IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED DATA ANALYTICS FOR LARGE-SCALE COMPLEX DATA ENVIRONMENTS

Received September 4, 2017, accepted November 2, 2017, date of publication December 6, 2017, date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2780109

From Partition-Based Clustering to Density-Based Clustering: Fast Find Clusters With Diverse Shapes and Densities in Spatial Databases

JIANG WANG, CHENG ZHU, YUN ZHOU, XIANQIANG ZHU, YILIN WANG, AND WEIMING ZHANG

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

Corresponding author: Yun Zhou (zhouyun@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 71571186, Grant 71471176, and Grant 61703416 and in part by the China Post-Doctoral Science Foundation under Grant 2016M593018.

ABSTRACT Spatial data clustering has played an important role in the knowledge discovery in spatial databases. However, due to the increasing volume and diversity of data, conventional spatial clustering methods are inefficient even on moderately large data sets, and usually fail to discover clusters with diverse shapes and densities. To address these challenges, we propose a two-phase clustering method named KMDD (clustering by combining K-means with density and distance-based method) to fast find clusters with diverse shapes and densities in spatial databases. In the first phase, KMDD uses a partition-based algorithm (K-means) to cluster the data set into several relatively small spherical or ball-shaped subclusters. After that, each subcluster is given a local density; to merge subclusters, KMDD utilizes the idea that genuine cluster cores are characterized by a higher density than their neighbor subclusters and by a relatively large distance from subclusters with higher densities. Extensive experiments on both synthetic and real-world data sets demonstrate that the proposed algorithm has a near-linear time complexity with respect to the data set size and dimension, and has the capability to find clusters with diverse shapes and densities.

INDEX TERMS Spatial clustering, partition-and-merge strategy, diverse shapes and densities, efficiency on large spatial databases.

I. INTRODUCTION

The advances in location-acquisition technologies, such as Global Position Systems (GPS), Automatic Identification Systems (AIS), sensor networks and mobile devices, have resulted in accumulating large amounts of spatial data related to people, vehicles, animals and natural phenomena. Knowledge discovery in spatial databases, or called as spatial data mining, is the process of extracting implicit knowledge about spatial objects, spatial relations and discovery of interesting spatial patterns that cannot be explicitly examined by people [1]. Spatial clustering, which groups spatial data into meaningful classes according to their similarities, is one of the major tools for spatial data mining. It has attracted attentions from diverse disciplines, given its importance in numerous applications such as epidemiology, crime analysis, intelligent transportation systems, urban computing, animal movement, and natural phenomena [2]–[7].

Due to the increasing amount and complexity of spatial data, a key challenge for clustering algorithms is to achieve good time efficiency [8]. Scalability becomes more and more important, and mainly manifests in two aspects: enormous data volume and high dimensionality [9]. In addition, due to the diverse nature and characteristics of spatial data sources, clustering algorithms should have the capability to find the clusters hidden in spatial datasets that may be of different shapes and densities [10]. Despite some well-known clustering algorithms try to offer the solution for these two requirements at the same time, most of them become inefficient even on moderately large datasets (both data volume and dimension) or ineffective on datasets with diverse densities and cluster shapes.

Recently, Rodriguez and Laio [28] proposed a significant clustering algorithm called Density and distance-based clustering (DD), of which the time consumption is high but the core idea is novel. In this paper, motivated by their work, we propose a new clustering algorithm KMDD for large spatial datasets to overcome the above limitations of existing clustering algorithms. KMDD is a two-phase algorithm, which

uses K-means to cluster the data into several relatively small subclusters in the partition phase, then uses the density and distance-based concept from [28] to aggregate these subclusters for finding the genuine clusters in the merging phase. The contributions of our study are summarized as follows:

(a) So far, to the best of our knowledge, this is the first work that employs the density and distance-based concept to aggregate subclusters, which is different from existing agglomerative algorithms that primarily measure the connectivity or disconnectivity between different sub-clusters or existing and merged subclusters by using functions of distance or probability.

(b) We introduce a new clustering method called KMDD as an alternative to DD on large spatial datasets. We show that the quality of its clustering result is closed to DD, and its running time linear to the dataset size and dimension.

(c) We perform experiments on datasets containing clusters with diverse shapes and densities and on datasets that significantly larger than those used in the previous work to our awareness. The experiments demonstrate that the new clustering method can effectively identify clusters of diverse shapes and densities for spatial datasets, and its scalability outperforms some of the widely used spatial clustering methods.

The rest of this paper is structured as: Section II reviews previous work related to ours. Section III provides preliminaries and proposes KMDD clustering method. In Section IV, to measure the performance of KMDD, extensive experiments on both synthetic and real-world datasets are conducted. Conclusions and possible further work on this research are given in Section V.

II. RELATED WORK

In this section, to introduce some basic ideas and concepts, we briefly review several kinds of widely used clustering methods related to our work.

A. PARTITION-BASED CLUSTERING

K-means is one of the most widely used partition-based clustering algorithms even though thousands of clustering algorithms have been published after it [11]. It can be deemed as a nice strategy to find subclusters in the partition phase because it is ease of implementation, efficient, and successfully applied in many real-world case studies. The core idea of K-means is to update the center of cluster which is represented by the centroid of data points, by iterative computation and the iterative process will be continued until the criteria for convergence is met.

Though K-means is simple and with a high computing efficiency in general, there still exist some drawbacks. K-means can easily converge to a local minimum. Other disadvantages include: K is hard to select; the clustering result is sensitive to K; not suitable for finding clusters with non-convex shape and relatively sensitive to the outliers. Like K-means, PAM [12], CLARA [13], CLARANS [14] and AP (Affinity Propagation) [15] are also typical partition-based clustering algorithms. However, their time complexities are reported to be higher than K-means and they also fail to find non-convex shape clusters [16].

B. DENSITY-BASED CLUSTERING

Density-based clustering methods assume that regions of points with high density in the data space are considered as clusters. DBSCAN [10] is the most well-known density-based clustering algorithm, which judges the neighborhood of a point is dense or not using two parameters: the radius *eps* of the neighborhood and the minimum number of points *MinPts* in the neighborhood.

DBSCAN algorithm has been claimed for years that it can terminate in $O(n \log n)$ time. However, Gan and Tao [17] proved that it requires at least $\Omega(n^{4/3})$ time to solve the problem with data dimension $\geq$ 3. Thus all varieties of DBSCAN, such as OPTICS [18], ST-DBSCAN [19], DENCLUE [20] are intolerably slow even on moderately large n in practice. To replace DBSCAN on big data, they proposed a grid-based approximation algorithm called ρ-approximate DBSCAN that can be solved in linear $O(n)$ expected time. DBSCAN can find clusters of arbitrary shapes but its performance is poor when clusters have greatly varied densities.

C. GRID-BASED CLUSTERING

Grid-based clustering methods partition the original data space into a grids structure with definite size for clustering. STING [21], CLIQUE [22] and Wavecluster [8] are typical algorithms of this kind and all their time complexity with respect to the data size n is low (near $O(n)$). However, scaling these methods to higher dimensional spaces is difficult [9]. Take Wavecluster as a representative of down-top partition methods: to partition the dataset with dimension d, m^d grids must be imposed (m is the number of segmentations in each dimension), which is a very large number for computation even d is not that large. As a top-down partition method, ρ-approximate DBSCAN uses a tree-like hierarchical grid partition structure that only keeps non-empty grids in each level. However, each non-empty grid must be divided into 2^d smaller grids of the same size until the side length of new grid is at most $eps * \rho\sqrt{d}$. In general, algorithms listed above lack the capability of managing high dimensionality datasets.

D. AGGLOMERATIVE HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering methods successively merge the most neighboring pair of clusters to form a cluster hierarchy from down to top. Typical algorithms include BIRCH [23], CURE [24], ROCK [25] and Chameleon [26]. BIRCH is to obtain an initial result by using a CF tree and then refining the result using iterative relocation. The latter three use hybrid methods that unite the pros of partition and hierarchical clustering, namely, partition-and-merge strategy. In the partition stage, CURE employs a combination of random sampling and a partitioning method, while the partition strategy of Chameleon is based on the k-nearest neighbor

graph. ROCK is an improvement of CURE for dealing with data of enumeration type. However, reported by [9] and [16], BIRCH can only detect convex shape clusters and the time complexity of CURE and Chameleon are relatively high $O(n^2 \log n)$ without sampling and $O(n^2)$ respectively).

There are many agglomerative strategies based on the different definitions for distance between two clusters, such as single linkage (nearest neighbor), complete linkage, group average linkage, median linkage, centroid linkage, Ward's method and probability-based method [27].

The partition-and-merge strategy is effective on discovering clusters with diverse shapes and densities, thus we integrate it in our work.

E. DENSITY AND DISTANCE-BASED CLUSTERING

DD in literature [28] is based on a straightforward idea about the cluster centers: (i) cluster centers must with high local density, namely, the number of data points near the cluster center within a certain range must be large enough; (ii) cluster center must be away from other data points that could be the center of a cluster, namely, cluster centers are away from other data points with high local density. A decision graph based on ρ (local density of a point) and δ (distance between a point and its nearest neighbor with a higher ρ) is used to select the cluster centers, and the remaining data points are assigned to their nearest cluster centers with higher densities. The border region of each cluster is defined as the set of points assigned to the cluster whose distances to data points of other clusters are within a cutoff distance d_c. The points in the cluster whose density is lower than the border region are considered as noises.

DD can deal with the datasets with arbitrary shape and is insensitive to the outliers, yet it is not suitable for large-scale datasets since the time complexity is $O(n^2)$ and the cluster centers are hard to be chosen when hundreds thousand points crowd into the decision graph.

F. OTHER RELATED CLUSTERING METHODS

Distribution-based clustering methods [41] [43] suppose that in the original datasets there exist some different distributions, then the data 'generated from the same distribution' should be assigned to the same cluster.

Advances in sensing and storage technology and the dramatic growth in applications are leading to the collection of many high-volume, high-dimensional, and real-time data sets. Clustering for large-scale data is aim at overcoming the challenge of clustering huge data points that with thousands of features [42]. This needs algorithms which are both efficient and with the capability of handling high-dimensions data, such as K-means, BIRCH, CLARA, CURE and etc. In addition, clustering for data streams and clustering in parallel environments are also faced with the problem of data explosion, for more detailed information, refer [40], [42].

III. MATERIALS AND METHODS

DD is feasible only in small datasets since its computation time is strongly quadratic to n such that it will take an intolerably long period to get the clusters precisely. In addition, K-means has an obvious drawback that it can only find spherical or ball-shaped clusters. In this section, we overcome their disadvantages by combining these two methods and introduce the concept of KMDD designed to replace DD on large datasets.

Take Fig. 1 to show the motivation of our method, in which the 2D dataset [29] has 7500 points and 50 spherical clusters. Fig. 1(a) shows the result of K-means that have been obtained by running 5000 times the algorithm and taking the best solution according to the objective function. Though K has been set to the right value, the best solution still failed to reach the global optimum. However, interestingly, if we obtain more clusters (Fig. 1(b), by running K-means once) and then merge them properly, the quality of result can be improved a lot. Fig. 1(c) shows the result of KMDD by merging clusters from Fig. 1(b), which approximates to the result of DD in Fig. 1(d). Both KMDD and K-means got results instantly on this dataset, yet DD consumed a longer time (288 seconds on our machine). Therefore, if we set a lager K for K-means and merge clusters properly, we can obtain a better result; even obtain clusters with diverse shapes and densities. In addition, as an approximate DD, KMDD can replace it on large-scale datasets due to its computational efficiency and with fewer points for choice in the decision graph.

A. NOTION OF CLUSTERS

Let $X = x_1, x_2, \ldots, x_n$ be a set of n points in d-dimensional space $\Re^d$. Let $S = s_1, s_2, \ldots, s_K$ be a set of K subclusters obtained by partitioning X with K-means. $|s_k|$ denotes the number of elements of s_k. D_k is a $|s_k| \times K$ matrix returned by K-means, whose entries are distances from each point of s_k to all K centroids. We denote by d_{ij} the distance between s_i and s_j. d_c is a cutoff distance. $C = c_1, c_2, \ldots, c_G$ denotes a set of genuine clusters hidden in X.

Lemma 1: When K is large enough, K-means can partition datasets with arbitrary shapes and densities into relatively small subclusters and ensure each belongs to only one genuine cluster (hard partition). Namely, subclusters are small enough so that all points in a subcluster will not belong to two or more genuine clusters.

Proof: Let $K = n$, each subcluster has only one point and it will not belong to two or more genuine clusters. □

Intuitively, for a dataset without noise and its clusters are well separated by space, K is not very large compared to n. When clusters are overlapping or datasets have too much noise, a small K may be insufficient. At the worst case, K must be set to n so that subclusters have the smallest size (one point). In this condition our algorithm will be precisely the same to DD. In practice, if less precision is demanded (i.e. a smaller K), KMDD becomes more efficient.

Definition 1: The minimum distance between points of s_i and the centroid of s_j is not always equal to that between points of s_j and the centroid of s_i. Thus, the distance between

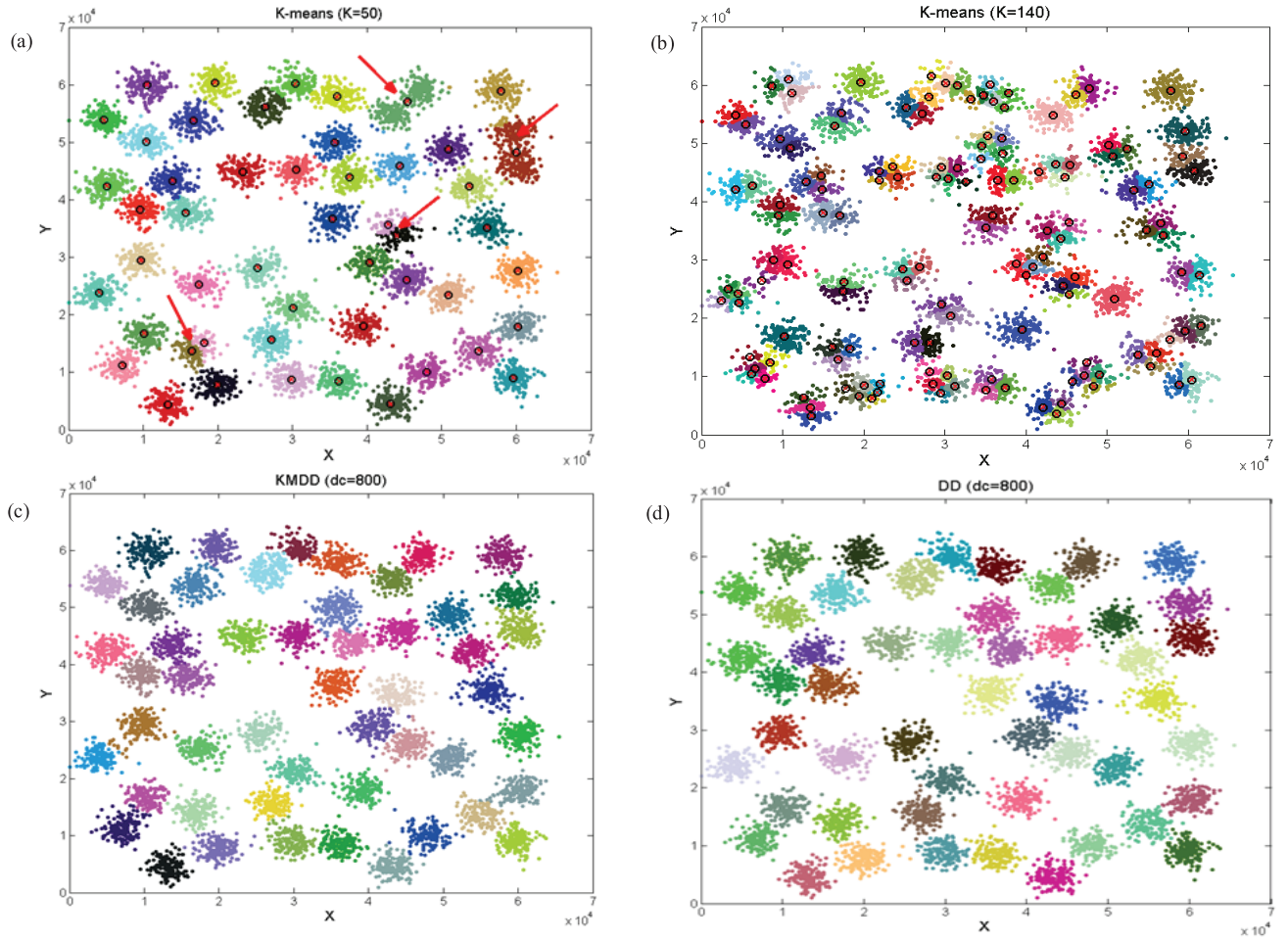

FIGURE 1. Results comparison among K-means, KMDD and DD. (a): K-means ($K = 50$). (b): K-means ($K = 140$). (c): KMDD ($d_c = 800$). (d): DD ($d_c = 800$).

s_i and s_j is denoted as the mean of the two:

$$d_{ij} = d_{ji} = \frac{1}{2}\left[\min_j (D_i.j) + \min_i (D_j.i)\right] \tag{1}$$

where $D_i.j$ is the j^{th} column of D_i and $D_j.i$ is the i^{th} column of D_j.

We utilize the distance defined in (1) instead of the distance between the centroids of subclusters. For finding non-convex shape clusters, the former provides more information about the closeness of two subclusters.

Definition 2: By rewriting the exponential kernel from [30], the local density ρ_i of subcluster i can be defined as:

$$\rho_i = \sum_{j=1}^{K} |s_k| \exp(\frac{d_{ij}^2}{d_c^2}) \tag{2}$$

Definition 3: The minimum distance δ between subcluster i and any other subclusters with higher density is measured by:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{3}$$

For the subcluster with highest density, $\delta_i = \max_j (d_{ij})$. The neighbor of subcluster i (exception of the subcluster with the highest density) is calculated by:

$$Ne(i) = arg \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{4}$$

Assumption 1: Each genuine cluster has only one core, which is: (i) the subcluster with high local density; (ii) away from other subclusters with high local density.

Based on this assumption, each non-core subcluster is assigned to the same cluster as its nearest neighbor with a higher ρ.

Assumption 1 suggests that one can choose cluster cores from subclusters instead of points in the decision graph, which makes the process of decision-making easier. In addition, since the subclusters extracted by K-means have nearly the same size, noise can be considered as a set made up of subclusters with low density.

Fig. 2 is an example illustrating how our method works. The 2D dataset has been taken from [31], with 240 points and 2 clusters. Since it has a non-convex shape, K-means is not able to deal with it. However, if we partition the data into some subclusters with K-means and merge them by the aggregation strategy above, this problem can be solved. Fig. 2(a) shows the original data and Fig. 2(b) shows 25 subclusters

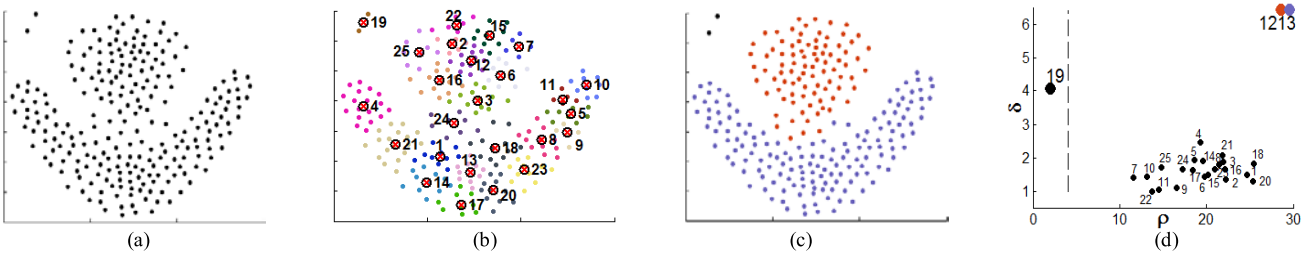

FIGURE 2. An example illustrating the clustering process. (a) Original data. (b) Subclusters. (c) Clustering results. (d) The decision graph.

found by K-means. Then we chose subcluster 12 and 13 as two cores and subcluster 19 as noise in the decision graph, shown in Fig. 2(d). It is obvious that cluster cores we selected are subclusters of relatively high δ and ρ in the decision graph, i.e., points lie in the right up corner in Fig. 2(d). A noticeable fact is that the decision graph here has only 25 points, while if we use DD, all original data will flood into the decision graph. Moreover, Fig. 2(c) shows that the 2 clusters were correctly found by our method. Note that if we didn't choose subcluster 19 as noise, it would be assigned to the cluster its nearest neighbor with a higher ρ (subcluster 25) belongs to, namely, the cluster colored red in Fig. 2(c).

TABLE 1. The proposed algorithm.

Algorithm 1 KMDD

Input: dataset X, number of subclusters K, number of clusters G, cutoff distance d_c.

Output: clusters C.

1. Use K-means to partition the dataset X into K subclusters, return subclusters $\{s_1, s_2, \ldots, s_K\}$ and distance matrices $\{D_1, D_2, \ldots, D_K\}$.
2. Compute the distance between each pair of subclusters by (1).
3. Compute ρ of each subcluster by (2).
4. Compute δ of each subcluster by (3), and get its neighbor by (4).
5. Use the decision graph to select G cores and (if need) noise subclusters.
6. Create G clusters for G cores respectively. If subcluster i (i=1, 2, …, K) is not a core or not belongs to noise, assign it to the cluster its neighbor belongs to. If noise subclusters have been chosen, merge them.
7. Label each point according to the cluster label of its belonging subcluster, and get the final clusters $C=\{c_1, c_2, \ldots, c_G\}$.

B. ALGORITHM AND PERFORMANCE ANALYSIS

Table 1 outlines the partition-and-merge process of KMDD, which mainly consists of four parts: K-means process (Step 1), calculation of δ and ρ for each subcluster (Steps 2-4), selection of cores (Step 5) and assignment of data (Step 6-7). Detailed illustrations are shown in the flowchart of the algorithm (Fig. 3), in which $cl(s_k)$ stands for the cluster label of s_k, $Ne(s_k)$ is the neighbor of s_k, and $point(s_k)$ returns all points from subcluster s_k.

Note that there is no neighbor for the subcluster whose density is the highest, so it is always considered as a core in the decision graph, which means that any dataset at least covers one cluster. In addition, if one sets $K = n$, KMDD is equivalent to DD; if one selects all points in the decision graph, KMDD is equivalent to K-means for no subclusters will be merged in the second phase.

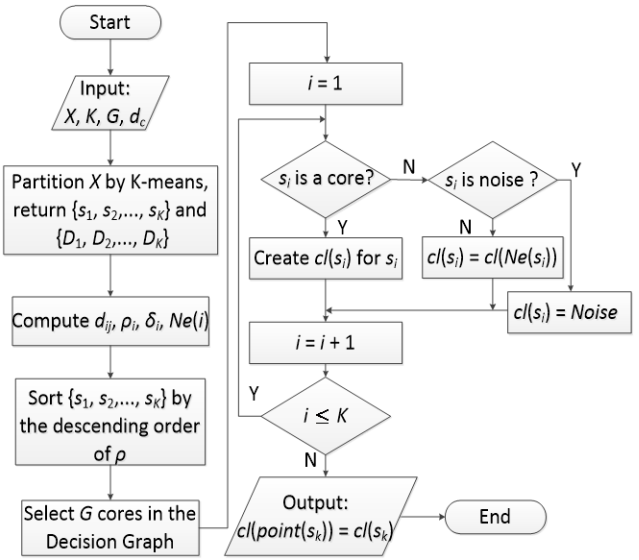

FIGURE 3. The algorithm flowchart.

Lemma 2: The algorithm can solve the KMDD problem in $O(Knt)$ expected time, where K is the number of subclusters, n is the data size and t is the maximum number of iterations allowed by K-means.

Proof: It takes $O(Knt)$ expected time to partition the dataset into K subclusters with K-means in Step 1. The expected time of computing the distance between each pair of subclusters in Step 2, computing ρ in Step 3, and computing δ in Step 4 are all $O(K^2)$. We don't consider the time consumption of Step 5, since it is a manual operation to select cluster centers in the decision graph. The run time for assignment process in Step 6 and Step 7 are $O(K)$ and $O(n)$ respectively. Therefore, the overall time complexity is $O(Knt+K^2+K+n)$, since $K < n$, the time complexity of the algorithm is about $O(Knt)$. □

According to our experiential knowledge, when K in the algorithm is several times as big as G, a good quality of result can be guaranteed. Since G is far smaller than n in most cases, KMDD can achieve a good efficiency. We perform extensive experiments in the next Section, showing that its time efficiency nears K-means, and like DD, it has the capability to find clusters with diverse shapes and densities.

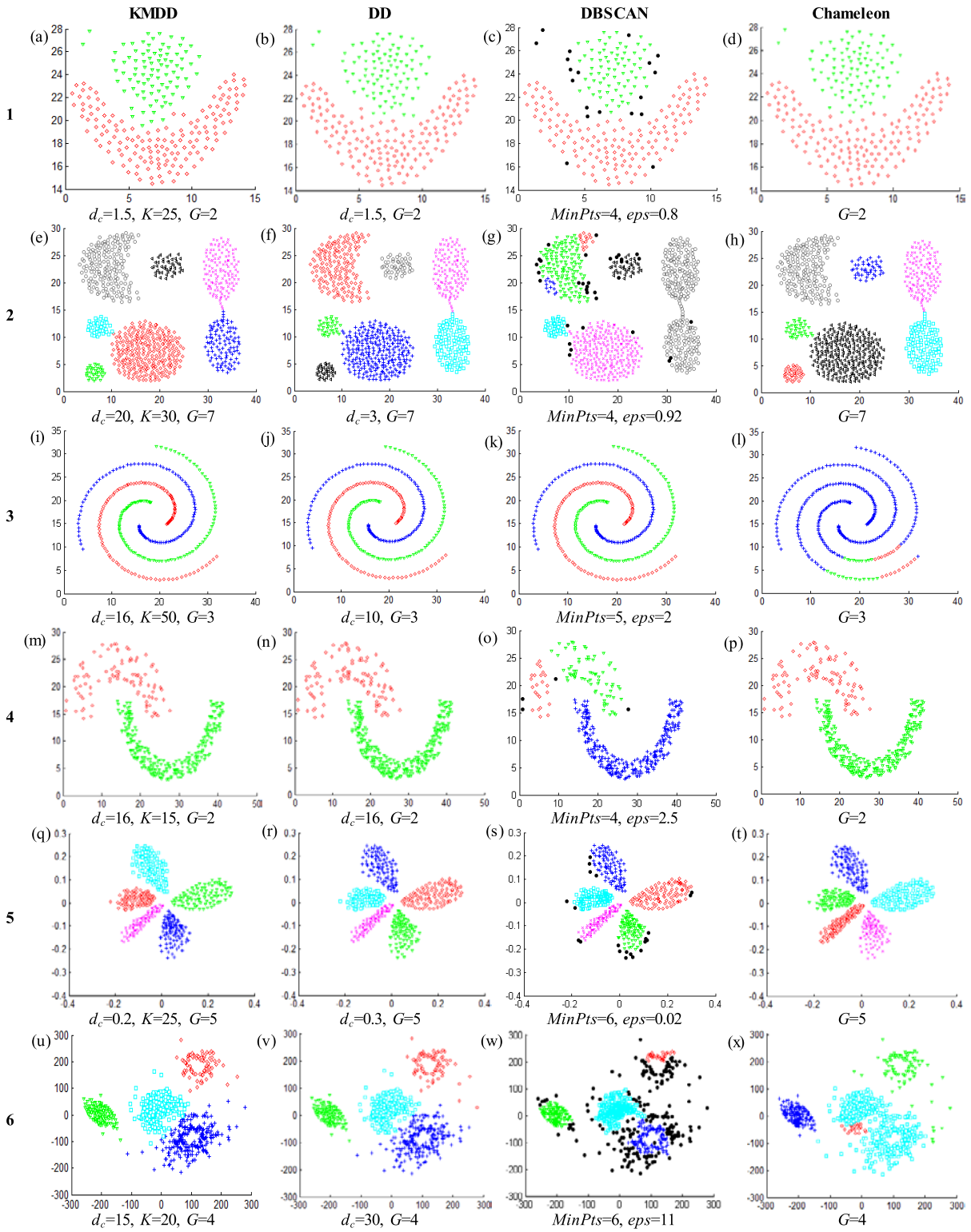

FIGURE 4. Clusters discovered by KMDD, DD, DBSCAN and Chameleon (columns) for DS 1-6 (rows).

IV. RESULTS AND DISCUSSION

This section presents an empirical evaluation of the proposed clustering method. Chameleon was implemented through CLUTO package [32] and the rest algorithms were implemented in the MATLAB R2013b 64-bit program. All the experiments were run on a PC with Intel Core i7–4790 CPU at 3.60 GHz, 32-GB RAM equipped with Windows 7.

A. DATASETS

We evaluate our method with both synthetic and real datasets whose details are explained as follows.

1) SYNTHETIC DATA

(a) DS1 [31], DS2 [33] and DS3 [34] are of diverse shapes. (b) DS4 [35], DS5 [36] and DS6 [37] are of diverse shapes and densities. (c) Seed Spreader (SS) [17], which was generated in a ''random walk with restart'' fashion. Each time a restart happens, the spreader begins to generate a new cluster. The underlying data space had a normalized domain of $[0, 10^5]$ for every dimension. We use it to analyze the impact of parameters of KMDD and test the computational efficiency of different algorithms on both data size and data dimension.

2) REAL SPATIAL DATA

We take a part set of taxicab data in New York City from [38], which has 36382 taxicab coordinates produced from June 2nd, 2014 to June 30th, 2014 within a range of longitude: 40.645° N ∼ 40.795° N and latitude: 73.85° W ∼ 74.02° W.

B. CLUSTERING RESULTS

To compare the effectiveness on clustering results, we benchmark KMDD against DBSCAN, Chameleon and DD on DS 1-6 and the taxicab dataset. Partition-based clustering methods are not discussed here for they lack the capability finding clusters with diverse shapes and densities, while DBSCAN and Chameleon are outstanding density-based clustering method and agglomerative hierarchical clustering method respectively and they are able to solve such problems. To achieve better results in terms of visual inspection, all parameters of the four algorithms were selected elaborately, in which parameters of Chameleon were set to recommended values by CLUTO package (-clmethod = graph, -sim = dist, -agglofrom = 30).

TABLE 2. Results in adjusted rand index of each method for DS 1-5.

Datasets	KMDD	DD	DBSCAN	Chameleon
DS1	0.9339	1	0.8792	0.9501
DS2	0.9949	1	0.7952	0.992
DS3	1	1	1	0.0144
DS4	1	1	0.9411	1
DS5	1	1	0.9456	1

1) DS 1-6

All results for DS 1-6 are demonstrated in Fig. 4. We evaluate the accuracy of clustering methods by visual inspection and the Adjusted Rand Index (ARI), which is designed to compare clustering results against the ground truth [39]. There's no available ARI for DS6 because it contains significant noise points and is without the ground truth. Table 2 demonstrates that DD has the highest ARI scores over all 5 datasets, which means all results got by DD are exactly the same as the ground truths. Followed by our KMDD, it also achieved high ARI scores and outperformed DBSCAN and Chameleon on most datasets.

For DS 1-3, both KMDD and DD can detect varied shapes of clusters and their results are almost the same. DBSCAN performs well on DS1 and DS3, but it detected some noise points on DS1 and failed to find the right clusters on DS2 no matter how to tune parameters. What's more, results obtained by DBSCAN are sensitive to parameters *MinPts* and *eps*, required tuning carefully. Chameleon discovered the right clusters in DS1 and DS2, but it has the worst performance on DS3, one explanation is that it may merge subclusters which are too close to each other.

We now proceed to inspect the performance of the four algorithms on DS 4-6, which are composed of clusters with diverse shapes and densities. KMDD and DD managed to find the right clusters on all the 3 datasets and they obtained the same results. Though DBSCAN discovered the right clusters in DS5, it identified 3 clusters in DS4 and partitioned too many points into noise in DS6. Chameleon outperformed DBSCAN on DS 4-5, but it failed to cluster correctly in the central region in DS6.

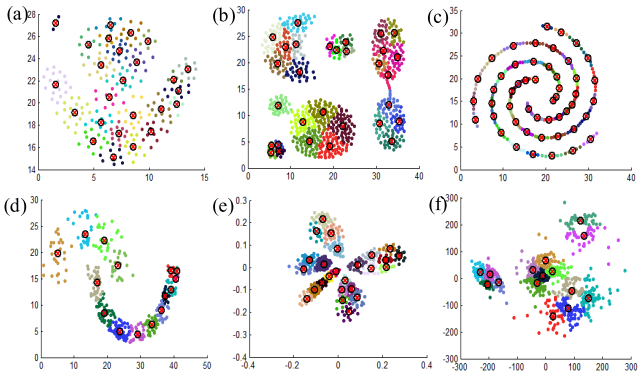

FIGURE 5. Subclusters and their centers discovered in the first phase of KMDD for DS 1-6.

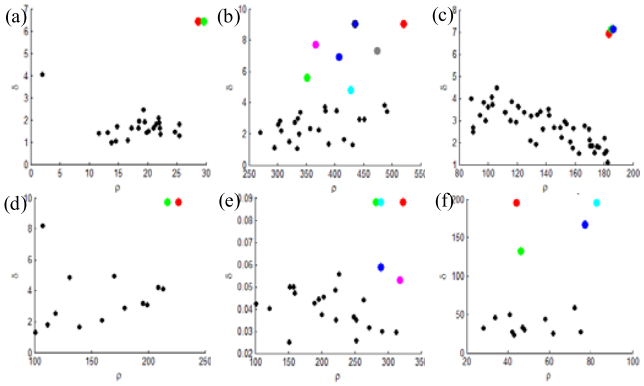

FIGURE 6. 'The decision graphs of KMDD for DS 1-6. Points (subclusters in Fig. 5) lie in the right up of decision graphs were considered as cores.

Fig. 5 shows subclusters and their centers discovered in the first phase of KMDD for each dataset. It can be seen from Fig. 6 that all subclusters selected as cores are those

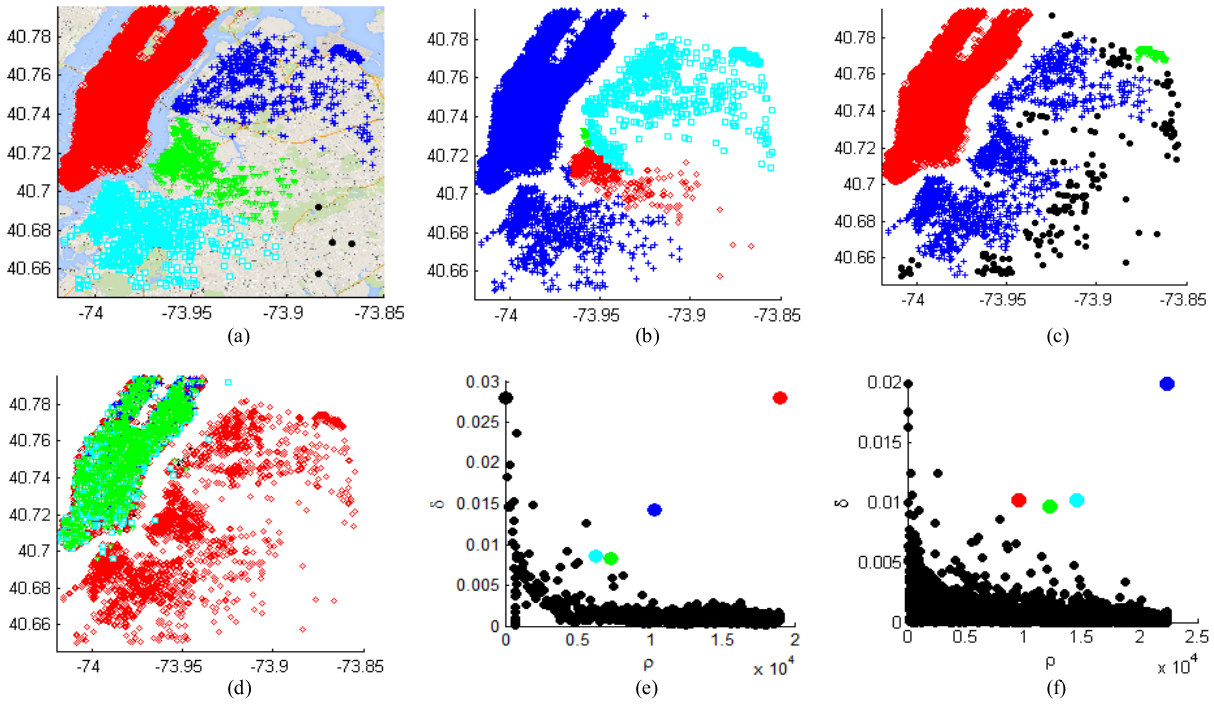

FIGURE 7. Clusters discovered by KMDD, DD, DBSCAN and Chameleon for the taxicab dataset. (a): 4 clusters and 4 noise points found by KMDD, plotting on the background map. (b): 4 clusters found by DD. (c): 3 clusters and some noise found by DBSCAN. (d): 4 Clusters and some noise found by Chameleon, highly overlapping in Manhattan region. (e) and (f): The decision graphs of KMDD and DD respectively.

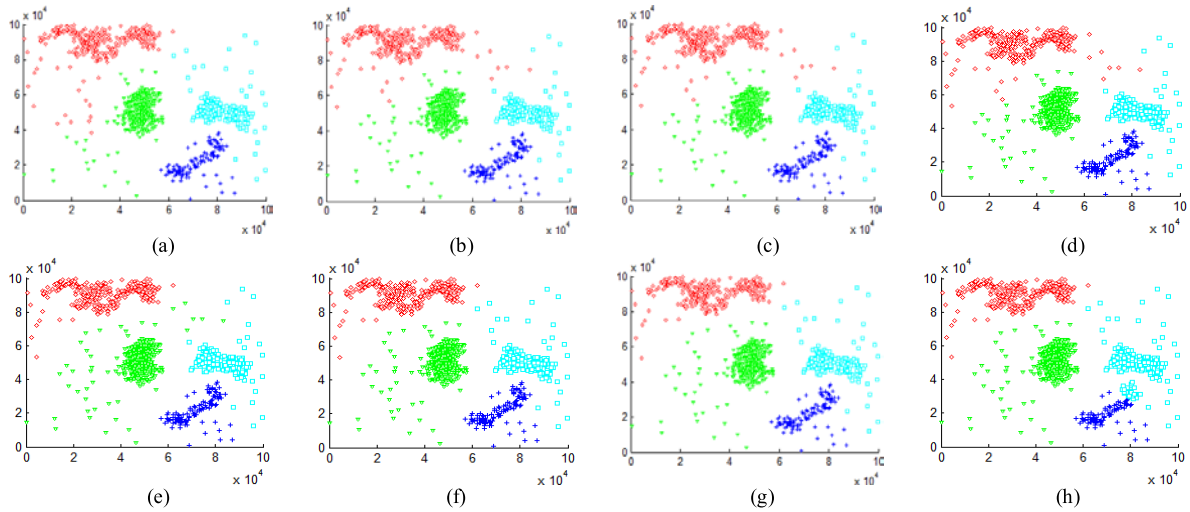

FIGURE 8. Results returned by fixing $K = 150$, increasing d_c from 100 to 14000 by a pace of 2000. (a) $d_c = 100$, $K = 150$. (b) $d_c = 2000$, $K = 150$. (c) $d_c = 4000$, $K = 150$. (d) $d_c = 6000$, $K = 150$. (e) $d_c = 8000$, $K = 150$. (f) $d_c = 10000$, $K = 150$. (g) $d_c = 12000$, $K = 150$. (h) $d_c = 14000$, $K = 150$.

points lie in the right up in the decision graphs, and it is easy to identify these cores for their relatively bigger ρ and δ comparing to other points. In addition, the results on most datasets are robust with respect to the choice of d_c and K. Therefore, if a dataset is not visible, DD and our KMDD can still perform well, but for DBSCAN and Chameleon, it is difficult to choose parameters since it is difficult to evaluate clustering results in this case.

2) REAL SPATIAL DATA

We also applied the 4 approaches to the New York City taxicab dataset, with the aim of identifying the possible functional regions on the map. This dataset poses a serious challenge because there's no ground truth can tell an ideal partition of these points; thus, we check the results by visual inspection. The results are presented in Fig. 7, in which KDMM, DD and Chameleon discovered 4 clusters while

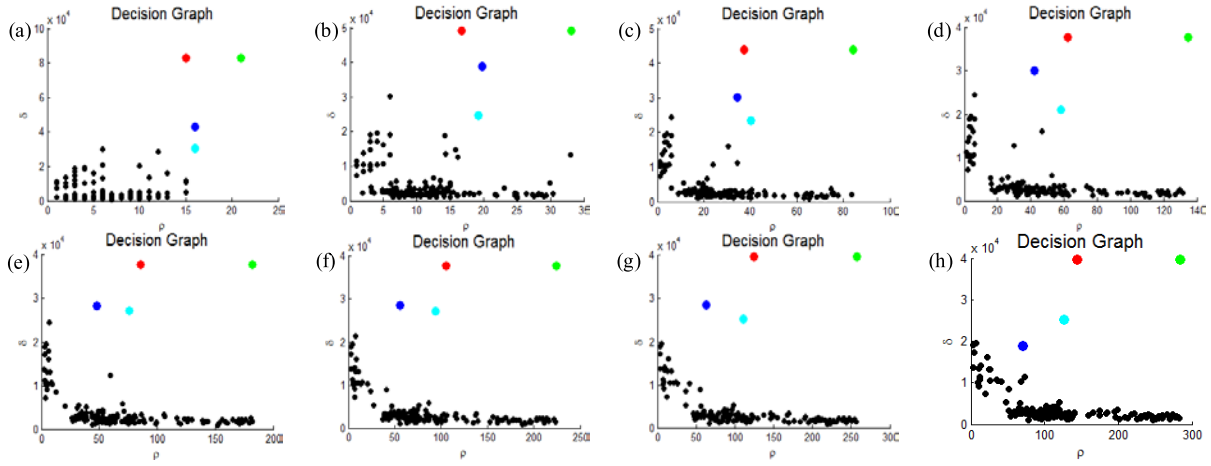

FIGURE 9. The decision graphs for $K = 150$, dc from 100 to 14000 by a pace of 2000.

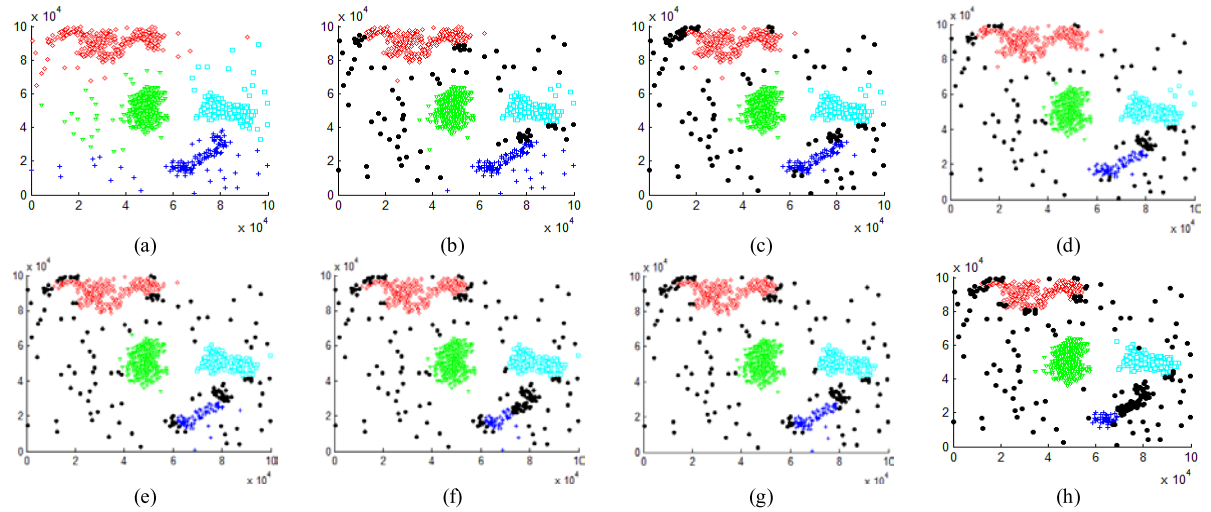

FIGURE 10. Results returned by fixing $d_c = 8000$, increasing K from 10 to 290 by a pace of 40. (a) $d_c = 8000$, K =10 (b) $d_c = 8000$, $K = 50$. (c) $d_c = 8000$, $K = 90$. (d) $d_c = 8000$, $K = 130$. (e) $d_c = 8000$, $K = 170$. (f) $d_c = 8000$, $K = 210$. (g) $d_c = 8000$, $K = 250$. (h) $d_c = 8000$, $K = 290$.

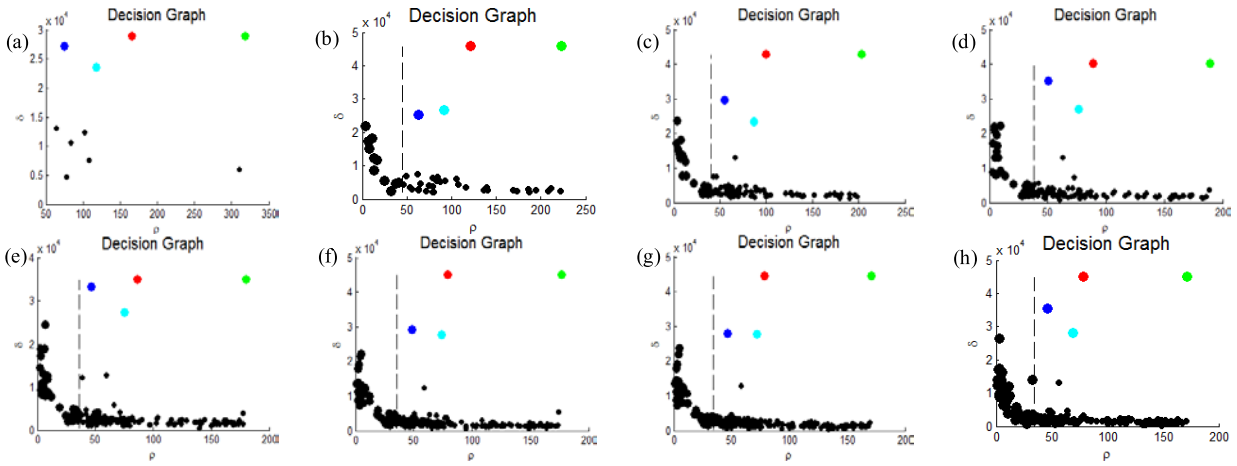

FIGURE 11. The decision graphs for $d_c = 8000$, K from 10 to 290 by a pace of 40. Subclusters in the left side of dashed line were considered as noise.

DBSCAN discovered 3. It seems like that Chameleon got the worst performance comparing to other 3 methods since the 4 clusters and noise are highly overlapping in Manhattan (region in the left up in Fig. 7(d)). DBSCAN clustered these points into two main clusters (red and blue in Fig. 7(c)), a small cluster (green in Fig. 7(c)) and some noise, showing

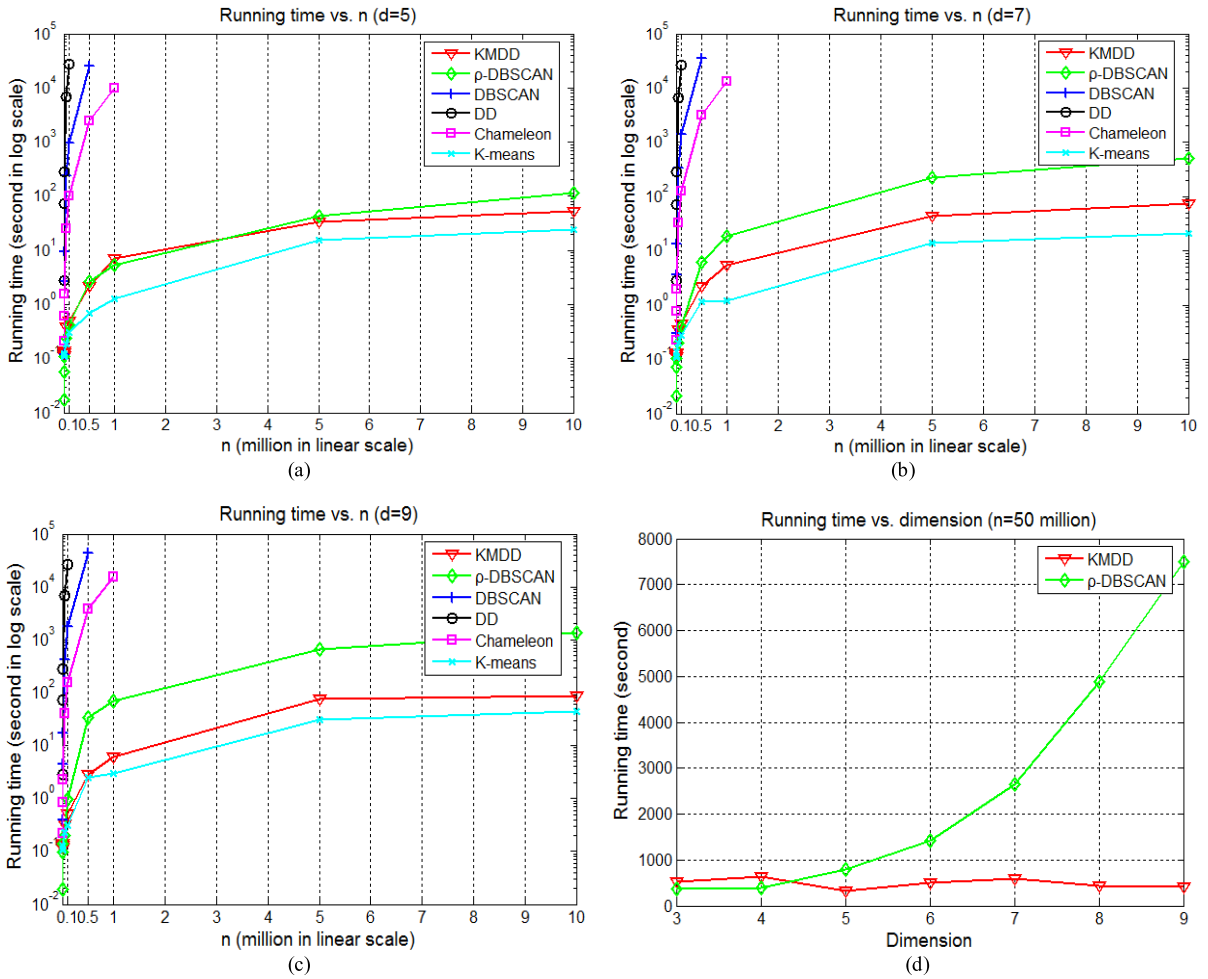

FIGURE 12. Running time comparison among KMDD, ρ-DBSCAN, DBSCAN, DD, Chameleon and K-means. (a) SS, $d = 5$, n from 1k to 10m. (b) SS, $d = 7$, n from 1k to 10m. (c) SS, $d = 9$s, n from 1k to 10m. (d) SS, d from 3 to 9, $n = 50$ million.

that its performance isn't good when data have greatly varied densities. For KMDD and DD, we selected 4 points along the diagonal line as cores in their decision graphs (see Fig. 7(e) and Fig. 7(f)) and present their clustering results in Fig. 7(a) and Fig. 7(b) respectively. Generally, this dataset was well partitioned by KMDD, and from Fig. 7(a) we can clearly see 4 noise points (the subcluster in left up corner in Fig. 7(e)), and clusters which may stand for 4 regions. However, it can be seen from Fig. 7(b) that DD merged points locating in the west regions and separated points locating in the central region.

All methods were fast when the dataset was small, e.g., DS 1-6. However, the running time of the 4 approaches for the real dataset had a big difference: 5.07 seconds for KMDD, 6149.91 seconds for DD, 21.82 seconds for DBSCAN and 13.60 seconds for Chameleon.

C. IMPACT OF CHOICE OF d_c AND K

We utilized the 2D SS dataset (1 *start* and 3 *restarts*, 1000 points) to analyze the robustness of results with respect to the choice of parameters d_c and K for KMDD.

First, we fixed K to 150 and increased d_c from 100 to 14000 by a pace of 2000, and the results are shown in Fig. 8. From Fig. 8(a) to Fig. 8(g), we can see that the 4 clusters returned by KMDD in every panel are almost the same except some difference on noise points. What's more, we can always easily find 4 cores in the decision graphs shown in Fig. 9. In other words, the clustering results are almost invariable for a large range of d_c (from 100 to 12000).

Next, we fixed d_c to 8000 and increased K from 10 to 290 by a pace of 40. Excluded the case of K =10, we regarded noise as a set of subclusters whose densities were lower than 20% of the highest density. The results and decision graphs are presented in Fig. 10 and Fig. 11 respectively. Again, we can easily find 4 cores in the decision graphs, and our method can effectively capture the main parts of 4 clusters for all Ks.

D. COMPUTATIONAL EFFICIENCY

This experiment examines how each method scales with the data size n and dimension d. For this purpose, we used SS datasets of 5D, 7D, and 9D by varying n from 1k to 10m to examine the running time of K-means, KMDD, DBSCAN

(without R-tree), ρ-approximate DBSCAN (ρ-DBSCAN following, the fastest existing variety of DBSCAN), DD and Chameleon. Parameters were set as: K-means ($G = 1 + restarts$), KMDD ($d_c = 5000$, $K = 20$, $G = 1 + restarts$), DBSCAN ($MinPts = 20$, $eps = 5000$), ρ-DBSCAN ($\rho = 0.001$, $MinPts = 20$, $eps = 5000$), DD ($d_c = 5000$, $G = 1 + restarts$) and Chameleon ($G = 1 + restarts$).

Fig. 12(a) to Fig. 12(c) present the running time of each method for 5D, 7D and 9D datasets, note that the axis of running time is in log scale. If an algorithm doesn't have results at a value of n, it means that it didn't terminate within 12 hours. We can see that when $n > 100$k, there's no results for DD, 500k for DBSCAN and 1m for Chameleon. However, KMDD, ρ-DBSCAN and K-means can all terminate in less than 1500 seconds even for $n = 10$m.

It's interesting that the time efficiency of our method isn't influenced by data dimension d too much, while the running time of ρ-DBSCAN increases obviously as d increases. Fig. 12(d) shows that for $n = 50$m, KMDD can always terminate in several hundred seconds no matter how large d is, while the computation time of ρ-DBSCAN is strongly polynomial to the dimension d.

V. CONCLUSIONS

In this paper, we proposed a novel spatial clustering method called KMDD, which imposes a partition-and-merge strategy to fast discover clusters with diverse shapes and densities in spatial databases. Unlike traditional agglomerative hierarchical clustering methods, KMDD is the first one that employs the density and distance-based concept to aggregate subclusters. Extensive experiments on both synthetic and real-world datasets demonstrate that KMDD does effectively identify clusters with diverse shapes and densities, and its scalability on data size and dimension outperforms some of the widely used spatial clustering methods. In addition, we also show that the clustering results are robust with respect to the choice of parameters.

However, there still remain some problems need to be further studied.

First, the selection of cores and noise is somewhat subjective, although KMDD reduces the number of points in the decision graph comparing to DD, cores and noise are still difficult to be determined in some cases. In a nutshell, we need to find criterions for the automatic choice of the cluster cores and noise as well as criterions for parameter selection.

Second, K-means is order-sensitive with respect to input data, so that it may converge to different local minima if we use random seeds (initialization centers), causing variation of location of points in the decision graph. Despite the final results of KMDD are nearly the same if we select proper cores in these different decision graphs, generally speaking, its results are unstable. In this paper, to ensure the reproducibility of experiments, on each dataset we produced the same random seeds for K-means by using default settings of the random number generator in our program. To overcome the problem of algorithm stability, we can run K-means multiple times and choose the best solution in practice or replace it with another efficient and stable partition method in the future.

REFERENCES

[1] K. Koperski and J. Han, "Discovery of spatial association rules in geographic information databases," in *Advances in Spatial Databases*. Berlin, Germany: Springer, 1995, pp. 47–66.
[2] H. Miller and J. Han, "Spatial clustering methods in data mining: A survey," *Geographic Data Mining Knowledge Discovery*. New York, NY, USA: Taylor & Francis, 2001.
[3] L. Wang and X. Li, "Spatial epidemiology of networked metapopulation: An overview," *Chin. Sci. Bull.*, vol. 59, no. 28, pp. 3511–3522, 2014.
[4] M. Sukanya, T. Kalaikumaran, and S. Karthik, "Criminals and crime hotspot detection using data mining algorithms: Clustering and classification," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 1, no. 10, pp. 225–227, 2012.
[5] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On detection of emerging anomalous traffic patterns using GPS data," *Data Knowl. Eng.*, vol. 87, pp. 357–373, Sep. 2013.
[6] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, p. 38, 2014.
[7] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. SIGMOD*, 2007, pp. 593–604.
[8] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. VLDB*, vol. 98. 1998, pp. 428–439.
[9] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.
[10] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," *SIGKDD*, vol. 96, no. 34, pp. 226–231, 1996.
[11] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.
[12] L. Kaufman and P. J. Rousseeuw, *Partitioning Around Medoids (Program PAM) Finding Groups Data: An Introduction to Cluster Analysis*. Hoboken, NJ, USA: Wiley, 1990, pp. 68–125.
[13] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344. Hoboken, NJ, USA: Wiley, 2009.
[14] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.
[15] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
[16] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015.
[17] J. Gan and Y. Tao, "DBSCAN revisited: Mis-claim, un-fixability, and approximation," in *Proc. SIGMOD*, 2015, pp. 519–530.
[18] 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, 1999.
[19] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.
[20] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. SIGKDD*, vol. 98. 1998, pp. 58–65.
[21] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proc. VLDB*, vol. 97. 1997, pp. 186–195.
[22] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *SIGMOD*, vol. 27, no. 2, pp. 94–105, 1998.
[23] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, 1996.
[24] 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.
[25] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.

[26] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.
[27] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Data Mining Knowl., Discovery*, vol. 2, no. 1, pp. 86–97, 2012.
[28] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.
[29] I. Kärkkäinen and P. Fränti, "Dynamic local search algorithm for the clustering problem," Ph.D. dissertation, Dept. Comput. Sci., Univ. Joensuu, Kuopio, Finland, 2002.
[30] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.
[31] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinformat.*, vol. 8, no. 1, p. 3, 2007.
[32] G. Karypis, "CLUTO: A clustering toolkit (release 2.1.1)," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. #02-017, Nov. 2003.
[33] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, p. 4, 2007.
[34] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, 2008.
[35] A. K. Jain and M. H. C. Law, "Data clustering: A user's dilemma," in *Proc. Int. Conf. Pattern Recognit. Mach. Intell.*, 2005, pp. 1–10.
[36] X. Chen, W. Liu, H. Qiu, and J. Lai, "APSCAN: A parameter free algorithm for clustering," *Pattern Recognit. Lett.*, vol. 32, no. 1, pp. 973–986, 2011.
[37] Y. Pei and O. Zaïane, "A synthetic data generator for clustering and outlier analysis," Dept. Comput. Sci., Univ. Alberta, Edmonton, AB, Canada, Tech. Rep. TR06-15, 2006.
[38] Y. Zheng, H. Zhang, and Y. Yu, "Detecting collective anomalies from multiple spatio-temporal datasets across different domains," in *Proc. SIGSPATIAL*, 2015, p. 2.
[39] K. Y. Yeung and W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 9, no. 17, pp. 763–774, May 2001.
[40] J. Silva, E. Faria, R. Barros, E. Hruschka, A. C. de Carvalho, and J. Gama, "Gama J Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, p. 13, 2013.
[41] S. Preheim, A. Perrotta, A. Martin-Platero, A. Gupta, and E. Alm, "Distribution-based clustering: Using ecology to refine the operational taxonomic unit," *Appl. Environ. Microbiol.*, vol. 79, no. 21, pp. 6593–6603, 2013.
[42] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[43] B. Jiang, J. Pei, Y. Tao, and X. Lin, "Clustering uncertain data based on probability distribution similarity," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 751–763, Apr. 2013.

JIANG WANG received the B.S. degree in information system engineering and the M.S. degree in management science and engineering from the National University of Defense Technology, China, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree with the Science and Technology on Information Systems Engineering Laboratory, under the supervision of Prof. W. Zhang. His current research interests include spatio-temporal data mining and trajectory mining.

CHENG ZHU received the Ph.D. degree in management science and engineering from the National University of Defense Technology (NUDT), China, in 2005. He is currently a Professor with the Science and Technology on Information Systems Engineering Laboratory, NUDT. His current research interest is computer network data mining.

YUN ZHOU received the Ph.D. degree from the Risk and Information Management Research Group, Queen Mary University of London, U.K., in 2015. He is currently a Lecturer with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology. His current research interests include Bayesian network and approximate reasoning.

XIANQIANG ZHU received the Ph.D. degree in computer science from Wuhan University, China, in 2011. He is currently a Lecturer with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology. His current research interests include spatial data mining and computer network data mining.

YILIN WANG received the B.S. degree in information management and information systems from the Dalian University of Technology, China. She is currently pursuing the master's degree with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology. Her current research interest is computer network data mining.

WEIMING ZHANG received the Ph.D. degree in management science and engineering from the National University of Defense Technology, China, in 2001. He is currently the Chief of the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology. His current research interest is command and control organizations.