Expert Systems With Applications 135 (2019) 219–236

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Estimating distance threshold for greedy subspace clustering

Bhagyashri Abhay Kelkar[a,*], Sunil F. Rodd[b], Umakant P. Kulkarni[c]

[a] Department of Computer Science & Engineering, Sanjay Ghodawat University, Atigre, Kolhapur, Maharashtra 416118, India
[b] Department of Computer Science & Engineering, Gogte Institute of Technology, Belgaum, Karnataka 590008, India
[c] Department of Computer Science & Engineering, SDMCET, Dharwar, Karnataka 580002, India

ARTICLE INFO

Article history:
Received 4 July 2018
Revised 3 April 2019
Accepted 5 June 2019
Available online 6 June 2019

Keywords:
Greedy subspace clustering
Parameter estimation
Single linkage clustering

ABSTRACT

Many approaches have been proposed to recognize clusters in subspaces. However, their performance is highly sensitive to input parameter values. The purpose and expected ranges of these parameters may not available to a non-expert user. The parameter setting producing optimal results can only be known after repeated execution of the clustering process every time with a different set, which is very time consuming. Most of the existing algorithms show high runtimes due to excessive data scans. In this work, we propose a subspace clustering technique that estimates the distance threshold parameter automatically from the data for each attribute and works on the basis of single linkage clustering, in bottom up, greedy fashion. The experimental results show that, the algorithm produces optimal results without accepting any input from the user, achieves up to 10 times better runtime and improved accuracy in a single run without requiring any tuning of parameter values.

1. Introduction

Many recent applications capture huge volumes of data that is big in both directions, i.e. terms of objects and attributes. Comprehending such huge datasets is now beyond human capability and requires use of powerful and automated data mining tools. This wealth of data is of no use unless the knowledge embedded within it, is not uncovered by applying appropriate data mining algorithms. Clustering is an unsupervised data-mining task and no class labels are present in the data except for the test data wherein these labels can be used to verify quality of clustering. The attributes describing the objects are also called as dimensions or variables and the objects represent vertices in multi-dimensional space described by these attributes. In the literature, datasets having more than ten dimensions are called as high-dimensional data (Han, Kamber, & Pei, 2012). Conventional clustering algorithms show unusual and incorrect behavior on such datasets, the reason being they compute inter-cluster similarity based on full attribute space. Such distance calculations result in a dissimilarity value which is nearly equal for all object pairs. Thus all the data points are equally similar or dissimilar due to an effect called “Curse of Dimensionality" (Bellman, 1961) observed in high dimensional data spaces. Hence the underlying clustering algorithm cannot find clusters or natural grouping of the objects based on this similarity value and show inability to mine meaningful clusters in high-dimensional data spaces. To overcome this problem, many data mining algorithms use a preprocessing step of dimensionality reduction. But it has been observed that, such global attribute reduction methods preserve the effects of curse of dimensionality. Dimensionality reduction methods may also result into loss of useful knowledge that may be hidden in subspaces.

Subspace clustering has gained attention due to its extensive applications in many areas. It is useful where the group of objects show similarity over few attributes; in business management for product recommendations, in the field of human-machine interaction, bioinformatics, health surveillance systems, text mining, customer segmentation, computer vision (Vidal, 2011) and for disease diagnosis leading to better medical treatments. Subspace clustering algorithms aim to identify most relevant features locally instead of globally. A cluster in subspace is group of data items that show similarity over given subset of attributes. The subset of attributes justifies the reasons behind the object being grouped together. It is possible that a subset of objects can show similarity for multiple reasons. Hence it is essential to search for all such reasons, i.e. all subsets of attributes. Thus, subspace clustering is very useful to uncover multiple hidden concepts for each item in the data. However the search for manifold concepts per object may end up in many analogous clusters. Hence it is desirable that, the clustering output should be useful for the purpose it is generated and should not have much redundant clusters.

Although existing subspace clustering approaches seem to be effective in identifying subspace clusters, they show major

* Corresponding author.
E-mail addresses: kelkarba@gmail.com (B.A. Kelkar), sfroddgit@git.edu (S.F. Rodd).

https://doi.org/10.1016/j.eswa.2019.06.011
0957-4174/

limitations. First, their working is controlled by many parameters (Sim, Gopalkrishnan, Zimek, & Cong, 2013). Intuitively setting the right values of parameters that will result in high quality clustering requires multiple runs and is very much time consuming. Hence there is a need to reduce number of parameters required by a subspace clustering algorithm. Second, it has been observed that, clustering output is very much sensitive to the input parameter values and the outcome varies drastically with minor changes in these values. Hence there is a need to estimate optimal values of the parameters required by the algorithm automatically. Third, generally, high quality clustering output is paid with high execution time of the clustering process. Still in some algorithms for dimensionality > 25 meaningful results are not obtained within tolerable timeframe due to many database scans and runtimes even up to several days are shown by some algorithms (Müller, Günnemann, Assent, & Seidl, 2009b). Hence practical application of such algorithms on high dimensional datasets is infeasible. A subspace clustering algorithm must have to find the trade-off between output quality and runtime. Fourth, especially for the algorithms producing overlapping clusters, the output may contain redundant clusters with no useful and added information. This is an undesirable effect as it may lead to too many interpretations of the same data and the output rapidly becomes unmanageable. Hence only meaningful, significant and interesting subspace clusters must be retained in the output. What is significant in terms of subspace clusters should be first defined and then accordingly the clustering algorithm should mine these clusters directly.

The concept of clustering is subjective. For a single dataset, various cluster representations are possible. The cluster structures are driven by the algorithm employed and purpose of clustering. It means that, most of the times clustering algorithms are biased towards the task for which clustering is done. The clustering results reflect assumptions and interest of the application. Hence such algorithms show poor results when used in other application areas. To address these issues, in this paper, we propose a parameter light subspace clustering algorithm called as CLUSLINK. The algorithm estimates the most essential parameter required by clustering algorithm – distance threshold automatically from the input data. It employs greedy approach to reduce computational complexity and still achieves highly accurate results on synthetic and real datasets. The algorithm is highly scalable with increase in number of data items and number of dimensions. The algorithm prunes the subspace search space by eliminating the attributes which do not contain any 1-dimensional subspace clusters. The closeness of entities in a dimension is determined by using Euclidean distance as similarity measure and all points belonging to a cluster are within the distance threshold estimated from the data. The algorithm is able to remove redundant clusters and it outputs maximal clusters. In contrast to other approaches in this area, the algorithm does not require parameter tuning and in only one run it outputs high quality clusters. As an application of the method, the output subspace clusters can be used to identify prominent relationships and rules existing in the data. The extracted rules can be very useful for building corresponding classification models without user intervention.

2. Review of approaches for subspace clustering

The first subspace clustering method called Clustering-In-QUEst (CLIQUE) was proposed by Agrawal, Gehrke, Gunopulos, and Raghavan (1998). Thereafter research in subspace clustering got momentum due to manifold applications in various domains. Developments in this field have been reviewed in many surveys (Kriegel, Kröger, & Zimek, 2009; Parsons, Haque, & Liu, 2004; Sim et al., 2013). Muller et al. put forward the OpenSubspace framework (Müller et al., 2011) which addresses the need for an open source evaluation solution for subspace clustering. It incorporates performance measures to evaluate new solutions and visualization of the results in order to promote focused research.

The categorization of existing subspace clustering algorithms can be done based on strategies employed to find subspaces and methods used to identify embedded clusters. One such classification can be possible by examining the nature of output produced by the algorithms – non-overlapping clusters or overlapping clusters. In non-overlapping clustering an object can be part of only one cluster over a set of dimensions. Overlapping clusters contain objects that may be member of many subspace clusters simultaneously. These algorithms try to find every possible object grouping in every possible subset of dimensions and some of the objects may be common to many subspace clusters. If each dimension in the data is equally important for cluster definition, such methods are called Hard Subspace Clustering (HSC) algorithms. These algorithms find exact subspaces. On contrary, Soft Subspace Clustering (SSC) algorithms work by assigning a factor or weight to each attribute indicating the extent of its membership in a subspace cluster (Deng, Choi, Jiang, Wang, & Wang, 2016b). Thus every subspace cluster has different subsets of attributes having different weightages and every attribute mandatorily takes part in detection of clusters. For scalability, these algorithms apply a clustering technique similar to k-Means (MacQueen, 1967) and iteratively compute attribute weights. Recently, soft subspace clustering is gaining attention and hence, many such algorithms have been proposed (Deng et al., 2016b; Zhu, Cao, Yang, & Lei, 2014; Deng, Choi, Chung, & Wang, 2010; Zhu, Cao, & Yang, 2011).

Further categorization of subspace clustering algorithms can be based on the search methodology used to find subspaces. The search can be performed in top-down or bottom-up manner. Top-down methods start the process with all available attributes and then iteratively reduce them while searching for candidate subspaces. Hence these approaches are also called as subspace clustering with dimensionality-reduction. Due to the nature of processing, they create partitions of the data in given subspace. Proper parameter tuning is an important issue in these algorithms for obtaining meaningful results. Some of the parameters like size of subspaces and the count of clusters are generally very hard to estimate before hand, when the information about underlying data distribution is not known. The parameter - subspace size compels the top-down algorithms to output subspace clusters that are of fixed sized. When sampling is involved, sample size is another important parameter that plays a vital role in determining quality of the output. These algorithms have a major limitation that, they are unable to find even small number of hidden clusters without exploring every possible combination of subspaces and objects and hence result into exponential complexity. They are not scalable and show poor quality output after few tens of attributes.

Bottom-up algorithms determine the prospective subspaces starting with each dimension as a one-dimensional subspace. Then they try to find clusters in these one-dimensional subspaces by using a search method analogous to mining of frequent item-sets. The problem resembles frequent item-set mining because each attribute can be called an item and the corresponding subspace cluster can be called transaction of attributes. The frequent item-set mining problem has exponential search space but can be reduced by proper heuristics. CLIQUE and ENCLUS (Cheng, Fu, & Zhang, 2004) are some of the important algorithms based on this approach, which use a fixed grid size to split each attribute into equal sized bins. CBF (Chang, 2005), MAFIA (Goil, Nagesh, & Choudhary, 1999), DOC (Procopiuc, Jones, Agarwal, & Murali, 2002) are some other examples of bottom-up approach. The parameters required for bottom-up approaches are hard to set since improper values may result into one cluster mistakenly reported as two or more smaller clusters. Even if the parameters are tuned properly,

some small and important clusters may be missed by the pruning stage.

Based on orientation of output clusters in given *N*-dimensional space, the algorithms are broadly grouped into two categories - axis parallel and arbitrarily oriented approaches. The advantage of axis-parallel approach is that, instead of infinite subspace search, searching for subspaces containing clusters can be reduced to subspaces excluding irrelevant attributes. This characteristic is also useful in dimensionality reduction and data compression as the attributes having high variance must only be preserved at a higher precision level for all cluster members. In case of similarity search, the high variance variables should only be searched individually and index should be constructed accordingly. Research in this field could be focused in the direction of identifying relevant and irrelevant attributes and reducing the exponential search space by applying proper heuristics. Kriegel et al. (2009) point out that axis-parallel paradigm is a favored approach in many existing algorithms

The algorithms producing arbitrary oriented clusters do not restrict the clusters to be axis-parallel. A dimensionality technique called as Principal Components Analysis (PCA) (Sapatnekar, 2011) can be used over neighboring collection of data items to define new vectors which are the principal components, i.e. set of vectors that best defines selected subset of data. The major disadvantage of techniques that use Principal Components Analysis is that they show high runtimes. These techniques show scalability linear or at the most quadratic in terms of number of data points. But in terms of number of dimensions in the data set, usually they show cubic time complexity. Another drawback of PCA based dimensionality reduction is that the interpretation of subspace clusters is hard and less intuitive as original attributes are transformed to a completely new set of derived attributes. The advantage is that, they are capable of expressing complex negative and positive correlations existing between different attributes. This knowledge is most useful when the cluster subspaces are arbitrarily oriented in the *D*-dimensional space and the attributes have a complex dependency between them. The rules identified by this dependency characterize the clusters and are useful for interpretation of the data.

Projected clustering is another variant of subspace clustering which separates the input data into mutually disjoint clusters. The strict object partitioning creates only few, non-redundant output clusters. Hence these algorithms do not face efficiency problems. But these methods have disadvantage that they cannot identify all unseen concepts in various subspace projections. Disjoint clusters may result into loss of meaningful clusters during the partitioning phase. Thus projected clustering methods represent a relatively restricted clustering output. Subspace clustering approaches overcome this drawback at the cost of increase in complexity and efficiency.

Depending on the method used to identify closeness of instances, subspace clustering algorithms are divided into three major variants - density based, window based and grid based. In all these variants, the distance measurement is generally specified in terms of Euclidean distance. The subspace clustering algorithms based on density notions work on local optimization principle. They examine the local structure of the objects to identify denser areas. The dense areas are separated from each other by areas with low data density. As dense regions are not restricted to be axis-parallel as in the case of grid based methods, density based methods are able to find clusters having arbitrary shape. These algorithms operate with parameters that specify - the distance measure to be employed, radius of the neighborhood and density threshold and are sensitive to these parameters.

Cell-based methods first create a grid and find density of objects falling inside each grid cell. All cells which show density above predefined threshold are joined to form clusters. These methods use important parameters such as density threshold and size of grid that highly impact output quality. Improper parameter setting may result into poor quality of the output. The clusters are always axis parallel and have polygon shape. It has been observed that, for medium range dimensionality, the algorithms show better performance. However, after certain dimensionality, the count of cells increases drastically which degrades the performance, resulting into poor quality output and increased runtime. In such situations, density based methods perform better.

It is generally observed that the subspace clustering algorithms that attempt to improve quality show high execution times, because they require multiple scans through the database to get optimal results. In a comparative evaluation, Muller et al. mention that, when dimensionality of the input datasets is more than 25, some of the well known algorithms did not produce meaningful output within acceptable timeframe (Müller, Assent, Krieger, Günnemann, & Seidl, 2009a). These algorithms took even several days to produce some output (SUBCLU (Kailing, Kriegel, & Kröger, 2004) took 6 days to finish on an arbitrary data set). As many of the recent datasets have hundreds of dimensions, such high time complexity limits practical use of these algorithms. The authors also mention that a subspace clustering algorithm should maintain balance between computational efficiency and output quality by applying proper heuristics.

3. Parameter-free subspace clustering

The thought of parameter-free data mining was first suggested by Keogh et al. The authors empirically show that parameter-laden algorithms tend to overfit the results particularly in the case of anomaly detection. These algorithms achieve high accuracy on a test dataset whereas completely fail on other datasets of similar kind. As a first step towards mitigating these problems, to devise parameter-free data mining algorithms, the authors proposed a Compression-based Dissimilarity Measure (CDM) (Keogh, Lonardi, & Ratanamahatana, 2004) which can be applied to clustering, anomaly detection and classification. CDM is designed as a dissimilarity measure and it can be used to prune search spaces. The empirical evaluation shows that such a parameter-free algorithmic approach outperforms parameter-laden algorithms. The authors claim that CDM can be useful to solve wide variety of data mining problems working on wide variety of data types.

The selection of parameter values for getting optimal subspace clustering output is not trivial and requires a trial-and-error approach. It involves repeatedly conducting clustering task every time with different set of parameter values and then the set resulting in optimal output is selected as best parameter combination for the given dataset. That means, the process should be repeated for every new dataset. In Müller et al. (2009a) the authors mention that, for fair evaluation of various algorithms and to get optimal results, they tried on average 100 parameter settings per algorithm per dataset. The authors also report runtimes of several days for some algorithms. When it involves high dimensional datasets which may be complex in three directions – in terms of count of objects, count of dimension and count of clusters embedded, this method of deciding parameter values is practically infeasible. Hence it is desirable to have parameter-free or if not possible at least parameter-light subspace clustering solutions for high dimensional datasets. Such algorithms can be designed by utilizing the knowledge hidden in the data itself and by avoiding human intervention. Table 1 highlights input parameters accepted by some prominent subspace clustering approaches.

There are some recent algorithms that tackle the problem of parameter sensitivity of subspace clustering methods. Yao, Cao, Zhao, Meng, and Xu (2018) modified the Expectation Maximization (EM) algorithm in order to identify parameter values for the

Table 1
Input parameters accepted by some prominent subspace clustering approaches.

Clustering method	Input parameters
CLIQUE	Number of Intervals and Unit Selectivity Threshold
ENCLUS	Entropy Threshold, Interest Gain Threshold
PROCLUS	Average Dimensions count, Clusters Count
MAFIA	Cluster Dominance Factor
DOC	Size of Grid, Density Threshold, Balance Factor Between Points and Dimensions
DENCOS	Equal Length Intervals, Unit Strength Factor, Maximum Subspace Cardinality
DUSC	Density Threshold
DENCLU	Density Threshold and Neighborhood Radius
OPTIGRID	Density Threshold and Neighborhood Radius
SUBCLU	Density Threshold and Neighborhood Radius
FIRES	Density Threshold and Neighborhood Radius
DiSH	Density Threshold and Neighborhood Radius
PreDeCon	Density Threshold and Neighborhood Radius, two Preference Parameters
INSCY	Neighborhood Radius and Density Threshold, Redundancy Factor

proposed PMoG-LRR algorithm. Subspace Memory Clustering (SuMC) (Struski, Tabor, & Spurek, 2018) is proposed which automatically determines optimal values of the parameters - number of clusters, dimensions of clusters and compression ratio. A density estimator to estimate object counts in denser areas is proposed by Müller et al. in the algorithm DensEst (Müller et al., 2009a). The estimation accuracy is improved by incorporating correlations between attributes. The authors claim that the density estimation method can be easily incorporated to improve accuracy and efficiency of subspace clustering and frequent itemset mining algorithms. Another efficient density based subspace clustering approach is proposed by Lakshmi, Madhuri, and Shashi (2017) by dynamically computing the value of ε.

In Zhu, Mozo, and Ordozgoiti (2016), the authors highlight that the subspace clustering algorithms that use cell-based approach show poor quality if the grid size if not set properly. Other parameters such as density threshold and location of denser units also affect the output. The authors propose a method that creates precise grids for given input data without accepting value of grid size from the user. They also estimate density threshold adaptively to avoid bias towards certain dimensionality in the output. Deng, Choi, Jiang, Wang, and Wang (2016a) present a comprehensive review of soft subspace clustering (SSC) algorithms. They observe that most of the SSC methods are sensitive to parameters. They suggest that for a given dataset, outputs of multiple runs on different parameter values can be combined by using ensemble learning and the best output can be selected. However they also mention that, it is difficult to determine volume of the results to be used for the merging process.

Wang et al. (2016) highlight that, choosing the right values of tuning parameters requires domain knowledge, which is rarely available. They emphasized further research for automatic estimation of parameter values producing optimal results. They mention that, the existing subspace clustering methods use a single distance metric for measuring dissimilarity between objects belonging to a single attribute. However such a global distance function cannot handle datasets having multifaceted structures. The authors propose a method to create Composite Kernel Space (CKS) by using basis kernels which integrates distance metric learning into the framework. To tackle the parameter sensitivity issue, the authors in Zhang et al. (2016) propose a parameter free subspace clustering algorithm that works on data cohesion model. The cluster purity is calculated by using entropy. The attributes having low entropy value and the most frequent values in the selected attributes are chosen as basis for finding pure clusters. In Lakshmi, Shashi and Madhuri (2017), the authors suggest an algorithm to output interesting and non-redundant subspace clusters. In Huang et al. (2016), an objective function is proposed for time series data clustering. They highlight the need of estimating the required parameters automatically.

4. Proposed algorithm

Similarity value amongst two data items is a reflection of the strength of relationship between them. Thus measurement of distances is mandatory in the clustering process. In case of numerical attributes, similarity/distance measurement is generally done by applying a distance measure such as Minkowski, Euclidean, Manhattan distance on the attribute values. Euclidean distance (straight line distance) is the most popular distance metric for numerical data. Conventional distance based clustering algorithms calculate similarity between objects over all dimensions. In case of high dimensional data, the similarity between objects is calculated over identified subspace i.e. on subset of attributes. In order to make the subspace clustering process less dependent on user expertise, it is essential to determine the distance threshold value automatically from the data to be clustered. As an attempt in this direction, a novel method is proposed in this paper.

The hierarchical algorithm that uses minimum distance criteria to merge two clusters is called as nearest-neighbor clustering algorithm or single-linkage algorithm. The clustering process stops when the distance amongst two nearest neighbors goes beyond user specified threshold. Thus single linkage clustering method minimizes the sum of linkages – the distances between clusters to be merged. The proposed method uses a single-linkage based bottom-up approach to find subspace clusters. As in CLIQUE, the proposed algorithm first identifies dense regions in each dimension based on the distance threshold value it has computed. Then it builds higher dimensional clusters by merging these 1-dimensional clusters. It operates with default values of parameters which specify how coarse or fine the resulting clusters should be. The steps followed by the algorithm are shown in Fig. 1 below:

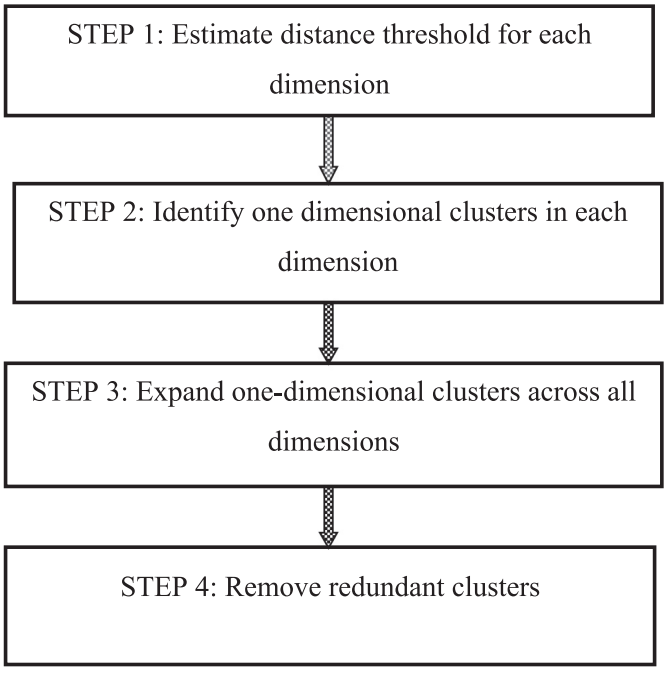


Fig. 1. Steps followed by the proposed algorithm.

Step 1: Estimation of distance threshold separately for each dimension

The proposed method estimates the distance threshold value automatically for every attribute. In this step, objects in each of the D attributes are arranged in non-decreasing order. Then a closely packed 5 elements region in it is identified and in case of tie, it is resolved randomly. The window size is set to five elements because subspace clusters having less than five objects are assumed to be non-significant (Zhang et al., 2016). The maximum separation between two consecutive elements in the group is rounded to next decimal point and is recorded as the distance threshold for that dimension. The estimation of distance threshold is highly accurate and produces near optimal results on synthetic datasets.

The complexity of this step is calculated as follows: the sorting of objects to find distance threshold can be performed by using an algorithm like quick sort having time complexity of the order O(N log N), where N is the count of objects. Then to find distance threshold for each dimension separately, consecutive elements in each attribute are compared, which requires ($N-1$) comparisons. Hence, total time complexity of this phase is $D \times (N \log N) + D \times (N-1)$ which is of the order O(N log N).

Step 2: Formation of 1-dimensional subspace clusters in each dimension

In this step, dense regions of objects in each dimension are found based on distance threshold value identified in Step 1. Initially each object in the dimension is placed in a separate cluster. If the distance between any one object from a cluster and any one object in another cluster is within the distance threshold then the two clusters are merged. The process of merging two clusters is repeated until no new clusters can be formed. This step works in a way similar to single linkage clustering. The 1-dimensional clusters which are having density of objects less than object_density_threshold are marked non-significant and pruned from further processing. The default value of object_density_threshold is set to 5. Time complexity of Step 2 is $D \times$ O(N), as N-1 comparisons are needed to find neighboring objects.

Step 3: Formation of higher dimensional subspace clusters

Higher dimensional clusters are formed by connecting 1-dimensional subspace clusters sharing common objects. If an outlier object is by chance becomes part of a one dimensional cluster, it will be absent in clusters present in remaining dimensions and its support will be below attribute_threshold. Such outlier objects get eliminated in this step. The attributes containing 1-dimensional clusters are arranged in non-decreasing order on the basis of percentage of coverage of the data items belonging to the clusters. The objects in a 1-dimensional cluster in an attribute are connected to objects in next attribute in the sequence if they contain common object indices to form higher dimensional subspace clusters. If the count of dimensions in a subspace cluster is less than attribute_threshold, those clusters are pruned from further processing. The default value of attribute_threshold is set 5. The time complexity of this step is O(N).

Step 4: Removal of redundant clusters

A subspace cluster is dispensable if it is subset of a larger subspace cluster. In step 3, many redundant clusters may be formed. In Step 4, redundant clusters are removed by applying an algorithm proposed in Kuhn (1955). As a high dimensional data is inherently sparse, it can be assumed that the subspace clusters cover less than 10% of the total sub-objects. Assuming that there are Q subspace clusters identified in Step 3, then intersection operation can be done by an algorithm of complexity O(N), which makes Q iterations to find non-redundant subspace clusters. Hence the time complexity of this Step is O(N). Here the points input for intersection are less than 10% of total number of objects (N).

Time complexity of the proposed method

After summing up the time complexities of Steps 1 to 4, the time complexity the proposed algorithm is estimated as O(N log N) + O(N) + O(N) + O(N) ≅ O(N log N).

Fig. 2 shows pseudo code of the algorithm.

```
Algorithm CLUSLINK( DS )
//Input: Dataset DS having N objects and D dimensions
//Output: k Subspace Clusters
Begin
   object_density_threshold = 5 ;
   attribute_threshold = 5 ;
   Dist[1:D]=Estimate_Distance(DS, N, D) ;
   Clusters[1:D] = Identify1D_Clusters(DS, N, D, Dist) ;
   Expanded_clusters[1:M]=Expand1D_Clusters(Clusters);
   Subspace_clusters[1:k]=Remove_Redundant_Clusters( Expanded_clusters);
   Return(Subspace_clusters);
End
```

(a) Pseudo code of the proposed algorithm

```
Algorithm Estimate_Distance(DS, N, D)
// Estimates of distance threshold for each attribute
//Input: Dataset DS having N objects and D dimensions
//Output: Distance threshold values for all D dimensions, Dist[1:D]
Begin
   A[1:D] = Attributes of DS;
   Dist [1:D]= ∞;
          for each attribute Ai ϵ A, 1 <= i <= D
   {
      // Ai is dimension for which distance threshold is to be estimated

                Sort unique values in ith dimension in non-decreasing order to form
                one-dimensional vector V ;
        Avg[1: n-4] = 0 ;
        for j = 1 to ( n – 4 )
        {
            sum_of_distance = 0 ;
           for k = 1 to 4
           {
              sum_of_distance = sum_of_distance + (V[j+k+1] - V[j+k] ) ;
           }
             Avg[j]=sum_of_distance/4 ;
        }
        //Find the densest 5 elements group of objects
        Least_avg_region=Location of the least element in vector Avg ;
        Dist[i]=maximum distance between two consecutive elements in
                the group pointed by Least_avg_region;
        Round Dist[i] to nearest decimal value ;
   }
   Return Dist;
End
```

(b) Pseudo code of STEP 1

```
Algorithm  Identify1D_Clusters(DS, N, D, Dist)
//Finds one-dimensional subspace clusters for each dimension
// Input: Dataset DS having N objects and D dimensions, Dist[1:D]
// Output: One-dimensional clusters in all D dimensions - Clusters[1:D]
//Clusters[i] is a list of one dimensional clusters
Begin
   for each attribute Ai ϵ A, 1 <= i <= D
   {
              //Put all values in Ai into singleton clusters
              CL1 = Ai[1] , CL2 = Ai[2], … CLN= Ai[N];
              Repeat until no new clusters are formed
              {
                Merge two singleton clusters generated above by applying single
                      linkage clustering method, i.e. Include CLj and CLk in one cluster
                    if dist(CLj , CLk) <= Dist[Ai] ;
              }
              Discard those one-dimensional clusters which have less than
                    object_density_threshold members ;
              Clusters[i] = clusters in dimension Ai;
   }
   Return Clusters;
End
```

Pseudo code of STEP 2

```
Algorithm  Expand1D_Clusters(Clusters)
// Expand one-dimensional subspace clusters to form multidimensional subspace
// clusters
//Input: One dimensional clusters identified in dimensions of DS
//Output: Multi-dimensional subspace clusters
Begin
      Arrange the dimensions of DS in non increasing order of percentage of
      coverage of objects and exclude dimensions which are not having any one-
      dimensional subspace clusters;

      //Now there are M <= D remaining dimensions for forming clusters
       If one-dimensional subspace clusters identified in step 2 share common
       object ids with one-dimensional subspace clusters in remaining
       dimensions, join them all to form k-dimensional subspace cluster.;

      Retain the subspace cluster for further processing only if the
      count of dimensions in the subspace cluster is greater than attribute_threshold;.

      Return Expanded_clusters[1:M];
End
```

(d) Pseudo code of STEP 3

```
Algorithm Remove_Redundant_Clusters( Expanded_clusters)
// Remove redundant subspace clusters from the output
// Input:  Multidimensional subspace clusters
// Output: Non-redundant subspace clusters
Begin
      Remove all subspace clusters Expanded_clusters[i] ⊆ Expanded_clusters[j] ;
      Return all k maximal subspace clusters;
End
```

(e) Pseudo code of STEP 4

Fig. 2. Pseudo-code of the steps followed by the proposed method.

5. Experimental results

5.1. Quality measures for subspace clustering

The performance of a clustering algorithm is evaluated in terms of execution time and quality of clustering results. If true results are already known, the quality of clustering can be measured by extrinsic methods and the clustering result can be compared with the ground truth. If such information is unavailable, intrinsic methods can be applied that evaluate how well the groups in the result are separated from each other. The information of objects participating in clusters is already known in case of synthetic datasets. If the true clusters hidden in a dataset are denoted as $H = \{H_1, ..., H_m\}$; the clusters found by given algorithm are denoted as $R = \{C_1, ..., C_k\}$; then the quality of clustering is evaluated by determining the score of conformity of output clusters to the true clusters.

Object and subspace based measures

For traditional algorithms, extrinsic clustering quality evaluation measures assume that all attributes in the given data are part of output clusters and the evaluation is done accordingly. However, for subspace clustering, the evaluation measures should account for a subset of attributes relevant to a given subspace cluster. For evaluation of subspace clustering result, each data point is divided into sub-objects annotated with attributes. If the dataset has D dimensions and N items, then each object O_i, is divided as sub-objects $O_{i1}, ..., O_{iD}$. Hence there will be $N \times D$ sub-objects. A subspace cluster then comprises of subset of these all sub-objects. As a result, two output subspace clusters will match only when they share the same sub-objects. Based on this notion of representation, the evaluation measures specific to subspace clustering are described in following paragraphs. The evaluation measures to be used for subspace clustering are based on knowledge of objects and attributes participating in the true subspace clusters. Hence these measures are called as subspace and object based evaluation scores.

Let a structured numerical dataset DS contains N data items and D variables.

$A = \{A_1, A_2, ..., A_D\}$ is the set of attributes and $O = \{O_1, O_2, ..., O_N\}$ where $O_i = \langle O_{i1}, O_{i2}, ..., O_{i\,D} \rangle$. A hidden or true cluster is denoted by $H_i = (I_i, S_i)\ \varepsilon\ H$, where $I_i \subseteq O$ and $S_i \subseteq A$. The quality of clustering produced by the proposed method is evaluated in terms of F-measure, accuracy, Clustering Error (CE) and Relative Non Intersecting Area (RNIA) (Patrikainen & Meila, 2006). The efficiency and scalability of the proposed algorithm is measured in terms of execution time measured in seconds. The true clusters are mapped to output clusters by applying method suggested in Kuhn (1955).

Precision: A high precision indicates that most of the sub-objects in the output cluster are true sub-objects. Optimal value of precision is 1.0.

Precision = True Positives / (True Positives + False Positives)

Recall: A high recall indicates that the identified clusters cover a large fraction of the true sub-objects. Optimal value of recall is 1.0.

Recall = True Positives / (True Positives + False Negatives)

F1-measure: F1-measure corresponds to extent of conformity of output to the true clusters and to what extent the algorithm is able to exclude false results. Optimal value of F1-measure is 1.0. If the true result contains m subspace clusters, the F1 value is calculated obtained as follows:

$$F1 = \sum_{i=1}^{m} \left(\frac{(2 * \text{Precision(i)} * \text{Recall(i)})}{(\text{Precision(i)} + \text{Recall(i)})} \right)$$

Accuracy: Accuracy is the measure of the extent to which the algorithm is able to mark true sub-objects in the clusters and out of the clusters. Optimal value of accuracy is 1.0.

Accuracy = (True Positives + True Negatives) / (count of all sub-objects)

RNIA: The subspace clustering quality is high if it covers all and only true sub-objects and does not include objects not supposed to be part of any subspace cluster. This aspect is measured by Relative nonintersecting area (RNIA) measure. Let U indicates count of objects in union of true and output clusters. Let I indicates count of objects common to true and output clusters. Then RNIA = (U−I)/U. Optimal value of RNIA is 0.0.

CE: RNIA measure does not reflect the case when a true cluster is partitioned into several small clusters in the output or several true clusters are merged to form an output cluster. CE measure reduces the clustering quality value in such cases. Here a mapping of true and output clusters is first generated. Let U indicates count of objects in union of true and output clusters. Let I' indicates count of objects common to mapped and output clusters. Then CE = (U – I')/U. Optimal value of CE is 0.0.

Extrinsic evaluation of output of a subspace clustering algorithm is generally done on synthetic as well as real datasets. In case of synthetic datasets, the information of embedded clusters (hidden clusters) is known a priori. However such information is missing for most of the real world datasets. Hence for evaluation of quality on real data, those datasets which contain class labels for all objects are chosen. Generally such datasets are available for validation of classification models. The objects I_i belonging to class i are then considered as members of cluster $H_i = (I_i, S_i)$, where S_i is the subspace of H_i. However as the real world dataset is intended for classification, all attributes should be assumed equally important for subspace clustering. Hence for real data set the relevant dimensions for all hidden subspace clusters is set to A, i.e. the whole set of attributes. By using this convention, the hidden clusters $H = \{H_1, ..., H_m\}$ can be known for real datasets also.

5.2. Experimental setup

5.2.1. Synthetic data generator

Evaluation of a clustering algorithm on artificially generated (synthetic) datasets is an effective and widely used technique in the data mining community. For synthetic datasets, the structure of embedded clusters is known precisely. Hence the performance of clustering algorithm under consideration can be determined by evaluating the conformity between true partitions and the grouping identified by the clustering algorithm. Another advantage of using synthetic datasets is that the structure and dimensions of the clusters to be embedded can be controlled as per the requirements of the clustering method.

The data generator used in this work is programmed in R, through which the structure and size of the data to be generated can be specified. The generator accepts following input from the user:

i. Number of objects (N)
ii. Count of dimensions (D)
iii. Attribute domain (minimum, maximum)
iv. Count of embedded subspace clusters (N_c)
v. Count of data items participating in each of the subspace clusters (C_o)
vi. Number of attributes in each subspace cluster (C_a)
vii. The standard deviation of objects participating in each subspace cluster (S_d)
viii. The percentage of outliers (N_o)

The experiments were done on synthetic datasets as described in Table 2.

Table 2
Description of synthetic datasets.

Number of dimensions (D)	100, 200, 300, 400, 500
Number of objects (N)	1000, 2000, 3000, 4000, 5000
Minimum value of objects in a dimension (minimum)	1.0
Maximum value of objects in a dimension (maximum)	100.0
Standard Deviation (S_d)	0.01
Percentage of outliers (N_o)	10
Size of subspace clusters (C_o and C_a)	10 objects and 10 dimensions 20 objects and 20 dimensions
Count of subspace clusters (N_c)	5, 10, 20, 50

5.2.2. Real datasets

UCI machine learning repository has many real world datasets that are being used for evaluation of data mining methods. Information about true classes of the objects in the datasets is provided as the last attribute in the data matrix. The Iris dataset contains 4 attributes and 140 objects which are classified into 3 classes. The *Ecoli* dataset has 7 attributes and 336 objects divided into 8 classes. The Glass dataset has 6 classes, 214 objects each described by 9 features. The Liver dataset has 2 classes, 345 objects and 6 attributes. Pima dataset contains 768 items each with 8 attributes and the objects are divided into 2 classes. The Vowel dataset has 11 classes, 990 instances each with 13 features.

5.3. Empirical evaluation of quality of output produced by CLUSLINK

This section discusses about the effects of accurate distance threshold estimation on quality of output produced by the proposed method. Empirical evaluation of the method was done on self generated synthetic datasets and real datasets available on UCI machine learning repository (Dua & Karra Taniskidou, 2017). The subspace clustering evaluation measures described in (Müller et al., 2009a) were used which are mainly of extrinsic type. All experiments were conducted on personal computer having Intel(R) Pentium® P6200 CPU @ 2.13 GHz, 2.00 GB RAM, Windows 7 Operating System, R version 3.4.3.

The output was assessed based on following criteria:

i. The quality of output produced by CLUSLINK is compared with other 5 well known algorithms. Synthetic data having 1000 objects and dimensions varied from 100 to 500 and varied count and dimensionality of embedded clusters was used for evaluation.
ii. The quality of output produced by CLUSLINK is compared with other 5 well known algorithms on 6 real datasets. The parameter values used are the default values suggested in “subspace” package of R (Hassani, 2015).
iii. The quality of output produced by CLUSLINK is compared with optimal results reported in Müller et al. (2009a) for 10 well known algorithms on 2 real datasets from UCI machine learning repository (Dua & Karra Taniskidou, 2017).

5.3.1. Clustering quality on synthetic datasets

5.3.1.1. Experiments on datasets having varied sizes. The quality was evaluated on synthetic datasets containing 5 subspace clusters containing 10 instances and 10 dimensions each. The object count was varied from 1000 to 5000 and dimension count was varied from 100 to 500.

Figs. 3–6 show following results:

Accuracy of the output is between 1 and 0.94. F1-score is approximately 1.0 in all runs.

The F1-score value is optimal i.e. 1.0 in most of the cases only due to accurate estimation of the distance threshold. RNIA value is between 0.0 and 0.06. CE value lies between 0.0 and 0.06.

As depicted in Fig. 7, the algorithm identifies all 5 clusters embedded in the dataset exactly. To check if the improved clustering quality is only due to accurate distance estimation, experiments were also carried out by keeping distance threshold value constant at 0.5 for all dimensions, instead of estimating it in the first Step of CLUSLINK. Synthetic datasets used for the experiments had 1000 objects and dimensions varied from 100 to 500. Tables 3 and 4 reflect the fact that, the algorithm could produce high quality output only due to accurate distance estimation in the first Step.

5.3.1.2. Comparison of output quality with other algorithms on synthetic datasets. Using the synthetic data generator mentioned above, five synthetic datasets with $N = 1000$ objects were generated each having 5 subspace clusters containing 10 objects and 10 attributes randomly were embedded in the data. The dimensionality of the five datasets was 100, 200, 300, 400 and 500 respectively. An implementation of CLIQUE, FIRES, P3C, PROCLUS and SUBCLU is available in package ‘Subspace’ of R (Hassani, 2015). The parameter setting used during the experiments for FIRES, PROCLUS and SUBCLU was as provided in the package. The setting was changed for CLIQUE and P3C as shown in Table 5 for getting some output, as the default values could not produce any result. CLUSLINK was executed with default values of object_density_threshold and attribute_threshold set to 5. A comparison of quality of the output of CLUSLINK with abovementioned subspace clustering algorithms is presented in Fig. 8.

Result analysis

CE: As shown in Fig. 8(a), the clustering error of the proposed method is 0.06 for synthetic dataset with 400 dimensions and 100 objects and in rest of the cases it is 0.0 which is the optimum. Other algorithms show clustering error of more than 0.98. This is because the experiments were done with default values of parameter specified in the package ‘Subspace’. As highlighted in Müller et al. (2009a), getting values of optimal parameter setting requires repeated runs of these algorithms every time with a new set of values. In contrast to this, the proposed algorithm could get optimal results in single run, without accepting any input from the user.

RNIA: In case of RNIA measure as shown in Fig. 8(b), the same observations hold true and RNIA value of the clustering produced by the algorithm is near optimal.

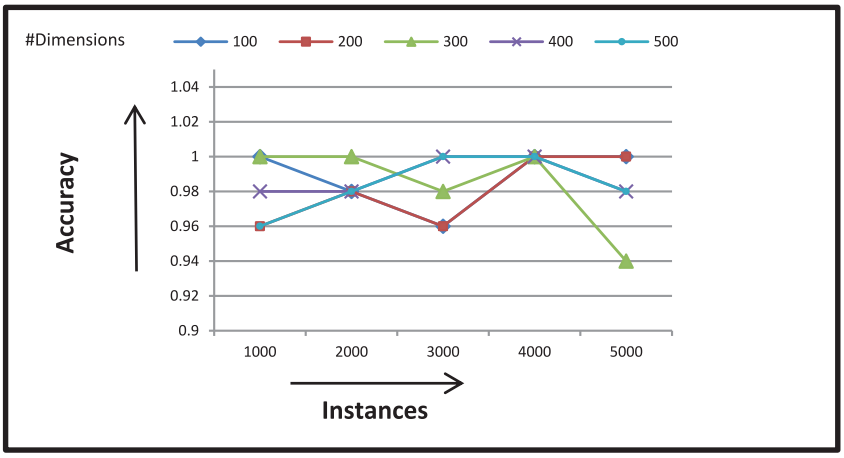
F1-measure: Fig. 8(c) highlights that, F1 value of the proposed method is more than 0.96 for $d = 400$ and in other cases it is 1.0 which is again the optimal value. CLIQUE could produce F1-value up to 0.3 and for rest of the algorithms it is less than 0.11.

Accuracy: As shown in Fig. 8(d), accuracy of the proposed method is optimal i.e. 1.0 in all cases except for $d = 400$ where it is 0.94. CLIQUE shows maximum accuracy as 0.59 on $d = 500$ and FIRES shows 0.78 on $d = 300$ which is better than remaining algorithms.

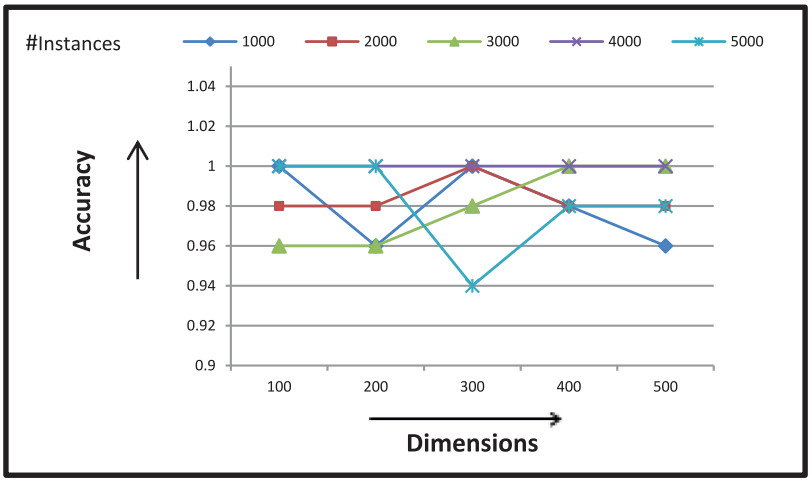
Count of output clusters: The algorithm CLUSLINK outputs exactly the same number of clusters as those are embedded in the data as shown in Table 6. CLIQUE and P3C output too many redundant clusters.

The results show that, the algorithm produces near optimal results on synthetic datasets. The optimal results are outcome of accurate distance threshold estimation.

5.3.1.3. Experiments on variable sized embedded subspace clusters. This section compares performance of CLUSLINK with other algorithms on synthetic datasets having 1000 objects and 100 dimensions, by varying cluster counts to 5, 10, 20 and 50. Each cluster has sizes of (i) 10 objects and 10 dimensions and (ii) 20 objects and 20 dimensions. The objective of the experiment was to check if varied cluster dimensionality affects the quality the output. The outcome of the experiments is shown in Fig. 9.



(a)



(b)

Fig. 3. Accuracy of CLUSLINK on synthetic datasets.

Table 3
Quality of clustering by CLUSLINK with and without distance estimation.

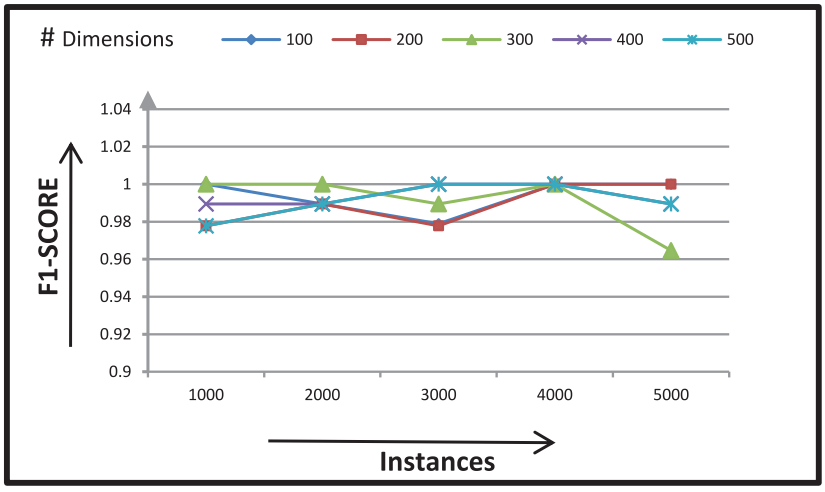
#Dimensions	CE after distance estimation	CE without distance estimation	RNIA after distance estimation	RNIA without distance estimation	F1 after distance estimation	F1 without distance estimation	Accuracy after distance estimation	Accuracy without distance estimation
100	0	0.99	0	0.99	1	0.04	1	0.21
200	0	0.99	0	0.99	1	0.02	1	0.10
300	0	0.99	0	0.99	1	0.014	1	0.11
400	0.06	0.99	0.06	0.99	0.96	0.014	0.94	0.09
500	0	0.99	0	0.99	1	0.008	1	0.13

Table 4
Number of output clusters with and without distance estimation.

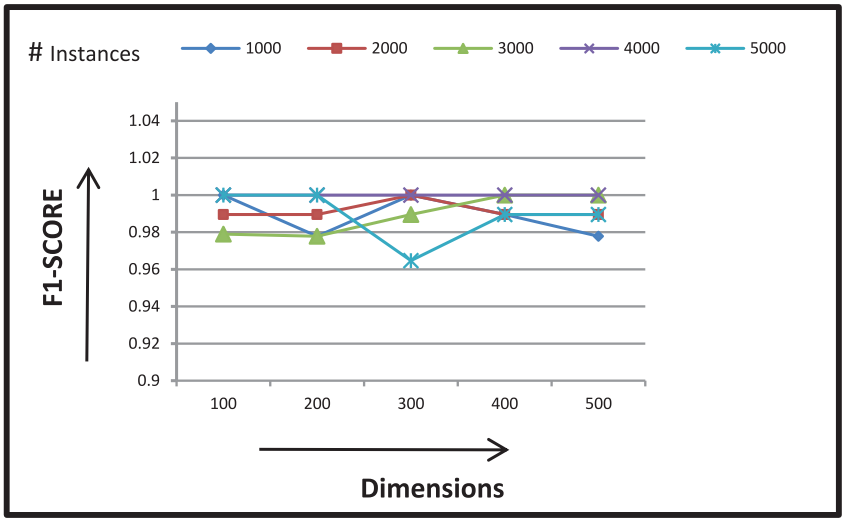
#Dimensions	Count of output clusters after distance estimation	Count of output clusters without distance estimation
100	5	650
200	5	1363
300	5	2014
400	5	2724
500	5	3338

Table 5
Parameter settings for the experiments.

Algorithm	Parameter values
CLIQUE	Xi = 50, tau = 0.03
FIRES	base_dbscan_minpts = 4, base_dbscan_epsilon = 1, $k = 1$, minimumpercent = 25, minclu = 1, mu = 1, post_dbscan_epsilon = 1, split = 0.66, post_dbscan_minpts = 1
P3C	ChiSquareAlpha = 0.50, PoissonThreshold = 2
PROCLUS	$k = 12$, $d = 2.5$
SUBCLU	epsilon = 1, minSupport = 5



(a)



(b)

Fig. 4. F1-SCORE of CLUSLINK on synthetic datasets.

Table 6
Number of output clusters on data having 1000 objects and 5 embedded clusters.

#Dimensions	CLUSLINK	CLIQUE	FIRES	P3C	PROCLUS	SUBCLU
100	5	117	1	121	12	4
200	5	191	1	152	12	4
300	5	349	1	195	12	4
400	5	440	1	177	12	4
500	5	534	1	297	12	4

Result analysis

CE and RNIA: The results in Fig. 9(a) and (b) highlight that, CLUSLINK has Clustering error and RNIA value less 0.02 in all cases, whereas for other algorithms, the error is very high.

F1-measure: CLUSLINK could produce F1-value in the range 1.00–0.98 which is optimal one as depicted in Fig. 9(c). CLIQUE achieves better results in terms of F1-value compared to other algorithms.

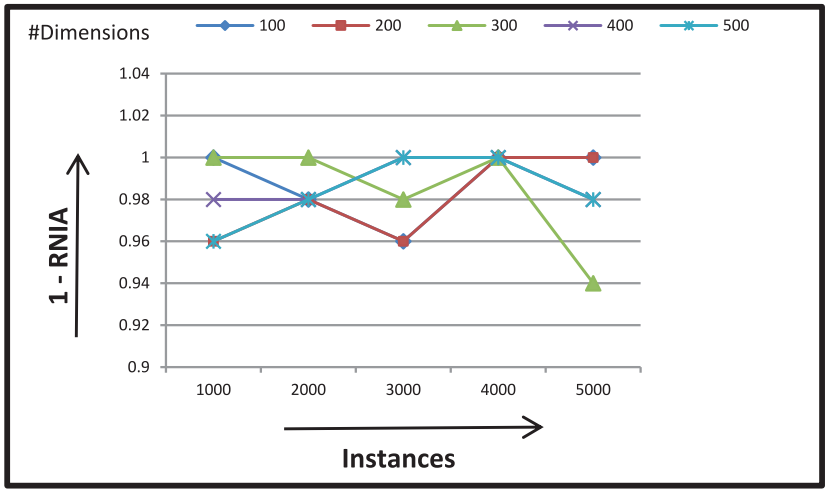
Accuracy: As shown in Fig. 9(d) the accuracy of the algorithm varies from 1.00 to 0.98 which is the best in the group. FIRES and CLIQUE are better in terms of accuracy compared to remaining algorithms.

Count of output clusters: As shown in Table 7, CLUSLINK outputs precisely the same number of clusters as those embedded in the dataset in all cases, whereas CLIQUE and P3C produced more than 100 clusters, FIRES could output only 1 cluster, PROCLUS 12 clusters and SUBCLU could output 4 clusters in each case.

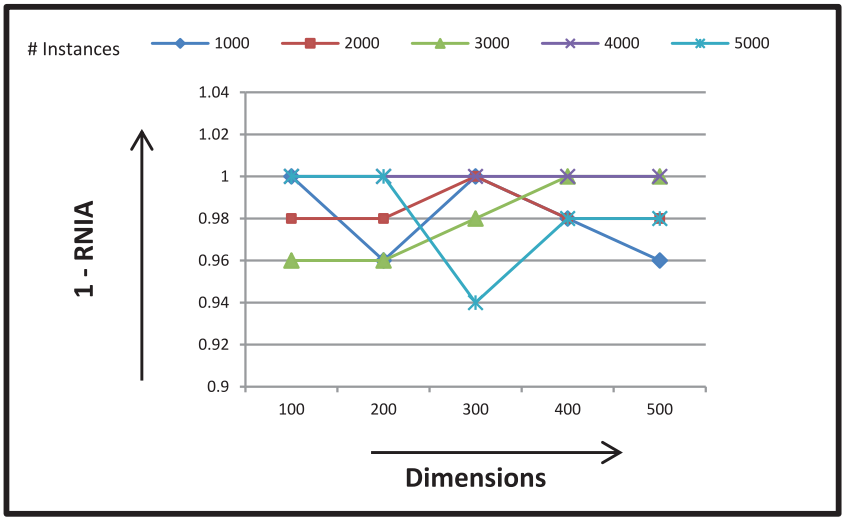
The results highlight that, CLUSLINK produces high quality results on synthetic datasets containing varied cluster dimensionality.

5.3.2. Clustering quality on real datasets

5.3.2.1. Comparison of quality with other algorithms on default parameter setting. This section describes experimental results on real datasets mentioned in Section 5.2.2. The performance of CLUSLINK was compared with other state of the art algorithms mentioned in Table 5. The parameter setting for these algorithms is also specified in the table. Default values of *object_density_threshold* and *attribute_threshold* for CLUSLINK were set to 5. The algorithms - CLUSLINK, CLIQUE, FIRES, P3C, PROCLUS, and SUBCLU were executed on each of datasets.



(a)



(b)

Fig. 5. RNIA of CLUSLINK on synthetic datasets.

Table 7
Number of clusters identified against embedded.

Cluster Dimensions Obj. X Dim.	Number of embedded Clusters	CLUSLINK	CLIQUE	FIRES	P3C	PROCLUS	SUBCLU
10 × 10	5	5	109	1	143	12	4
10 × 10	10	10	156	1	157	12	4
10 × 10	20	20	159	1	188	12	4
10 × 10	50	50	272	1	107	12	4
20 × 20	5	5	182	1	163	12	4
20 × 20	10	10	378	1	N. A.*	12	4
20 × 20	20	20	706	1	N. A.*	12	4
20 × 20	50	50	1104	1	N. A.*	12	4

* N.A.- P3C could not finish within 30 min; hence data of clusters output is not available.

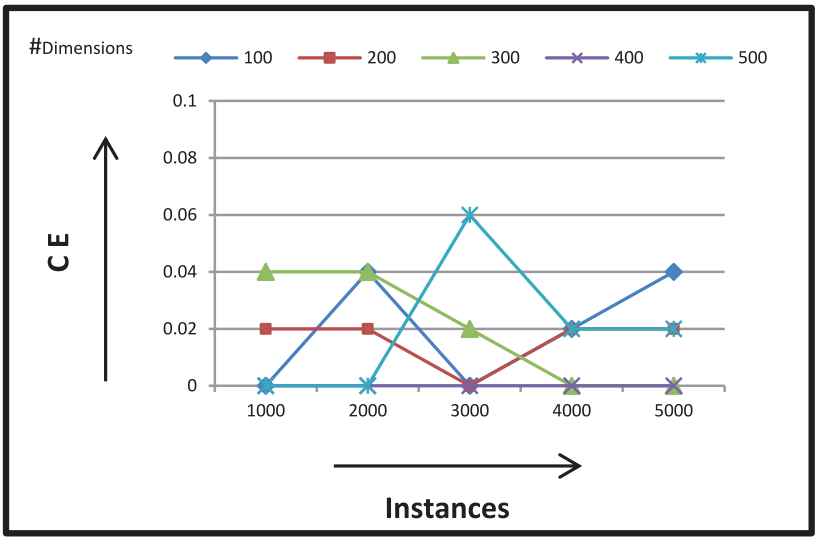
CE: Fig. 10(a) shows the clustering error on various real datasets by each of the algorithms. The clustering error of CLUSLINK is the minimum compared with other algorithms.

RNIA: Fig. 10(b) shows that, RNIA value of CLUSLINK and FIRES is found to be the least in the group. However on Pima dataset FIRES shows poor performance in terms of RNIA.

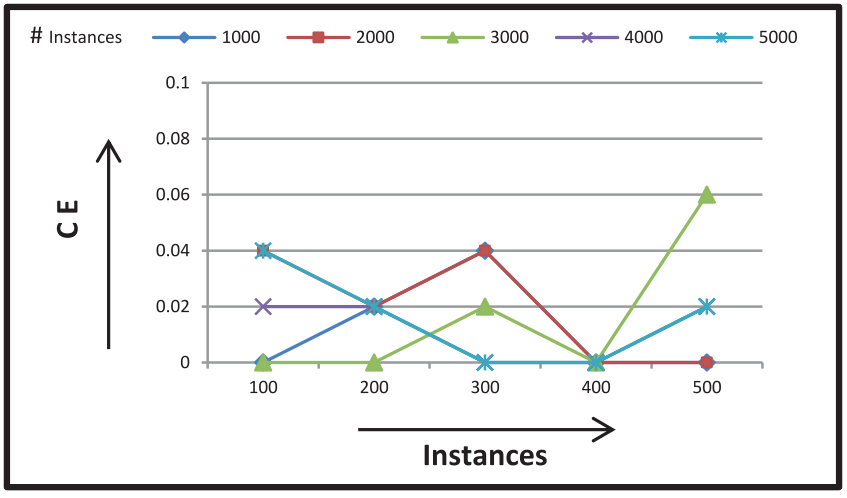
Accuracy: Accuracy of FIRES is the optimal i.e. 1.0 on 4 datasets. CLUSLINK shows Accuracy in the range of 0.81 to 0.99 over 4 datasets. FIRES shows very poor performance on Pima dataset in terms of Accuracy (=0.13), whereas CLUSLINK shows the best performance with Accuracy = 0.81. On Iris Accuracy of CLUSLINK is 0.99 which is the best amongst all. Fig. 10(c) shows graph of Accuracy.

F1-measure: CLUSLINK performs the best in terms of F1-value for all datasets compared to other algorithms. Fig. 10(d) reflects F1-score.

Count of clusters output: Table 8 displays the number of clusters output by all the algorithms against the classes actually present in the data. The table shows that no algorithm could output the same number of clusters as those are present in the data.



(a)



(b)

Fig. 6. CE of CLUSLINK on synthetic datasets.

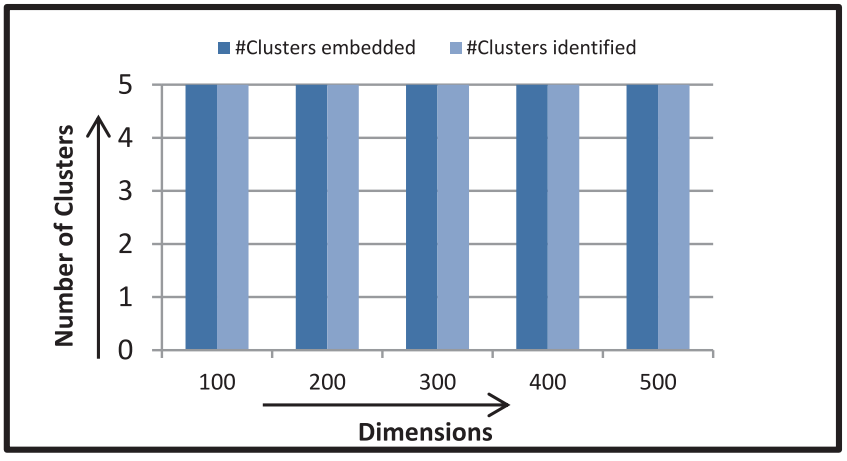
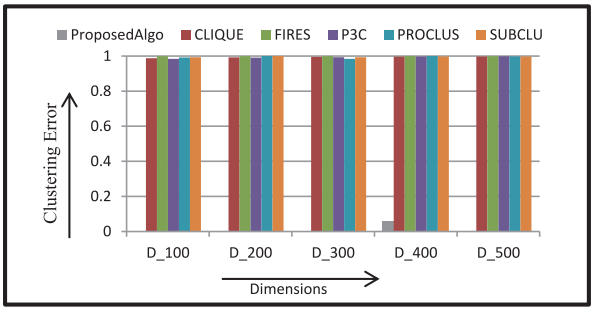
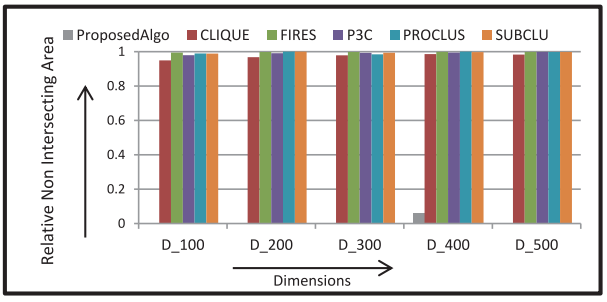


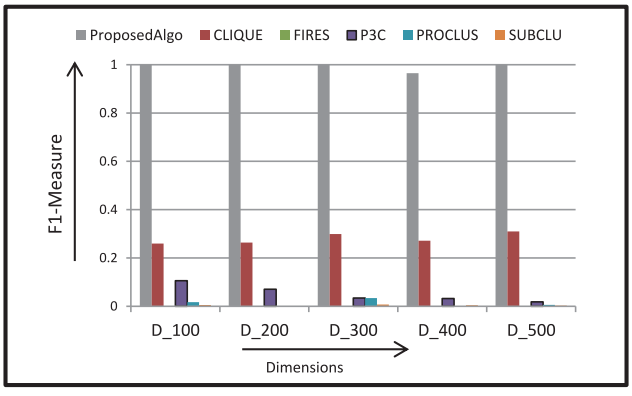
Fig. 7. Clusters identified by CLUSLINK against actually present.



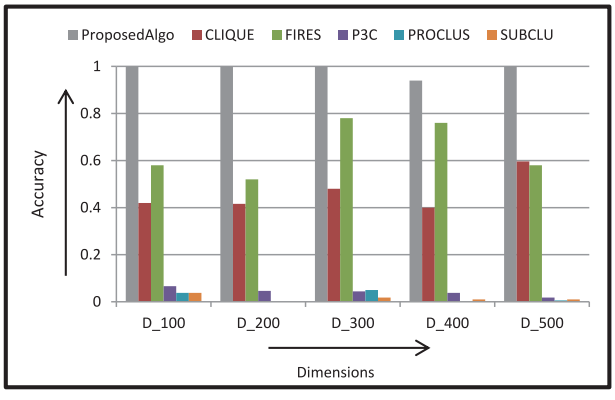
(a) Clustering Error



(b) Relative Non Intersecting Area



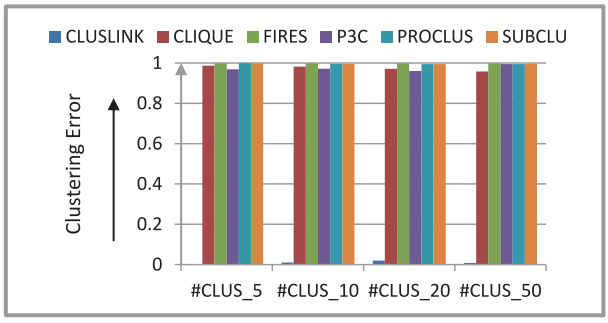
(c)F1-Score



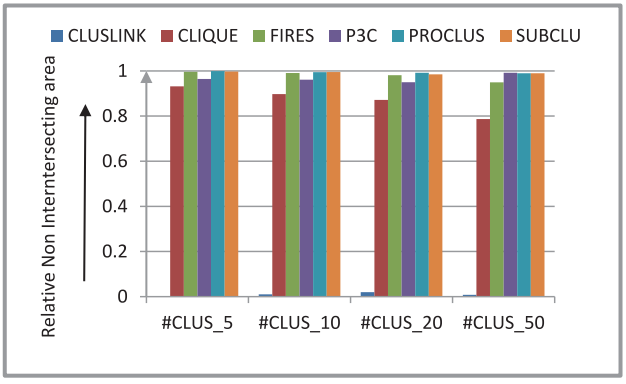
(d) Accuracy

Fig. 8. Comparison of quality of output on synthetic datasets.

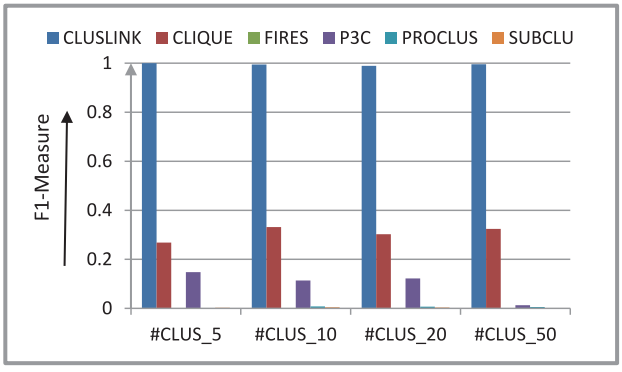
Some of the algorithms – FIRES, PROCLUS, SUBCLU output a constant number of clusters irrespective of the dataset input to the algorithm. CLUSLINK outputs comparatively less number of clusters than CLIQUE and it can be further improved to output all and only true clusters.



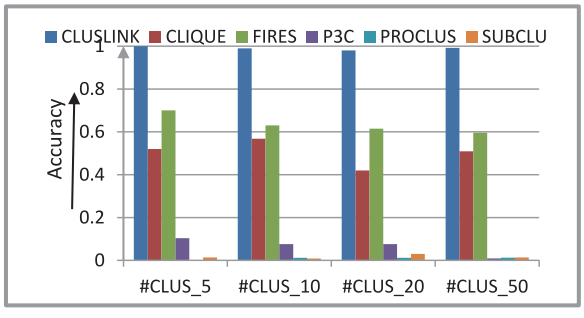
(a) Clustering Error



(b) Relative Non Intersecting Area



(c) F1-Score

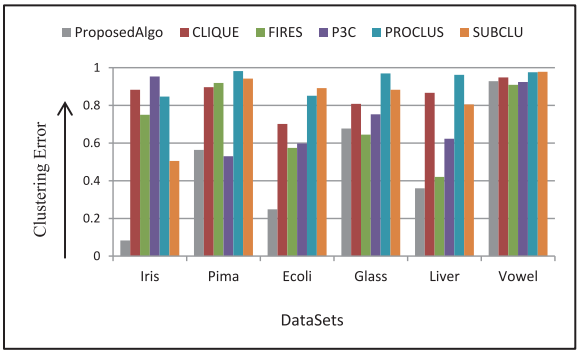


(d) Accuracy

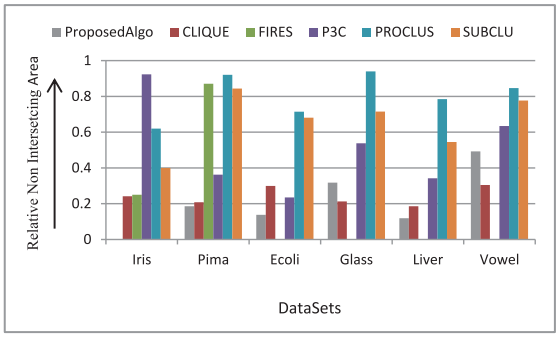
Fig. 9. Comparison of CLUSLINK on synthetic data having varied cluster sizes.

Above results show that CLUSLINK is highly effective on real datasets also. It could produce the high quality results in a single run without requiring any parameter setting by the user.

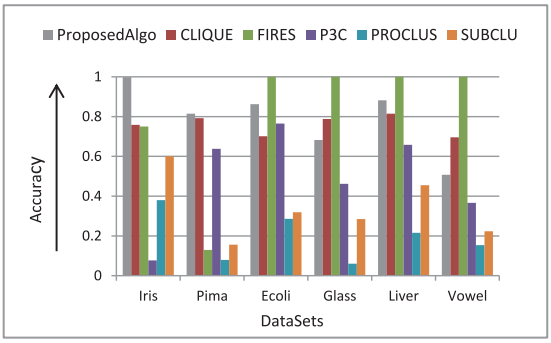
5.3.2.2. Comparison of quality with other algorithms on optimal parameter setting. The results obtained in section 5.3.2.1 show



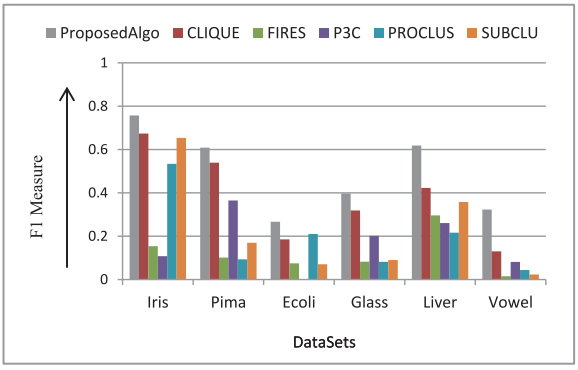
(a) Clustering error



(b) Relative Non Intersecting Area



(c) Accuracy



(d) F1-Score

Fig. 10. Comparison of quality of output on real datasets.

that, most of the algorithms could not perform well on the default parameter setting specified in "subspace" package of R. In Müller et al. (2009a) the authors proposed a methodical framework to evaluate various subspace clustering approaches. They performed large number of experiments each with different parameter setting (around 100 parameter settings per algorithm) and reported the optimal results for each algorithm on real world data sets. These results can be used as baseline for evaluation of new methods. CLUSLINK was executed only once on each real dataset and with no input parameters to set by the user. Fig. 11 presents the comparison of quality of results by CLUSLINK and optimal results shown by other algorithms on PIMA and VOWEL datasets as reported in Müller et al. (2009a). Table 9 shows the number of clusters output by each algorithm against those actually present in the datasets.

Results analysis

In the works by Huang et al. (2014) and Müller et al. (2009a), the authors mention that they executed 100 runs of each algorithm under consideration with different set of parameters to arrive at the best values of F1-score, accuracy, and other quality measures. Opposite to this, CLUSLINK requires only a single run without any parameter setting. As shown in Fig. 11, it outperforms other algorithms in terms of Accuracy and produces comparable results for F1-measure and RNIA. Still the algorithm can be further improved to reduce CE values. The CE measure penalizes for splitting or merging of true clusters in different output clusters. Table 9 shows the comparison of number of clusters reported. Most of the algorithms output many small clusters when only 2 true clusters are present in the input datasets. CLUSLINK can be improved in this issue by applying adaptive merging strategy. This will also result in improvement of CE values. The improved clustering accuracy shown by CLUSLINK is the outcome of accurate distance estimation. Overall the algorithm outperforms in terms of accuracy and produces comparable results for other evaluation measures due to accurate estimation of the most important parameter – the distance threshold.

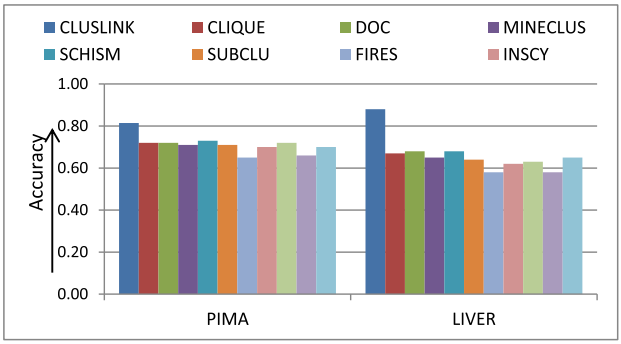
5.4. Improving execution time by using greedy clustering approach

Optimization problems try to minimize or maximize given objective functions. These problems generally belong to NP-hard category (Manning, Raghavan, & Schütze, 2008). The clustering problem falls in the category of optimization problems, as it attempts to minimize inter-cluster distances and maximize intra-cluster distances. A subspace clustering algorithm that aims to optimize the clustering results always shows exponential time complexity on datasets with few tens of dimensions. Sometimes, these algorithms also fail to output any results in reasonable time. Today, most of the datasets have hundreds of dimensions. However, existing subspace clustering algorithms show poor scalability on high dimensional datasets having more than 25 dimensions (Müller et al., 2009a). This happens due to the reason that, they iterate multiple times over the dataset to optimize results (Nagesh, Goil, & Choudhary, 2000; Zhu, Mara, & Mozo, 2015). To obtain a trade-off between execution time and quality of output, the proposed method uses greedy approach. Greedy algorithms find locally optimal choices. These approaches are significantly scalable than other approaches like backtracking and dynamic programming, because they do not reiterate through the input data to revise their earlier choices. Although greedy approaches do not generally produce optimal solutions, as discussed in Section 5.3, the proposed algorithm has proven its effectiveness in terms of high quality of the results.

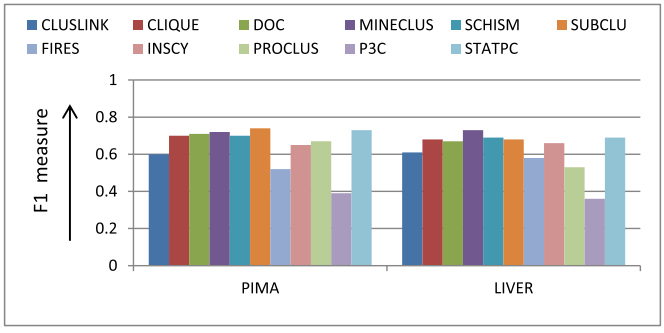
5.4.1. Empirical evaluation of scalability of the algorithm

This section presents evaluation results of the proposed method based on following criteria:

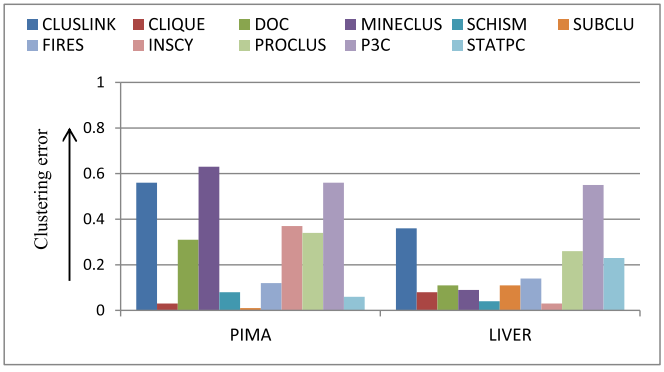
i. The scalability of CLUSLINK by increasing dimensionality and number of instances in the synthetic data.
ii. The execution time of CLUSLINK in comparison with other 5 well known algorithms on synthetic data having 1000 objects and dimensions varied from 100 to 500. The experiments were



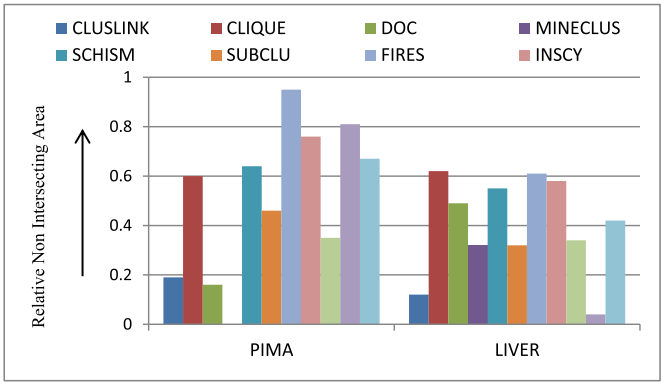
(a) Comparison of Accuracy



(b) Comparison of F1-Score



(c) Comparison of Clustering Error



(d) Comparison of Relative Non Intersecting Area

Fig. 11. Comparison of CLUSLINK with optimal results of other algorithms.

then repeated by varying count and dimensionality of embedded clusters.

iii. The comparison of execution time of CLUSLINK with other 5 well known algorithms on 6 real datasets. The default values of parameters suggested in "subspace" package of R were used during these experiments (Hassani, 2015).

iv. The comparison of execution time of CLUSLINK with optimal results reported in Müller et al. (2009a) for 10 well known algorithms on 2 real datasets from UCI machine learning repository (Dua & Karra Taniskidou, 2017).

5.4.2. Execution time on synthetic datasets

5.4.2.1. Scalability of CLUSLINK on datasets having varied sizes. Scalability results of the proposed algorithm are presented in this section. The objective was to check changes in runtime by increasing dimensionality and cardinality of the datasets. The object count was varied from 100 to 500. 5 subspace clusters containing 20 data items and 20 attributes were embedded. The empirical results shown in Fig. 12 reveal that, the execution time increases linearly with increase in N up to 4000 data items. For $N > 4000$, due to formation of many one-dimensional subspace clusters. Hence in process of joining one-dimensional clusters to form multidimensional subspace clusters, many set matching operations are performed. This results into noticeable increase in the execution time. The graph at $N = 5000$ resembles graph of time complexity $O(N \times \log(N))$.

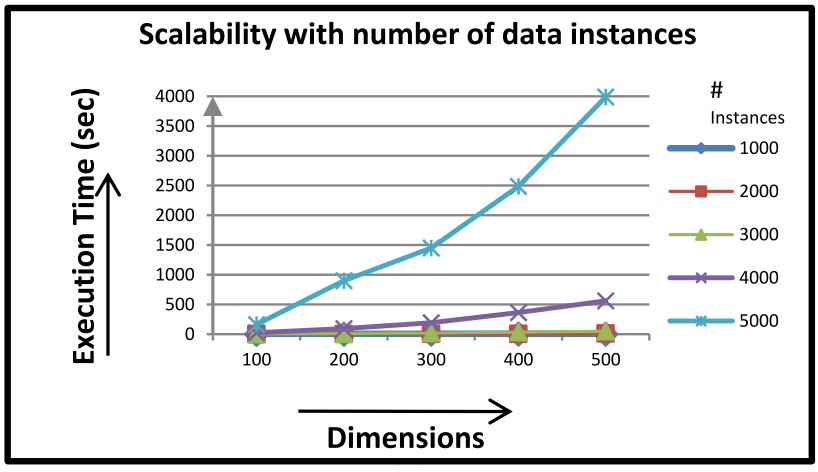
Experiments were also carried out by keeping distance threshold value constant at 0.5 for all dimensions, instead of estimating it through the first Step of CLUSLINK. Synthetic datasets used for the experiments had 1000 objects and dimensions varied from 100 to 500. Table 10 reflects that, the excellent runtime shown by CLUSLINK is only due to accurate distance estimation in the first Step of the algorithm.

5.4.2.2. Comparison of execution time with other algorithms. Using the synthetic data generator mentioned earlier, five synthetic datasets having 1000 objects were generated. Each dataset was having 5 subspace clusters containing 10 objects and 10 attributes randomly embedded in the data. The dimensionality of the five datasets was set to 100, 200, 300, 400 and 500 respectively. An implementation of CLIQUE, FIRES, P3C, PROCLUS and SUBCLU available in package 'Subspace' of R (Hassani, 2015) was used for the experiments. The parameter setting used during the experiments for these algorithms is mentioned in Table 5. CLUSLINK was executed with default values of object_density_threshold and attribute_threshold set to 5. A comparison of execution time of CLUSLINK with abovementioned subspace clustering algorithms is presented in Table 11.

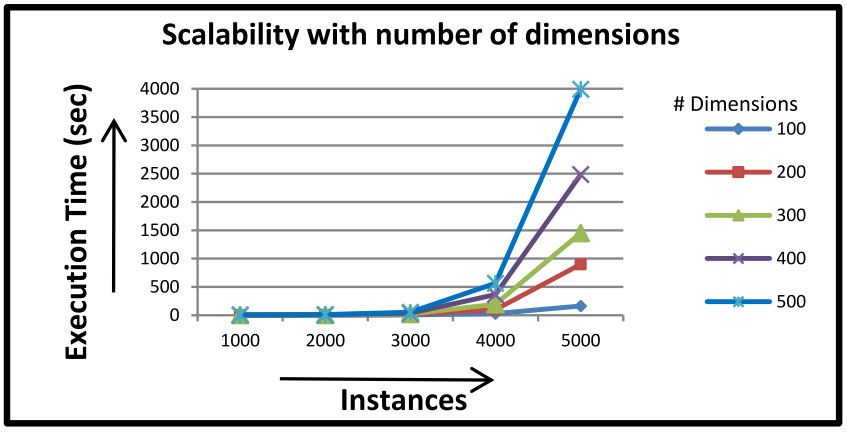
Result analysis

CLUSLINK has the best execution time in the group for all datasets. The method is highly scalable and shows negligible increase in runtime with increased dimensionality. The excellent execution time shown by CLUSLINK is due to application of greedy approach for selection of cluster members. The assignment of objects to clusters is done in a single pass over each attribute. As the one-dimensional clusters are formed by using distance threshold estimated automatically by the algorithm, the clusters are optimal and need no revision. Thus the best execution time shown by CLUSLINK is outcome of accurate distance threshold estimation and use of greedy technique.

5.4.2.3. Comparison of execution time on variable sized subspace clusters. This section shows comparison of performance on synthetic datasets having 1000 objects and 100 dimensions, by embedding 5, 10, 20 and 50 clusters each having sizes (i) 10 objects and 10 dimensions and (ii) 20 objects and 20 dimensions. The objective



(a)



(b)

Fig. 12. Scalability of CLUSLINK.

Table 8
Number of clusters output by various algorithms.

Dataset	Classes in dataset	#Clusters CLUSLINK	#Clusters CLIQUE	#Clusters FIRES	#Clusters P3C	#Clusters PROCLUS	#Clusters SUBCLU
Iris	3	3	47	1	3	12	4
Pima	2	66	71	2	1	12	4
Ecoli	8	10	87	1	1	12	4
Glass	6	86	175	1	3	12	4
Liver	2	26	31	1	1	12	4
Vowel	11	183	122	1	8	12	4

of the experiments is to check if varied dimensionality and count of embedded clusters affects the execution time. The outcome of the experiments is shown in Table 12. The results highlight that, CLUSLINK has the best execution time in the group and the algorithm shows linear growth with increased cluster count and dimensionality.

5.4.3. Execution time on real datasets

5.4.3.1. Comparison of execution time on default parameter setting. This section describes experimental results on real datasets mentioned in Section 5.2.2. The execution time of CLUSLINK was compared with other algorithms mentioned in Table 5. The parameter setting for the algorithms is also specified in the table. Default values of object_density_threshold and attribute_threshold for CLUSLINK were set to 5. Table 13 shows the comparison of processing time on real datasets. The table reflects that CLUSLINK shows the least execution time in the group in most of the cases. The execution time of CLUSLINK is comparable to SUBCLU. However, as shown in Fig. 11, the quality of results produced by SUBCLU is very poor on default parameter setting.

5.4.3.2. Comparison of execution time with other algorithms on optimal parameter setting. In this section, the results of CLUSLINK are compared with the optimal results reported in Müller et al. (2009a). Table 14 presents the comparison of execution time taken by CLUSLINK and optimal results for other algorithms on PIMA and VOWEL. The authors in Müller et al. (2009a) executed the experiments on a compute cluster containing compute nodes having four Opteron quad core CPUs @ 2.3 GHz, 1.5GB RAM. They restricted runtime for each processing to 30 min. CLUSLINK was executed a computer having CPU P6200, 2 GB RAM and 2.13 GHz. frequency. The hardware configuration used for executing CLUSLINK is low

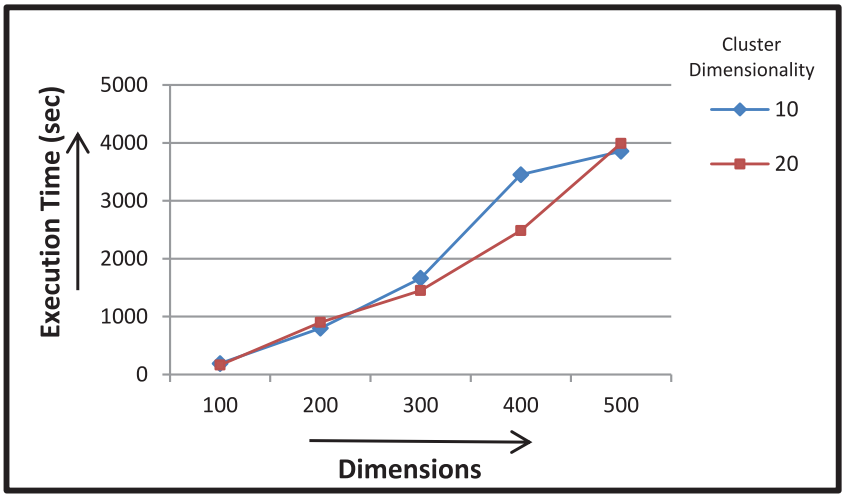


Fig. 13. Scalability with changed cluster dimensionality.

Table 9
Number of output clusters for real datasets.

Algorithm	Number of output clusters	
	PIMA (2 Classes)	LIVER (2 Classes)
CLUSLINK	66	26
CLIQUE	201	19
DOC	17	13
MINECLUS	3	32
SCHISM	21	68
SUBCLU	325	64
FIRES	1	1
INSCY	3	130
PROCLUS	3	2
P3C	1	1
STATPC	27	4

Table 10
Execution time by CLUSLINK with and without distance estimation.

#Dimensions	Execution time (second) after distance estimation	Execution time (second) without distance estimation
100	0.57	12.86
200	1.37	81.65
300	1.76	210.07
400	2.79	348.32
500	3.46	426.62

compared to that used in Müller et al. (2009a). Still the execution times of all algorithms are compared with CLUSLINK to draw conclusion about the efficiency approximately.

6. Conclusion and future scope

Every data mining model operates based on a set of parameters. The processing control is generally encapsulated in many arbitrary parameters. These parameters are the deciding factors to obtain accurate results and in an efficient manner. However, such threshold-based data mining poses another requirement on the process i.e. need of an expert user who has extensive knowledge of the algorithm as well as data to be processed. Although parameter-laden algorithms are necessary to control noise and unwanted results, they work as a double-edge sword. First, it restricts the capability of a data mining algorithm to find novel and interesting knowledge due to constraining parameters values which are based on incomplete knowledge of a non-expert. Second, it is difficult to compare different approaches which work on different kind of parameterizations. Generally the parameters are estimated using the full dataset and hence tend to overfit the test data and are not generalized for real unseen data. While there are some recent techniques aiming at automatic tuning of parameters, these techniques are themselves based on some parameters which may result into an infinite regression.

The work presented in this paper is a step towards parameter free data mining. The proposed algorithm makes use of knowledge embedded in the data itself for estimating distance threshold for numerical data. The method uses separate distance threshold for each dimension rather than using global distance threshold for all attributes. The algorithm has proven to be very is very effective on sparse high dimensional datasets and produces highly accurate results on synthetic and real datasets. The experimental evaluation of the quality and runtime of the algorithm demonstrates that, the algorithm accurately estimates the distance threshold for clustering in subspaces. The greedy approach employed by the method reduces exhaustive database scans required to find candidate subspace projections. The algorithm has time complexity of $O(N \times \log(N))$. On some of the real datasets, the count of output clusters is comparatively high. CLUSLINK can be improved further to adaptively merge neighboring clusters if they fall at a short distance from each other.

Future direction in this field could be estimation of other important parameters such as density threshold for clustering. The

Table 11
Execution time (in second) on synthetic datasets having 1000 objects.

Dimensions	CLUSLINK	CLIQUE	FIRES	P3C	PROCLUS	SUBCLU
100	0.98	0.58	40.34	334.37	24.23	4.74
200	1.37	1.65	246.33	648.10	25.18	37.05
300	1.76	7.49	520.78	2502.03	54.48	62.80
400	2.79	14.07	1757.3	2768.17	244.50	156.86
500	3.46	25.15	5262.71	1888.22	123.56	74.18

Table 12
Execution time (in second) by varying cluster count and dimensionality.

Cluster Dimensions Obj. X Dim.	Number of embedded Clusters	CLUSLINK	CLIQUE	FIRES	P3C	PROCLUS	SUBCLU
10 × 10	5	0.86	2.45	65.98	164.63	12.51	27.12
10 × 10	10	1.06	1.95	83.43	258.72	54.07	14.73
10 × 10	20	1.64	2.12	74.72	115.69	23.03	18.67
10 × 10	50	5.03	8.24	78.80	844.73	19.93	11.76
20 × 20	5	1.16	2.78	26.53	669.02	39.72	11.31
20 × 20	10	1.83	6.26	82.81	N. A.*	25.82	20.53
20 × 20	20	4.48	22.76	79.24	N. A.*	15.21	14.87
20 × 20	50	19.10	227.46	91.63	N. A.*	18.26	13.3

N.A.*- P3C could not finish within 30 min; hence execution time is not available.

Table 13
Execution time (in second) on real datasets (with default parameter setting).

Dataset	CLUSLINK	CLIQUE	FIRES	P3C	PROCLUS	SUBCLU
Iris	0.01	0.05	0.05	0.03	0.05	0.01
Pima	0.11	0.25	0.20	1.06	0.16	0.11
Ecoli	0.03	0.20	0.08	0.14	0.10	0.03
Glass	0.11	0.21	0.05	0.18	0.05	0.04
Liver	0.03	0.06	0.06	0.07	0.06	0.05
Vowel	0.61	1.05	1.17	43.89	0.47	0.25

Table 14
Execution time (in second) on real datasets (with optimal parameter setting).

Algorithm	PIMA Runtime(sec)	LIVER Runtime(sec)
CLUSLINK	0.11	0.03
CLIQUE	203	15
DOC	51,640	1625
MINECLUS	62	1954
SCHISM	250	Not available
SUBCLU	58,718	47
FIRES	360	46
INSCY	33,531	234
PROCLUS	109	31
P3C	141	32
STATPC	4657	781

distance threshold estimation method can be used for other traditional and subspace clustering algorithms. The method can be extended to handle categorical and binary data. The algorithm is intended to process data sets containing no missing values. The distance threshold estimation function can be modified to handle data that may have missing values. In this work, one-dimensional subspace clusters are expanded across remaining dimensions by applying set intersection operation on object indices, which are of integer type. The object matching step can be made faster by using graph data structures and a suitable graph searching method. The algorithm can be further modified to output hierarchy of subspace clusters which can reveal interesting relationships between output clusters.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Competing Interest

None.

Credit authorship contribution statement

Bhagyashri Abhay Kelkar: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing - original draft. **Sunil F. Rodd:** Project administration, Resources, Supervision, Validation, Writing - original draft, Writing - review & editing. **Umakant P. Kulkarni:** Project administration, Resources, Supervision, Validation, Writing - original draft, Writing - review & editing.

References

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Record, 27*(2), 94–105. https://doi.org/10.1145/276305.276314.

Bellman, R. (1961). *Adaptive control processes*. Princeton.

Chang, J.-W. (2005). *A new cell-based clustering method for high-dimensional data mining applications* (pp. 391–397).

Cheng, C.-H., Fu, A. W., & Zhang, Y. (2004). *Entropy-based subspace clustering for mining numerical data* (pp. 84–93).

Deng, Z., Choi, K.-S., Jiang, Y., Wang, J., & Wang, S. (2016a). A survey on soft subspace clustering. *Information Sciences, 348*, 84–106. https://doi.org/10.1016/j.ins.2016.01.101.

Deng, Z., Choi, K. S., Chung, F. L., & Wang, S. (2010). Enhanced soft subspace clustering integrating within-cluster and between-cluster information. *Pattern Recognition*. https://doi.org/10.1016/j.patcog.2009.09.010.

Deng, Z., Choi, K. S., Jiang, Y., Wang, J., & Wang, S. (2016b). A survey on soft subspace clustering. *Information Sciences*. https://doi.org/10.1016/j.ins.2016.01.101.

Dua, D., & Karra Taniskidou, E. (2017). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.

Goil, S., Nagesh, H., & Choudhary A. (1999). MAFIA: Efficient and scalable subspace clustering for very large data sets. *Discovery and Data Mining* https://doi.org/CPDC-TR-9906-010.

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: concepts and techniques* Data Mining: Concepts and Techniques https://doi.org/10.1016/C2009-0-61819-5 .

Hassani, M. (2015). Package "subspace" - Interface to OpenSubspace. Retrieved from https://cran.r-project.org/web/packages/subspace/index.html

Huang, X., Ye, Y., Guo, H., Cai, Y., Zhang, H., & Li, Y. (2014). DSKmeans: A new kmeans-type approach to discriminative subspace clustering. *Knowledge-Based Systems, 70*, 293–300. https://doi.org/10.1016/j.knosys.2014.07.009.

Huang, X., Ye, Y., Xiong, L., Lau, R. Y. K., Jiang, N., & Wang, S. (2016). Time series k-means: A new k-means type smooth subspace clustering for time series data. *Information Sciences, 367–368*, 1–13. https://doi.org/10.1016/j.ins.2016.05.040.

Kailing, K., Kriegel, H.-P., & Kröger, P. (2004). Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 2004 SIAM international conference on data mining* (pp. 246–256). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972740.23.

Keogh, E., Lonardi, S., & Ratanamahatana, C. A. (2004). Towards parameter-free data mining. In *Proceedings of the 2004 ACM SIGKDD international conference on knowledge discovery and data mining - KDD '04* (p. 206). ACM Press. https://doi.org/10.1145/1014052.1014077.

Kriegel, H.-P., Kröger, P., & Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*. https://doi.org/10.1145/1497577.1497578.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(1–2), 83–97. https://doi.org/10.1002/nav.3800020109.

Lakshmi, B. J., Madhuri, K. B., & Shashi, M. (2017a). An efficient algorithm for density based subspace clustering with dynamic parameter setting. *International Journal of Information Technology and Computer Science, 9*(6), 27–33. https://doi.org/10.5815/ijitcs.2017.06.04.

Lakshmi, B. J., Shashi, M., & Madhuri, K. B. (2017b). A rough set based subspace clustering technique for high dimensional data. *Journal of King Saud University - Computer and Information Sciences* https://doi.org/10.1016/j.jksuci.2017.09.003.

MacQueen, J. (1967). *Some methods for classification and analysis of multivariate observations*. 5th Berkeley Symposium on Mathematical Statistics and Probability 1967 https://doi.org/citeulike-article-id:6083430.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY: Cambridge University Press.

Müller, E., Assent, I., Günnemann, S., Gerwert, P., Hannen, M., Jansen, T. et al. (2011). A framework for evaluation and exploration of clustering algorithms in subspaces of high dimensional databases. (pp. 347–366).

Müller, E., Assent, I., Krieger, R., Günnemann, S., & Seidl, T. (2009a). DensEst: density estimation for data mining in high dimensional spaces. In *Proceedings of the 2009 SIAM international conference on data mining* (pp. 175–186). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972795.16.

Müller, E., Günnemann, S., Assent, I., & Seidl, T. (2009b). Evaluating clustering in subspace projections of high dimensional data. In *VLDB Endow* (pp. 1270–1281). https://doi.org/10.14778/1687627.1687770.

Nagesh, H. S., Goil, S., & Choudhary, A. (2000). A scalable parallel subspace clustering algorithm for massive data sets. In *Proceedings 2000 international conference on parallel processing* (pp. 477–484). IEEE Comput. Soc.. https://doi.org/10.1109/ICPP.2000.876164.

Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: A review. *SIGKDD Explorations Newsletter, 6*(1), 90–105 https://doi.org/10.1145/1007730.1007731.

Patrikainen, A., & Meila, M. (2006). Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering, 18*(7), 902–916. https://doi.org/10.1109/TKDE.2006.106.

Procopiuc, C. M., Jones, M., Agarwal, P. K., & Murali, T. M. (2002). A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD '02* (p. 418). ACM Press. https://doi.org/10.1145/564691.564739.

Sapatnekar, S. S. (2011). Overcoming variations in nanometer-scale technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 1*(1), 5–18. https://doi.org/10.1109/JETCAS.2011.2138250.

Sim, K., Gopalkrishnan, V., Zimek, A., & Cong, G. (2013). A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-012-0258-x.

Struski, Ł., Tabor, J., & Spurek, P. (2018). Lossy compression approach to subspace clustering. *Information Sciences, 435*, 161–183. https://doi.org/10.1016/j.ins.2017.12.056.

Vidal, R. (2011). A tutorial on subspace clustering. *IEEE Signal Processing Magazine*. https://doi.org/10.1109/MSP.2010.939739.

Wang, J., Deng, Z., Choi, K.-S., Jiang, Y., Luo, X., Chung, F.-L., et al. (2016). Distance metric learning for soft subspace clustering in composite kernel space. *Pattern Recognition, 52*, 113–134. https://doi.org/10.1016/j.patcog.2015.10.018.

Yao, J., Cao, X., Zhao, Q., Meng, D., & Xu, Z. (2018). Robust subspace clustering via penalized mixture of Gaussians. *Neurocomputing, 278*, 4–11. https://doi.org/10.1016/j.neucom.2017.05.102.

Zhang, H., Tang, Y., He, Y., Mou, C., Xu, P., & Shi, J. (2016). A novel subspace clustering method based on data cohesion model. *Optik, 127*(20), 8513–8519. https://doi.org/10.1016/j.ijleo.2016.06.004.

Zhu, B., Mara, A., & Mozo, A. (2015). CLUS: Parallel subspace clustering algorithm on spark. *Communications in Computer and Information Science, 539*, 175–185. https://doi.org/10.1007/978-3-319-23201-0_20.

Zhu, B. O., Mozo, A., & Ordozgoiti, B. (2016). *PSCEG: an unbiased parallel subspace clustering algorithm using exact grids*. ESANN.

Zhu, L., Cao, L., & Yang, J. (2011). *Soft subspace clustering with competitive agglomeration*. IEEE International Conference on Fuzzy Systems https://doi.org/10.1109/FUZZY.2011.6007424.

Zhu, L., Cao, L., Yang, J., & Lei, J. (2014). Evolving soft subspace clustering. *Applied Soft Computing Journal*. https://doi.org/10.1016/j.asoc.2013.03.002.