IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2018, accepted November 28, 2018, date of publication December 11, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2884902

MulSim: A Novel Similar-to-Multiple-Point Clustering Algorithm

MEI CHEN[1,2], XIAOFANG WEN[1], ZHICHONG YANG[1], MING LI[1], AND MEI ZHANG[1]
[1]School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China
[2]School of Information Science & Engineering, Lanzhou University, Lanzhou 730000, China

Corresponding author: Mei Chen (mei.chen.lzjtu@hotmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61762057, Grant 61602225, and Grant 61762077, and in part by the Foundation of A Hundred Youth Talents Training Program of Lanzhou Jiaotong University.

ABSTRACT Finding clusters in datasets with different distributions and sizes is challenging when clusters are of widely various shapes, sizes, and densities. Based on a similar-to-multiple-point clustering strategy, a novel and simple clustering algorithm named MulSim is presented to address these issues in this paper. MulSim first defines a new distance which can automatically adapt different densities when clustering. Then, the MulSim groups two points together if and only if one point is similar to another point and its similar neighbors. Our comprehensive experiments on both multi-dimensional and two dimensional datasets representing different clustering difficulties, show that the MulSim performs better than classical and state-of-the-art baselines in most cases. Besides, when increasing the size of datasets, MulSim can still ensure good clustering quality. In addition, the impact of the two MulSim parameters on clustering quality as well as the way of the parameter estimation are analyzed. In the end, the practicability and feasibility of the algorithm are tested through a face recognition example.

INDEX TERMS Clustering algorithm, distance-based clustering, similarity.

I. INTRODUCTION

With the rapid development of the information technology, many complex datasets have been emerging, which possibly differ from each other in the number of points and dimensions or have different data densities or patterns, for example, spacial data [1], image data [2] and air pollution data [3]. How to mine rich information from these datasets has received much attention in recent years. Clustering is an explorative way to investigate underlying structures in datasets, which groups a set of data by maximizing the similarities within clusters and minimizing the similarities between two clusters. Serving as the foundation of further data analysis techniques, clustering is facing a critical problem, that is to say, how to effectively detect clusters in these various complex datasets.

Many clustering algorithms are distance-based clustering methods, of which two or more data points are grouped to one cluster if they are close enough according to a given distance. Among them, the most representatives are hierarchical and partitioning-based clustering methods. However, both of them have certain defects when facing complex datasets.

Hierarchical clustering [4]–[6] methods are based on the core idea that points are more related to nearby points than to points farther away. However, these algorithms do not provide a unique partition of the dataset, but provide an extensive hierarchy of clusters that can be merged with each other at certain distances instead. The distance between two clusters is hard to canculate [7], and most of the methods cannot identify non-spherical shaped clusters [8]. Furthermore, most of the hierarchical clustering methods suffer from a high computational cost in handling big size datasets. In general cases, the complexity is $O(n^3)$ for agglomerative clustering and $O(n^2)$ for divisive clustering.

The partition-based algorithms decompose a dataset into a set of disjoint clusters where the number of the clusters is predefined by the users. The most well-known partition-based algorithms are k-Means and k-Medoids [9]. The k-Means algorithm uses a single mean vector to represent a cluster centre, while the k-Medoids chooses a data point which is the median or an exemplar within a cluster as a cluster centre. Both of the methods need a fixed k as the number of clusters, and update the k cluster centres iteratively based on a distance measure and assign the points to the nearest cluster centre, such that the sum of the squared distances from the cluster is minimized. Both can but can only find a local optimum, and commonly run multiple times with different random initializations. Due to its efficiency and easy-to-use

characteristic, a host of variations of k-Means have been presented to improve the performance, such as k-Means++ and CLARA. Although most k-Means-type algorithms require the number of clusters to be specified in advance, they do not provide any information about how to set the number of clusters, which is considered to be one of the biggest drawbacks of these algorithms. Moreover, since partition-based algorithms always assign a point to the nearest cluster centre, they can only find clusters with approximately similar size, and cannot detect non-convex clusters.

In this paper, a novel distance-based clustering algorithm, named MulSim (clustering by finding MULtiple points being SIMilar to one point), is devised to discover clusters with arbitrary shapes, sizes and densities more effectively on datasets with different distributions and sizes. MulSim is based on an interesting observation: two similar points from one cluster tend to have shared similar neighbors, but two similar points from different clusters usually have no such feature. Inspired by this observation, first we define a new distance which can automatically match diverse densities when clustering. Then we group two points together if one is similar to another and its similar neighbors. The clustering strategy of this novel clustering method is more restrictive by considering a point being similar to multiple points at the same time, which ensures to find arbitrary shaped clusters with different densities effectively. Whereas, traditional similarity-based clustering methods just group two points into one cluster so long as the two points are similar to each other. For example, on the Aggregation dataset shown in Figure 1, if we simply put two similar points together one by one, the purple bridge and the yellow bridge will not be disconnected. But if we adopt the clustering strategy of MulSim, the two bridges will be disconnected naturally. Apart from the new distance and the new clustering strategy, MulSim has two more advantages compared with traditional similarity-based clustering methods: first, MulSim has an acceptable time complexity; second, the two input parameters of MulSim can be determined according to the distribution of a dataset.

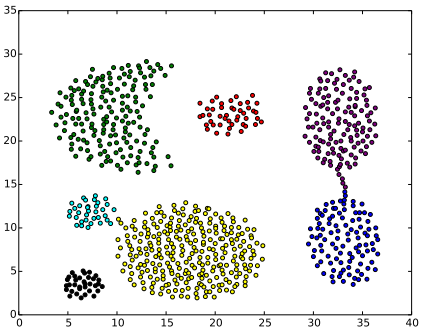


FIGURE 1. An example dataset with different shaped clusters.

The remaining sections are organized as follows. We review the related works in Section II. Then, we give preliminary of MulSim in Section III and elaborate MulSim algorithm in Section IV. After that, in Section V, we present the performances of MulSim on both two-dimensional and multi-dimensional datasets, which show how effective our method is, compared with state-of-the-art methods. Section V also provides the way of the parameter estimation, and shows the practicability and feasibility of the algorithm as well through a face recognition example. Finally, Section VI concludes the work.

II. RELATED WORKS

Clustering algorithms can be categorized according to their clustering models, such as centroid-based, density-based, distribution-based, the hierarchical, and the spectral clustering methods. The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally. Unless in terms of the characteristics of a dataset, users may prefer one specific clustering model to others. The performances and limitations of the above clustering models are discussed in detail as follows.

To begin with, centroid-based clustering methods, or known as partition-based clustering methods, such as k-Means [10], [11], k-Medoids [9] and Fuzzy C-Means [12], have been widely used in many domains, using an iterative way to determine k clusters for n points by minimizing the dissimilarities between each point and their corresponding centres [13]. Thus, centroid-based clustering methods can only detect spherical-shaped compact clusters and can only find a local optimum with different random initializations.

Next, density-based clustering methods can find non-spherical shaped clusters by grouping the data points spreading over a contiguous region of high density together and taking the points locating in low-density regions as outliers. For instance, DBSCAN [14] is one of the most well-known density-based clustering algorithms. By using a density criterion, i.e., a minimum number of other points within a radius, DBSCAN connects points to one cluster while these pioints are density-reachable. If points lie alone in a low-density regions, DBSCAN marks them as outliers. Although DBSCAN can discover clusters with different shapes, it has difficulty in detecting clusters with significant differences in densities, because the density cannot be chosen appropriately for all clusters [15]. OPTICS [16] is an improved method of DBSCAN, which aims to remove the need of choosing an appropriate value for the density criterion. However, it still cannot identify clusters of varying densities. Another popular density-based method, DENCLUE [17], also suffers from failures in detecting the clusters with arbitrary densities, which clusters data points by attracting them to the density-attractors that are local maxima of density function.

Then, distribution-based clustering algorithms [18], [19] are based on formal models and are not largely heuristic. They model clusters using statistical distributions. For example, multivariate normal distributions are used in the expectation-maximization algorithm. But most of the distribution-based clustering methods put an extra burden on users, that is, users need to choose the best model whose

parameters need to be iteratively optimized to better fit the dataset.

Besides, the hierarchical clustering algorithms group data points in two ways. Either start by regarding each single point in the dataset as an individual cluster and then aggregate them into clusters (agglomerative), or start by taking the initial dataset as a whole and then divide it into partitions (divisive). CURE [20] is a classical hierarchical clustering algorithm, which carries out a hierarchical agglomerative clustering after randomly sampling a set of representative points. CHAMELEON [7] is also a prominent hierarchical clustering algorithm. It takes a two-phase approach. First, CHAMELEON uses a graph partition method to divide the dataset into a set of individual clusters. Second, it uses an agglomerative hierarchical clustering to merge the clusters. Both CURE and CHAMELEON can identify non-spherical shaped clusters. However, they suffer from high time complexities.

Last but not least, spectral clustering [21], [22] is one of the most prominent clustering approaches. It makes use of eigenvalues of the similarity matrix of the data to perform dimensionality reduction. Then, it constructs clusters only depending on a similarity graph. Whereas it is highly sensitive to noisy input data and takes high time cost.

In recent years, considerable efforts have been made to improve the performance of detecting clusters with arbitrary shapes, sizes and densities [15]. Such as, ABACUS [23] is a globbing-based method which identifies the intrinsic clusters by iteratively globbing points from dense regions and moving the representative points simultaneously. AnyDBC [24] is a novel anytime approach to cope with the cost problem for very large datasets by reducing both the range query and the label propagation time of DBSCAN. Perch [25] is a new incremental algorithm for hierarchical clustering to solve the problem of extreme clustering. BOOL [26] is a novel hierarchical clustering algorithm, which first discretizes all points in a dataset and then iteratively merges small clusters to construct final clusters. Although BOOL can basically identify the structures of clusters in most cases, it wrongly identifies a few normal points as outliers. CLASP [27] is a shrinking-based clustering algorithm and detects clusters by effectively preserving the shape information of clusters. SPARCL [28] works in two stages. It first generates many small representative clusters and then merges these small clusters to get final clusters. A widely concerned algorithm, CFDP [29], combines the advantages of both centroid-based and density-based clustering methods. As a local-density-based method, it can achieve good performances in most cases. But as a centroid-based method, it is unable to group points correctly when a cluster has more than one centres.

III. PRELIMINARY OF MulSim ALGORITHM

In this section, we first prepare the necessary notions about clustering. Then we present a new distance based on the nearest-neighbor relationship. We also introduce the clustering strategy used in MulSim algorithm.

A. NOTION OF CLUSTERING

Clustering is the process of partitioning a set of points into subsets. A *Dataset* is denoted as follows,

$$D = \{x_1, x_2, \cdots, x_i, \cdots, x_n\} \tag{1}$$

where $x_i (1 \leq i \leq n) \in D$ is the ith data point.

Clustering methods operate on this dataset and group the points in D into $m (1 \leq m \leq n)$ clusters, $C_1, \cdots, C_m$. Each subset is a cluster. Thus the points in one cluster are similar to one another but dissimilar to points in other clusters.

B. A NEW DISTANCE

Clustering is based on a measure of similarity. Thus, choosing an appropriate similarity has a critical impact on clustering quality. The most used dissimilarity measures are absolute distances, such as Euclidean distance and Manhattan distance. However, if we choose a similarity-based clustering method to partition a dataset, we will have to face the problem that absolute distances cannot adapt to diverse densities.

Intuitively, two similar points in one cluster tend to be the nearest neighbors to each other. This motivates us to define a new metric to measure the similarity between two points based on the nearest-neighbor relationship. First, we give the notion of the nearest neighbors.

Let x_1,x_2,..., x_n be n points in dataset D. For each x_i in D, the most similar neighbors or the nearest neighbors of x_i are called *nearest neighbors (NN)* of x_i, denoted as $N(x_i)$, $N(x_i) \subseteq D$.

Then, we introduce a new distance to measure the similarity of two points. Formally, given two points x_i and x_j, we first obtain the sort orders $o_{x_i}(x_j)$ and $o_{x_j}(x_i)$, where $o_{x_i}(x_j)$ is the sort order of x_j in x_i's NN order list getting by sorting points according to an absolute distance, and so is $o_{x_j}(x_i)$. Next, we define a symmetric distance $bigger(x_i, x_j)$ between x_i and x_j,

$$bigger(x_i, x_j) = bigger(o_{x_i}(x_j), o_{x_j}(x_i)) \tag{2}$$

For example, if x_i is the l_{th} neighbor in the NN order list of x_j, x_j is the m_{th} neighbor in the NN order list of x_i, and $m > l$, then the $bigger(x_i, x_j)$ is m. The smaller the $bigger(x_i, x_j)$ is, the more possible that x_i and x_j are top neighbors to each other.

No matter what densities in a dataset are, each point shall have its nearest neighbors. Therefore, this new distance can automatically adapt to different densities while the absolute distances cannot.

C. SIMILARITY MEASUREMENT

In order to construct clusters, we need a criterion to decide when to assign a point to one of the clusters. This can potentially be solved by adopting an appropriate threshold. In MulSim, the threshold is a positive integer k. If

$$bigger(x_i, x_j) \leq k, \tag{3}$$

then the pair of points are similar. Otherwise, they are dissimilar.

D. A SIMILAR-TO-MULTIPLE-POINT CLUSTERING STRATEGY

When grouping two points into a cluster, MulSim not only considers the similarity between these two points according to Eq. 3, but also considers the similarities between one and the other's neighbors. If and only if one point is similar to the other point and its similar neighbors, MulSim groups the two points into one cluster.

Therefore, MulSim takes a much stricter strategy compared with that of traditional similarity-based clustering methods. Traditional similarity-based clustering methods thoughtlessly group two points into one cluster so long as the two points are similar to each other. This clustering strategy may wrongly merge two or more different clusters into one big cluster just because a few points are similar to points located in the different communities. Whereas the strategy MulSim takes is pretty similar to the rules in human society. If two persons are in one community, they both have great potential to be familiar with each other's neighbors in the same community. This phenomenon properly mirrors the scientific nature of our clustering strategy. We call this clustering strategy the similar-to-multiple-point clustering strategy. It is owing to the similar-to-multiple-point clustering strategy that MulSim can detect various shaped clusters.

To reduce the time cost, when clustering, MulSim employs an equivalent clustering strategy by using the notion of Mutual k-nearest neighbors. The definition of Mutual k-nearest neighbors is given as follows,

For two points x_i and x_j, if and only if $bigger(x_i, x_j) \leq k$, we call x_i and x_j *mutual k-nearest neighbors (MkNN)*. MkNN of x_i is denoted as a set $MkNN(x_i)$.

Theorem 1: For a pair of points $\langle x_i, x_j \rangle \in D$, if x_j is similar to x_i, the strategy that x_j is similar to at least m similar neighbors of x_i, is equivalent to the following equation,

$$|MkNN_{x_i} \cap MkNN_{x_j}| \geq m \tag{4}$$

where m indicates the required number of similar neighbors of point x_i when clustering, $|MkNN(x_i) \cap MkNN(x_j)|$ is the number of points in the intersection of the set $MkNN(x_i)$ and $MkNN(x_j)$.

Proof: Since x_j is similar to m similar neighbors of x_i, the m similar neighbors of x_i are in the set $MkNN(x_j)$. Besides, the m similar neighbors of x_i are in the set $MkNN(x_i)$. Thus, the m similar neighbors of x_i are in the set $MkNN(x_i) \cap MkNN(x_j)$. Then, the strategy that x_j is similar to at least m similar neighbors of x_i is equivalent to the equation $|MkNN_{x_i} \cap MkNN_{x_j}| \geq m$. □

Following the above conditions, we can also know that x_i, x_j and each member in the set $MkNN(x_i) \cap MkNN(x_j)$ are similar to each other.

IV. MulSim ALGORITHM

We introduce the MulSim algorithm in this section. First, we describe the clustering process of MulSim in detail. Then, the time complexity of MulSim is analyzed.

A. MulSim CLUSTERING PROCESS

MulSim needs two input parameters, the threshold of the distance $bigger(x_i, x_j)$, k, and the required number of similar neighbors of a point when clustering, m. It starts with a point that has not been visited. The detailed steps involved in clustering using MulSim are described as follows.

Step 1 Obtain MkNN List: We calculate the distance $bigger(x_i, x_j)$ of any two points by equation 2. If $bigger(x_i, x_j) \leq k$, we put the two points into each other's MkNN list.

Step 2 Clustering: While clustering, as the Theorem 1 proved, if a point x_i is similar to another point x_l and the neighbors of x_l, we apply the similar-to-multiple-point clustering strategy to group these two points and the points in the set $MkNN(x_i) \cap MkNN(x_l)$ into one cluster. If a point has no nearest neighbors, this point will be detected as an outlier. To label the outlier and visualize it in the demonstration of experimental result, we mark the outlier as a single cluster.

Algorithm 1: MulSim

Input: D: a dataset with n data points; k: the threshold of distance *bigger*, and m: the required number of similar neighbors of a point
Output: C: a set of clusters
Step 1: Put the similar points of each point into it's MkNN list

```
for i=1 to n do
    for x_j ∈ N(x_i) do
        calculate distance bigger(x_i, x_j) by Equation 2
        if bigger(x_i, x_j) ≤ k then
            put x_i to the set MkNN_{x_j}
            put x_j to the set MkNN_{x_i}
```

Step 2: Clustering

```
C ← ∅
for i=1 to n do
    for x_l ∈ MkNN_{x_i} do
        if |MkNN_{x_i} ∩ MkNN_{x_l}| ≥ m then
            if x_i or x_l is already existed in a cluster then
                put x_i, x_l and each point in
                MkNN(x_i) ∩ MkNN(x_l) to the cluster
            else
                put x_i, x_l and each point in
                MkNN(x_i) ∩ MkNN(x_l) to a new cluster c
                put c into C
    take the point without label as a single cluster g
    put g into C
Output C
```

B. TIME COMPLEXITY ANALYSIS

In the first step of MulSim, when searching the nearest neighbors of each point, since we use k-d tree [30], [31], the time

TABLE 1. Datasets statistics and the corresponding parameters of each method.

	Name	Dataset Statistics				Input Parameter						
		Points	*Dim.*	*Clusters*	Characteristic	MulSim	BOOL	CLASP	OPTICS	k-Means	DBSCAN	CFDP
Two Dimensional Datasets	Aggregation	788	2	7	Uniform Density	(44,10)	(7,2,0)	(7,4,20,0,20)	(8)	(7)	(1.08,5)	(200,7)
	Flame	240	2	2	Uniform Density	(9,1)	(2,11,0)	(2,4,20,0,20)	(8)	(2)	(0.85,4)	(200,2)
	Compound	399	2	6	Various Density	(10,1)	(6,3,0)	(6,4,20,0,20)	(2)	(6)	(1.00,5)	(200,6)
	Toy	1204	2	2	Multi-centre	(8,1)	(2,5,0)	(2,4,20,0,20)	(4)	(2)	(0.80,6)	(200,2)
	Spiral	312	2	3	Spiral Shapes	(5,1)	(3,1,0)	(3,4,20,0,20)	(9)	(3)	(2.00,4)	(200,3)
	R15	600	2	15	Convex Shapes	(34,5)	(15,5,0)	(15,4,20,0,20)	(7)	(15)	(0.30,3)	(200,15)
Multi-dimensional Datasets	Elico	336	7	8	-	(36,5)	(8,1,0)	(8,4,20,1,20)	(8)	(8)	(1.2,27)	(200,8)
	Haberman	306	3	2	-	(18,3)	(1,2,0)	(2,4,20,1,20)	(4)	(2)	(0.5,6)	(200,2)
	Iris	150	4	3	-	(10,1)	(3,0,0)	(3,4,20,1,20)	(4)	(3)	(1.6,15)	(200,3)
	Wpbc	198	30	2	-	(39,7)	(7,2,0)	(2,4,20,1,20)	(1)	(2)	(2.0,2)	(200,2)
	Spectef	80	44	2	-	(13,2)	(2,0,0)	(2,4,100,1,20)	(36)	(2)	(0.1,83)	(200,2)
	User	258	5	4	-	(13,2)	(15,4,0)	(4,4,70,1,20)	(1)	(4)	(0.9,1)	(200,4)

complexity is $O(n \cdot \log n)$, where n is the number of data points in the original dataset D. Then MulSim takes $O(k \cdot n)$ time to obtain the distance $bigger(x_i, x_j)$, the set $MkNN_{x_j}$ and $MkNN_{x_i}$, where k is the threshold of distance $bigger$. Since $k \ll n$ holds, the time complexity is usually $O(n)$. In the second step of MulSim, the time complexity is also $O(n \cdot k)$. Because $k \ll n$, the time complexity of the second step is reduced to $O(n)$. Thus, the overall time complexity of MulSim is approximate to $O(n \cdot \log n)$.

V. EXPERIMENTS AND ANALYSIS

In this section, we first evaluate the clustering performance of MulSim on two dimensional and multi-dimensional datasets which contain various shaped clusters with different densities, by comparing it with several state-of-the-art clustering algorithms. Then we discuss the way of selecting the parameters k and m that are used in MulSim. Finally, we apply MulSim to the Olivetti Face dataset [32] to demonstrate its feasibility and practicality.

A. BASELINES AND BENCHMARKS

1) BASELINES

To evaluate the performance of MulSim, we compare it with several representative state-of-the-art clustering algorithms.

k-Means [10] is the most well-known algorithm as a partition-based clustering method. The input parameter k is the number of clusters.

DBSCAN [14] is a classical density-based clustering algorithm. DBSCAN needs two input parameters: ε is the radius of a neighborhood for each point, and *MinPts* is the least number of points within the ε-neighborhood of the points.

OPTICS [16] is another well-known representative density-based clustering algorithm. The parameter k is the number of nearest neighbors.

BOOL [26] is a new and very fast hierarchical clustering algorithm. BOOL has three input parameters: k is the lower bound on number of clusters, l is the distance parameter, and o is the outlier parameter.

CLASP [27] is a new clustering algorithm. CLASP has five input parameters: m is the number of clusters, k is the number of nearest neighbors, k_for_Lof is the adjusting parameter according to the size of dataset, d is the dimension reducing flag, and t_{max} is the maximal number of iterations for position adjusting.

CFDP [29] is the current most popular clustering algorithm. CFDP needs two parameters when clustering: k is the number of nearest neighbors, m is the number of selected centres.

OPTICS, k-Means and DBSCAN are obtained from scikit-learn[1], which is a Python module for machine learning built on top of SciPy and distributed under the 3-Clause BSD license. The source codes of BOOL, CLASP and CFDP are provided by their authors. MulSim is implemented in Python.

2) BENCHMARKS

We evaluate MulSim and the baselines on a wide range of datasets including two dimensional datasets and multi-dimensional datasets. Table 1 lists the description of all the 12 datasets and the corresponding parameters of each algorithm, where *Points* stands for the number of points, *Dim.* stands for the number of dimensions of each point, and *Clusters* stands for the number of clusters. Among the two dimensional datasets, Aggregation, Compound, D31, Spiral, Flame and R15 are obtained from University of Eastern Finland website[2]; Toy is made by ourselves, which contains a cluster with two centres. Compound represents datasets containing arbitrary shaped clusters with various densities. Toy represents datasets containing a two-centre cluster. Spiral represents dataset containing special shaped clusters, i.e. spiral shaped cluster. R15 represents datasets containing concave-shaped clusters. Aggregation and Flame represent datasets containing arbitrary shaped clusters with uniform density, in which the borders between two clusters are composed of points with relatively sparse densities.

The multi-dimensional datasets are taken from UCI website, and the dimensions are range from three to forty-four.

3) TUNING AND VALIDATION

For each dataset, we determine the relatively optimal clustering result by tuning their corresponding input parameters. For BOOL, CLASP and k-Means, we input the right number of clusters. The appropriate noise parameter o of BOOL on

[1] http://scikit-learn.org/stable/index.html

[2] http://cs.joensuu.fi/sipu/datasets/

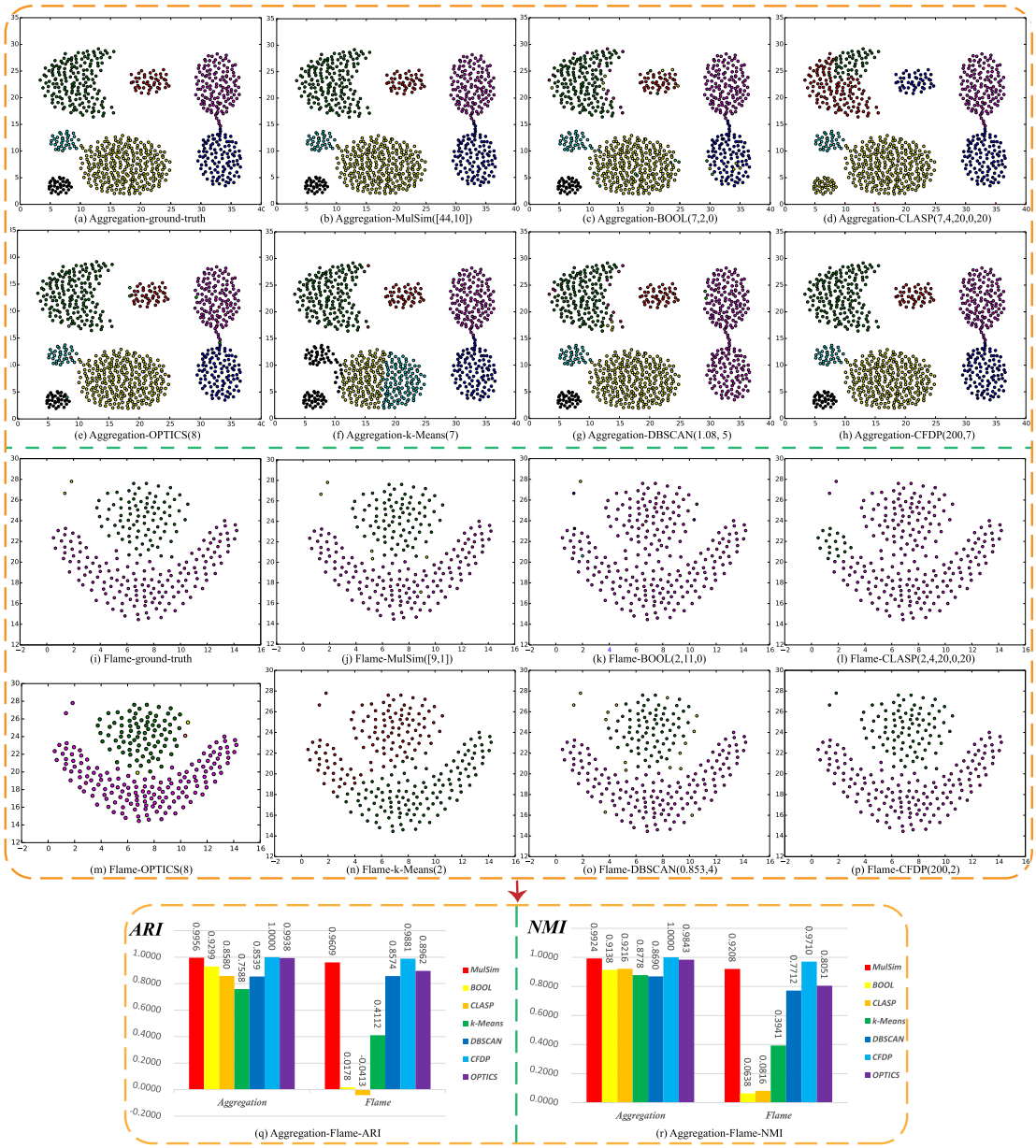


FIGURE 2. A comparison of clustering results on the uniform density datasets with any-shaped clusters.

all the benchmarks is obtained by iteratively running. The parameters k and t_{max} of CLASP are set as the authors suggested. For CFDP, the density is determined by the average distance of 2 percent of neighbors as in the code the authors provided, and cluster centres are selected manually. When the exact number of cluster centres on the decision graph is bigger than the correct cluster number, we select points with both relatively lager minimum-distances and larger densities as centres of clusters' according to the ground-truth. We choose Euclidean distance as the absolute distance for MulSim.

Since all the 12 datasets already have known clusters, the performances of MulSim and the six baselines are quantitatively measured by two widely used evaluation measures: Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI).

B. TWO DIMENSIONAL BENCHMARK DATASETS

To demonstrate that MulSim can be applied to dataset containing clusters with widely different shapes, sizes and varying densities, we select datasets which can represent different clustering instances. The clustering results of the seven algorithms on the datasets are exhibited from Figure 2 to 6, and the corresponding parameters for each of the seven algorithms on each of the datasets are given in the bracket respectively. The corresponding ARI and NMI are listed in Table 2.

1) *On the Uniform Density Datasets With Any-Shaped Clusters:* We select two representative datasets which

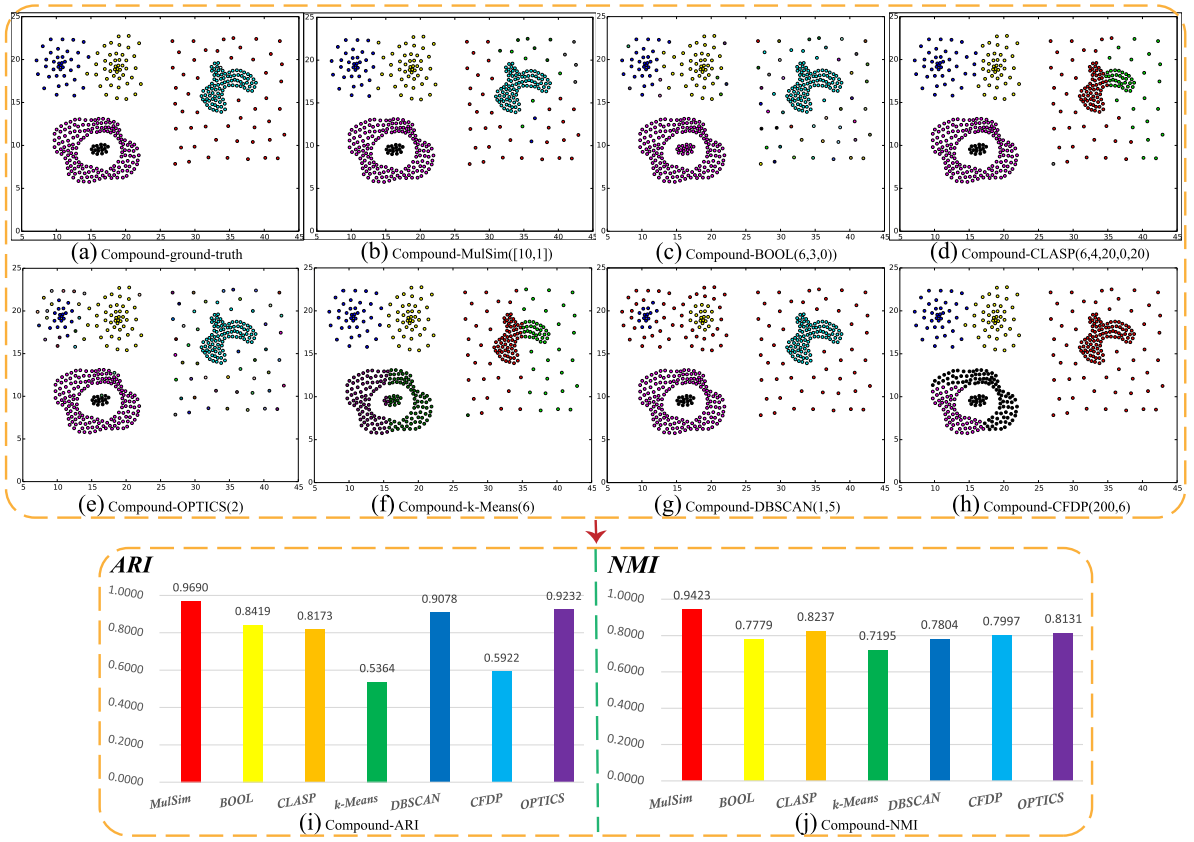


FIGURE 3. A comparison of clustering results on the various density datasets with any-shaped clusters.

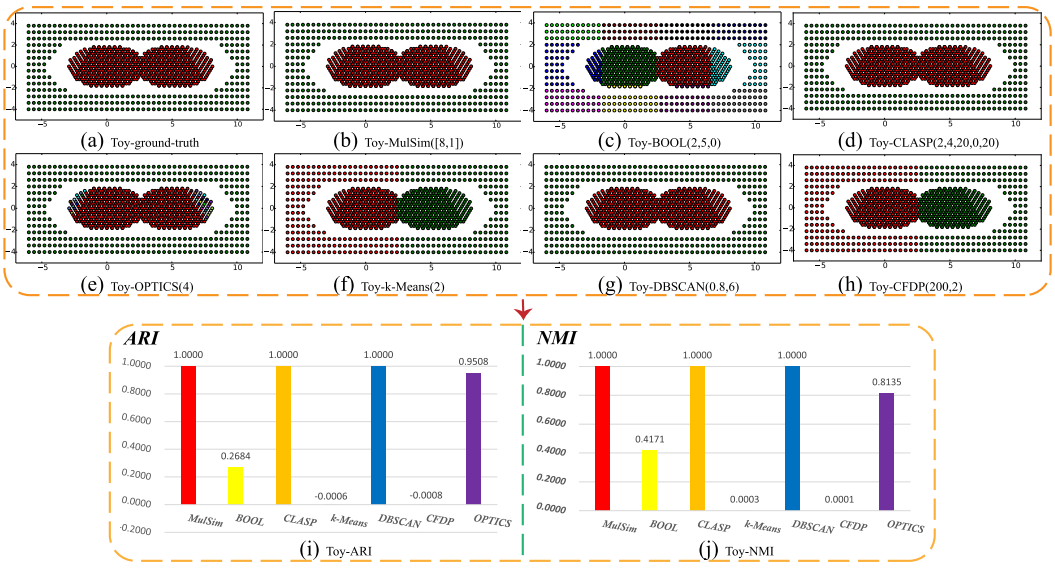


FIGURE 4. A comparison of clustering results on the dataset containing multi-centre cluster.

represent two different difficulties in clustering. Aggregation is a uniform density dataset containing clusters with different sizes and shapes. The key to identify this dataset is to disconnect the two bridges, i.e., the yellow and the purple bridges shown in Figure 2(a). The graphical descriptions and the quantitative comparison of the clustering results are shown in Figure 2. As illustrated from Figure 2(b) to Figure 2(h), MulSim and CFDP ideally identify the cluster structures of Aggregation. OPTICS and BOOL get the basically correct clustering results except for mistaking very few points for

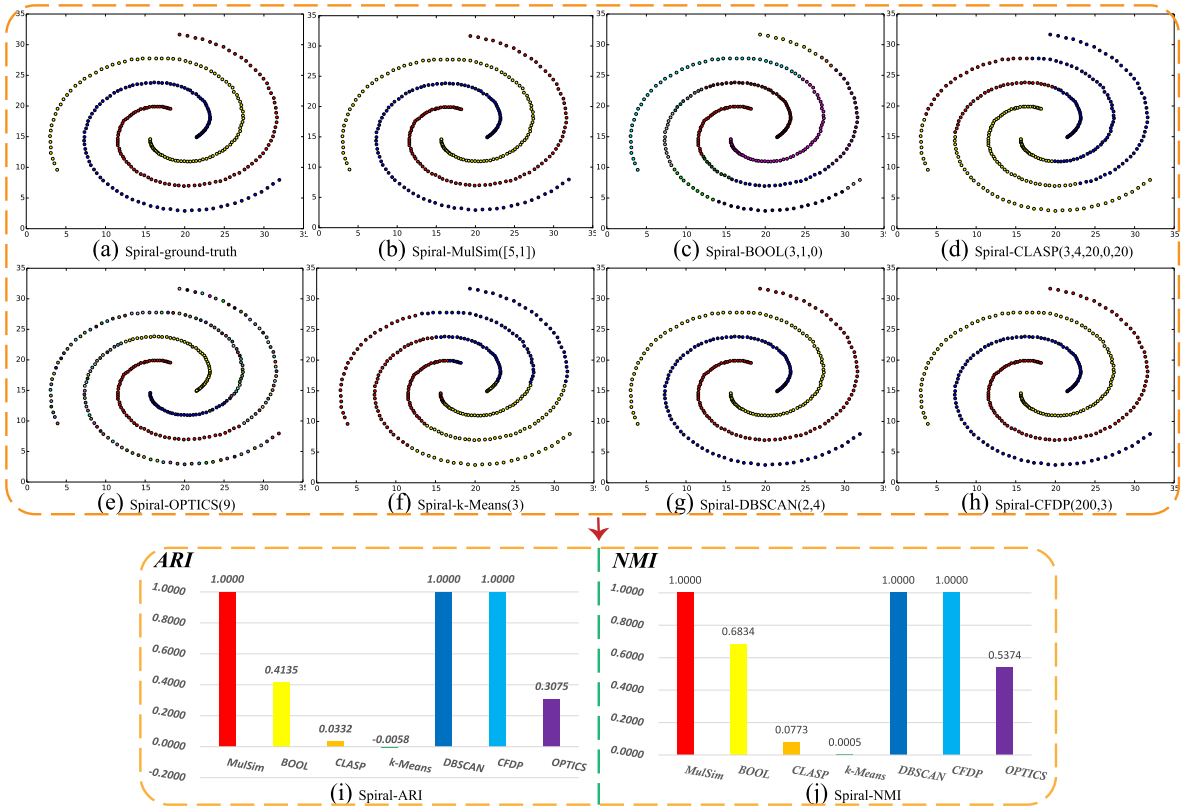


FIGURE 5. A comparison of clustering results on the dataset containing the spiral shaped cluster.

TABLE 2. The ARI and NMI of different methods on two dimensional datasets.

Algorithm	Aggregation		Compound		Flame		R15		Spiral		Toy	
	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
MulSim	0.9956	0.9924	0.9690	0.9423	0.9609	0.9208	0.9837	0.9861	1.0000	1.0000	1.0000	1.0000
BOOL	0.9299	0.9138	0.8419	0.7779	0.0178	0.0638	0.8997	0.9557	0.4135	0.6834	0.2684	0.4171
CLASP	0.8580	0.9216	0.8173	0.8237	-0.0413	0.0816	0.6388	0.8542	0.0332	0.0773	1.0000	1.0000
OPTICS	0.9938	0.9843	0.9232	0.8131	0.8962	0.8051	0.9600	0.9693	0.3075	0.5374	0.9508	0.8135
k-Means	0.7588	0.8778	0.5364	0.7195	0.4112	0.3941	0.9928	0.9942	-0.0058	0.0005	-0.0006	0.0003
DBSCAN	0.8539	0.8690	0.9078	0.7804	0.8574	0.7712	0.9160	0.9424	1.0000	1.0000	1.0000	1.0000
CFDP	1.0000	1.0000	0.5922	0.7997	0.9881	0.9710	0.9928	0.9942	1.0000	1.0000	-0.0008	0.0001

outliers. While k-Means roughly partitions the dataset to seven clusters without finding the correct cluster centres. DBSCAN cannot disconnect the bridge and is also frustrated at slightly non-uniform density points, and CLASP only detects four out of the seven clusters.

Flame is a dataset containing two clusters and two outliers. One cluster is convex shaped and the other cluster is non-convex shaped. Since the density of points between the upper cluster and the lower cluster is relatively sparse, the key to identify clusters from Flame is to break the dataset at this sparse area. Figure 2(i) to 2(p) exhibit the clustering results on dataset Flame. According to both Figure 2 and Table 2, we can find that CFDP and MulSim generate the satisfying clustering results. CFDP gets the best ARI and NMI and MulSim gets the second best ARI and NMI. OPTICS and DBSCAN also basically find the upper and the lower clusters except for regarding some normal points as outliers. Nevertheless, the other three methods, BOOL, CLASP and k-Means cannot partition the dataset properly.

Thus, MulSim can find the borders between two clusters on the uniform density dataset and further partition the dataset correctly.

2) *On the Various Density Dataset With Any-Shaped Clusters:* Compound is a very typical dataset which contains a variety of touchy situations for many clustering algorithms. As the ground truth shows in Figure 3(a), the two clusters on the upper left corner represent the situation that densities in one cluster are various, and the two clusters on the right represent the situation that densities in different clusters are various in one dataset, and the outer cluster at the bottom of left corner represents the situation that there is no clear centre in a cluster. Hence, identifying the clusters in this dataset is rather challenging. From Table 2 and Figure 3, we can see that MulSim gets the biggest ARI and NMI, and only MulSim

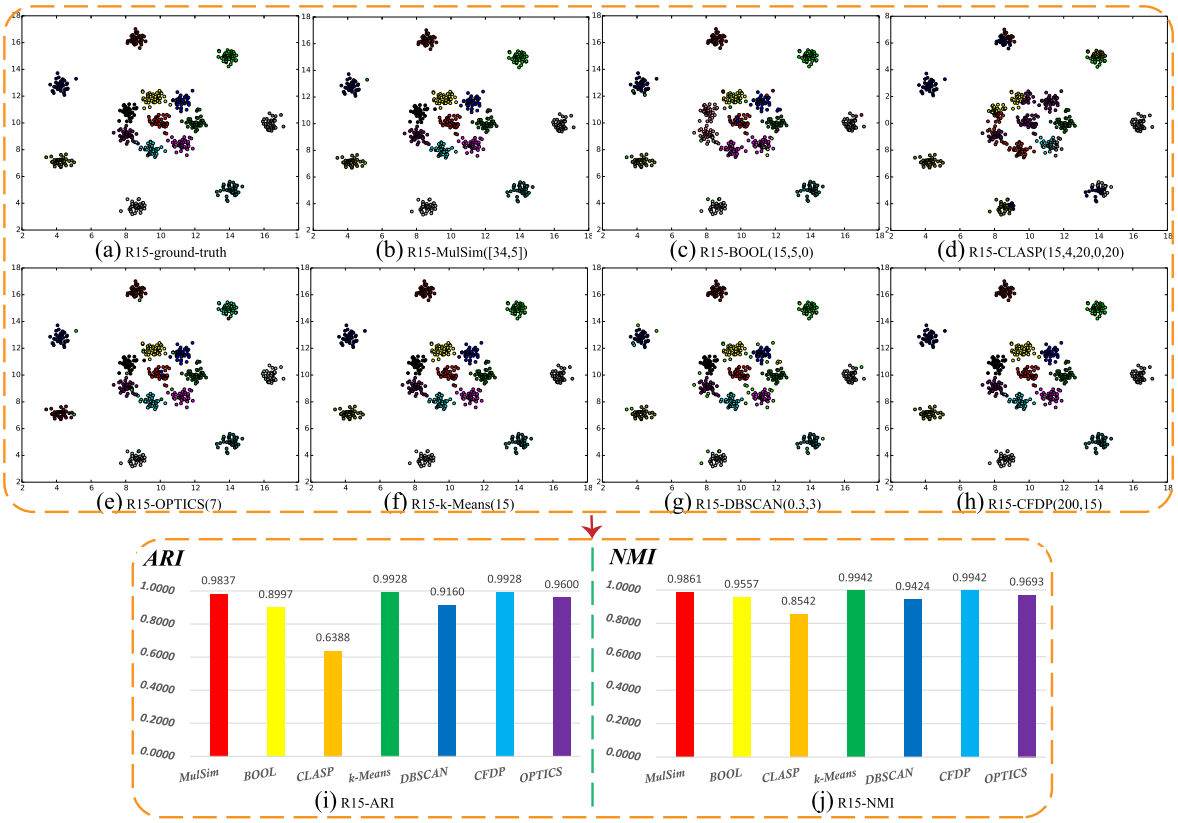


FIGURE 6. A comparison of clustering results on the dataset containing convex shaped clusters.

can basically identify all the clusters except for a few points. While OPTICS, BOOL and DBSCAN perform not so well as MulSim when densities vary, and CLASP, k-Means and CFDP encounter difficulty in determining the cluster centres.

3) *On the Dataset Containing Multi-Centre Cluster:* As depicted in Figure 4, MulSim, DBSCAN and CLASP generate the ideal clustering results. OPTICS identifies the basically correct shapes. While k-Means and CFDP inappropriately partition the dataset according to the two centroids, and BOOL mistakenly identifies the clusters. Hence, on this type of dataset, if a method detects clusters purely basing on centres, it will fail to find the correct cluster structures.

4) *On the Dataset Containing the Spiral Shaped Cluster:* Figure 5 demonstrates the clustering results on the Spiral dataset. Among the seven methods, only MulSim, DBSCAN and CFDP can get the correct results. For MulSim, the similar-to-multiple-point clustering strategy makes a big difference to its great performance. DBSCAN performs well because the dataset has uniform density. CFDP works properly owing to its combination of centre-based and density-based clustering methods.

5) *On the Datasets Containing Convex Shaped Clusters:* From both Figure 6 and Table 2, we can conclude that MulSim, CFDP, OPTICS and k-Means obtain the preferable cluster structures. Of them, k-Means and CFDP, as centre-based clustering methods, show the best performance. MulSim gets the second best ARI and NMI slightly behind the k-Means and CFDP. While BOOL and DBSCAN basically identify the structures of clusters, and CLASP only identifies seven correct clusters out of fifteen.

C. MULTI-DIMENSIONAL BENCHMARK DATASETS

In this section, six widely-used multi-dimensional datasets which have ground truths are used to demonstrate that MulSim is capable of clustering multi-dimensional datasets.

Table 3 lists the corresponding ARI and NMI of the clustering results generated by MulSim, BOOL, CLASP, k-Means, DBSCAN, CFDP and OPTICS on the six multi-dimensional datasets respectively. The corresponding parameters of each method on each dataset are listed in Table 1. Note that on dataset Wpbc and Spectef, DBSCAN cannot get valid partition, because it regards each single point as an individual cluster. OPTICS also gets invalid partition on dataset User. Figure 7 shows a quantitative comparison of clustering results of different methods on the 6 multi-dimensional datasets. On each dataset, MulSim obtains the best ARI, as shown in both Table 3 and Figure 7. In addition, on Elico, MulSim shows the second best NMI, and k-Means shows the best NMI while its ARI is the sixth best. On Haberman, MulSim gets the third biggest NMI, and OPTICS gets the biggest NMI and the second best ARI. On Iris, MulSim shows the third biggest

TABLE 3. The ARI and NMI of different methods on multi-dimensional datasets.

Algorithm	Elico		Haberman		Iris		Wpbc		Spectef		User	
	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
MulSim	0.6561	0.6321	0.2415	0.1156	0.6418	0.6727	0.0712	0.0477	0.3220	0.2540	0.2241	0.2304
BOOL	0.3112	0.3521	0.0392	0.0341	0.2802	0.3788	0.0368	0.0729	0.0190	0.0588	0.0146	0.0963
CLASP	0.5062	0.5648	-0.0237	0.0028	0.1267	0.2642	-0.0316	0.0273	0.0267	0.1430	0.0987	0.1367
OPTICS	0.5088	0.5462	0.1940	0.2329	0.5657	0.7452	0.0000	0.0000	0.0006	0.0579	-	-
k-Means	0.5060	0.6420	-0.0011	0.0009	0.6201	0.6595	0.0282	0.0237	0.0076	0.0970	0.1572	0.2160
DBSCAN	0.6437	0.6237	0.1906	0.2295	0.5681	0.7612	-	-	-	-	0.1089	0.4401
CFDP	0.6332	0.5995	0.0448	0.0112	0.4531	0.6586	0.0136	0.0009	0.0285	0.0318	0.1089	0.1412

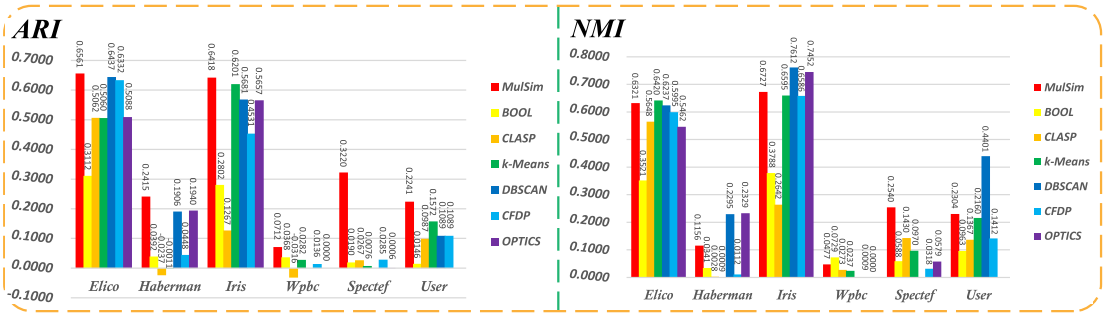


FIGURE 7. A quantitative comparison of clustering results of different methods on the 6 multi-dimensional datasets.

NMI, and DBSCAN shows the biggest NMI and the third best ARI. On Wpbc, only MulSim, BOOL, k-Means and CFDP get positive values in ARI and NMI. On Spectef, MulSim gets the biggest ARI and NMI, and the performances of the other algorithms are far behind that of MulSim's. On User, MulSim gets the second biggest NMI, and DBSCAN gets the biggest NMI and the third best ARI.

In most cases, on these multi-dimensional datasets, just like the performances on the two-dimensional datasets, clustering results of MulSim outperform that of the other six algorithms. Therefore, MulSim is capable of clustering multi-dimensional datasets.

D. COMPREHENSIVE ASSESSMENT

To comprehensively analyze the performances of MulSim and the baselines, in this section, we use box plot as a descriptive statistics means. Figure 8(a)is the box plot of ARI on two dimensional datasets. MulSim shows the best performance by a landslide in quartile, median, minimum and maximum of the ARI respectively. Figure 8(b) is the box plot of ARI on multi-dimensional datasets, and Figure 8(c) is the box plot of ARI on all these benchmarks including the six two dimensional and the six multi-dimensional datasets. From Figure 8(b), we can see that DBSCAN gets the best ARI in the upper quartiles, median and minimum. From Figure 8(c), we can see that DBSCAN gets the best ARI in the lower quartiles, median and minimum. The main reason is that DBSCAN has two missing values of ARI on datasets Wpbc and Spectef. For a data group containing a few members, the missing values can greatly affect the statistical performance of the group. Therefore, taking these factors into account, MulSim shows the best statistics performance of ARI.

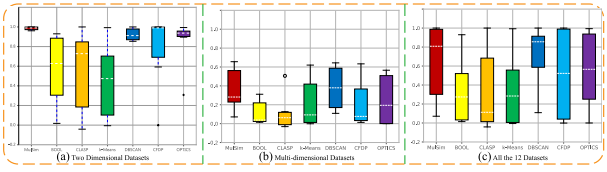


FIGURE 8. Three box plots of ARIs on two dimensional, multi-dimensional and all the 12 datasets.

According to all the above experiments, we analyze and compare the characteristics of different algorithms respectively. Among them, k-Means can only find the convex clusters. It can not effectively identify non-convex clusters as in datasets Aggregation, Compound, Toy, Flame and Spiral. DBSCAN runs well on the dataset with uniform density, but it detects the normal points with relatively low density as outliers on the dataset with various densities. Even on uniform density dataset, if there are no clear borders of clusters, it will mistakenly connect two clusters together. That is why DBSCAN cannot correctly detect the cluster structures on Compound and Aggregation. The centre-based clustering strategy leads to the failure of CFDP on Compound and Toy which contain clusters with two centres. That is the reason why CFDP does not show obvious advantages in the plot boxes, although it shows good performance on the other datasets containing the cluster with only one centre. OPTICS, as an improved algorithm of DBSCAN, generates relatively good clustering results on the datasets except on Spiral, but it still suffers the pain in dealing with various density datasets. BOOL regards a few normal points as outliers on some datasets, and its clustering results are especially not satisfying on Flame, Toy, and Spiral. CLASP has poor performances on

Flame and Spiral. As for MulSim, not only can it find cluster structures correctly but also it can obtain the best clustering results on most of the datasets, except for a few flaws on Compound.

Therefore, MulSim is an effective clustering method in mining clusters with arbitrary shapes and various densities on datasets with different dimensions and cluster numbers. The reason for the excellent performance of MulSim is that both the new distance and the similar-to-multiple-point clustering strategy are used.

E. STABILITY OF CLUSTERING QUALITY AS DATASET SIZES CHANGE

Modern clustering applications require algorithms run gracefully with different dataset sizes, especially on datasets with large number of data points. In this section, we have generated synthetic datasets as the way illustrated in Figure 9. We can see that the main shapes of the three clusters stay still while their densities are changing from sparse to dense. By extending the dataset size within a 5K interval from 0 to 100k, we have generated 20 datasets. Figure 10 demonstrates the quantitative clustering results of MulSim when varying the size of the datasets. As can be seen, the ARI and NMI of MulSim always stay the best value on all the synthetic datasets. On account of this, we say MulSim can adapt to varying dataset size, and keep fine stability of clustering quality as the size of dataset increases. Besides, from Section IV-B, we can know the time complexity of MulSim is $O(n \cdot \log n)$. Thus, we can draw the conclusion that MulSim has the ability to cope with large datasets.

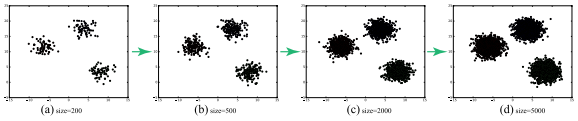


FIGURE 9. Demonstration of the process of generating different size datasets.

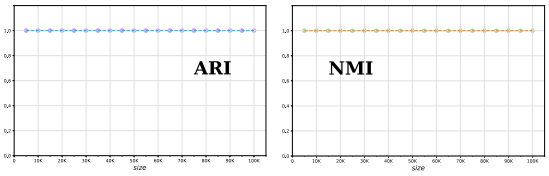


FIGURE 10. Quantitative results of MulSim on different size datasets.

F. PARAMETER ESTIMATION

In this section, we evaluate MulSim in terms of its sensitivity to the regularization parameter k and m. The dataset used is the above synthetic dataset with 20K points. Figure 11 shows the clustering ARI and NMI with varying k and m. B is a figure of 3D view of the variation tendency of ARI, in which k is [1,500] and m is [1,10]. A is a part of profile of B on how m affects the clustering results, in which we use three m values to show three representative situations. C is a part of profile of B on how k affects the clustering results, in which we use three k values to show three representative situations. We can observe that MulSim always keeps the best clustering result with a long range of both k and m, in other words, MulSim is robust to the two input parameters. The bigger the m is, the wider range of k is. To reduce time cost, we suggest setting m to [1,5] in real applications.

Next, since CV(coefficient of variation) is a standardized measure of dispersion of a probability distribution for a dataset, we use CV to suggest the range of input k in real applications. CV is defined as the ratio of the standard deviation σ to the mean μ. In this paper, μ is the average distance between any point and its first nearest neighbor.

To be more feasible, we have performed experiments on the 23 datasets we used above, including the six two dimensional, the six multi-dimensional datasets, as well as the eleven synthetic datasets. As shown in Figure 12, there is an approximately linear relationship between the right k and CV,

$$k = 23CV + 5 \tag{5}$$

Note that the k in Figure 12 is one of the ks set in MulSim when ARI ranges from its maximum to 0.9 times of the maximum on each dataset respectively.

Therefore, we suggest finding the right input k near the line (Eq. 5).

G. AN APPLICATION ON FACE RECOGNITION

In this section, we group the faces of the same person into a cluster on the Olivetti Face dataset [32]. Olivetti Face dataset is one of the popular face datasets, which is widespreadly used as a benchmark for machine learning algorithms. There are ten different images for each of 40 distinct persons in the Olivetti Face dataset. Each face is treated as a long vector of 10304 features. As in [29], the similarity between two images is calculated by [33].

In Figure 13, we show the clustering results performed by MulSim of the first ten persons in the dataset, where the images of the same color correspond to one cluster, and the gray images mean outliers which are not assigned to any cluster. As shown in Figure 13, except for the third and the fourth persons who are assigned into one cluster, each of the other eight persons are approximately identified, which shows that the MulSim can essentially identify 8 persons out of ten, and the ARI is 0.5531. If we take CFDP [29] into consideration, where the ARI is 0.3244, the performance of MulSim is even more impressive. Therefore, MulSim is applicable on face recognition.

H. THE CONTRIBUTIONS OF MulSim

From the above comprehensive studies and experiments, we can conclude that MulSim has the following contributions.

(1) MulSim is a new clustering algorithm which can detect clusters with various densities, shapes and sizes from different dimensional datasets.

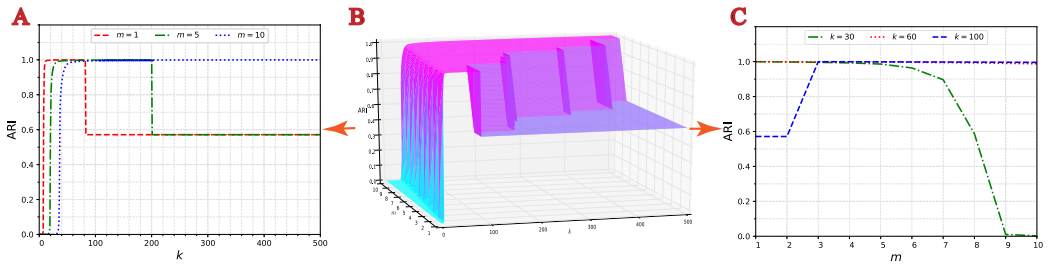


FIGURE 11. Parameter sensitivity evaluation.

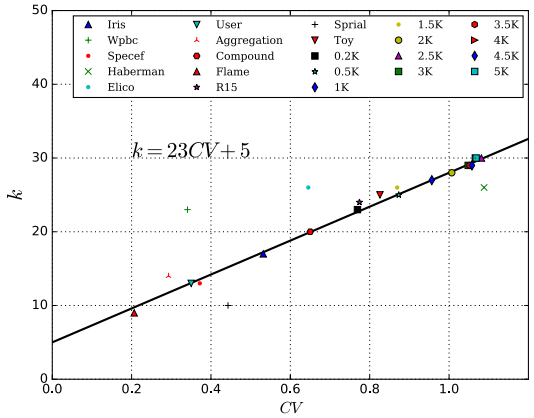


FIGURE 12. *k* estimation.

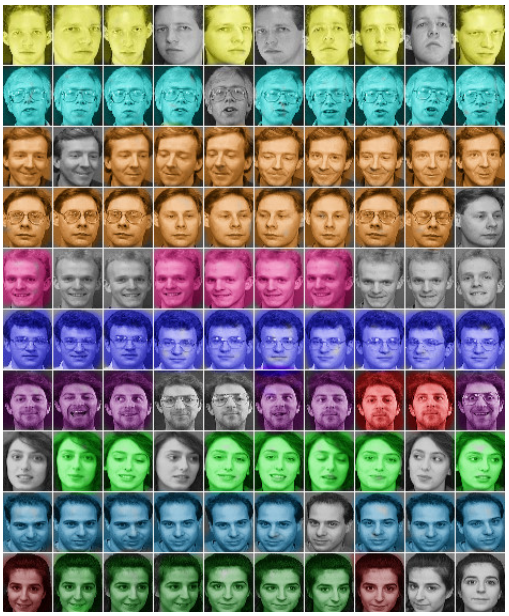


FIGURE 13. Cluster analysis of the first 100 images in the Olivetti Face dataset.

(2) MulSim defines a new distance which differs from the absolute distances. Our new distance can adaptively change along with the change of densities when clustering.

(3) MulSim adopts a similar-to-multiple-point clustering strategy, which makes discovering arbitrary shaped clusters more effective compared with the traditional similarity-based clustering algorithms.

(4) MulSim can keep fine stability of clustering quality as the size of dataset increases.

(5) MulSim is robust to the input parameters, and we have suggested the way to estimate the input parameters.

(6) Comprehensive experiments on various datasets have been conducted to evaluate the effectiveness of MulSim.

VI. CONCLUSION

In this paper, to mine clusters with widely different shapes, sizes and densities, we have presented an effective and efficient algorithm known as MulSim. The algorithm defines a novel distance based on nearest neighbor relationship, which can automatically adapt to different densities when clustering. More remarkable, MulSim adopts a similar-to-multiple-point clustering strategy to group points together. Extensive experiments further demonstrate that MulSim is capable of finding clusters on both two-dimensional and multi-dimensional datasets with high quality, and it also shows attractive superiorities comparing with several state-of-the-art methods. In future work, we will develop our method to be more adaptable to large size datasets.

ACKNOWLEDGMENT

We are hugely grateful to the authors who provide their source codes for us.

REFERENCES

[1] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, *Big Data Clustering: A Review*. Cham, Switzerland: Springer, 2014.
[2] C. Zhu, F. Wen, and J. Sun, "A rank-order distance based clustering algorithm for face tagging," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 481–488.
[3] M. Chen, P. Wang, Q. Chen, J. Wu, and X. Chen, "A clustering algorithm for sample data based on environmental pollution characteristics," *Atmos. Environ.*, vol. 107, pp. 194–203, Apr. 2015.
[4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
[5] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdiscipl. Rev. Data Mining Knowl. Discovery*, vol. 2, no. 1, 86–97, 2012.
[6] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, "Efficient agglomerative hierarchical clustering," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2785–2797, 2015.
[7] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.

[8] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, pp. 103–114, Jun. 1996.

[9] L. Kaufman and P. J. Rousseeuw, "Partitioning around medoids (program PAM)," *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, NY, USA: Wiley, 1990, pp. 68–125.

[10] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 1, 1967, pp. 281–297.

[11] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[12] J. C. Bezdek, W. Full, and R. Ehrlich, "FCM: The fuzzy C-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, 1984.

[13] J. Han, M. Kamber, and J. Pei, *Data Mining, Southeast Asia Edition: Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96. 1996, pp. 226–231.

[15] M. Chen, L. Li, B. Wang, J. Cheng, L. Pan, and X. Chen, "Effectively clustering by finding density backbone based-on *K*NN," *Pattern Recognit.*, vol. 60, pp. 486–498, Dec. 2016.

[16] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD. Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999.

[17] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th. Int. Conf. Knowl. Discovery. Data Mining*, vol. 98, Aug. 1998, pp. 58–65.

[18] Z. Yu, X. Zhu, H.-S. Wong, J. You, J. Zhang, and G. Han, "Distribution-based cluster structure selection," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3554–3567, Nov. 2017.

[19] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. Int. Conf. Data Eng.*, Feb. 1998, pp. 324–331.

[20] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, 1998.

[21] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[22] A. Bojchevski, Y. Matkovic, and S. Günnemann, "Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 737–746.

[23] V. Chaoji, G. Li, H. Yildirim, and M. J. Zaki, "Abacus: Mining arbitrary shaped clusters from large datasets based on backbone identification," in *Proc. SDM*. Philadelphia, PA, USA: SIAM, 2011, pp. 295–306.

[24] S. T. Mai, I. Assent, and M. Storgaard, "AnyDBC: An efficient anytime density-based clustering algorithm for very large complex datasets," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1025–1034.

[25] A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum, "A hierarchical algorithm for extreme clustering," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 255–264.

[26] M. Sugiyama and A. Yamamoto, "A fast and flexible clustering algorithm using binary discretization," in *Proc. IEEE 11th Int. Conf. Data Mining (ICDM)*, Dec. 2011, pp. 1212–1217.

[27] H. Huang, Y. Gao, K. Chiew, L. Chen, and Q. He, "Towards effective and efficient mining of arbitrary shaped clusters," in *Proc. IEEE 30th Int. Conf. Data Eng. (ICDE)*, Mar./Apr. 2014, pp. 28–39.

[28] V. Chaoji, M. Al Hasan, S. Salem, and M. J. Zaki, "SPARCL: An effective and efficient algorithm for mining arbitrary shape-based clusters," *Knowl. Inf. Syst.*, vol. 21, no. 2, pp. 201–229, 2009.

[29] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[30] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[31] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1997, pp. 1000–1006.

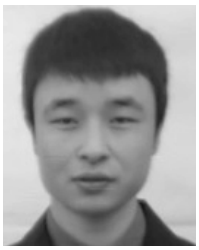
[32] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop Appl. Comput. Vis.*, Dec. 1994, pp. 138–142.

[33] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: A new image similarity index," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–2401, Nov. 2009.

MEI CHEN received the Ph.D. degree in computer science from Lanzhou University in 2016. He is currently a Professor with the School of Electronic and Information Engineering, Lanzhou Jiaotong University. She has published over 20 research papers in many conferences and journals, such as WAIM, *Pattern Recognition*, *Atmospheric Environment*, and *Frontiers of Computer Science*. Her research interests include artificial intelligence and data mining. She is a member of CCF.

XIAOFANG WEN is currently pursuing the master's degree in computer software and theory with the School of Electronic and Information Engineering, Lanzhou Jiaotong University. Her research field is data mining.

ZHICHONG YANG is currently pursuing the master's degree in computer technology with the School of Electronic and Information Engineering, Lanzhou Jiaotong University. His research fields are community detection and clustering.



MING LI is currently pursuing the master's degree in computer technology with the School of Electronic and Information Engineering, Lanzhou Jiaotong University. His research fields are community detection and clustering.

MEI ZHANG is currently pursuing the master's degree in computer technology with the School of Electronic and Information Engineering, Lanzhou Jiaotong University. Her research fields are community detection and clustering.