# A Meta-heuristic Density-based Subspace Clustering Algorithm for High Dimensional Data

<sup>1\*</sup>Parul Agarwal, <sup>2</sup>Shikha Mehta, <sup>3</sup>Ajith Abraham

<sup>1,2</sup>Department of Computer Science and Information Technology, Jaypee Institute of Information Technology, Sector-62, Noida, <sup>3</sup>Washington, United States Email- Address: parul.agarwal@jiit.ac.in, shikha.mehta@jiit.ac.in, abraham.ajith@gmail.com

\*Corresponding Author

Abstract: Subspace clustering is one of the efficient techniques for determining the clusters in different subsets of dimensions. Ideally, these techniques should find all possible non-redundant clusters in which the data point participates. Unfortunately, existing hard subspace clustering algorithms fail to satisfy this property. Additionally, with the increase in dimensions of data, classical subspace algorithms become inefficient. This work presents a new density-based subspace clustering algorithm (S\_FAD) to overcome the drawbacks of classical algorithms. S\_FAD is based on a bottom-up approach and finds subspace clusters of varied density using different parameters of the DBSCAN algorithm. The algorithm optimizes parameters of the DBCAN algorithm through a hybrid meta-heuristic algorithm and uses hashing concepts to discover all non-redundant subspace clusters. The efficacy of S\_FAD is evaluated against various existing subspace clustering algorithms on artificial and real datasets in terms of F\_Score and rand\_index. Performance is assessed based on three parameters: average ranking, SRR ranking, and scalability on varied dimensions. Statistical analysis is performed through the Wilcoxon signed-rank test. Results reveal that S\_FAD performs considerably better on the majority of the datasets and scales well up to 6400 dimensions on the actual dataset.

Keywords: subspace clustering algorithm; high dimensional clustering; meta-heuristic algorithm; Wilcoxon signedrank test; success rate ratio ranking; average ranking.

#### 1. Introduction

High-dimensional data means data with numerous features. High-dimensional data exist in various domains like recommendation systems, microarray data, social media data, and many more. Its rapid growth is seeking the attention of researchers and scientists for the last 2 decades(Steinbach et al., 2003)(Hans-Peter Kriegel et al., 2009). Clustering is a technique of finding groups of similar data based on their attributes. Effective clustering of the high-dimensional dataset is an important research issue in the field of data mining (L. Abualigah et al., 2020; L. M. Q. Abualigah, 2019). It has many challenges. Traditional clustering algorithms like K-Means, DBSCAN, OPTICS, etc.(Fahad et al., 2014) perform clustering in full-dimensional space. These algorithms attempt to find the cluster using all attributes given for each object of data. However, it becomes computationally expensive to apply these algorithms in the case of a large number of attributes/dimensions. This problem is called the "*curse of dimensionality*" (Steinbach et al., 2004). One of the reasons for this problem is that distance measure loses its importance as data points are sparse in high dimensional space. Clusters in such high dimensional space remain hidden under few relevant dimensions. Such relevant dimensions i.e., subsets of features are called subspaces. Irrelevant dimensions and noise completely mask the true clusters. Thus, classical clustering algorithms fail to determine clusters lying in different subspaces. One of the efficient ways of performing clustering in high-dimensional data is subspace clustering.

Data mining research communities have given a number of techniques to perform clustering in high dimensional data (Ira Assent, 2012) (L. Abualigah, Gandomi, et al., 2021). To determine clusters lying in different subsets of dimensions, subspace clustering algorithms (Domeniconi et al., 2004; Parsons et al., 2004) are employed. Subspace clustering determines a similar group of objects in a set of relevant dimensions of the dataset. Subspace clustering algorithms are broadly classified into two categories: Hard subspace and soft subspace (Deng et al., 2016). Hard subspace determines precise subspaces for various clusters. However, subspaces can be overlapping on various clusters. Müller et. al (Emmanuel Müller et al., 2009) divided hard subspace clustering algorithms. In order to obtain better subspace clusters, researchers developed a number of algorithms in the literature. Some of the relevant work is discussed below.

The evolutionary algorithm has been incorporated to form effective subspace clusters (Agarwal & Mehta, 2014, 2017) (L. Abualigah, Diabat, et al., 2021). The first study in this field was made by Sarafiset al. (Sarafis et al., 2003). Authors inculcated genetic operators for determining subspaces in the subspace clustering algorithm. The experiment is performed on 80-dimensional datasets but not compared with existing algorithms. In 2011, Li et al. (Lu et al., 2011) introduced a technique for soft projected clustering of high dimensional data using particle swarm optimization algorithm (PSO). The algorithm addresses the problem of variable weighting in the projected clustering approach. The algorithm is evaluated up to a 2000-dimensional dataset. Timmerman et al. (Timmerman et al., 2013) presented a new variant of the subspace k-means algorithm for shaping clusters in all dimensions. The proposed algorithm was assessed against k-means, factorial k-means, mixture factor analysis, and reduced k-means. The evaluation was made on adjusted rand index and cluster variance on a 9-dimensional dataset only. Lin et al.(Lin et al., 2014) suggested an evolutionary approach for determining subspaces. In order to improve clustering quality, the algorithm uses a hybrid genetic algorithm where local search is performed by PSO. The efficacy of the algorithm is assessed upon an error rate of up to 13-dimensional problems. Kaur and Datta(Kaur & Datta, 2015) presented an extended version of the SUBSCALE algorithm. The algorithm is examined based on F Score up to 6144-dimensional dataset. However, no result of 6144-dimensional dataset is depicted in the paper. In 2016, Kumar et al., (Kumar et al., 2016) projected the clustering algorithm to address the issues of big data. The algorithm (clusiVat) is compared with four existing algorithms on the basis of rand index and time for 500-dimensional data only. A survey on natureinspired algorithms with evolutionary strategies and applications is illustrated in (Agarwal & Mehta, 2014). A comprehensive analysis of nature-inspired algorithms is shown in (Agarwal & Mehta, 2017). Comparative analysis of these algorithms on clustering is depicted in (Agarwal & Mehta, 2015). To improve clustering quality, an enhanced version of the flower pollination algorithm is also employed (Agarwal & Mehta, 2016). Zhong and Pun (Zhong & Pun, 2020) have given a subspace algorithm to find similarities in data points and perform feature selection. The algorithm normalizes the column values and does not find overlapping clusters. The maximum dimensions evaluated were 6000. Yan et al. (Yan et al., 2020) have proposed a multi-view subspace clustering algorithm which is an extended version of the K-means algorithm. Clusters formed are mostly of spherical shape and include all samples hence maximal subspaces along with overlapping clusters and noisy points could not be found. An extended version of high dimensional data clustering using multivariate t-distribution is given by (Pesevski et al., 2018). The algorithm is evaluated on low dimension datasets i.e. maximum 8 dimensions. A novel subspace clustering technique for large number of samples is given by (Liu et al., 2020). Though algorithm discovers clusters in low time complexity, yet overlapping clustering could not be found. Another low rank subspace clustering algorithm is given by (Zhao et al., 2019). Maximum dimensions evaluated are in hundreds only. Also, the algorithms developed are not compared with conventional subspace clustering algorithms.

Though various subspace clustering algorithms have been developed yet there exist many challenges in finding subspace clusters in high dimensional data. There is scope for improvement in clustering quality and finding overlapping subspace clusters. Existing algorithms are unable to find maximal subspace i.e., subspaces without redundant information. It should exclude noisy data points from clusters. Existing hard subspace clustering algorithms are not able to deal with high dimensional data i.e., data with thousands of attributes. Most of the existing studies for high dimensional clustering have been assessed on few hundreds of attributes only. Additionally, these algorithms require normalization of data in range 0-1 before performing clustering. Normalization is usually min-max type and doesn't handle outliers properly. Also, existing algorithms have limited capability to find clusters of varied densities. Generally, clusters found in subsets of dimensions are of the same density. This might cause few redundant data points to be included in a cluster or some relevant points left out from the cluster.

The above reasons draw the motivation for developing a new subspace clustering algorithm. This paper is an extended version of work presented in (Agarwal & Mehta, 2019b). In previous work (Agarwal & Mehta, 2019b), the subspace algorithm presented is integrated with a differential evolution algorithm. However, the algorithm gets stuck in local solutions due to its limited capability to explore the complete search space. Also, it lags in maintaining a judicious balance of exploration and exploitation. One of the reasons is that the parameters of DBSCAN are not well found. These shortcomings are taken care in the presented work. Thorough analysis of the results and statistical analysis substantiates the performance of S\_FAD with respect to subspace\_DE (subspace with differential evolution) (Agarwal & Mehta, 2019b) and other subspace clustering variants.

In this work a hybrid meta-heuristic subspace clustering algorithm named S\_FAD is proposed. In S\_FAD, a self-tuned DBSCAN algorithm is used to perform clustering. It begins clustering with one-dimensional data, once clusters in each dimension are formed, their details are stored in the hash table. S FAD uses the concept of hashing for finding maximal subspace clusters. In each maximal subspace, DBSCAN algorithm is executed to form clusters. It employs a bottom-up subspace search method to determine subspaces. The self-tuned version of the DBSCAN algorithm is introduced in this work, wherein input parameters of the DBSCAN algorithm are automatically determined based on the dataset. This is achieved by a hybrid meta-heuristic algorithm named as FAD algorithm(Agarwal & Mehta, 2019a). The FAD is a combination of flower pollination, artificial bee colony, and differential evolution algorithm (named ABC\_DE\_FP in (Agarwal & Mehta, 2019a)). Experimentally it is observed that the S FAD algorithm can find clusters up to 6400-dimensional dataset. It overcomes the shortcomings of the bottom-up approach by automatically determining the optimized parameters suitable for a given dataset. It is successful in determining the overlapping subspace clusters. Also, the algorithm does not need any parameter priory such as the number of clusters or number of subspaces. S\_FAD eliminates duplicate subspaces by determining the maximal subspaces through hashing. It obtains all possible clusters where each data point participates. Along with it, S\_FAD does not normalize the original dataset as done by most of the subspace clustering algorithms and finds arbitrary shaped clusters of varied densities.

Performance of S\_FAD is evaluated on standard artificial and actual datasets and compared with various conventional subspace clustering algorithms. Evaluation parameters for assessing the performance of the S\_FAD algorithm are F\_Score and rand\_index. Using F\_Score and rand\_index, S\_FAD is judged on the following measures: a)Average ranking, b) Success rate ratio ranking, c) Wilcoxon signed-rank test, d) Scalability in terms of dimensions. Results demonstrate that the proposed algorithm (S\_FAD) is able to handle the challenges of subspace and high dimensional clustering considerably.

The paper is organized under various sections. Section 2 describes the proposed algorithm in detail along with pseudocode. The experimental setup is depicted in section 3. Section 4 shows experimental results and the analysis of algorithms. Section 5 finally concludes the paper.

### 2. Subspace Clustering in High Dimensional Data

Clusters in high-dimensional data are mostly present in low dimension data. These dimensions may vary from cluster to cluster. Hence subspace clustering plays an important role in performing clustering in high dimensional data. Algorithms on subspace clustering explore the subspaces existing in datasets with two different search techniques (Parsons et al., 2004). These are top-down and bottom-up subspace search methods. The top-down approach is an iterative method that starts by finding clusters with full dimensions assuming all dimensions are of equal weights. Thereafter, according to the clusters formed, each dimension is allotted a certain weight. The modified weights of dimensions are used in the successive iteration for generating new clusters. The most crucial input parameter is the size of subspaces which is difficult to determine at an early stage. This approach has the drawback of finding disjoint subspaces of equal size. Some top-down-based search algorithms (Parsons et al., 2004) are PROCLUS, FINDIT, ORCLUS, etc. The bottom-up search method starts clustering from a single dimension. It follows the Apriori principle to reduce the search space. Only those dimensions containing dense units participate in the formation of higher subspaces. This approach finds overlapping clusters and subspace clusters of arbitrary shape. Input parameters used in this approach are density threshold and grid size. CLIQUE, DOC, etc. are subspace algorithms of a bottom-up approach.

### 2.1 Proposed Algorithm (S\_FAD)

A novel density-based meta-heuristic subspace clustering algorithm is proposed for clustering high-dimensional data. The naming convention provided to this algorithm is S\_FAD (subspace FAD) for ease of representation and comparison. S\_FAD algorithm is inspired by (Kaur & Datta, 2015) and uses a bottom-up strategy of subspace clustering (Parsons et al., 2004) which is based on the Apriori principle (Agrawal et al., 1996). Accordingly, the data points that do not participate in lower-dimensional clustering are eliminated from high dimensional subspace clustering

As shown in Figure 1, the algorithm begins by performing clustering on each attribute of the dataset (taking all tuples) using a self-tuned DBCAN algorithm (explained in the next subsection). Prior to clustering, every data point is allotted

a large unique arbitrary integer named as the signature. In the S\_FAD, this signature is considered as 15 digits natural number generated randomly for each record/tuple in the dataset. The signature of a cluster is the sum of signatures of data points belonging to the cluster. The large number is considered to avoid identical signatures of clusters. Cluster signatures act as the key to the hash table. The clustered data points with summation of signatures (of respective data points) and dimensions are noted in the hash table. The signature of dense points forms the search key of the hash table. If any entered signatures in the hash table are the same then attributes are combined in a single entry as shown in figure 2. (Signatures are merely used to match the same clusters formed in the various attributes. Instead of matching each data point, signatures are matched.) This step ensures that attributes merged are part of the same subspace. Thereafter rows with the same subspaces/dimensions in the hash table are merged. Thus, relevant maximum subspaces are created with dense points. Thereafter, clustering in each subspace is performed to find dense units by a self-tuned DBSCAN algorithm. Final clusters in high-dimensional datasets are obtained in each subspace.



(A1=attribute 1 of dataset;

P1,P2,P3.... = data point or Records;

C1,C2,C3 = clusters formed ;

X1, X2, X3= sum of signatures of data points associated with each cluster C1,C2,C3

e.g. X1=sum(sig(P1)+sig(P2)+sig(P3)) where sig(P1) is the signature composed of a 15 digit unique number)

Figure 1: Clustering in Single Dimensional Data



Figure 2: S\_FAD Algorithm

Pseudocode of the S\_FAD is presented in Algorithm 1. Assume a given dataset is X with m tuples and n attributes represented as  $\{d_1, d_2, d_3, ..., d_n\}$ .

Algorithm 1 (S\_FAD)

Randomly generate m large integers (each of 15 digits) called signatures and map them to every data point. Initialize a clear hash table with columns as the sum of signatures *sum*, set of dense points *D*, and subspace *S*.  $H = \{sum, D, S\}$ For every dimension *j* of dataset X Run self-tuned DBSCAN algorithm with its parameters extracted from FAD.

Add each clustered data point D (formed from step 4) and their respective sum of signatures sum(D) to cell array (data structure) DU.

```
DU = \{sum(D), D\}
```

```
For all entries of DU
```

```
if any key of hash table H.h<sub>x</sub>==sum //(h_x, h_y, h_z \text{ are entries of hash table})
add j dimension to subspace S of a hash table in row h<sub>x</sub>.S
```

else

```
adda new row in hash table H as { sum, D, j }
```

end if

```
end for
```

end for

```
for all rows of hash table H as \{h_x, h_y, h_z, ...\}
```

```
if subspaces of some entries are the same i.e. h_x.S == h_y.S == h_z.S == ...
append dense points D of h_y.D, h_z.D... to h_x.D
```

```
end if
```

end for for every entry  $h_x$  of hash table HExecute a self-tuned DBSCAN algorithm with its parameters extracted from FAD. end for Return clusters of maximal subspaces

### 2.2 Self-tuned DBSCAN using FAD

In the proposed algorithm, the DBSCAN (Density-based spatial clustering of applications with noise) algorithm is used to perform clustering in each dimension of the data as well as in maximal subspaces formed in a hash table. The advantage of the DBSCAN algorithm over the partition-based algorithm (Fahad et al., 2014) is that it has the ability to find clusters of arbitrary shapes and detect noisy points. Moreover, it clusters the datasets even without any former information of a number of clusters. Two input parameters used in the DBSCAN algorithm are *epsilon* ( $\varepsilon$ ) and *MinPts* ( $\tau$ ).  $\varepsilon$  is the distance measured between two points to form a neighbourhood. *MinPts* is the minimum number of points to form a cluster in the neighbourhood of any points within  $\varepsilon$  distance. The efficacy of the DBSCAN algorithm largely depends on the input parameters  $\varepsilon$  and *MinPts*. These are the sensitive parameters that vary from data to data and are hard to determine priory. Thus, with respect to datasets, the optimized values of these parameters are determined by a proposed meta-heuristic algorithm named FAD algorithm. Hence DBSCAN is named here as self-tuned DBSCAN as parameters are self-tuned in the algorithm.

FAD is a swarm intelligence based algorithm which is an amalgamation of flower pollination (FP)(X.-S. S. Yang, 2012), artificial bee colony algorithm (ABC)(Karaboga & Basturk, 2007), and differential evolution (DE) (Storn & Price, 1997) algorithm. FAD is a name given to the binary version of ABC\_DE\_FP algorithm (Agarwal & Mehta, 2019a) which was developed for a continuous optimization problem. It was established that ABC\_DE\_FP performed better as compared to existing meta-heuristic algorithms (Agarwal & Mehta, 2019a) on complex benchmark functions. Hence binary version of the algorithm is developed to optimize the parameter values of the DBSCAN algorithm i.e., *Minpts* and  $\varepsilon$ .

The FAD algorithm encodes each individual of the population in binary form. Each individual represents *MinPts* in bit string format (Karami & Johansson, 2014). (Since *MinPts* represent a discrete value hence binary version is applied instead of continuous version). The number of dimensions 'D' is the number of bits required to represent a decimal number (*MinPts*). In FAD, the population of individuals is randomly initialized. Each individual represents the food source. The fitness of the food source is computed through an objective function. Here *purity* (shown in Equation 1 is the objective function used to get the best values of *MinPts* and  $\varepsilon$ . Purity is an external validation criterion for measuring the quality of clusters formed. Higher the purity, better the *Minpts* and  $\varepsilon$ . To compute purity, the most frequent class in a cluster is assigned to that cluster. Thereafter, the correctness of the class assignment to a cluster is determined by counting the number of data points assigned appropriately divided by a total number of points in the dataset 'N'.

$$purity = \frac{\sum_{j=1}^{k} \max_{1 \le i < l} (class_i \cap cluster_j)}{N}$$
(1)

Where k is the actual cluster number,  $i^{th}$  is the class already defined in the dataset and  $j^{th}$  is the cluster formed from the clustering algorithm. The numerator term of Equation 1 signifies that  $j^{th}$  cluster has a majority of data points of  $i^{th}$  class such that  $i^{th}$  class is assigned to  $j^{th}$  cluster.

The working of FAD algorithm is divided into three phases: Employed bee, onlooker bee, and scout bee. In the employed bee phase, each food source is updated with mutation (Equation 2) and crossover strategies (Equation 3) of differential evolution algorithm:

$$v_i = x_a + F.(x_b - x_c)$$
 (2)

Where  $v_i$  is the mutant food source, i=1,2,...N,  $x_a$ ,  $x_b$ , and  $x_c$  are all distinct initial (target) food sources of the same population such that  $x_a$ ,  $x_b$ ,  $x_c \in N$ . Also, *i* is different from *a*, *b* and *c*. *F* is a scaling factor. In the present algorithm, *F* is randomly generated between the uniform distribution range *betamin* and *betamax*. Crossover is performed on mutant and target food source. New food source generated is called trial food source  $u_{ij}$ .

$$u_{ij} = \begin{cases} v_{ij}, & \text{if } rand[0,1] \le Cr \text{ or } j == j_{rand} \\ x_{ij}, & \text{otherwise} \end{cases}$$
(3)

Where *j* is the selected index of dimension *D*,  $j_{rand}$  is randomly chosen index from *I* to *D*. This is made to ensure that resultant vector  $u_{ij}$  receive the minimum single mutant vector  $v_{ij}$ . *Cr* is the rate of crossover which controls the choice of target or mutant vectors. Fitness of the updated solution is computed in the form of purity. If the purity value (fitness value) of the new solution is better than solution is updated in the population else its trial value is incremented by 1. Trial represents a counter maintained for each food source in the population. If a food source gets updated in the population, then its respective trial is reset to 0 else it is incremented. After the employed bee phase, food sources are updated in the onlooker bee phase. In this phase, the global and local search of food sources is controlled by switch probability *p*. Global search is made through the global pollination process of flower pollination algorithm (using Equation 4)

$$x_i^{t+1} = x_i^t + L (x_i^t - g_*)$$
(4)

Where  $x_i^t$  is the *i*<sup>th</sup> food source at *t*<sup>th</sup> generation,  $g_*$  is the current best food source and *L* is a levy flight distribution given in (Pavlyukevich, 2007; X. S. Yang, 2012). If the algorithm switches to the local search process, then the mutation strategy of DE given in Equation 2 is applied. Thereafter crossover technique of DE given in Equation 3 is applied to obtain the new food source. If the fitness of the food source is improved then it is updated in a population otherwise its trial counter is incremented. Food sources with the highest fitness value (nectar amount) are memorized. If the trial counter of any food source outstrips the *Limit* value, then that particular food source is discarded and the scout bee explores for a new food source. The process continues unless the termination condition is satisfied. Here termination condition is maximum iterations.

For calculating the fitness value of any food source,  $\varepsilon$  is calculated using respective *Minpts*. Epsilon value ( $\varepsilon$ ) is determined analytically (Daszykowski et al., 2001) from *Minpts* and data matrix *x* using the following Equation 5:

$$Eps = \left(\frac{(\prod_{i=1}^{\max(x) - \min(x)} i) * k * \Gamma(0.5n+1)}{m\sqrt{\pi^n}}\right)^{1/n}$$
(5)

Where *m* is the number of tuples and *n* is the dimension of each tuple of data matrix *x*. *k* is the *MinPts*,  $\Gamma$  is a gamma function which generalizes the factorial of a given argument.

FAD algorithm initializes the following input parameters:

- Iterations- Number of times each individual is updated by an algorithm.
- *D* Number of bits used to represent MinPts.
- *N* Population size
- betamin, betamax- Range of solution
- Trial- a counter incremented when a solution does not improve
- *Limit* threshold value for calling scout bees (randomly initializing individual)
- *p* switch probability for selecting local and global search

The flowchart of FAD is shown in figure 3. In DE, there are only two update functions of chromosomes i.e. mutation and crossover. While in FAD, there are three phases where mutation and crossover are used for exploitation, levy flight distribution is used for exploration. These steps in FAD algorithms helps in maintaining an appropriate balance between the local and global search and hence give better results as compared to DE.



Figure 3: FAD Algorithm

### 3. Experimental Setup

The proposed algorithm (S\_FAD) is compared with various well-known subspace clustering algorithms on actual as well as artificial datasets. Existing subspace algorithms (Emmanuel Müller et al., 2009) compared are SCHISM (Sequeira & Zaki., 2004), CLIQUE (Road & Jose, 1998), MINECLUS (Yiu & Mamoulis, 2003), DOC (Procopiuc, 2002), INSCY(I. Assent et al., 2008), SUBCLU (Kailing et al., 2004), FIRES(H.-P. Kriegel et al., 2005), P3C (Moise et al., 2006), PROCLUS (Aggarwal et al., 1999), and STATPC (Moise & Sander., 2008). S\_FAD is also compared with the subspace clustering algorithm given by (Kaur & Datta, 2015) called SUBSCALE algorithm and (Agarwal & Mehta, 2019b) named SUBSPACE\_DE. Conventional subspace clustering algorithms are implemented on an extended WEKA toolbox provided by (E Müller et al., 2009). This toolbox provides space for executing various subspace clustering algorithms. S\_FAD, SUBSCALE, and SUBSPACE\_DE algorithms are implemented on MATLAB R2013a. Evaluation metrics, parameter setting and dataset description used for comparison are described in subsections.

#### 3.1 Evaluation Metrics

Performance evaluation of the proposed algorithm (S\_FAD) against various subspace clustering algorithms is made through classification. The true cluster labels (T) for data items in each dataset are already known. Each algorithm predicts the label for each data item in the dataset. Predicted and true labels for each data item in the dataset form a confusion matrix. This confusion matrix helps in determining various evaluation measures of clustering. In this study, evaluation measures used for testing the performance of S\_FAD against various subspace algorithms are rand\_index and F\_Score. Subsequently average ranks and success rate ratio ranks are computed using F\_Score and rand\_index. Evaluation measures employed are briefly described as below:

- Rand index- It is the evaluation measure for determining the quality of clusters formed by the clustering algorithm. It is defined by the ratio of correctly labelled data items to the total number of data items. Higher the rand\_index, betters the algorithm. A good clustering algorithm predicts the cluster that best portrays the true cluster and thus posses' high-quality clusters.
- F Score- This measure defines that the predicted cluster should cover maximum data items from the true cluster and minimum items from other clusters (Emmanuel Müller et al., 2009). F\_Score is expressed in Equation 6:

$$F \ score = \frac{1}{m} \sum_{i=1}^{m} \frac{2*recall(T_i)*precision(T_i)}{recall(T_i)+precision(T_i)} \tag{6}$$

where 'm' is the number of true clusters. High precision corresponds to the least number of items from another cluster while high recall signifies maximum coverage of items from the true cluster. High F\_Score denotes good clustering quality.

• Average ranks- Average ranking is the primitive and simple method to rank algorithms. This ranking is defined in (Brazdil & Soares, 2000). According to this method, the rand\_index and F\_Score defined for each dataset by every algorithm are sorted and assigned the ranks. It is worth noting that the F\_Score and rand\_index are treated independently while ranking the algorithms. The algorithm possessing the highest value will be assigned rank 1, the second-highest will be assigned rank 2, and so on for each dataset independently. Thereafter, the overall average rank of each algorithm is computed by taking the mean of ranks on all datasets. Let us consider  $r_j^i$  be the  $j^{th}$  algorithm rank for  $i^{th}$  dataset. The average rank of each algorithm on total 'n' datasets is computed using the following Equation 7:

$$r_{j} = \frac{\sum_{i=1}^{n} r_{j}^{i}}{n}$$
(7)

• Success Rate Ratio Ranks (SRR)- SRR is a ranking method where the ratio of success rates is considered between the pairs of algorithms (Brazdil & Soares, 2000). This method is useful in estimating the magnitude of difference in rand\_index (RI) obtained by algorithms. Also, this method aids in determining the significant differences in algorithms. If the difference is not significant then the success rate ratio is close to 1. SRR ranking starts by taking one algorithm and one dataset at a time and calculating its rand\_index ratio with the rest of the algorithms. This ratio is computed by following Equation 8:

$$SRR_{j,k,j\neq k}^{i} = \frac{RI_{j}^{i}}{RI_{k}^{i}}$$

$$\tag{8}$$

Where '*i*' is the dataset, '*j*' is the algorithm for which success rate is calculated and '*k*' is the compared algorithm different from '*j*'. In this way, the success rate ratio is computed for the algorithm '*j*' with respect to algorithm '*k*' on *i*<sup>th</sup> dataset. Similarly, SRR is computed for all datasets taking the same pair of algorithms. Thereafter, SRR for all datasets are added so as to obtain an overall SRR ratio for the given pair of algorithms using the following Equation 9:

$$SRR_{j,k,j\neq k} = \frac{\sum_{i=1}^{n} SRR_{j,k,j\neq k}^{i}}{n}$$

$$\tag{9}$$

Where 'n' is the number of datasets. In this way, the success rate of the algorithm 'j' is computed over the algorithm 'k'. Similarly, the success rate of the algorithm 'j' is computed over each of the left algorithms. After computing the success rate ratio for all datasets and summing them, the mean success rate ratio is calculated for an algorithm 'j' using the following Equation 10:

$$SRR_j = \frac{\sum_k SRR_{j,k,j\neq k}}{m-1} \tag{10}$$

Where 'm' is the total number of compared subspace algorithms. In this way, SRR for each algorithm on all datasets is computed and ranked in descending order as the higher the rand\_index or F\_Score, the better the algorithm.

#### **3.2 Parameter Tuning**

Parameter values used in the S\_FAD algorithm are shown in Table 1. Values of these parameters are decided after replicating many experiments. The best values are chosen on the basis of an algorithm's performance.

Parameter	Values
Signature	15 digit number
Crossover(cr)	0.25
[betamin, betamax]	[0.2,0.8]
Switch probability(p)	0.7
Limit	Population Size* number of decision variables
D	8
D	

### Table 1: S\_FAD Parameter Settings

The two other parameters used in the FAD algorithm are population size and iterations. These parameters are tuned by repeating the set of experiments on datasets. Population size and the number of iterations are decided by fixing any one component and varying the other. Table 2 to Table 5 depict the parameters for the best population size and iteration value to be selected for further experiments of the S\_FAD algorithm. Datasets used for tuning the parameters are 'D50' artificial dataset and 'shape' actual dataset of 50 and 17 dimensions respectively.

Population	10	20	30	40	50
F_Score	0.46	0.71	0.61	0.78	0.61
Rand_index	0.77	0.92	0.87	0.94	0.88

Table 5. Varying nerations with 40 population size (for artificial dataset)										
Iterations	10	20	30	40	50	100				
F_Score	0.78	0.85	0.82	0.82	0.80	0.82				
Rand_index	0.94	0.97	0.95	0.95	0.95	0.96				

Table 4: Varying population size with 10 iterations (for actual dataset)									
Population	10	20	30		40	50			
F_Score	0.29	0.37	0.4	6	0.47	0.47			
Rand_index	0.4	0.59	0.7	1	0.72	0.71			
	Tab	le 5: Varying Ite	rations with 40 po	pulation size (f	or actual datas	et)			
Iterations	10	20	30	40	50	100			
F_Score	0.47	0.52	0.46	0.51	0.48	0.5			
Rand index	0.72	0.78	0.71	0.76	0.72	0.75			

Tables 2 and 3 correspond to the parameter tuning on an artificial dataset. It is perceived from Table 2 that for the population size of 40, the algorithm shows better rand\_index and F\_Score. Using population size as 40, Table 3 depicts variation in iterations. 20 iterations on the artificial dataset are selected for further experiments. In the case of the actual dataset (assume shape dataset), S\_FAD's performance variation in population size is shown in Table 4. It is observed that the algorithm perform better for 40 population size. Adopting this population size, variation in iterations is shown in Table 5. 20 iterations on the actual dataset are selected for further experiments. For the ease of implementation on higher-dimensional datasets (500 and above dimensions) (Bache & Lichman, 2006), population size is assumed to be 10 with 20 iterations.

Parameter values of other compared algorithms are obtained from their respective work. For SUBSCALE algorithm, parameter values are taken from the original author's work (Kaur & Datta, 2015). Set of parameter values for existing subspace clustering algorithms are defined in (E Müller et al., 2009).

### 3.3 Dataset Description

In order to establish the efficacy of the S\_FAD algorithm against various subspace clustering algorithms, it is evaluated on artificial and actual datasets. The description of datasets is shown in Table 6 and Table 7. Artificial datasets used in this research work are composed of 10 to 75 dimensions. These datasets are used in work (Emmanuel Müller et al., 2009). Table 6 (also used in (Agarwal & Mehta, 2019b)) gives the name of artificial datasets along with their size. Real datasets used in this research work are divided into two categories: small dimensions and high dimensions. These datasets are the standard datasets and have been used by various researchers for evaluation of the algorithm's performance (I. Assent et al., 2008; Moise & Sander., 2008; Sequeira & Zaki., 2004). Table 7 depicts the number of dimensions and instances of each actual dataset used in experiments. The maximum dimensional dataset evaluated by S\_FAD is DrivFace with 6400 attributes.

Dataset	Dimensions	Records	No. of clusters
D10	10	1596	11
D15	15	1596	11
D20	20	1596	11
D25	25	1596	11
D50	50	1596	12
D75	75	1596	12
1			

Table 7: Description of Actual Datasets

Name of Dataset	No. of attributes(dimensions)	No. of Instances
LIVER DISORDER	6	345
DIABETES	8	768
GLASS	9	214
VOWEL	10	990
PENDIGITS	16	7494
SHAPE	17	160
BREAST	33	198
MADELON	500	4400
MICROMASS	1300	931
GISSETTE	5000	13500
DRIVFACE	6400	606

### 4. Results and Analysis

Thorough experiments are carried out for evaluating the performance of the proposed algorithm (S\_FAD) on small and high dimensional datasets. S\_FAD takes 30 independent runs on small dimensional datasets and the mean value is considered for comparison. For high-dimensional datasets (Madelon, Micromass, Gissette, and Drivface), 10 independent runs of S\_FAD are considered due to hardware limitations. F\_Score and rand\_index of various conventional algorithms except for SUBSCALE and SUBSPACE\_DE are obtained by re-implementing on the extended WEKA framework provided by (E Müller et al., 2009). While S\_FAD, SUBSCALE and SUBSPACE\_DE are implemented on Matlab.

In order to explore the experimental outcome, this section is comprised of two subsections: section 5.1 illustrates the comparison of S\_FAD algorithm with various conventional subspace algorithms on different datasets. Section 5.2 depicts results of S\_FAD on high-dimensional actual datasets.

### 4.1 Comparison of Proposed Algorithm (S\_FAD) with Conventional Subspace Clustering Algorithms

Figure 4 and 5 depict the actual values of S\_FAD and conventional subspace clustering algorithms for rand\_index and F\_Score respectively on artificial datasets. In these figures, results of conventional subspace clustering algorithms are adapted from (Agarwal & Mehta, 2019b; Emmanuel Müller et al., 2009). It can be observed from Figure 4 that S\_FAD attains more than 90% accuracy on the majority of artificial datasets. On small dimensions i.e. D10 dataset S\_FAD gives better performance than all existing algorithms except SCHISM where it lags by approximately 4%. On the D15 dataset, percentage enhancement in the accuracy of S\_FAD is much higher than percentage decrease from DOC (approximately 5%) and SCHISM (approximately 4%). S\_FAD lags by 4% only from INCY while superior to all other algorithms for D20. On the D25 dataset, S\_FAD depicts better performance than compared algorithms except for MINECLUS and SCHISM where percentage lag is by 8% and 2% respectively. On D50 and D75 dimensions, S-FAD outperforms all algorithms in terms of rand\_index. SCHISM and INSCY algorithms are capable of achieving good rand\_index but fail to provide results for 50 and 75-dimensional datasets. A similar kind of trend is exhibited on F\_Score which is shown in Figure 5.



Figure 4: Rand\_index of algorithms on artificial datasets



Figure 5: F\_Score of algorithms on artificial datasets

Figure 6 and 7 shows the results of proposed algorithm against various conventional algorithms on liver disorder, glass, diabetes, shape, breast and vowel datasets in terms of rand\_index and F\_Score respectively. It can be witnessed from Figure 6 that the proposed algorithm (S\_FAD) gives similar or better performance as compared to other various subspace clustering algorithms. For the glass dataset, S\_FAD lags by 7.1% total from CLIQUE, SUBCLU, and INCY, while enhances by overall 171.3% from the rest of the algorithms. On the Diabetes dataset, S\_FAD gives a comparable performance. It improves by overall 33.2% and worse by 6.9%. For the liver disorder dataset, S\_FAD outperforms all other subspace clustering algorithms in terms of rand\_index. For breast cancer and shape dataset, performance is very much similar to other algorithms. In the vowel dataset, S\_FAD total percentage enhancement is much higher than total percentage decrement (11% from CLIQU, SCHISM, INSCY). For the pendigits dataset, S\_FAD remains near to

the best performing algorithm. It can be witnessed from the above observation that though  $S_FAD$  does not outperform on all actual dataset; however, the average percentage enhancement of the algorithm is much higher than average percentage decrement. Thus,  $S_FAD$  performs above average on actual datasets. Analysis for  $F_score$  is similar to that of rand\_index.



Figure 6: Rand\_index of algorithms on actual datasets



Figure 7: F\_Score of algorithms on actual datasets

It can be noticed that SUBCLU could not cope with pendigits dataset as the dataset has 7494 instances. Thus, it can be concluded that in the terms of F\_Score and rand\_index, S\_FAD gives a good performance as compared to other subspace clustering algorithms.

In order to further establish the results, two well-recognized ranking methods - F\_Score and rand\_index are used for analysis (Brazdil & Soares, 2000). Also, the statistical significance of algorithms is portrayed using a Wilcoxon signed-rank test. The subsections included in this section are as follows. Section 4.1.1 depicts the average ranking of algorithms on artificial and actual datasets, section 4.1.2 shows the success rate ratio ranking of algorithms, section 4.1.3 describes Wilcoxon signed-rank test and final section 4.1.4 illustrates the scalability of S\_FAD in terms of data dimensionality.

### 4.1.1 Analysis on Average ranking

The average ranking of S\_FAD and various existing subspace algorithms on artificial datasets and actual datasets are computed independently based on F\_Score and rand\_index. Table 8 presents the average ranking of subspace algorithms on rand\_index and F\_Score values for artificial datasets (depicted in Figures 4 and 5). S\_FAD attains the first rank based on both rand\_index and F\_Score. SCHISM occupies the second rank in terms of rand\_index. Based on F\_Score, DOC occupies the second rank which belongs to cell-based category. It can be concluded from the above analysis that next to S\_FAD, cell-based algorithms depict approximately similar performances on artificial datasets while clustering-oriented-based algorithms occupy ranks in second-half positions (i.e. after 5<sup>th</sup> position).

Table 9 represents the average ranking of subspace algorithms on actual datasets in connection with rand\_index and F\_Score. CLIQUE algorithm is a winning algorithm based on rand\_index but in terms of F\_Score, it stands in the third last position. However, S\_FAD gives a very consistent behaviour and holds on the second position concerning both rand\_index and F\_Score. MINECLUS depicts the first and fifth positions with respect to F\_Score and rand\_index respectively. It can also be observed that two cell-based algorithms i.e., CLIQUE and MINECLUS depict extreme performance whereas S\_FAD is consistent. Additionally, except for these two cell-based algorithms (CLIQUE and MINECLUS), none of the subspace algorithms gives good performance in connection with rand\_index and F\_Score. Thus, S\_FAD shows sufficiently better results for both measures.

	Rand_index	F_Score		Rand_index	F_Score
S_FAD	1	1	CLIQUE	1	11
SCHISM	2	9	S_FAD	2	2
INSCY	3	7	DOC	3	4
SUBSPACE_DE	4	3	SCHISM	3	8
FIRES	5	4	MINECLUS	5	1
SUBSCALE	6	8	INSCY	5	5
MINECLUS	7	5	STATPC	7	6
DOC	8	2	PROCLUS	8	7
P3C	9	10	SUBCLU	9	10
PROCLUS	10	5	SUBSPACE_DE	10	3
StatPC	11	11	SUBSCALE	11	9
State	11	11	P3C	12	12
			FIRES	13	13

Table 8: Average rank of algorithms on artificial datasets Table 9: Average rank of algorithms on actual datasets

### 4.1.2 Analysis of Ranking on Success Rate Ratios

Success rate ratio (SRR) ranks of S\_FAD and other existing subspace clustering algorithms are calculated on artificial and actual datasets. Table 10 depicts the SRR rank of all subspace algorithms concerning rand\_index and

F\_Score on artificial datasets. It can be seen that the S\_FAD wins the among various subspace algorithm and stands at the first position. The second position is occupied by SUBSPACE\_DE with respect to rand\_index and F\_Score. These results indicate that density-based algorithms win over cell-based and clustering-oriented based algorithms.

Table 10:SRR Rank	of Algorithms on .	Artificial		Rand_index	F_Score
Datasets			S_FAD	4	5
	Rand_index	F_Score	CLIQUE	1	10
S_FAD	1	1	SCHISM	2	8
SUBSPACE_DE	2	2	INSCY	3	1
FIRES	3	5	STATPC	5	7
SUBSCALE	4	7	SUBSPACE_DE	6	6
DOC	5	3	DOC	7	3
MINECLUS	6	4	PROCLUS	8	4
РЗС	7	10	MINECLUS	9	2
PROCLUS	8	6	SUBCLU	10	11
SCHISM	9	9	РЗС	11	12
INSCY	10	8	FIRES	12	13
STATPC	11	11	SUBSCALE	13	9

Table 11: SRR Rank of Algorithms on Actual Datasets

Table 11 depicts the ranking of various subspace algorithms based on the success rate ratio on actual datasets. S\_FAD stands at the fifth and fourth position corresponding to F\_Score and rand\_index respectively. Conversely, its average rank is 2. The reason for such difference in average and SRR ranks is that S\_FAD could not perform very well for few low dimensional datasets like pendigits, shape, and vowel. However, S\_FAD gives a consistent performance as its ranks are approximately the same for rand\_index and F\_Score. The performances of CLIQUE, MINECLUS, DOC, PROCLUS, INSCY vary largely to rand\_index and F\_Score. CLIQUE algorithm holds the first position and gives the highest rand\_index but at the same time, it gives poor F\_Score on actual datasets. Whereas S\_FAD that gives a comparable performance for both rand\_index and F\_Score on actual datasets.

Also, it has been observed from the analysis that the overall percentage enhancement of S\_FAD is better as compared to the percentage decrease from other subspace clustering algorithms. Thus, it can be concluded that S\_FAD overall presents considerably good efficacy, better consistency, and reliability in view of rand\_index and F\_Score for the majority of small dimensional datasets. S\_FAD is a hard subspace clustering algorithm that performs clustering on high-dimensional datasets (shown in section 5.2). The next section discusses the statistical significance of obtained results using the Wilcoxon signed-rank test.

# 4.1.3 Statistical Significance of Results of proposed Algorithm (S\_FAD) vs. Other Subspace Clustering Algorithms

To further strengthen the performance of S\_FAD over conventional subspace clustering algorithms, a statistical hypothesis test is performed. This test is used to establish the statistical significance of results obtained through experiments. The test starts with the null hypothesis  $H_0$  and alternative hypothesis  $H_1$ . These two hypothesis claims as follows:

 $H_0$  = results of compared algorithms are statistically the same

 $H_1$  = results of compared algorithms are statistically not the same

For the problem given at hand, it is suitable to use a non-parametric test (Demšar, 2006) as there are no assumptions on population distribution. Wilcoxon signed-rank statistical test is used to determine the significant difference in results obtained by various subspace algorithms on each dataset. It finds the difference, overlooking the signs in rand\_index and F\_Score on artificial and actual datasets. Thereafter, algorithms are ranked and compared considering positive and negative differences. The significance level is assumed 5% i.e. the probability of rejecting the null hypothesis when it is true is 5%. In this work, the significant difference of S\_FAD algorithm with other subspace algorithms is determined based on rand\_index and F\_Score.

### • Results of Wilcoxon Signed Rank Test on Artificial Datasets

Tables 12, 13, and 14 show the Wilcoxon signed-rank test of the proposed algorithm (S\_FAD) versus density-based, cell-based, and clustering-oriented based subspace algorithms respectively on artificial datasets. It is examined from Table 12 that S\_FAD shows a significant difference from SCHISM on F\_Scores. However, no significant difference (H0 TRUE) is obtained in performances of S\_FAD versus cell-based algorithms on artificial datasets. This is because the value of parameter 'W' surpasses the critical value of the Wilcoxon signed-rank test.

Table 13 illustrates the outcomes of the Wilcoxon signed-rank test of S\_FAD against density-based algorithms. It is observed that for FIRES, SUBSCALE, and SUBSPACE\_DE algorithms, parameter 'W' is equal to critical value i.e. 0, which means S\_FAD outperformed all artificial datasets. Hence, it shows a significant difference with respect to these two algorithms. There is no significant difference between the S\_FAD and INSCY algorithm with respect to rand\_index and F\_Score. S\_FAD shows better results for the majority of datasets. Thus, it can be concluded that S\_FAD shows overall better performance than existing density-based algorithms. Table 14 represents a statistical test of S\_FAD against clustering-oriented subspace algorithms. It is observed that S\_FAD surpasses subspace clustering-oriented based algorithms i.e., P3C, PROCLUS, and, STATPC in regards to rand\_index and F\_Score. It is also seen that W-=0 which means S\_FAD outperforms on all datasets and depicts higher rand\_index and F\_Score than clustering-oriented based algorithms.

	Table 1	Table 12: Wilcoxon signed-rank test of S_FAD vs. Cell-based							
	DOC		MINECLUS		SCHISM				
	Rand_index	F_Score	Rand_index	F_Score	Rand_index	F_Score			
W+	20	16	18	18	15	21			
<b>W-</b>	1	5	3	3	6	0			
W (min(W+,W-)	1	5	3	3	6	0			
Critical Value(From Table 24)	0	0	0	0	0	0			
Significant Difference	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 False			

	Table 13:Wilcoxon signed-rank test of S_FADvs. Density-based							
	FIRES	INCY			SUBSCAI	Æ	SUBSPACE_DE	
	Rand_index	F_Score	Rand_index	F_Score	Rand_in dex	F_Score	Rand_in dex	F_Score
W+	15	21	16.5	18	17	21	21	21
<b>W-</b>	0	0	4.5	3	0	0	0	0
W (min(W+,W-)	0	0	4.5	3	0	0	0	0
Critical Value(From Table 24)	0	0	0	0	0	0	0	0
Significant Difference	H0 False	H0 False	H0 TRUE	H0 TRUE	H0 False	H0 False	H0 False	H0 False

#### Table 14:Wilcoxon signed-rank test of S\_FAD vs. Clustering Oriented

	PROCLUS		P3C		STATPC		
	Rand_index	F_Score	Rand_index	F_Score	Rand_index	F_Score	
W+	21	21	21	21	21	21	
W-	0	0	0	0	0	0	
W (min(W+,W-)	0	0	0	0	0	0	
Critical Value(From Table 24)	0	0	0	0	0	0	
Significant Difference	H0 False	H0 False	H0 False	H0 False	H0 False	H0 False	

#### **Results of Wilcoxon Signed Rank Test on Actual Datasets**

Table 15 shows the statistical significance test in form of the Wilcoxon signed-rank test of S\_FAD versus the cell-based algorithm on actual datasets. Since the number of actual datasets considered for evaluation is 7 and twotailed tests have been employed, therefore critical value (seen from Wilcoxon signed-rank test table) is 2. It has been observed that S\_FAD shows the statistical difference in results from the CLIQUE algorithm based on F\_Score. Since W-=0, that means S\_FAD performs better on all datasets. For the rest of the cell-based algorithms, S\_FAD shows no significant difference (H0 TRUE).

	Table 15:Wilcoxon signed-rank test of S_FAD vs. Cell-based								
	DOC		MINECLU	MINECLUS SCHISM			CLIQUE	CLIQUE	
	Rand_index	F_Score	Rand_in dex	F_Score	Rand_in dex	F_Score	Rand_inde x	F_Score	
W+	27	11	10	9	16	23	13	27	
<b>W-</b>	7	17	5	19	12	5	15	0	
W (min(W+,W-)	7	11	5	9	12	5	13	0	
Critical Value(From Table 24)	2	2	2	2	2	2	2	2	
Significant Difference	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 TRUE	H0 False	

Table 16 demonstrates the Wilcoxon signed-rank test of S\_FAD against density-based subspace algorithms on actual datasets. S\_FAD depicts better performance with respect to F\_Score and rand\_index against FIRES, SUBCLU, SUBSCALE, and SUBSPACE\_DE algorithms. While on INSCY, no statistically significant difference is pointed. Also for SUBCLU, SUBSCALE, and SUBSPACE\_DE algorithms W-=0 which means S\_FAD is superior on all datasets.

T 11 16 XX/1

			Table 16:v	vilcoxon si	gned-ran	k test of S	_FAD VS	. Density-	based	
	FIRES		INSCY		SUBCLU	J	SUBSCA	ALE	SUBSPAC	E_DE
	Rand_in dex	F_Score	Rand_in dex	F_Score	Rand_ index	F_Score	Rand_ index	F_Score	Rand_in dex	F_Score
W+	27	27	19.5	10	28	28	21	28	21	11
W-	1	1	8.5	18	0	0	0	0	0	0
W	1	1	8.5	10	0	0	0	0	0	0
(min(W+,W-) Critical	2	2	2	2	2	2	2	2	2	2
Value(From Table 24)										
Significant	H0 False	H0	H0	H0	H0	H0 False	H0	H0 False	H0 False	H0 False
Difference		False	TRUE	TRUE	False		False			

Table 16:Wilcoxon	signed-rank test of S	FAD vs. Density-based
	0 -	

#### Table 17: Wilcoxon signed-rank test of S FAD vs. Clustering Oriented

	PROCLUS		P3C		STATPC	
	Rand_index	F_Score	Rand_index	F_Score	Rand_index	F_Score
W+	22	16	26	24	15	20
W-	6	12	1	4	6	8
W (min(W+,W-)	6	12	1	4	6	8
Critical Value(From Table 24)	2	2	2	2	2	2
Significant Difference	H0 TRUE	H0 TRUE	H0 False	H0 TRUE	H0 TRUE	H0 TRUE

Table 17 represents the statistical difference of S\_FAD versus clustering-oriented based subspace algorithms. S\_FAD gives a statistically better rand index than the P3C algorithm. However, value of W' exceeds the critical value of PROCLUS and STATPC and hence there is no significant difference in the performance of S\_FAD in connection with rand\_index and F\_Score. It is noticeable that W+>W- even if there is no significant difference, that means S\_FAD

is a better performer for clustering-oriented based algorithms. From the above discussions, it can be established that S\_FAD performs considerably superior to the majority of algorithms on most of the datasets.

### 4.1.4 Algorithm's Scalability Analysis

Scalability of algorithms means the performance of algorithms with increasing dimensions of data. Scalability could be shown only on artificial datasets as these datasets have the same number of instances with varied attributes. In this work, scalability has been shown with respect to both rand\_index and F\_Score. The scalability of S\_FAD versus cell-based is shown in Figures 8 and 9 respectively. The scalability is represented in terms of rand\_index/F\_Score (y-axis) on data dimensionality (x-axis). It has been scrutinized that the proposed algorithm (S\_FAD) depicts comparable performance from 10 to 25 dimensional datasets while it gives better performance on 50 and 75 dimensional datasets. DOC and MINECLUS reveal random behaviour with an increase in dimensionality. Figures 10 and 11 show the scalability of S\_FAD on density-based algorithms. INCY performed very well with 25-dimensional dataset, however, an algorithm could not handle higher dimension dataset. S\_FAD takes the lead on the 50 and 75 dimensional dataset. The scalability of S\_FAD versus clustering-oriented based algorithms is depicted in figures 12 and 13. S\_FAD gives better results in almost all datasets against clustering-oriented algorithms like STATPC, P3C, and PROCLUS.



Figure 8: Scalability of S\_FAD vs. Cell based on Rand\_index





Figure 9: Scalability of S\_FAD vs. Cell based on F\_Score



Figure 11: Scalability of S\_FAD vs. density based on F\_Score





Figure 13: Scalability of S\_FAD vs. clustering oriented on F\_Score

Thus, S\_FAD shows a good sign of improvement with scalability on data dimensionality as compared to various subspace clustering algorithms. The next section illustrates the performance of S\_FAD on very high dimensional datasets.

### 4.2 S\_FAD on High Dimensional Actual Dataset

To validate the efficacy of the proposed algorithm (S\_FAD) on high-dimensional data, it is implemented on actual datasets with a large number of attributes. It is found that S\_FAD is successful in forming subspace clusters on high-dimensional datasets. Table 18 and 19 present results in the form of best (maximum), worst (minimum), mean (average), median and standard deviation of F\_Score and rand\_index at 10 independent runs. Other subspace clustering algorithms could not cope up with high-dimensional datasets. Also, the results of any other clustering algorithm are not available on such high-dimensional datasets. SUBSCALE algorithm (Kaur & Datta, 2015) was attempted on the MADELON dataset but exact values of results are not revealed in their work. The actual datasets included for the study are MADELON, MICROMASS, GISSETTE, and DRIV FACE with 500, 1300, 5000, and 6400 dimensions respectively.

		1 able 10. 1_5	$core or S_1 m$	9 oli iligii alili	ensional actual e	utusets
	Dimension	Best	Worst	Mean	Median	Std. Deviation
MADELON	500	0.9905	0.9434	0.974	0.9797	0.0165
MICROMASS	1300	0.8166	0.8141	0.8159	0.8166	0.0008
GISSETTE	5000	0.5888	0.5668	0.5768	0.5772	0.0087
DRIVFACE	6400	0.8864	0.8844	0.8852	0.8854	0.0008
	Та	ble 19: Rand_	_index of S_F.	AD on high di	mensional actua	l datasets
	Ta Dimension	ble 19: Rand_ Best	_index of S_F. Worst	AD on high di Mean	mensional actua Median	l datasets Std. Deviation
MADELON	Ta Dimension 500	ble 19: Rand_ Best 0.9905	index of S_F. Worst 0.94	AD on high di Mean 0.9731	mensional actua Median 0.9795	l datasets Std. Deviation 0.0176
MADELON MICROMASS	<u>Ta</u> Dimension 500 1300	ble 19: Rand_ Best 0.9905 0.69	<u>index of S_F.</u> Worst 0.94 0.6865	AD on high di Mean 0.9731 0.6891	mensional actua Median 0.9795 0.69	l datasets Std. Deviation 0.0176 0.0012
MADELON MICROMASS GISSETTE	<u>Ta</u> Dimension 500 1300 5000	ble 19: Rand_ Best 0.9905 0.69 0.4181	index of S_F. Worst 0.94 0.6865 0.3972	AD on high di Mean 0.9731 0.6891 0.4053	mensional actua Median 0.9795 0.69 0.4052	l datasets Std. Deviation 0.0176 0.0012 0.0077

It is perceived from Tables 18 and 19 that the proposed algorithm (S\_FAD) shows a high rand\_index on MADELON, MICROMASS, and DRIV FACE datasets. In the case of the GISSETTE dataset, the algorithm's efficacy is low because of the sparse dataset. Since the standard deviation is very low, algorithms do not require many independent runs for obtaining the best results.

### Discussion

S\_FAD is assessed against 11 subspace clustering algorithms on a total of 13 datasets including actual and artificial with respect to rand index and F Score. On small dimensions, it is found that the total percentage enhancement of S FAD is higher than the total percentage lag (as compared to other algorithms). Subspace algorithms are also ranked on the basis of average ranking and SRR ranking on artificial and actual datasets independently. The Wilcoxon signed-rank test, is performed to validate the significant variation in results obtained by subspace algorithms. From the above experimental results and analysis, it is inferred that S\_FAD reveals better performance in terms of F\_Score and rand\_index against most of the existing subspace algorithms on the majority of datasets. Additionally, S\_FAD is executed on high-dimensional actual datasets. The results exhibit that the proposed algorithm scales very well on high-dimensional thin data sets. As the size of the dataset increases, some other measures like sampling are required to be inculcated to improve the efficacy. The time complexity of the S FAD algorithm is  $O((2^m).(nlogn))$  where m is the number of dimensions and n is the number of records in the dataset. Hence, there is a trade-off between rand index and time complexity. Although the time complexity of the proposed S FAD algorithm is relatively high, it can provide near-optimal solutions on high-dimensional problems. On the contrary traditional subspace clustering algorithms are unsuccessful in providing the results for high dimensional data. Also, S FAD finds overlapping subspace clusters with no redundant information. This property is hardly satisfied by any of the other subspace clustering algorithms. The unique property of S FAD is it can determine subspace clusters of varied densities.

Hence it can be established that conventional subspace clustering algorithms are more suited for applications involving small dimensions and have time-sensitive requirements. However, for high dimension applications adopting the S\_FAD algorithm would be a better choice.

#### 5. Conclusion

Subspace clustering in a large number of attributes is a computational challenge in the data mining field. This challenge includes an unknown number of subspaces and dimensions involved in each subspace before clustering. Additionally, there can be an exponential number of subspaces for the high-dimensional dataset. To resolve these issues, a novel subspace clustering algorithm S\_FAD is proposed. The efficacy of S\_FAD is better in terms of F\_Score and rand\_index. Using these evaluation measures, the proposed algorithm is compared with various conventional subspace clustering algorithms on basis of 4 parameters i.e. average ranking, success rate ratio ranking, Wilcoxon signed-rank test, and scalability on dimensions. S\_FAD provides considerably good performance when analysed on these parameters. It does not give any redundant information of subspaces as it performs clustering in maximal subspaces. S\_FAD takes input data in its original form without normalization. It could determine overlapping subspace clusters of varied densities. Thus, S\_FAD provides several advantages over existing subspace clustering algorithms and is successful in determining clusters in 6400-dimensional actual datasets. In the future, the S\_FAD algorithm can be applied in applications of distributed databases such as vertical and horizontal fragmentation. Also, some useful techniques can be incorporated into the algorithm to cluster large datasets and improve clustering quality.

#### **Compliance with ethical standards**

Conflict of interest All author declares that he/she has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

### References

- Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The Arithmetic Optimization Algorithm. *Computer Methods in Applied Mechanics and Engineering*, 376, 113609. https://doi.org/https://doi.org/10.1016/j.cma.2020.113609
- Abualigah, L., Gandomi, A. H., Elaziz, M. A., Hamad, H. Al, Omari, M., Alshinwan, M., & Khasawneh, A. M. (2021). Advances in Meta-Heuristic Optimization Algorithms in Big Data Text Clustering. *Electronics*, 10(2). https://doi.org/10.3390/electronics10020101
- Abualigah, L., Gandomi, A. H., Elaziz, M. A., Hussien, A. G., Khasawneh, A. M., Alshinwan, M., & Houssein, E.

H. (2020). Nature-Inspired Optimization Algorithms for Text Document Clustering—A Comprehensive Analysis. *Algorithms*, *13*(12). https://doi.org/10.3390/a13120345

- Abualigah, L. M. Q. (2019). *Feature selection and enhanced krill herd algorithm for text document clustering*. Springer.
- Agarwal, P., & Mehta, S. (2014). Nature-Inspired Algorithms: State-of-Art, Problems and Prospects. *International Journal of Computer Applications*, 100(14), 14–21. https://doi.org/10.5120/17593-8331
- Agarwal, P., & Mehta, S. (2015). Comparative Analysis of Nature Inspired Algorithms on Data Clustering. *IEEE International Conference on Research in Computational Intelligence and Communication Networks* (*ICRCICN*), 119–124.
- Agarwal, P., & Mehta, S. (2016). Enhanced flower pollination algorithm on data clustering. *International Journals of Computers and Applications, Taylor and Francis*, 7074(August), 144–155. https://doi.org/10.1080/1206212X.2016.1224401
- Agarwal, P., & Mehta, S. (2017). Empirical analysis of five nature-inspired algorithms on real parameter optimization problems. *Artificial Intelligence Review*, 1–57. https://doi.org/10.1007/s10462-017-9547-5
- Agarwal, P., & Mehta, S. (2019a). ABC\_DE\_FP: A Novel Hybrid Algorithm for Complex Continuous Optimisation Problems. *International Journal of Bio-Inspired Computation*, 14(1), 46–61. https://doi.org/10.1504/ijbic.2018.10014476
- Agarwal, P., & Mehta, S. (2019b). Subspace Clustering of High Dimensional Data Using Differential Evolution. In *Nature-Inspired Algorithms for Big Data Frameworks* (First, pp. 47–74). IGI Global. https://doi.org/10.4018/978-1-5225-5852-1.ch003
- Aggarwal, C., Wolf, J., Yu, P., C. Procopiuc, & Park., J. (1999). Fast Algorithm for Projected Clustering. *SIGMOD*, 61–72.
- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. (1996). Fast discovery of association rules. *Adv Knowl Discov Data Min*, *12*, 307–328.
- Assent, I., Krieger, R., M<sup>\*</sup>Uller, E., & Seidl., T. (2008). INSCY: Indexing subspace clusters with in-process-removal of redundancy. *ICDM*, 719–724.
- Assent, Ira. (2012). Clustering high dimensional data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(4), 340–350. https://doi.org/10.1002/widm.1062
- Bache, K., & Lichman, M. (2006). UCI machine learning repository. http://archive.ics.uci.edu/ml
- Brazdil, P., & Soares, C. (2000). A Comparison of Ranking Methods for Classification Algorithm Selection. *Machine Learning: ECML 2000, 1810,* 63–75. https://doi.org/10.1007/3-540-45164-1\_8
- Daszykowski, M., Walczak, B., & Massart, D. L. (2001). Looking for natural patterns in data: Part 1. density-based approach. *Chemometrics and Intelligent Laboratory Systems*, 56, 83–92.
- Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7, 1–30. https://doi.org/10.1016/j.jecp.2010.03.005
- Deng, Z., Choi, K. S., Jiang, Y., Wang, J., & Wang, S. (2016). A survey on soft subspace clustering. *Information Sciences*, 348, 84–106. https://doi.org/10.1016/j.ins.2016.01.101
- Domeniconi, C., Papadopoulos, D., Gunopulos, D., & Ma, S. (2004). Subspace clustering of high dimensional data. *SIAM Int. Conf. on Data Mining*, 31–40.
- Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., & Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3), 267–279. https://doi.org/10.1109/TETC.2014.2330519
- Kailing, K., Kriegel, H.-P., & Oger, P. K. (2004). Density-connected subspace clustering for high-dimensional data. *SDM*, 246–257.
- Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x
- Karami, A., & Johansson, R. (2014). Choosing DBSCAN Parameters Automatically using Differential Evolution. International Journal of Computer Applications, 91(7), 1–11.
- Kaur, A., & Datta, A. (2015). A novel algorithm for fast and scalable subspace clustering of high-dimensional data. *Journal of Big Data*, 2(1), 17. https://doi.org/10.1186/s40537-015-0027-y
- Kaur, A., & Datta, A. (2014). SUBSCALE: Fast and Scalable Subspace Clustering for High Dimensional Data. *IEEE International Conference on Data Mining Workshops*, *{ICDM}*, 621–628. https://doi.org/10.1109/ICDMW.2014.100
- Kriegel, H.-P., Kroger, P., Renz, M., & Wurst., S. (2005). A generic framework for efficient subspace clustering of high-dimensional data. *ICDM*, 250–257.

- Kriegel, Hans-Peter, 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, 3(1), 1–58. https://doi.org/10.1145/1497577.1497578
- Kumar, D., Bezdek, J. C., Palaniswami, M., Rajasegarar, S., Leckie, C., & Havens, T. C. (2016). A Hybrid Approach to Clustering in Big Data. *IEEE Transactions on Cybernetics*, 46(10), 2372–2385. https://doi.org/10.1109/TCYB.2015.2477416
- Lin, L., Gen, M., & Liang, Y. (2014). A hybrid EA for high-dimensional subspace clustering problem. Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, 2855–2860. https://doi.org/10.1109/CEC.2014.6900313
- Liu, X., Wang, J., Cheng, D., Shi, D., & Zhang, Y. (2020). Non-convex low-rank representation combined with rank-one matrix sum for subspace clustering. *Soft Computing*. https://doi.org/10.1007/s00500-020-04865-0
- Lu, Y., Wang, S., Li, S., & Zhou, C. (2011). Particle swarm optimizer for variable weighting in clustering highdimensional data. *Machine Learning*, 82(1), 43–70. https://doi.org/10.1007/s10994-009-5154-2
- Moise, G., & Sander., J. (2008). Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. *KDD*, 533–541.
- Moise, G., Sander, J., & Ester, M. (2006). P3C: A robust projected clustering algorithm. ICDM, 414-425.
- Müller, E, Günnemann, S., Assent, I., Seidl, T., & Färber, I. (2009). *Evaluating Clustering in Subspace Projections* of High Dimensional Data. http://dme.rwth-aachen.de/en/OpenSubspace/evaluation
- Müller, Emmanuel, Günnemann, S., Assent, I., & Seidl, T. (2009). Evaluating clustering in subspace projections of high dimensional data. *Proceedings of the VLDB Endowment*, 2(1), 1270–1281. https://doi.org/10.14778/1687627.1687770
- Parsons, L., Haque, E., & Liu, H. (2004). Subspace Clustering for High Dimensional Data: A Review. ACM SIGKDD Explorations Newsletter, 6(1), 90–105. https://doi.org/10.1145/1007730.1007731
- Pavlyukevich, I. (2007). Lévy Flight, non local search and simulated annealing. *J.Computational Physics*, 226(January), 1830–1844. https://doi.org/10.1016/j.jcp.2007.06.008
- Pesevski, A., Franczak, B. C., & McNicholas, P. D. (2018). Subspace clustering with the multivariate-t distribution. *Pattern Recognition Letters*, *112*(2002), 297–302. https://doi.org/10.1016/j.patrec.2018.07.003
- Procopiuc, C. e. a. (2002). A monte carlo algorithm for fast projective clustering. SIGMOD, 418-427.
- Road, H., & Jose, S. (1998). Automatic Subspace Clustering Mining of High Dimensional Applications for Data. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, 27, 94–105. https://doi.org/10.1145/276305.276314
- Sarafis, I. A., Trinder, P. W., & Zalzala, A. M. S. (2003). Towards effective subspace clustering with an evolutionary algorithm. 2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings, 2, 797–806. https://doi.org/10.1109/CEC.2003.1299749
- Sequeira, K., & Zaki., M. (2004). SCHISM: A new approach for interesting subspace mining. ICDM, 186–193.
- Steinbach, M., Ertoz, L., & Kumar, V. (2003). The Challenges of Clustering High Dimensional Data. *New Vistas in Statistical Physics, Applications in Econophysics, Bioinformatics, and Pattern Recognition*, 1–33.
- Steinbach, M., Levent, E., & Kumar, V. (2004). The Challenges of Clustering High Dimensional Data. New Vistas in Statistical Physics, Applications in Econophysics, Bioinformatics, and Pattern Recognition, 273–309. https://doi.org/10.1007/978-3-662-08968-2\_16
- Storn, R., & Price, K. (1997). Differential evolution a simple and efficient heutistic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.
- Timmerman, M. E., Ceulemans, E., De Roover, K., & Van Leeuwen, K. (2013). Subspace K-means clustering. *Behavior Research Methods*, 45(4), 1011–1023. https://doi.org/10.3758/s13428-013-0329-y
- Yan, F., Wang, X. dong, Zeng, Z. qiang, & Hong, C. qun. (2020). Adaptive multi-view subspace clustering for highdimensional data. *Pattern Recognition Letters*, 130, 299–305. https://doi.org/10.1016/j.patrec.2019.01.016
- Yang, X.-S. S. (2012). Flower Pollination Algorithm for Global Optimization. Unconventional Computation and Natural Computation, 7445, 240–249. https://doi.org/10.1007/978-3-642-32894-7\_27
- Yang, X. S. (2012). Flower pollination algorithm for global optimization. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7445 LNCS, 240–249. https://doi.org/10.1007/978-3-642-32894-7\_27
- Yiu, M. L., & Mamoulis, N. (2003). Frequent-pattern based iterative projected clustering. ICDM, 689-692.
- Zhao, X., An, G., Cen, Y., Wang, H., & Zhao, R. (2019). Robust discriminant low-rank representation for subspace clustering. *Soft Computing*, 23(16), 7005–7013. https://doi.org/10.1007/s00500-018-3339-y
- Zhong, G., & Pun, C. M. (2020). Subspace clustering by simultaneously feature selection and similarity learning. *Knowledge-Based Systems*, 193, 105512. https://doi.org/10.1016/j.knosys.2020.105512



(A1=attribute 1 of dataset;

P1,P2,P3.... = data point or Records;

C1,C2,C3 = clusters formed ;

X1, X2, X3= sum of signatures of data points associated with each cluster C1,C2,C3

e.g. X1=sum(sig(P1)+sig(P2)+sig(P3)) where sig(P1) is the signature composed of a 15 digit unique number)

# Figure 1

Clustering in Single Dimensional Data



table)

•••

•••

•••

....

....

....

...

•••

...

# Figure 2

••

....

•••

S\_FAD Algorithm



FAD Algorithm





Rand\_index of algorithms on artificial datasets

# Figure 5

F\_Score of algorithms on artificial datasets



Rand\_index of algorithms on actual datasets





F\_Score of algorithms on actual datasets

# Figure 8



Scalability of S\_FAD vs. Cell based on Rand\_index

# Figure 9

Scalability of S\_FAD vs. Cell based on F\_Score



Scalability of S\_FAD vs. density based on Rand\_index



# Figure 11



Scalability of S\_FAD vs. clustering oriented on RI



# Figure 13

Scalability of S\_FAD vs. clustering oriented on F\_Score