uses a dimension voting technique to find subspace clusters. Dimension oriented distance is defined to measure the distance between points based on not only the value information but also the dimension information. DENCOS (density conscious subspace clustering) [32] tackles the density divergence problem; in this algorithm, authors 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.

Grid based subspace clustering algorithms consider the data matrix as a high-dimensional grid and the clustering process as a search for dense regions in the grid. ENCLUS (entropy based clustering) [33] uses entropy instead of density and coverage as a heuristic to prune away uninteresting subspaces. It finds correlated, high density and high coverage subspaces using level wise search that is used in CLIQUE [25]. However, this algorithm finds only subspaces within which meaningful clusters exist, without explicitly finding the actual clusters. A more significant modification of CLIQUE is presented in MAFIA that extends the base units in CLIQUE to utilize adaptive and variable-sized units in each dimension. These variable-sized bins are taken as building blocks to form units in higher subspaces. MAFIA [34] also utilizes candidate generate-and-test scheme to generate candidate dense units in higher subspaces, thus resulting in unavoidable information overlapping in the clustering result. pMAFIA (merging adaptive finite intervals and is more than a clique) [35] proposes to use adaptive units instead of the rigid ones

used in CLIQUE [25]. Each dimension is partitioned into windows of small size, and then adjacent windows having similar distribution are merged to form larger windows. However, in the CLIQUE the search complexity increases exponentially as a function of the highest dimensionality of the dense units. pMAFIA may have the difficulties in discovering clusters with high qualities in all subspace cardinalities. DOC (density-based optimal projective clustering) [36] proposes a mathematical definition of an optimal projective cluster along with a Monte Carlo algorithm to compute approximations of such optimal projective clusters. DOC tries different seeds and neighboring data points, in order to find the cluster that optimizes the quality function. The entire process is repeated to find other projected clusters. It is clear that since DOC scans the entire data set repetitively, its execution time is very high. O-Cluster (orthogonal partitioning clustering) [37] clustering method combines a novel partitioning active sampling technique with an axis parallel strategy to identify continuous areas of high density in the input space. EWKM (entropy weighting K -means) algorithm [38] is a new K -means type subspace clustering algorithm for high-dimensional sparse data. This algorithm simultaneously minimizes the within cluster dispersion and maximizes the negative weight entropy in the clustering process.

3. Multiobjective Subspace Clustering

A genetic algorithm, a particular class of evolutionary algorithms, has been recognized to be well suited to multi-objective optimization problems. In our work, we employ multi-objective subspace clustering (MOSCL) algorithm for clustering data sets based on subspace approach. In this section, we discuss important concepts of preprocessing phase and design concepts of MOSCL.

3.1. Preprocessing Phase. The goal of preprocessing step is to identify all dimensions in a data set which exhibit some cluster structure by discovering dense regions and their location in each dimension [39]. By cluster structure we mean a region that has a higher density of points than its surrounding regions. Such dense region represents the one-dimensional projection of some cluster. Hence, it is clear that, by detecting dense regions in each dimension, we are able to discriminate between dimensions that are relevant to clusters and irrelevant ones. The identified dimensions represent potential candidates for relevant dimensions of the subspace clusters. The irrelevant attributes contain noise/outliers and sparse data points [23].

Let us first give some definitions. Let D be a data set of n data points of dimensionality d . Let $A = \{A_1, A_2, \dots, A_d\}$ be the set of all attributes A_i of the data set D , and let $S = \{A_1 \times A_2 \times \dots \times A_d\}$ be the corresponding d -dimensional data space. Any k -dimensional subspace of $S \subseteq A$ is the space with the k dimensions drawn from the d attributes, where $k \leq d$. The cardinality of the subspace is defined as the number of dimensions forming this subspace. The input consists of a set of d dimensional points $D = \{x_1, x_2, \dots, x_n\}$, where $x_i = \{x_{i1}, x_{i2}, \dots, x_{id}\}$. The projection of a point $x_i \in D$

into a subspace $S \subseteq A$ is denoted by $\pi_S(x_i)$. The distance function between the data points D is denoted by dist . It is assumed that dist is one of the L_p -norms. The k -nearest neighbors of a data point x_i for any $k \in NN$ are denoted by $NN_k(x_i)$. More formally, the set of k -nearest neighbors of a data point x_i is the smallest set $NN_k(x_i) \subseteq D$ that contains at least k data points from data set D . In order to detect densely populated regions in each attribute we compute variance of the local neighborhood of each data point by measuring the variance of its k nearest neighbors [40]. The variance of local neighborhood of data point x_{ij} ($i = 1, \dots, n$ and $j = 1, \dots, d$) is defined as

$$\lambda_{ij} = \frac{\sum_{q \in p_j^i(x_{ij})} (q - C_i^j)^2}{k + 1}, \quad (2)$$

where $p_j^i(x_{ij}) = \{nn_k^j(x_{ij}) \cup x_{ij}\}$ and $|p_j^i(x_{ij})| = k + 1$. Here $nn_k^j(x_{ij})$ denotes the set of k -nearest neighbors of x_{ij} in attribute A_j and C_i^j is the center of the set $p_j^i(x_{ij})$; the center is calculated as

$$C_i^j = \frac{\sum_{q \in p_j^i(x_{ij})} q}{k + 1}. \quad (3)$$

A large value of λ_{ij} means that data point x_{ij} belongs to a sparse region, while a small one indicates that it belongs to a dense region. Calculation of the k nearest neighbors is an expensive task, especially when the number of data points n is very large. We can search the k nearest neighbors in an efficient way by presorting the values in each attribute and limiting the number of distance comparisons to a maximum of $2k$ values. The major advantage of using the variance degree is that it provides a relative measure on which the dense regions are more easily distinguishable from sparse regions. On the other hand, when a dimension contains only sparse regions, all the estimated λ_{ij} for the same dimension tend to be very large. Our objective now is to determine whether or not dense regions are present in a dimension. In order to identify dense regions in each dimension, we are interested in all sets of x_{ij} having a small variance degree by a predefined density threshold $\varepsilon \in \mathbb{R}$. Therefore, setting $0 < \varepsilon \leq 0.1$ is a reasonable choice. We have applied this density threshold for estimation of binary weight z_{ij} of data point x_{ij} . If $\lambda_{ij} < \varepsilon$, then $z_{ij} = 1$ and x_{ij} belongs to a dense region; else $z_{ij} = 0$ and x_{ij} belongs to a sparse region. We obtain a binary matrix $Z_{(n*d)}$ which contains the information on whether each data point falls in a dense region of an attribute. Table 1 shows a binary matrix $Z_{(n*d)}$ which contains the information on whether each data point falls in a dense region of an attribute or sparse region. It is clear that the computation of z_{ij} depends on the input parameters k and ε threshold. We set the value of variance $\lambda_{ij} \leq 0.1$, which indicates that data point x_{ij} belongs to a dense region. In practice, since the variance degree λ_{ij} is the indicator for dense regions and its values vary significantly depending on the attributes, first we normalize all the λ_{ij} for each attribute A_j by mapping them onto the interval $[0, 1]$ before applying the threshold. In fact, the role of the parameter k is intuitively easy to understand and it can

TABLE 1: Binary weight matrix Z of data points.

Data points	Attributes									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
x_1	1	0	1	0	0	1	0	1	0	1
x_2	1	0	1	0	0	1	0	1	0	1
x_3	1	0	1	0	0	1	0	1	0	1
x_4	0	1	0	1	0	0	0	1	0	1
x_5	0	0	0	0	0	0	0	0	0	0
x_6	1	0	1	1	0	1	0	1	1	0
x_7	1	0	1	1	0	1	0	1	1	0
x_8	1	0	1	1	0	1	0	1	1	0
x_9	0	1	1	0	0	1	1	1	0	1
x_{10}	0	1	1	0	0	1	1	1	0	1

be set by the variance degrees λ_{ij} which are not meaningful enough because few neighbors would have a distance close to zero.

Obviously, the parameter k is related to the expected minimum cluster size and should be much smaller than the number of objects n in the data. To gain a clear idea of the variance of the neighborhood of a point, we have chosen $k \leq \sqrt{n}$ in this phase. In order to capture irrelevant attributes and outliers, we used binary similarity coefficients to measure the similarity between binary data points z_i for ($i = 1, \dots, n$) in the matrix Z . One commonly used similarity measure for binary data is the Jaccard coefficient [41]. This measure is defined as the number of variables that are coded as 1 for both states divided by the number of variables that are coded as 1 for either or both states. In preprocessing phase, we require a similarity measure that can reflect the degree of overlap between the binary data points z_i in the identified dense regions in the matrix Z . Since dense regions are encoded by 1 in the matrix Z , we believe that the Jaccard coefficient is suitable for our task because it considers only matches on 1's to be important. The Jaccard coefficient is given as

$$\text{sim}(z_1, z_2) = \frac{a}{a + b + c}, \quad (4)$$

where a indicates the number of dimensions in which the two objects have the same binary value of 1. Similarly, c and b count the number of dimensions in which the two objects have different binary values. The Jaccard coefficient [41] has values between 0 and 1. A pair of points is considered similar if the estimated Jaccard coefficient between them exceeds a certain threshold α . We set the value of threshold α equal to 0.8, for measuring degree of similarity between two binary vectors in preprocessing phase. On the other hand, it is clear that when all of the binary weights for a binary data point z_i in the matrix Z are equal to zero, the related data point x_i is systematically considered as an outlier because it does not belong to any of the discovered dense regions. The identified outliers $|o|$ are discarded from the data set and their corresponding rows are eliminated from the matrix. By eliminating outliers the size of the data set D is reduced with size $N_d = n - |o|$ and now its new associated matrix of binary weights $U_{(N_d*d)}$ (see Table 2). MOSCL uses

TABLE 2: Binary weight matrix U of data points after removing outliers.

Data points	Attributes									
	A_1	A_2	A_3	A_4	A_6	A_7	A_8	A_9	A_{10}	
x_1	1	0	1	0	1	0	1	0	1	
x_2	1	0	1	0	1	0	1	0	1	
x_3	1	0	1	0	1	0	1	0	1	
x_4	0	1	0	1	0	0	1	0	1	
x_6	1	0	1	1	1	0	1	1	0	
x_7	1	0	1	1	1	0	1	1	0	
x_8	1	0	1	1	1	0	1	1	0	
x_9	0	1	1	0	1	1	1	0	1	
x_{10}	0	1	1	0	1	1	1	0	1	

these binary weights of the matrix U for representation of chromosomes. MOSCL uses a modified distance function that considers contributions only from relevant dimensions when computing the distance between a data point and the cluster center. In concrete terms, we associate the binary weights u_{ij} $\{i = 1, \dots, N_d; j = 1, \dots, d\}$ in matrix U which the Euclidean distance.

This makes the distance measure more effective because the computation of distance is restricted to subsets where the data point values are dense. Formally, the distance between a point x_i and the cluster center v_s $\{s = 1, 2, \dots, N_c\}$ is defined as

$$\text{dis}(x_i, v_s) = \sqrt{\sum_{j=1}^d u_{ij} (x_{ij} - v_{sj})^2}. \quad (5)$$

3.2. Multiobjective Subspace Clustering Algorithm. In this subsection, we will dwell on the important design issues of MOSCL, including individual representation, fitness function, selection operator, search operator, and elitism. The basic steps of MOSCL are presented in Procedure 1.

3.2.1. Representation. For any genetic algorithm, a chromosome representation is needed to describe each individual in the population of interest [42]. The representation method determines how the problem is structured in the algorithm and the genetic operators that are used. Each chromosome is made up of a sequence of genes from certain alphabet. An alphabet can consist of binary digits (0 and 1), floating-point numbers, integers, and symbols [43]. A straightforward yet effective representation scheme for subspaces is the standard binary encoding; all individuals are represented by binary strings with fixed and equal length θ , where θ is the number of dimensions of the dataset. Using a binary alphabet set $B = \{0, 1\}$ for gene alleles, each bit in the individual will take on the value of “0” and “1”, respectively, indicating whether or not its corresponding attribute is selected (“0” indicates the corresponding condition is absent and vice versa for “1”). As a simple example, the individual 110101 when the dimension of data stream $\theta = 6$ represents a 4-dimensional subspace cluster containing the 1st, 2nd, 4th, and 6th attributes of the dataset. In our method, each chromosome is described by a sequence

of matrix $M = d \times S_c$, where d is the dimension of the feature space and S_c is the number of subspace clusters described by the chromosome. That is to say, the chromosome of the algorithm is written as when the first d values represent the weights of d dimensions of the first subspace cluster, the next d points represent the weights of second subspace cluster and so on.

3.2.2. Objective Functions. In the single-objective optimization problem, the objective function and fitness function are often identical and are usually used interchangeably [43]. Hence, quality evaluation of individuals in single-objective optimization problems can be conducted directly based on their objective function values. Single objective methods of data clustering can be categorized based on the criterion they optimize [44]. One group of clustering algorithms try to form compact spherical clusters by minimizing the intracluster distance between data points and cluster representatives, as in K -means [12] average link agglomerative clustering [13] and self-organizing maps [45]. Another group of clustering algorithms tries to form clusters in which data items close to each other fall into the same cluster, hence optimizing connectedness. This category of clustering algorithms can find clusters of arbitrary shapes; however, they might fail when clusters are close to each other. Two examples of this group of clustering algorithms are the single link agglomerative clustering, since clustering algorithms with single criterion can find one category of cluster shapes and fail on the other category. In contrast, the objective function and fitness function differ in genetic multiobjective optimization. The objective function f , by applying the decision vector R , produces a k -dimensional objective vector S . MOSCL uses a multiobjective genetic algorithm during fitness evaluation including multi objective functions for finding subspace clusters. First objective measures the density of each subspace cluster C_s . On the other hand minimizing compactness tries to cluster dataset in as many subspace clusters as possible. Therefore the two objectives, density and compactness, can interact towards making the semioptimal clustering solutions. The mathematical definitions of the objectives of MOSCL are as follows.

Let $A = \{A_1, \dots, A_d\}$ be set of d dimensional attributes. We define the density of a single attribute A_i as the fraction of “1” entries in the column A_i , denoted as $\text{dens}(A_i)$. The density of a subspace cluster C_s denoted as $\text{dens}(C_s)$ is the ratio between the number of data points contained in the subspace cluster $|C_s|$ and the total number of data points in the dataset $|D|$. We propose a weighted density measure, which finds density of all subspace clusters. The weighted density of a subspace cluster C_s denoted as $\text{dens}_w(C_s)$ is defined as the ratio between $\text{dens}(C_s)$ and the average density over all attributes contained in A ; that is, $\text{dens}_w(C_s) = \text{dens}(C_s) / ((1/|A|)(\sum_{A_i \in A} \text{dens}(A_i)))$, where $|A|$ is the number of attributes contained in subspace cluster. We call the denominator $(1/|A|)(\sum_{A_i \in A} \text{dens}(A_i))$ weight. The second objective is to minimize intracluster variance or compactness. To measure the compactness between subspaces clustering, we use the Euclidean distance in weighted form as the distance measure. In order to define weighted Euclidean distance

```

Procedure: Multi Objective Subspace Clustering algorithm
Begin
Apply preprocessing phase on data set  $D$ 
 $S_{\text{pop}}$  = initial population of relevant subspaces /* Popsiz =  $|P| * /$ 
While (the termination criterion is not satisfied)
  for  $i = 1$  to  $|P|$  // for each individual in subspace  $S_{\text{pop}}$ 
    Randomly select chromosomes from the population
    Call subspace_update ( )
    Compute objective functions for current chromosomes
    Apply crossover operation with probability  $p_c$ 
    Apply mutation operator with mutation probability  $p_m$ 
    Compute objective functions for new offsprings
  end for
end while
Select the best solution from population
End

Procedure subspace_update ( )
  Choose cluster centers  $\{s = 1, \dots, N_c\}$  from data set generated
  by preprocessing phase

  Repeat
    Compute the membership matrix  $M_{(N_d * N_c)}$ 

  for  $i = 1$  to  $N_d$  do
    for  $j = 1$  to  $N_c$  do //  $N_c$ —number of cluster centers
      if  $\text{dist}(x_i, v_s) < \text{dist}(x_i, v_j)$  then
         $m_{ij} = 0$ ;
      else
         $m_{ij} = 1$ ;
      end if
    end for
  end for

  Compute the cluster center:
  
$$v_s^1 = \frac{\sum_{i=1}^{N_d} (m_{ij} \times u_i \times x_i)}{\sum_{i=1}^{N_d} m_{ij}} \quad (j = 1, \dots, N_c);$$


  Compute cluster weights  $u_{ij}$  encoded in each chromosome using
  membership matrix  $M_{(N_d * N_c)}$  and cluster center  $v_s^1$ 
  until convergence

```

PROCEDURE 1: Steps of MOSCL algorithm.

for subspace clustering, the data points x_i and v_s can be thought of as the projections $\pi_S(x_i)$ and $\pi_S(v_s)$ in their own subspace S . The equation of compactness is

$$\begin{aligned}
 \text{S_Comp}(C_s) &= \frac{1}{|C_s|} \sum_{x_i \in C_s} \|\pi_S(x_i) - \pi_S(v_s)\|^2 \\
 &= \frac{1}{|C_s|} \sum_{x_i \in C_s} \sum_{j=1}^d u_{ij} (x_{ij} - v_{sj})^2,
 \end{aligned} \tag{6}$$

where C_s is the subspace cluster and u_{ij} is binary weights associated with each attribute. This makes the distance measure more effective because the computation of distance is restricted to subsets where the object values are dense. Thus

by minimizing this objective we give higher fitness to compact subspace clusters.

3.2.3. Selection. The selection operator supports the choice of the individuals from a population that will be allowed to mate in order to create a new generation of individuals. Genetic algorithm methods attempt to develop fitness methods and elitism rules that find a set of optimal values quickly and reliably [46]. But, ultimately, it is the selection method that is responsible for the choice of a new generation of parents. Thus, the selection process is responsible for making the mapping from the fitness values of the input population to a probability of selection and thus the probability of an individual's genetic material being passed to the next generation. Here we adopt the tournament selection method [47] because its time complexity is low. The basic concept of the tournament

```

begin
  for  $i = 1$  to  $P_{\text{pop}}$  do
     $P'$  best fitted item among  $N$  tour elements randomly
    selected from  $P$ ;
  return  $P'$ 
end

```

PROCEDURE 2: Procedure of tournament selection method.

method is as follows: randomly select a positive number N tour of chromosomes from the population and copy the best fitted item from them into an intermediate population. The process is repeated P times, and here P is the population size. The algorithm of tournament selection is shown in Procedure 2.

3.2.4. Crossover. Crossover is the feature of genetic algorithms that distinguishes it from other optimization techniques. As with other optimization techniques genetic algorithms must calculate a fitness value, select individuals for permutation, and use mutation to prevent convergence on local maxima. But, only genetic algorithms use crossover to take some attributes from one parent and other attributes from a second parent. We used the uniform crossover [48] in the proposed algorithm. The uniform crossover is applied to the genes of the chromosome, simultaneously. Two chromosomes are randomly selected as parents from the current population. The crossover creates the offspring chromosome on a bitwise basis, copying each allele from each parent with a probability p_i . The p_i is a random real number uniformly distributed in the interval $[0, 1]$. Let P_1 and P_2 be two parents, and let C_1 and C_2 be offspring chromosomes; the i_{th} allele in each offspring is defined as.

$$\begin{aligned} C_1(i) &= P_1(i), & C_2(i) &= P_2(i) & \text{if } p_i \geq 0.5, \\ C_1(i) &= P_2(i), & C_2(i) &= P_1(i) & \text{if } p_i < 0.5. \end{aligned} \quad (7)$$

For example chromosome P_i and chromosome P_j are selected for uniform crossover as follows:

$$\begin{aligned} P_i &= 101010, \\ P_j &= 010101. \end{aligned} \quad (8)$$

The random number p_i is generated for crossover $[0.3, 0.7, 0.80.4, 0.2, 0.5]$. Now copy each allele from each parent with a probability p_i as discussed in the uniform crossover operation. The resulting offsprings are

$$\begin{aligned} C_i &= 001100, \\ C_j &= 110011. \end{aligned} \quad (9)$$

3.2.5. Mutation. Unlike the crossover and selection operators, mutation is not necessarily applied to all individuals [49]. Multiple mutation specifications are available: bit, uniform, and normal. Each mutation type has an independent

probability of being applied to each individual. Therefore, some individuals may be mutated by several mutation methods, some individuals may be mutated by a single mutation method, and some individuals may not be mutated at all. Here we apply bit mutation to the genes [48]. This results in some bits in genes of the children being reversed: “1” is changed to “0” and “0” is changed to “1”. The gene values are modified as follows:

$$\hat{p} = p \pm N(0, \sigma) \times p, \quad (10)$$

where \hat{p} is the mutated gene, p is the existing gene value, and $N(0, \sigma)$ is a normally distributed random number with mean 0 and standard deviation (σ). After some trial and error procedures, we find that when (σ) is equal to 0.1.

3.2.6. Elitism. When creating a new population by crossover and mutation, we have a high chance to lose the best subspace clusters found in the previous generation due to the nature of randomness of evolution [50]. Hence, besides Pareto-based selection method, we also use elitism method to achieve a constantly improving population of subspace clusters. We adopt the archive fashion to implement elitism in MOSCL. In our elitism strategy, the best or a few best subspaces are directly copied to the new population, together with other new individuals generated from the mating pool. In addition, all the solutions obtained in MOSCL in different generations will be stored in an external archive. In the current implementation of MOSCL, there is no limit on the archive that is used to store all the solutions that are obtained in the MOSCL.

4. Experimental Evaluation

The experiments reported in this section used a 2.0 GHz core 2 duo processors with 2 GB of memory. After data preparation has been finished, we can conduct an experimental evaluation on MOSCL and compare the performance of MOSCL with other competitive methods. We use both synthetic and real-life datasets for performance evaluation in our experiments. There are four major parameters that are used in the MOSCL, that is, the total number of generations that are to be performed in the MOSCL (denoted by n_g), the number of individuals in the population of each generation (denoted by n_p), the probability that the crossover between two selected individuals will be performed (denoted by p_c), and finally the probability that each bit of the individual will be mutated (denoted by p_m). The typical parameter setup in MOSCL is $n_g = 100$, $n_p = 50$, $p_c = 0.8$, and $p_m = 0.1$. One can

change the value specification of these parameters in order to obtain different search workloads and search strengths for the MOSCL.

4.1. Performance Measure. The performance measure used in MOSCL is the clustering error (CE) distance [24] for subspace clustering. It is an intuitive way to compare clustering because it uses the proportion of points which are clustered differently in generated subspace clusters and real subspace clusters after optimal matching of clusters. In other words, it is the scaled sum of the nondiagonal elements of the confusion matrix, minimized over all possible permutations of rows and columns. The CE distance has been shown to be the most desirable metric for measuring agreement between partitions in the context of projected/subspace clustering [24].

Let us have two subspace clusters: $GS = \{C_1, C_2, C_3, C_{N_c}\}$ is a collection of experimental generated projected subspace clusters and $RS = \{C'_1, C'_2, C'_3, \dots, C'_{N_c}\}$ is a collection of real subspace clusters. In order to construct a clustering error distance measure for subspace clusters, we need to define the union, and the intersection of subspace clusters in GS and RS. We denote the union of two subspace clusters GS and RS as

$$\cup = \cup(GS, RS) = \text{sup}(GS) \cup \text{sup}(RS), \quad (11)$$

where $\text{sup}(GS)$ and $\text{sup}(RS)$ are the support of subspace clustering GS and RS. The support of RS is defined as $\text{sup}(GS) = \cup_{s=1, \dots, n} \text{sup}(C_s)$, where $\text{sup}(C_s)$ is support of subspace cluster C_s , given by $\text{sup}(C_s) = \{x_{ij} \mid x_i \in DS \wedge A_j \in AS\}$. The DS is a subset of data points of D , and AS is a subset of dimensions of A . Let $M = (m_{ij})$ denote confusion matrix, in which m_{ij} represents number of data points in the intersection of subspace clusters C_i and C_j ; we can represent m_{ij} as

$$m_{ij} = |\text{sup}(C_i) \cap \text{sup}(C_j)|. \quad (12)$$

We use the Hungarian method [51] to find a permutation of the cluster labels such that the sum of the diagonal elements of M is maximized. D_{\max} denotes this maximized sum. The clustering error (CE) distance for subspace clustering is given by

$$\text{CE}(GS, RS) = \frac{|U| - D_{\max}}{|U|}. \quad (13)$$

The value of cluster error (CE) is always between 0 and 1. The more similar the two partitions RS and GS, the smaller the CE value.

4.2. Data Sets

4.2.1. Synthetic Data Sets. The synthetic data sets were generated by data generator [23]. It permits controlling the size and structure of the generated data sets through parameters such as number and dimensionality of subspace clusters, dimensionality of the feature space, and density parameters for the whole data set as well as for each cluster. The parameters used in the synthetic data generation are the size of the data set

n , the number of clusters N_c , the data set dimensionality d , the average cluster dimensionality d_{avg} , the domain of the values of each dimension $[\min_j, \max_j]$, the standard deviation value range $[\text{sd}_{\min}, \text{sd}_{\max}]$, which is related to the distribution of points in each cluster, and the outlier percentage. Using these parameters, the data generator proceeds by defining seed points among which the clusters will be distributed, as well as the dimensions associated with each such seed point [23]. Then, it determines how many points will be associated with each cluster and finally it generates the cluster points. Projected values of cluster points are generated according to the normal distribution in their relevant dimension, with the mean randomly chosen from $[\min_j, \max_j]$ and the standard deviation value from $[\text{sdv}_{\min}, \text{sdv}_{\max}]$. The experiments were analyzed by using 10 data sets with $n = 1000$ to 50000, number of dimensions $d = 20$ to 250, $\min_j = -100$, and $\max_j = 100$. The values of sd_{\min} and sd_{\max} were set to 1 percent and 5 percent of the standard deviation on that dimension, respectively.

4.2.2. Real Data Sets. We also tested our proposed algorithm on the breast cancer, the mushroom data sets, and multiple features data (MF). All three data sets are available from the UCI machine learning repository [<http://archive.ics.uci.edu/ml/>]. The cancer data set contains 569 instances with 30 features. The mushroom data set contains 8124 instances and 22 attributes. The multiple features data (MF) consists of features of handwritten numerals ("0"–"9") extracted from a collection of Dutch utility maps. Two hundred patterns per cluster (for a total of 2,000 patterns) have been digitized in binary images. These digits are represented in terms of the following six feature sets (files):

- (1) mfeat-fou: 76 Fourier coefficients of the character shapes;
- (2) mfeat-fac: 216 profile correlations;
- (3) mfeat-kar: 64 Karhunen-Love coefficients;
- (4) mfeat-pix: 240 pixel averages in 2×3 windows;
- (5) mfeat-zer: 47 Zernike moments;
- (6) mfeat-mor: 6 morphological features.

For experimental results we used five feature sets (mfeat-fou, mfeat-fac, mfeat-kar, mfeat-zer, and mfeat-mor). All the values in each feature were standardized to mean 0 and variance 1.

4.3. MOSCL Results Analysis. The scalability of MOSCL with increasing data set size and dimensionality are discussed in this section. In all of the following experiments, the performance of the results returned by MOSCL is better in comparison with PROCLUS.

4.3.1. Scalability with the Data Set Size. The scalability of MOSCL with the size of the data set is depicted in Figure 2. The results for scaling by the data size n verify that the algorithm is approximately linear in n . The experiments were obtained by using data set varying the number of data points

from 1000 of 50000. The dimensionality value of the data set is set to 20. The data points were grouped in 5 clusters with 10 percentages of data points generated as outliers. For this set of experiments, the distributions of points for the bounded dimensions of all clusters follow a uniform distribution. MOSCL successfully locates all input clusters within the course of 40 generations, on average. The execution time of MOSCL is comparable to that of PROCLUS [23]. The slight nonlinearity comes from the final stage of extracting the points from the clustered dimensions.

4.3.2. Scalability with Dimensionality of the Data Set. Figure 3 shows the scalability of MOSCL as the dimensionality of the data set increases from 50 to 250. In this series of experiments, each dataset has 5000 data points and there are 5 clusters being embedded in some 5-dimensional subspace. Additionally, as the data dimensionality increases, the application of the genetic operators becomes increasingly more expensive mainly due to the increasing number of constraint-checking computations which require yielding individuals without overlapping candidate solutions. As in the scalability experiments with the dimensionality of data set, the execution of MOSCL is usually better than that of PROCLUS [23]. The result shows a linear increase of runtime when increasing the dimensionality of the data set.

4.3.3. Scalability with Average Cluster Dimensionality. Figure 4 shows the dependency of the execution time on the average cluster dimensionality where the latter increases from 10 to 50 in a 100-dimensional data space. In each case, the dataset has 50000 points distributed over 5 clusters with a fixed 10 percent level of outliers. The superlinear speedup in the execution time with the dimensionality of the hidden clusters is explained as follows: the higher the dimensionality of the hidden clusters is, the more likely it is for the evolutionary search operators to produce chromosomes along the bounded dimensions. It required extra operations to obtain a feasible chromosome. Additionally, as the dimensionality of hidden clusters increases, more space becomes available for the mutation and crossover operators. The higher average cluster dimensionality increased the computational overhead by genetic operators.

The scalability experiments show that MOSCL scales well also for large and high-dimensional data sets.

4.3.4. Results on Real Data Sets. The quality of the clustering result is evaluated in terms of accuracy. The accuracy achieved by the PROCLUS algorithm on breast cancer data is 71.0 percent, which is less than 94.5 percent achieved by MOSCL clustering algorithm. The enhanced performance given by MOSCL in comparison with that of the PROCLUS suggests that some dimensions are likely not relevant to some clusters in this data. The accuracy achieved by MOSCL on mushroom data set is 93.6 percent, which is greater than the accuracy (84.5 percent) achieved by PROCLUS on the same data. PROCLUS and MOSCL are used to cluster the MF data. The accuracy achieved by these algorithms on this data is 94.5 percent and 83.5 percent, respectively. Such results

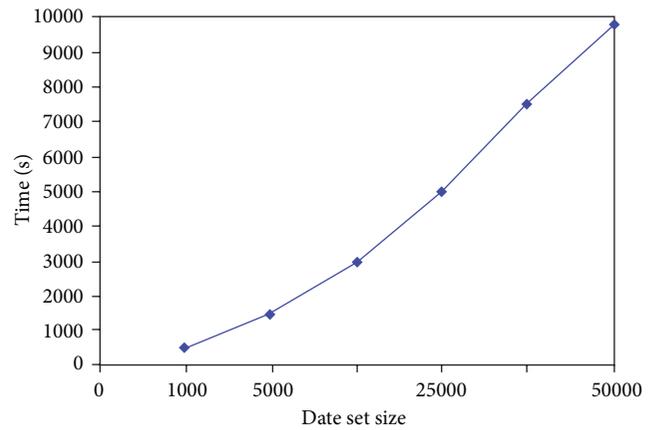


FIGURE 2: Scalability with the size of the data set.

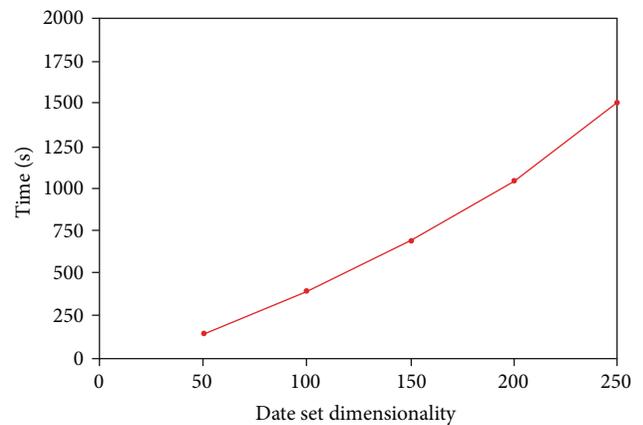


FIGURE 3: Scalability with the dimensionality of the data set.

suggest that the moderate number of dimensions of this data does not have a major impact on algorithms that consider all dimensions in the clustering process. MOSCL is executed for a total of 50 generations. The experimental results show the superiority of MOSCL algorithm in comparison to PROCLUS. The proposed algorithm MOSCL detects high quality subspace clusters on all real data sets but PROCLUS is unable to detect them. The accuracy results of the real data sets are presented in Table 3.

4.3.5. Performance Measure of Clustering Results. We have computed the CE distance for all clustering results using the subspace clustering CE distance. MOSCL is able to achieve highly accurate results and its performance is generally consistent. As we can see from Figure 5, MOSCL is more robust to cluster dimensionality than the PROCLUS algorithm. The experiments show that our algorithm is able to detect clusters and their relevant dimensions accurately in various situations. MOSCL successfully avoids the selection of irrelevant dimensions in all the data sets used in these experiments. The results of PROCLUS are less accurate than those given by MOSCL. When data sets contain projected clusters of low dimensionality, PROCLUS performs poorly.

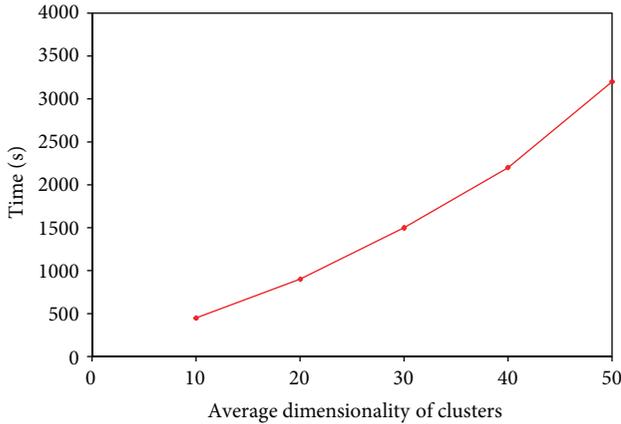


FIGURE 4: Scalability with average dimensionality of clusters.

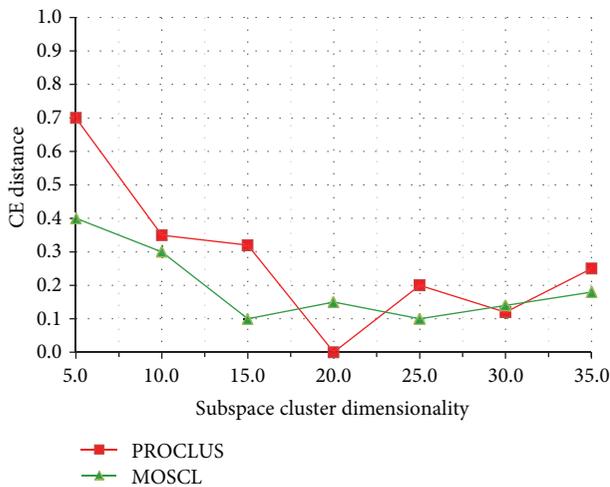


FIGURE 5: Distances between MOSCL output and the true clustering.

TABLE 3: Average accuracy results on real data sets.

Data set	PROCLUS	MOSCL
Cancer	71.5	94.5
Mushroom	84.5	93.6
Multiple features	83.5	94.5

5. Conclusion

In this paper we have presented a robust MOSCL (multi-objective subspace clustering) algorithm for the challenging problem of high-dimensional clustering and illustrated the suitability of our algorithm in tests and comparisons with previous work. The first phase of MOSCL performs subspace relevance analysis by detecting dense and sparse regions and their locations in data set. After detection of dense regions it eliminates outliers. MOSCL discovers subspaces in dense regions of data set and produces subspace clusters. The discussion details key aspects of the proposed methodology, MOSCL, including representation scheme, maximization fitness functions, and novel genetic operators. Our experiments

on large and high dimensional synthetic and real world data sets show that MOSCL outperforms subspace clustering as PROCLUS by orders of magnitude. Moreover, our algorithm yields accurate results when handling data with outliers. There are still many obvious directions to be explored in the future. The interesting behavior of MOSCL on generated data sets with low dimensionality suggests that our approach can be used to extract useful information from gene expression data that usually have a high level of background noise. Another interesting direction to explore is to extend the scope of preprocessing phase of MOSCL from attribute relevance analysis to attribute relevance.

References

- [1] L. C. Hamilton, *Modern Data Analysis: A First Course in Applied Statistics*, Brooks/Cole, 1990.
- [2] D. H. Fisher, "Iterative optimization and simplification of hierarchical clustering," in *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD '95)*, pp. 118–123, Montreal, Canada, 1995.
- [3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [4] N. Kumar and K. Kummamuru, "Semisupervised clustering with metric learning using relative comparisons," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 4, pp. 496–503, 2008.
- [5] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice Hall, 2003.
- [6] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. C. Tseng, "Evaluation and comparison of gene clustering methods in microarray analysis," *Bioinformatics*, vol. 22, no. 19, pp. 2405–2412, 2006.
- [7] K. Wang, Z. Du, Y. Chen, and S. Li, "V3COCA: an effective clustering algorithm for complicated objects and its application in breast cancer research and diagnosis," *Simulation Modelling Practice and Theory*, vol. 17, no. 2, pp. 454–470, 2009.
- [8] J. Sander, M. Ester, H. Kriegel, and X. Xu, "Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 169–194, 1998.
- [9] B. R. S. A. Young, *Efficient algorithms for data mining with federated databases [Ph.D. thesis]*, University of Cincinnati, 2007.
- [10] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: a wavelet-based clustering approach for spatial data in very large databases," *Journal of Very Large Data Bases*, vol. 8, no. 3-4, pp. 289–304, 2000.
- [11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2011.
- [12] J. McQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics*, vol. 1, pp. 281–297, 1967.
- [13] E. M. Voorhees, "Implementing agglomerative hierarchic clustering algorithms for use in document retrieval," *Information Processing and Management*, vol. 22, no. 6, pp. 465–576, 1986.
- [14] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Proceedings of the ACM SIGMOD International Conference on Management*

- of *Data of ACM SIGMOD Record*, vol. 25, no. 2, pp. 103–114, June 1996.
- [15] S. Guha and R. Rastogi, “CURE: an efficient clustering algorithm for large databases,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 27, no. 2, pp. 73–84, ACM Press, 1998.
 - [16] F. Korn, B. Pagel, and C. Faloutsos, “On the “dimensionality curse” and the ‘self-similarity blessing–,’” *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 96–111, 2001.
 - [17] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is nearest neighbors meaningful,” in *Proceedings of the 7th International Conference on Database Theory (ICDT ’99)*, pp. 217–235, London, UK, 1999.
 - [18] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, UK, 1986.
 - [19] M. Ester, H. P. Kriegel, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD ’96)*, pp. 226–223, Portland, Ore, USA, 1996.
 - [20] A. Hinneburg and H. Gabriel, “DENCLUE 2. 0: fast clustering based on kernel density estimation,” in *Advances in Intelligent Data Analysis VII*, pp. 70–80, Springer, Berlin, Germany, 2007.
 - [21] A. H. Pilevar and M. Sukumar, “GCHL: a grid-clustering algorithm for high-dimensional very large spatial data bases,” *Pattern Recognition Letters*, vol. 26, no. 7, pp. 999–1010, 2005.
 - [22] W. Wang, J. Yang, and R. Muntz, “STING: a statistical information grid approach to spatial data mining,” in *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB ’97)*, vol. 97, pp. 186–195, 1997.
 - [23] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, and J. S. Park, “Fast algorithms for projected clustering,” in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, vol. 28, no. 2, pp. 61–72, 1999.
 - [24] A. Patrikainen and M. Meila, “Comparing subspace clusterings,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 7, pp. 902–916, 2006.
 - [25] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, “Automatic subspace clustering of high dimensional data for data mining applications,” in *Proceedings of the ACM-SIGMOD Conference on the Management of Data*, vol. 27, no. 2, pp. 94–105, 1998.
 - [26] K. Kailing, H. P. Kriegel, and P. Kroger, “Density-connected subspace clustering for high-dimensional data,” in *Proceedings 4th SIAM International Conference on Data Mining (SDM ’04)*, p. 4, 2004.
 - [27] C. C. Aggarwal and P. S. Yu, “Finding generalized projected clusters in high dimensional spaces,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, pp. 70–81, 2000.
 - [28] H. Kriegel, P. Kröger, M. Renz, and S. Wurst, “A generic framework for efficient subspace clustering of high-dimensional data,” in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM ’05)*, pp. 250–257, November 2005.
 - [29] M. J. Zaki, M. Peters, I. Assent, and T. Seidl, “CLICKS: an effective algorithm for mining subspace clusters in categorical datasets,” *Data and Knowledge Engineering*, vol. 60, no. 1, pp. 51–70, 2007.
 - [30] J. H. Friedman and J. J. Meulman, “Clustering objects on subsets of attributes,” *Journal of the Royal Statistical Society B*, vol. 66, no. 4, pp. 815–839, 2004.
 - [31] K. Woo, J. Lee, M. Kim, and Y. Lee, “FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting,” *Information and Software Technology*, vol. 46, no. 4, pp. 255–271, 2004.
 - [32] Y. Chu, J. Huang, K. Chuang, D. Yang, and M. Chen, “Density conscious subspace clustering for high-dimensional data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 16–30, 2010.
 - [33] C. H. Cheng, A. W. Fu, C. H. Cheng, A. W. Fu, and Y. Zhang, “Entropy-based subspace clustering for mining numerical data,” in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 84–93, 1999.
 - [34] S. Goil, H. S. Nagesh, and A. Choudhary, “MAFIA: efficient and scalable subspace clustering for very large data sets,” in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 443–452, 1999.
 - [35] H. S. Nagesh, S. Goil, and A. Choudhary, “Adaptive grids for clustering massive data sets,” in *Proceedings of the 1st SIAM International Conference on Data Mining (SDM ’01)*, pp. 1–17, 2001.
 - [36] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, “A Monte Carlo algorithm for fast projective clustering,” in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 418–427, June 2002.
 - [37] B. L. Milenova and M. M. Campos, “O-Cluster: scalable clustering of large high dimensional data sets,” in *Oracle Data Mining Technologies*, pp. 290–297, Oracle Corporation, 2002.
 - [38] L. Jing, M. K. Ng, and J. Z. Huang, “An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1026–1041, 2007.
 - [39] E. K. K. Ng, A. W. Fu, and R. C. Wong, “Projective clustering by histograms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 369–383, 2005.
 - [40] A. Elke, C. Böhm, H. P. Kriegel, P. Kröger, I. M. Gorman, and A. Zimek, “Detection and visualization of subspace cluster hierarchies,” in *Advances in Databases: Concepts, Systems and Applications*, pp. 152–163, Springer, Berlin, Germany, 2007.
 - [41] S. Robert and P. H. A. Sneath, *Principles of Numerical Taxonomy*, W. H. Freeman, 1963.
 - [42] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
 - [43] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, “A survey of evolutionary algorithms for clustering,” *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 39, no. 2, pp. 133–155, 2009.
 - [44] J. Handl and J. Knowles, “An evolutionary approach to multiobjective clustering,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.
 - [45] T. Kohonen, *Self-Organizing Maps*, vol. 30, Springer, 2001.
 - [46] D. E. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” in *Foundations of Genetic Algorithms*, vol. 51, pp. 61801–62996, Urbana, 1991.
 - [47] J. E. Baker, *An analysis of the effects of selection in genetic algorithms [Ph.D. thesis]*, Graduate School of Vanderbilt University, Nashville, Tenn, USA, 1989.
 - [48] J. Yeh and J. C. Fu, “A hierarchical genetic algorithm for segmentation of multi-spectral human-brain MRI,” *Expert Systems with Applications*, vol. 34, no. 2, pp. 1285–1295, 2008.
 - [49] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

- [50] D. Thierens and D. Goldberg, "Elitist recombination: an integrated selection recombination GA," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence (ICEC '94)*, pp. 508–512, June 1994.
- [51] J. H. Friedman, L. B. Jon, and A. F. Raphael, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, 1977.




Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

