Appl Intell (2010) 33: 39–53
DOI 10.1007/s10489-010-0239-y

PHD: an efficient data clustering scheme using partition space technique for knowledge discovery in large databases

Cheng-Fa Tsai · Heng-Fu Yeh · Jui-Fang Chang · Ning-Han Liu

Published online: 23 June 2010

Abstract Rapid technological advances imply that the amount of data stored in databases is rising very fast. However, data mining can discover helpful implicit information in large databases. How to detect the implicit and useful information with lower time cost, high correctness, high noise filtering rate and fit for large databases is of priority concern in data mining, specifying why considerable clustering schemes have been proposed in recent decades. This investigation presents a new data clustering approach called PHD, which is an enhanced version of KIDBSCAN. PHD is a hybrid density-based algorithm, which partitions the data set by K-means, and then clusters the resulting partitions with IDBSCAN. Finally, the closest pairs of clusters are merged until the natural number of clusters of data set is reached. Experimental results reveal that the proposed algorithm can perform the entire clustering, and efficiently reduce the run-time cost. They also indicate that the proposed new clustering algorithm conducts better than several existing well-known schemes such as the K-means, DBSCAN, IDBSCAN and KIDBSCAN algorithms. Consequently, the proposed PHD algorithm is efficient and effective for data clustering in large databases.

Keywords Data mining · Data clustering · Density-based clustering · Partitioning clustering

C.-F. Tsai (✉) · H.-F. Yeh · N.-H. Liu
Department of Management Information Systems,
National Pingtung University of Science and Technology,
91201 Pingtung, Taiwan
e-mail: cftsai@mail.npust.edu.tw

J.-F. Chang
Department of International Business,
National Kaohsiung University of Applied Sciences,
80778 Kaohsiung, Taiwan

1 Introduction

The rapid progress of information technology has led to increasing amounts of data produced and stored in databases. How to extract the implicit and useful information with lower time cost and high correctness is of priority concern in data mining, explaining why many clustering methods have been developed in recent decades. Data clustering is the most widely employed data mining technique for various business applications. Data clustering categorizes objects into groups of high similar. Conversely, objects in different groups are quite dissimilar. Many clustering algorithms have been presented in recent years [2–7, 13, 14, 16].

Most clustering methods can be divided into partitioning, density-based, hierarchical, grid-based or hybrid algorithms. A partitioning clustering algorithm partitions a data set into k clusters containing similar objects. The best-known partitioning clustering algorithm is K-means [12]. K-means can be easily and quickly implemented, and quickly identifies data clustering, but cannot accurately recognize arbitrary shapes. The clustering concept of a dense region is unique for density-based clustering, which can handle arbitrarily shaped clusters and can filter out noise, and produces stable clustering results. However, it consumes a large run-time cost. Typical density-based clustering algorithms are DBSCAN [6] and IDBSCAN [2]. Hierarchical clustering constructs a hierarchical structure from the data set, and there are two clustering procedures for hierarchical structure, namely bottom-up (agglomerate) and top-down (divisive). Well known agglomerate clustering algorithms, which merge similar objects to a cluster, include BIRCH [18], CURE [9], and ROCK [10]. Divisive clustering algorithms, such as CHAMELEON [11], decompose the cluster of difference into numerous clusters. Nevertheless, hierarchical clustering needs to compare all objects before they are merged or decomposed, which is a time-consuming operation. Grid-based clustering adopts grid-cell structure in data set to cluster. Each grid-cell is treated as a point, thus all clustering procedures are performed on a grid-cell structure. Gird-based clustering has the merit of a fast clustering time, but cannot work efficiently in high-dimensional space. CLIQUE [1] is a classic grid-based clustering algorithm. Mixed clustering algorithms, such as KIDBSCAN [13], GDH [16] and ANGEL [14], can be placed in more than one category.

To reduce the time taken by density-based clustering, this investigation presents a new algorithm **PHD** (a new clustering scheme hybridizing Partitioning, Hierarchical, and Density-based clustering techniques) by hybridizing partitioning, density-based and hierarchical clustering approaches, an advanced version of KIDBSCAN. This new algorithm is enhanced in two important directions, namely partition space of data set for searching and efficient agglomeration. Simulation results demonstrate that the proposed PHD algorithm is a highly effective and efficient clustering method.

The rest of this paper is organized as follows. Section 2 describes several clustering algorithms related to our work. The idea of partition space technique and the proposed algorithm PHD are presented in Sect. 3 and 4 respectively. Next, Sect. 5 shows the simulation results. Conclusion is drawn in Sect. 6.

2 Related works

2.1 K-means

McQueen designed **K-means** in 1967. It is one of most usually utilized clustering algorithms in data mining, due to its efficiency and scalability in clustering large databases. K-means involves the following procedures. (1) Initialize the K cluster centers with random sampling. (2) Assign each object to the nearest cluster center. (3) Compute the new K cluster centers. (4) Repeat steps 2 and 3 until K cluster centers convergence. However, K-means has some weaknesses: it cannot filter out noise; it cannot discover effectively in arbitrary shapes, and its clustering result is instable. Figure 1 shows the data clustering result using K-means, where cluster center A, B, and C are automatically yielded by running K-means. Besides, each partition in Fig. 1 represents a region referring to a cluster.

2.2 Density-based algorithms

The first density-based clustering algorithm was **DBSCAN**, which takes two input parameters *Eps* and *Minpts*. The main concept of DBSCAN is that a neighborhood around a point of a given radius (*Eps*) must include at least minimum number of points (*Minpts*). DBSCAN begins cluster action from an arbitrary point P. The neighborhood of P is obtained by executing a region query. If the neighborhood of P includes greater than or equal to *Minpts* points, then a new cluster with P as the *core point* is created, and all data points in neighborhood of P as *seed* are assigned to this cluster, as displayed in Fig. 2a. Otherwise, the point P is labeled as *noise*, as described in Fig. 2b. The neighborhood of each of P's neighbors is then examined to see whether it can be assigned into the cluster. The process is repeated for every point in this neighborhood. If a cluster cannot be expanded further, then DBSCAN chooses another arbitrary unclassified point, and repeats the process. This procedure is iterated until all points in the data set have been categorized or labeled as *noise*. Furthermore, if a point is labeled as *seed* and *noise*, then the point is named a *border point*, as presented in Fig. 2c. Point Q is considered as a *border point* because it is in the neighborhood of P that satisfies the specified *Minpts* threshold and its neighborhood contains less than *Minpts* points. DBSCAN can locate arbitrary shapes and separate noises. Nevertheless, the complexity of DBSCAN is very high for large databases, since each point must check all data points to discover its neighborhood.

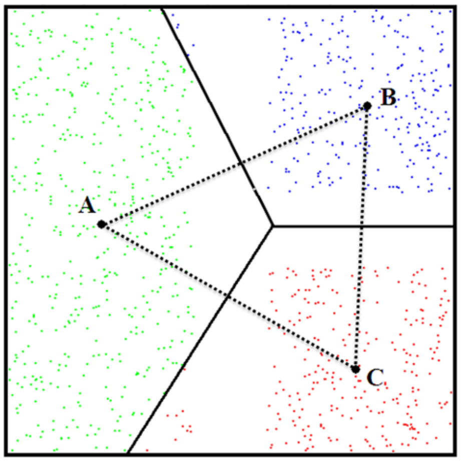


Fig. 1 The concept of K-means for data clustering

IDBSCAN is an advanced version of DBSCAN, which adopts a sampling-based approach to lower the number of expansion queries owing to the large number of expansion queries of DBSCAN. For neighborhood of each point exceeds *Minpts* points, DBSCAN considers all points in neighborhood as seed to expand cluster, thus generating redundant actions of expansion that query the same neighborhood many times. The technique sets eight distinct points in the neighborhood region (a circle with radius *Eps*) as Marked Boundary Objects (MBOs), as depicted in Fig. 3, and then chooses representative points nearest MBOs as a *seed*. If the representative point is the nearest of more than one MBO, then it is selected as a *seed* only once. Hence, the number

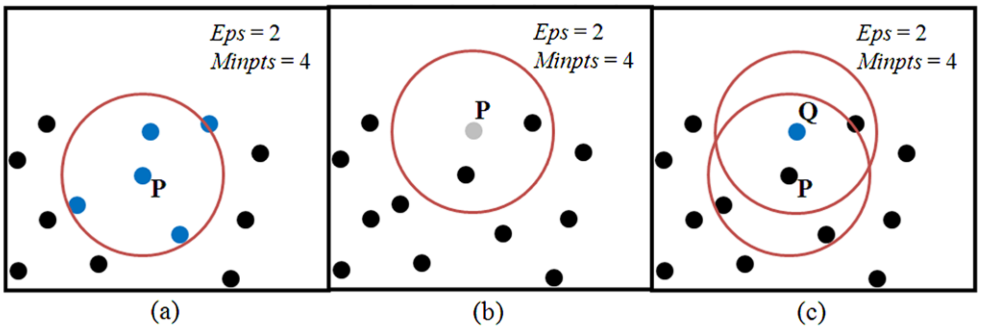


Fig. 2 The concept of core point, border point, and noise in DBSCAN algorithm

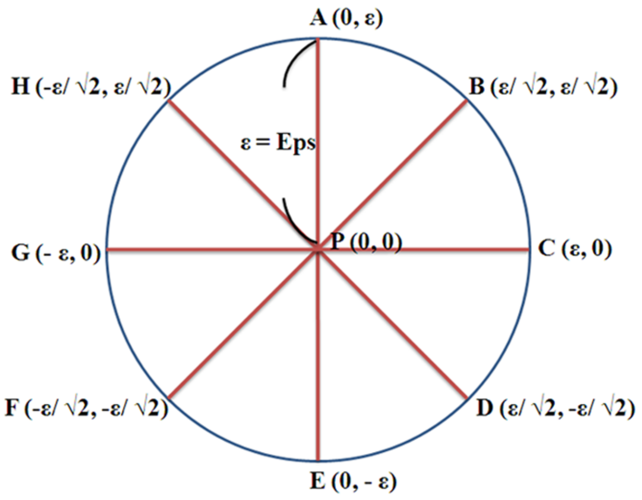


Fig. 3 Circle with eight distinct points in IDBSCAN

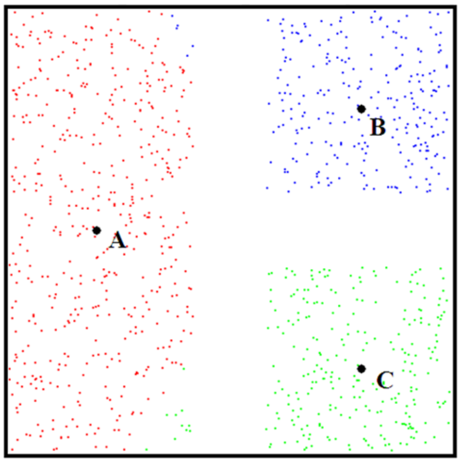


Fig. 4 The high-density center points in the data set of KIDBSCAN

of seeds chosen is at most eight for the neighborhood region of each *core point*. This scheme decreases the number of expansion queries and execution time dramatically. IDBSCAN still generates the same accurate clustering result of DBSCAN.

IDBSCAN carries out data clustering based on the sequence of data set, making it inefficient due to the time taken to expand a cluster from low-density points or border points. Accordingly, **KIDBSCAN** is presented to solve this problem. KIDBSCAN is a hybrid algorithm that combines K-means and IDBSCAN. K-means is applied to yield high-density center points, which are then moved to the front of the data set. Clusters are expanded from high-density center points by performing IDBSCAN. Figure 4 demonstrates the high-density center points in the data set which are nearest to their cluster center. The expansion of clusters from high-density center points then decreases the number of redundant steps and raises efficiency. KIDBSCAN provides high-quality data clustering, and significantly reduces the I/O cost.

3 The idea of partition space technique

DBSCAN, IDBSCAN, and KIDBSCAN look for clusters by exploring the neighborhood of each unclassified point, and expand a cluster from a core point P by conducting neighborhood queries for each *seed* point. However, querying all points in a data set to find neighborhood of each unclassified and seed point is very expensive. Figure 5a depicts clustering starting from P, where the whole region must be searched to find the neighborhood of P. If a data set contains n points, then each point must query n points. The number of queries is n^2. Consequently, the query time is long.

Decreasing the query time and number of queries is a critical issue. This investigation presents a partition space technique that employs data set partitioning to lower the search zone of the clustering technique. The data set is partitioned fast into K clusters by K-means. Figure 5b illustrates a data set partitioned into five clusters as A, B, C, D and E. Points in region B, where point P is located, is scanned to discover the neighborhood of P. Therefore, the number of queries becomes nk, where k denotes the number of data points of clustering.

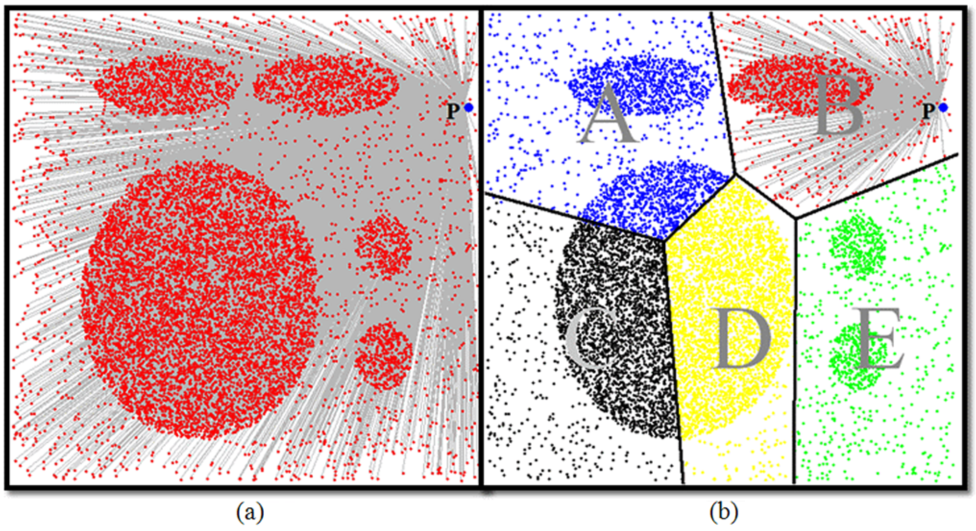


Fig. 5 The concept of neighborhood search (a) A search region covers all points. (b) Region B, in which point *P* is located

Fig. 6 The proposed PHD algorithm

```
PHD(Dataset, K, Eps, Minpts)
  Initialization(); //Initialize all arguments.
  K_means(Dataset, K);
  FOR each point in the data set as p Do
    Assign p into PartitionData according to K_ClusterID;
  END FOR
  ClusterID = 1;
  FOR each point in the data set as p Do
    curPoint = p;
    IF curPoint is unclassified THEN
      IF ExpandCluster(PartitionData, Eps, Minpts, ClusterID) = true THEN
        ClusterID++;
      END IF
    END IF
  END FOR
  ClusterNumber = ClusterID – 1;
  MergeID = 1;
  WHILE ClusterNumber > K DO
    MergeCluster(borderPointData, MergeID);
    ClusterNumber--;
  END WHILE
  Output all points of the data set along with MergeID or "noise" mark;
END PHD
```

4 The proposed PHD clustering algorithm

This section describes the proposed PHD clustering algorithm in detail, and presents an illustrative example to explain the clustering process. This algorithm can be described in terms of the following three stages.

(1) *Partitioning the data set:* This step decreases the number of queries. DBSCAN, IDBSCAN and KIDBSCAN search for *corepoint* to extract dense regions and form a new cluster. Checking each point for core conditions means searching the entire data set to obtain the neighborhoods within the specified *Eps* threshold, and is thus a computationally expensive process. However, partitioning the data set can enhance clustering efficiency, where the neighborhood search is applied on the whole partition only. It reduces the number of data set queries by restricting the search space for each unclassified or *seed* point to the partitions, rather than to the whole data set.

K-means is one of the most efficient partitioning techniques for very large data sets, according to the related work introduced in the previous sections. The proposed algorithm utilizes K-means as a preprocessing stage, and IDBSCAN adopts the resulting partitions to detect their dense regions separately. The data set partitions yielded by K-means depend on one parameter, which is the required number of clusters of data set.

(2) *Clustering each partition using IDBSCAN:* IDBSCAN identifies clusters of data set for each partition separately to improve the performance and lower the number of scans. Since K-means is a centroid-based scheme, its output partitions may contain points from other clusters that occur as a result of assigning each point to its nearest centroid. This case generally occurs in arbitrary shapes, and

Fig. 7 K_means function

```
K_means(Dataset, K)
   FOR i = 1 to K DO
      Center_i = randomGenerate();    //Randomly select K points from the data set as centers.
   END FOR
   WHILE Center <> previousCenter DO
//The current cluster centers are not same to the previous centers.
      FOR each point in the data set as p DO
         FOR i = 1 to K DO
            Compute the distance between p and Center_i;
         END FOR
         Assign p into the nearest center with K_ClusterID according to the distance;
         //K_ClusterID = K, K = 1, 2, 3, .., K
      END FOR
      Recalculate each cluster center;
   END WHILE
   Output all points of the data set along with K_ClusterID;
END K_means
```

Fig. 8 ExpandCluster function

```
ExpandCluster(PartitionData, Eps, Minpts, ClusterID)
   neighbors = FindNeighborsOfcurPoint(PartitionData, curPoint, Eps);
   //The searching of curPoint's neighbors only runs on the partition, rather than on the whole data set.
   IF neighbors.Size < Minpts THEN
      Mark curPoint as noise;
      RETURN false;
   ELSE
      Mark all point in neighbors with ClusterID;
      DefineMBOofcurPoint(curPoint);
      FindSeedsNearestMBO(neighbors);
      Storage all seed points into SeedList;
      WHILE SeedsList <> Empty DO
         curPoint = SeedsList.first;
         neighbors = FindNeighborsOfcurPoint(PartitionData, curPoint, Eps);
    //The searching of curPoint's neighbors only runs on the partition, rather than on the whole data set.
         IF neighbors.Size >= Minpts THEN
            DefineMBOofcurPoint(curPoint);
            FindSeedsNearestMBO(neighbors);
            Append all seed points into SeedList;
            Mark all points in neighbors with ClusterID;
         ELSE
            Mark curPoint as border point and assign it into borderPointData;
         END IF
         SeedList.delete(first);
      END WHILE
      RETURN true;
   END IF
END ExpandCluster
```

even in spherical shapes with different sizes. Under the specified IDBSCAN parameters Eps and $MinPts$, these points are not reachable from others within the same partition. Depending on the accepted solution from the partitioning stage, IDBSCAN can detect more than one dense region in each partition, where the number of dense regions is larger than the final required number of clusters that can be detected by applying IDBSCAN directly onto the data set. This introduces the need for a new stage to merge the nearest dense regions detected by IDBSCAN from the different partitions to reach the final required number of clusters. Furthermore, IDBSCAN filters out the noise from each dense region.

(3) *Merging closest clusters:* A merging stage is stipulated to merge the nearest dense regions detected by applying IDBSCAN separately on each partition in order to reach the final number of clusters. Single linkage is employed as the basis for combining dense regions from various clusters. The single linkage means the shortest distance between the closest pair of data points in two different dense regions. This approach is not efficient if all objects in a dense region are applied to measure a single linkage between pairs of dense regions. Hence, border points are used instead of using all dense region objects for this purpose. Border points are extracted automatically when running IDBSCAN, and are defined in the section of related work. As the dense regions are merged, the final required number of clusters is reached exactly as the output clusters detected by applying the IDBSCAN directly on the whole data set. The *ClusterID* of two dense regions i and j are set to the same value if they conform to the following equation:

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \|P - P'\| \tag{1}$$

```
MergeCluster(borderPointData)
  FOR each point in $C_i$ as p DO    //$C_i$ in borderPointData
    FOR each point in $C_j$ as p' DO    //$C_j$ in borderPointData, $C_i$ <> $C_j$
      Find the shortest distance (p, p') between $C_i$ and $C_j$;
    END FOR
  END FOR
  IF $C_i$.status = NotMerged and $C_j$.status = NotMerged THEN
    $C_i$.MergeID = MergeID;
    $C_j$.MergeID = MergeID;
    $C_i$.status = Merged;
    $C_j$.status = Merged;
    MergeID++;
  ELSE IF $C_i$.status = Merged and $C_j$.status = NotMerged THEN
    $C_j$.MergeID = $C_i$.MergeID;
    $C_j$.status = Merged;
  ELSE IF $C_i$.status = NotMerged and $C_j$.status = Merged THEN
    $C_i$.MergeID = $C_j$.MergeID;
    $C_i$.status = Merged;
  ELSE
    Evaluate the size of $C_i$.MergeID and $C_j$.MergeID;
    The larger MergeID is set to the same value with the smaller MergeID;
  END IF
END MergeCluster
```

Fig. 9 MergeCluster function

Where C_i and C_j represent two different clusters referring to two dense regions. P and $P^{'}$ denote the data points in C_i and C_j respectively.

The PHD algorithm is illustrated in Fig. 6. *Dataset* indicates an input data set. K represents the required number of partitions of data set, *Eps* denotes the radius of neighborhood of each point and *Minpts* is the minimum number of points in the radius of neighborhood of each point. *PartitionData* stands for a list to store the partitioned data, which is outputted by K-means function. *K_ClusterID*, *ClusterID* and *MergeID* denote the clustering location marked by K-means, IDBSCAN and the merging stage respectively. *ClusterNumber* represents the current number of clusters on which the merging stage relies. *BorderPointData* stands for a list to store the points which are labeled as "border" by IDBSCAN. There are three main functions in the PHD algorithm, namely, *K_means()*, *ExpandCluster()* and *MergeCluster()*.

K_means function segments the data set into K clusters, as described in Fig. 7. Notably, *Center* means the current center of cluster, while *previousCenter* represents the past center of cluster. When K centers equals K past centers, end this function.

The purpose of ExpandCluster is improvement of the performance for K-means and separation of the noise for each partition. Figure 8 displays ExpandCluster function. Notably, *neighbors* stands for a list to store the neighbors of *curPoint*. *DefineMBOofcruPoint()* function defines eight MBO (Marked Boundary Objects) location coordinates of the current point. *FindSeedsNearestMBO()* function discovers at most eight points as seed points which are nearest to some MBOs. *SeedList* is a list to place the seed points. The procedure repeats until all points in the data set are categorized into the cluster or labeled as noise.

The third function in PHD algorithm is MergeCluster, as depicted in Fig. 9. Notably, this stage may merge the nearest pair of clusters for outputting the accurate clustering result. This process continues until the current number of clusters equals K.

An illustrative example for the proposed PHD clustering algorithm is then presented to explain the clustering process below.

Figure 10a reveals the original data set 3 [14], which contain 115,000 points in two dimensional space representing 5 clusters with spherical shapes. The first stage of the proposed PHD algorithm is to partition the data set using K-means. In this example, the required number of partitions is set to 5. Because K-means is a centroid-based approach, it fails to detect the final solution, but it can split some clusters into different partitions, where the points are assigned to their closest centroid as displayed in Fig. 10b. Figure 10c presents the seven dense regions, which exceeds the required number of 5 clusters, obtained by applying IDBSCAN separately on each partition. Border objects are detected automatically during the running of IDBSCAN as shown in Fig. 10d. Finally, as the dense regions are merged, the final required number of clusters is reached exactly as the output clusters detected by applying the IDBSCAN directly on the whole data set. Figure 10e depicts the clustering result by the proposed PHD algorithm.

5 Experiment and analysis

Experiments were conducted on a desktop computer with 2 GB RAM and an Intel 3.4 GHz Dual-CPU and running Microsoft Windows XP Professional. All algorithms were implemented in a Java-based program.

5.1 The configuration of K

How to determine the value of K is an important issue for K-means. Many investigations adopt Ward's minimum variance method [17] to solve this problem. Ward's method can provide an appropriate range of value of K as the reference source for an investigator. The following equation is the error sum of squares (*ESS*) given by Ward's method.

$$ESS = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2 \tag{2}$$

Where x_i is the score of ith individual and n means the number of individuals in a cluster. This work uses (2) on the data set 3 to observe the variation of the total error sum of squares after every merging stage. Figure 11 illustrates the variation of the total error sum of squares on the data set 3. As for how to determine the pair of clusters which must be

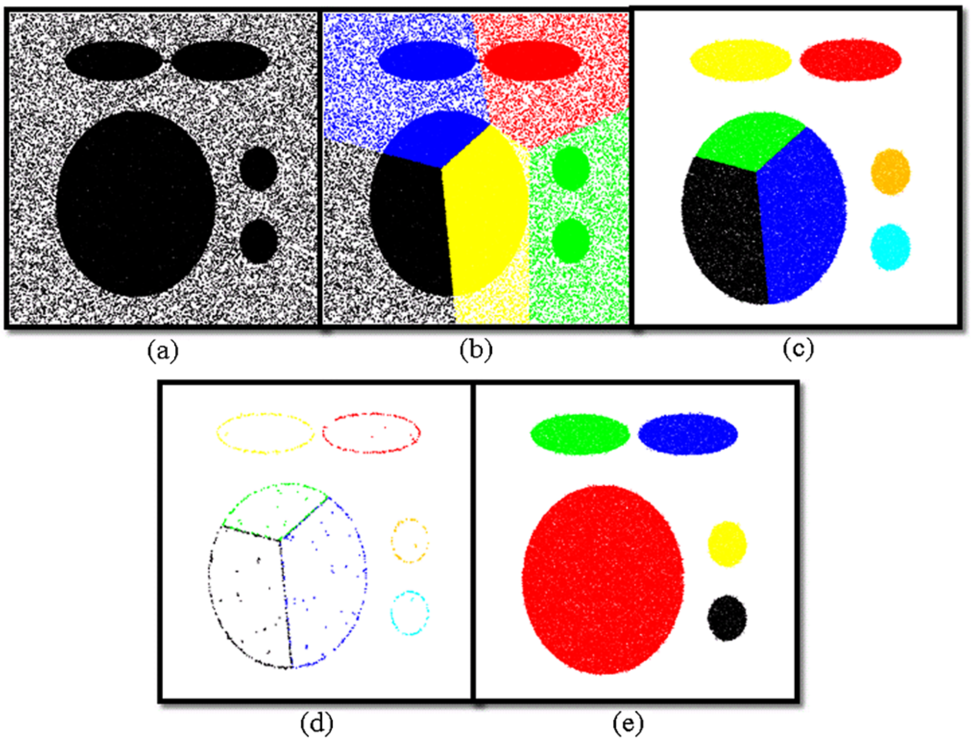


Fig. 10 (**a**) Original data set. (**b**) Data clustering result by K-means. (**c**) Data clustering result by IDBSCAN. (**d**) Border objects detected automatically during the running of IDBSCAN. (**e**) Data clustering result by PHD

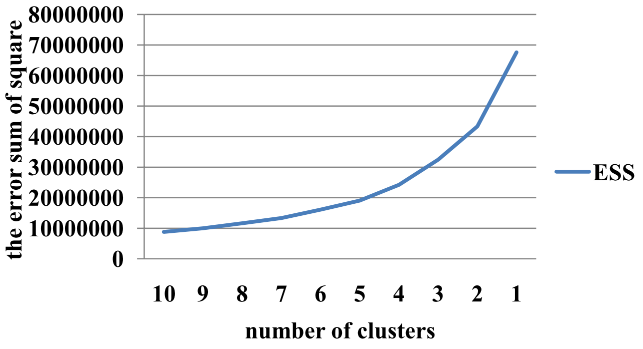


Fig. 11 The variation of the total error sum of squares on the data set 3; x-axis indicates the number of clusters; y-axis represents the error sum of squares

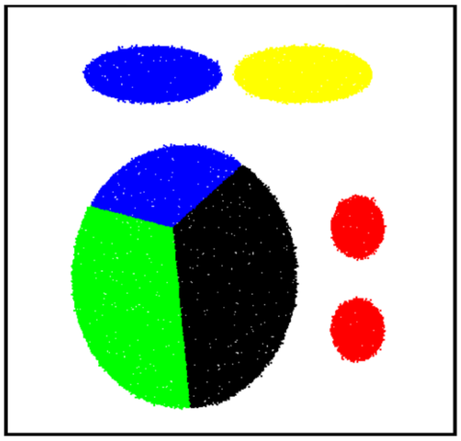


Fig. 12 The clustering result with data set 3 using Ward's method in the merging stage

merged, the increment of its *ESS* must be the smallest in all combinations of merge. In Fig. 11, it can be seen that the appropriate value of K lies between 2 and 6. The PHD algorithm examines every value of K to find the most appropriate value of K, 5. The value of K for other data sets also follows this model to find the most suitable value of K.

5.2 Using Ward's method in the merging stage

This study attempts to use Ward's method in the merging stage. The experimental result on the data set 3 is presented in Fig. 12. It can be seen that the clustering result is incorrect. Before the merging procedure, the data set has been divided into several clusters by K-means. These clusters have different shapes, scales and sizes. The evaluation of *ESS* is affected by the shape, scale and size of cluster. It causes that the first merge involving two small round clusters with red color and then in the second merge involving two regions with blue color. However, the inaccurate result also occurs on other data sets. Thus, the use of Ward's method in the merging process of PHD algorithm is inappropriate.

5.3 Experiment on synthetic data

This simulation employed six synthetic data sets. The data set 1, containing 100,000 points with no noise in 2-D, was

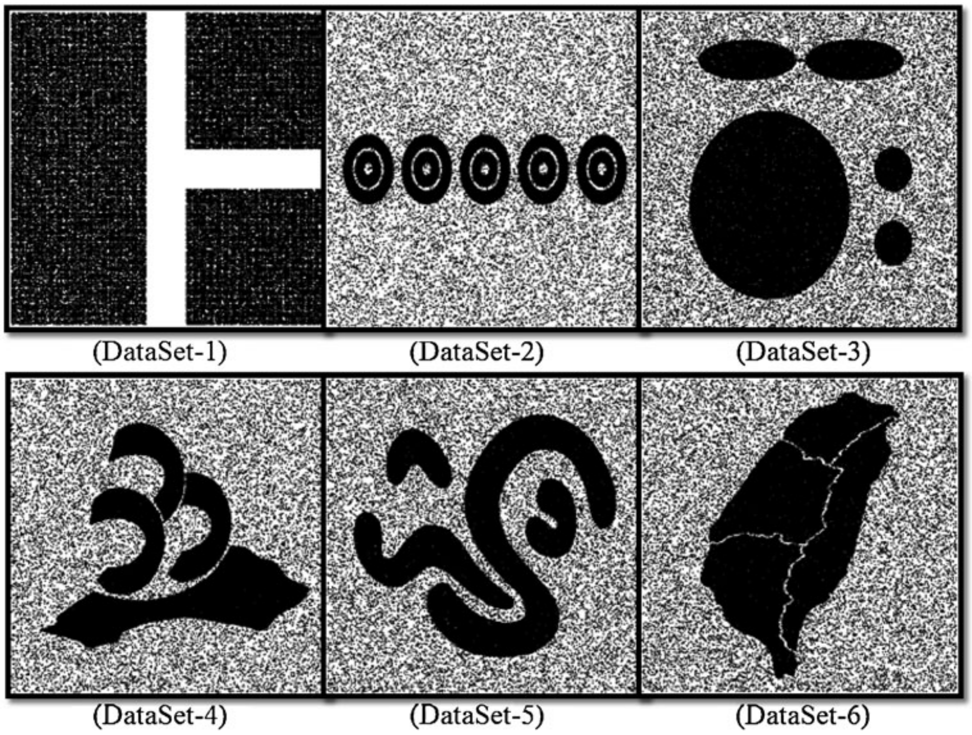


Fig. 13 The original data sets

Table 1 Comparisons of PHD, K-means, DBSCAN, IDBSCAN and KIDBSCAN using 115,000 data sets with 15% noise with Data sets 2–6; the Data set 1 has 100,000 points with noise; item 1 represents run-time cost (in seconds); item 2 denotes CCR (%), while item 3 indicates NFR (%) (N/A means that the simulations were not performed)

Algorithm	Item	DataSet-1	DataSet-2	DataSet-3	DataSet-4	DataSet-5	DataSet-6
	1	0.255	2.563	0.696	0.9	0.918	1.362
K-means	2	93.387%	49.175%	76.766%	38.765%	57.825%	57.243%
	3	N/A	0%	0%	0%	0%	0%
	1	123.631	152.9	150.085	151.294	150.503	150.556
DBSCAN	2	100.000%	100.000%	99.965%	99.995%	99.999%	99.976%
	3	N/A	95.020%	96.960%	95.627%	95.407%	96.147%
	1	21.136	30.016	39.966	35.905	37.636	37.897
IDBSCAN	2	99.997%	100.000%	99.927%	99.984%	99.938%	99.938%
	3	N/A	95.413%	97.787%	96.133%	96.400%	96.647%
	1	20.903	29.717	39.558	35.919	37.347	37.627
KIDBSCAN	2	99.982%	99.989%	99.909%	99.982%	99.949%	99.924%
	3	N/A	95.340%	97.725%	96.132%	96.403%	96.633%
	1	9.432	8.72	12.516	12.499	13.199	14.018
PHD	2	99.956%	99.960%	99.897%	99.966%	99.720%	99.908%
	3	N/A	95.373%	97.707%	96.215%	96.491%	96.695%

obtained from Tsai [13]. The patterns of data sets 2, 3, 4, 5 and 6 were sampled from Tsai [14]. These data sets have three data sizes with 115,000, 230,000 and 575,000 points in 2-D, and with 15% noise. Figure 13 illustrates the original data sets. For data Set 1, the parameters K, *Eps* and *Minpts* were set to 3, 5 and 35. For data sets 2-6, the *Eps* all were set to 6; K were set to 10, 5, 4, 4, and 4, respectively, and *Minpts* were set to 95, 40, 50, 50, and 45, respectively. Furthermore, the configuration of K was based on how many clusters are in the data set and the configu-

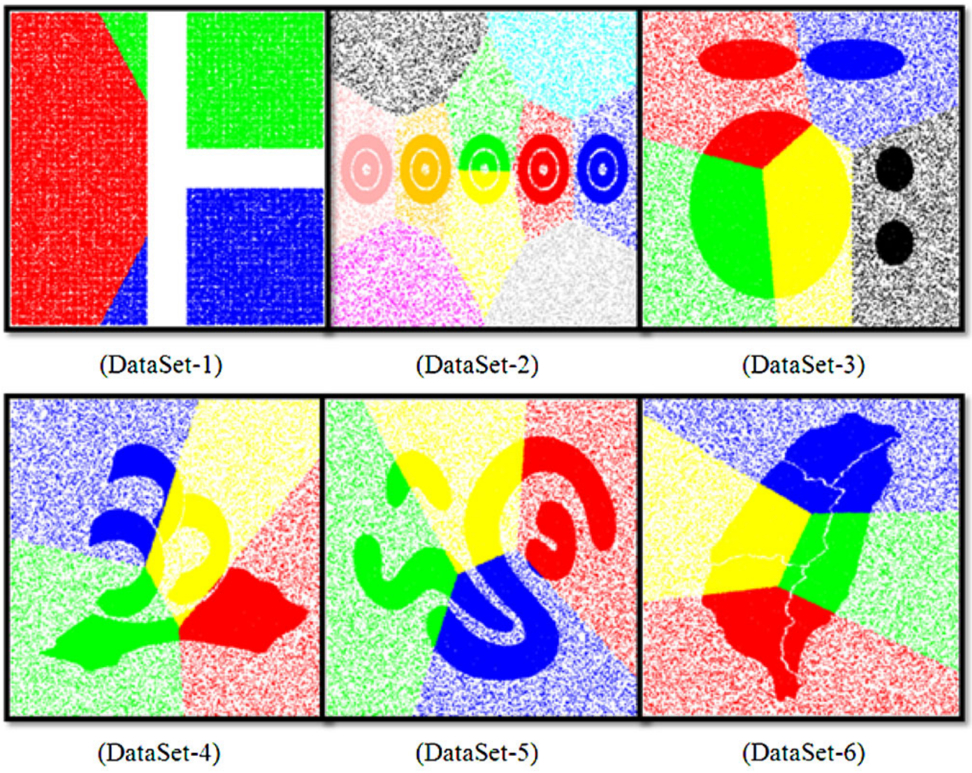


Fig. 14 The clustering results using several different data sets by K-means

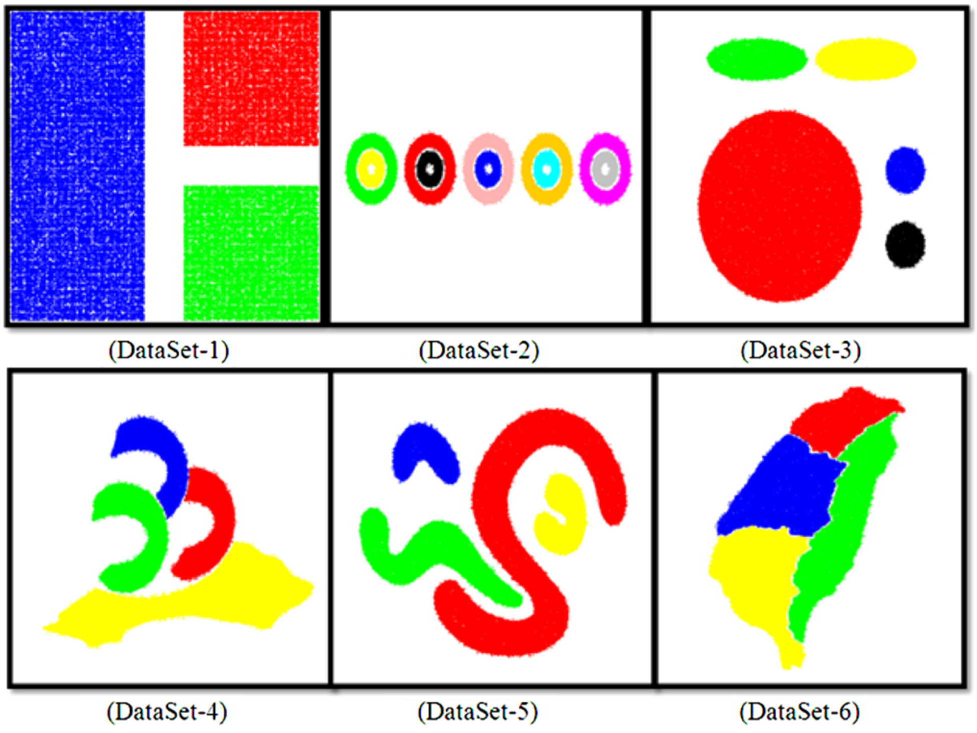


Fig. 15 The clustering results using several different data sets by DBSCAN

rations of *Eps* and *Minpts* were determined by a series of experiments to discover appropriate values for each data set. The results of the proposed PHD algorithm were compared with those of K-means, DBSCAN, IDBSCAN and KIDBSCAN. For clustering performance comparisons, the clustering correctness rate (CCR) and noise filtering rate (NFR) are shown, where CCR represents the percentage of cluster points correctly recognized by algorithm, and NFR denotes the percentage of noise points correctly filtered by algorithm.

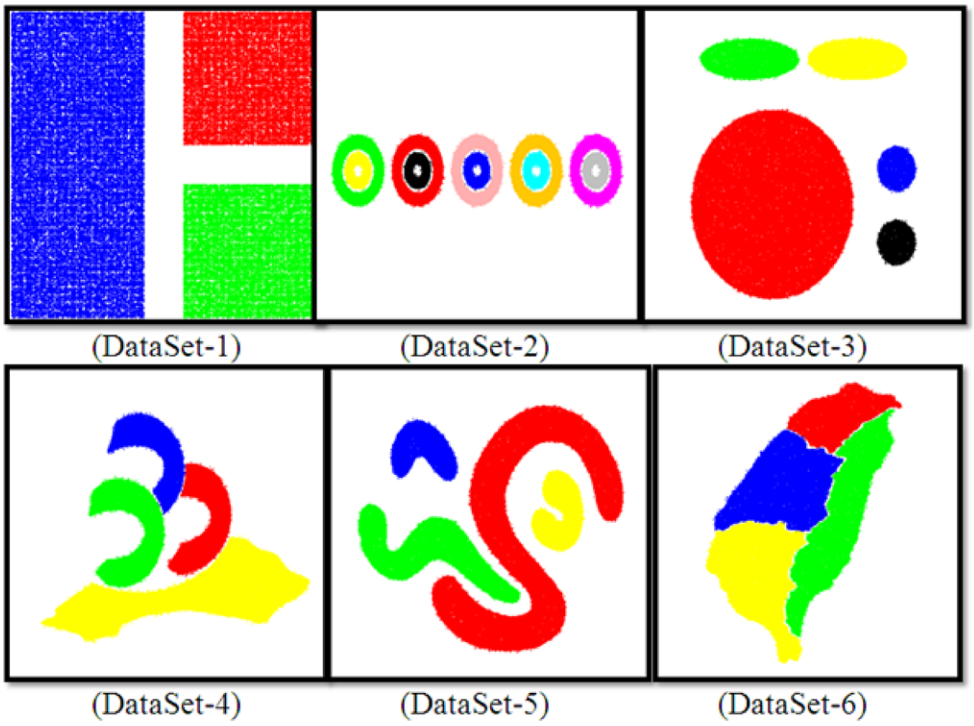


Fig. 16 The clustering results using several different data sets by IDBSCAN

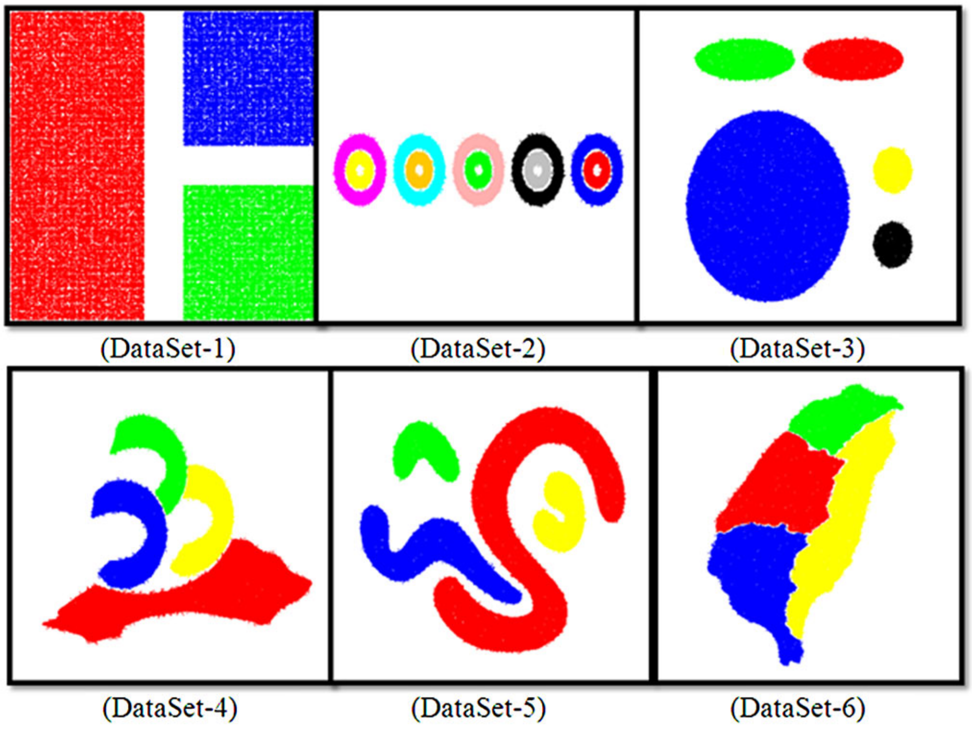


Fig. 17 The clustering results using several different data sets by KIDBSCAN

Table 1 presents the clustering experimental results from PHD, K-means, DBSCAN, IDBSCAN and KIDBSCAN by using 100,000 (Data set 1) and 115,000 (Data sets 2–6) points data sets, and indicates that the proposed PHD algorithm had a lower run-time cost than DBSCAN, IDBSCAN and KIDBSCAN. Additionally, PHD also yielded almost the same high quality of data clustering as DBSCAN, IDBSCAN and KIDBSCAN. Figures 14–18 depict the data clustering results by K-means, DBSCAN, IDBSCAN, KIDBSCAN and PHD, respectively. Experimental results demonstrate that DBSCAN, IDBSCAN, KIDBSCAN, and PHD can accurately recognize clusters for arbitrary shapes. Al-

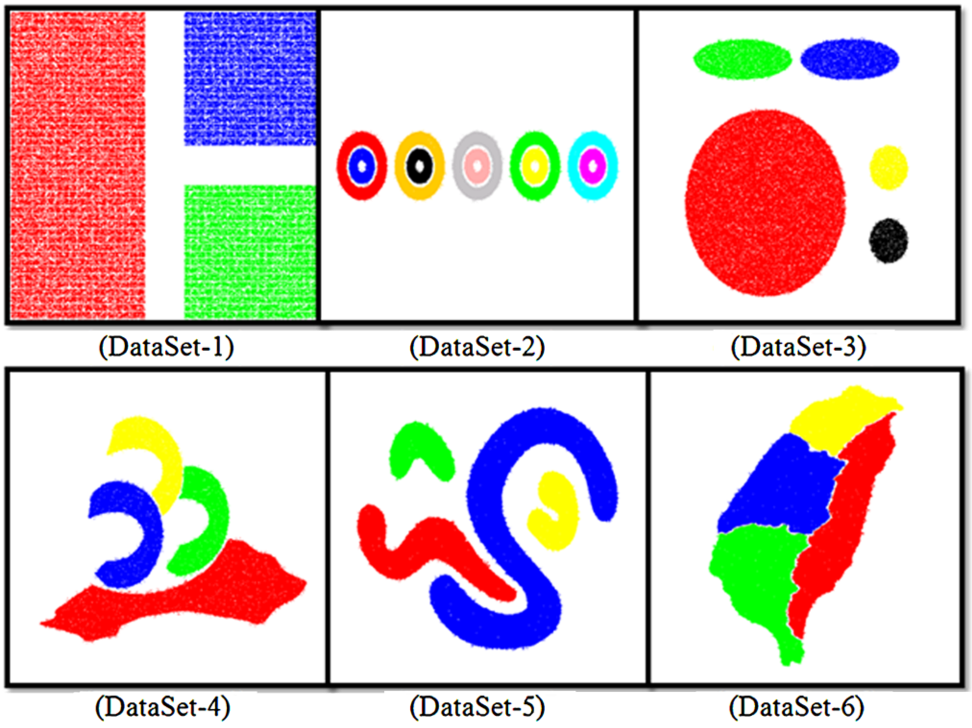


Fig. 18 The clustering results using several different data sets by PHD

Table 2 Comparisons of number of queries (in millions) of PHD, DBSCAN, IDBSCAN and KIDBSCAN using 115,000 points with data sets 2–6; the data set 1 has 100,000 points

Algorithm	DataSet-1	DataSet-2	DataSet-3	DataSet-4	DataSet-5	DataSet-6
DBSCAN	10000.000	12194.255	12079.140	12136.755	12104.555	12086.040
IDBSCAN	1710.800	2394.070	3212.525	2872.585	3016.680	3039.680
KIDBSCAN	1695.900	2391.655	3141.685	2895.585	2997.705	3023.120
PHD	603.671	337.329	724.203	730.411	764.681	773.961

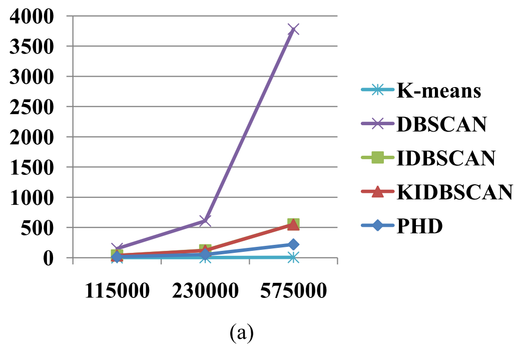
though K-means cannot identify arbitrary shapes, and filter out noise, it has the lowest run-time cost.

Table 2 lists the number of queries run by PHD, DBSCAN, IDBSCAN and KIDBSCAN with 100,000-point (Data set 1) and 115,000-point (Data sets 2–6) data sets, revealing that the proposed PHD has the lowest number of queries owing to its use of the partition space technique. The number of queries in DBSCAN is almost always n^2. IDBSCAN and KIDBSCAN sample the points as the *seed* to decrease the huge number of queries, while KIDBSCAN start cluster from high-density points. Figure 19a shows the run-time comparison of PHD, K-means, DBSCAN, IDBSCAN and KIDBSCAN using Data set 6 with various data sizes. These experimental results indicate that increasing data size raises the difference in run time between K-means, DBSCAN, IDBSCAN, KIDBSCAN and PHD. Figure 19b displays a comparison of the number of queries run by DBSCAN, IDBSCAN, KIDBSCAN and PHD using data Set 6 with various data sizes. An increasing data size leads to increasing difference in the number of queries run by DBSCAN, IDBSCAN, KIDBSCAN and PHD, as revealed in Fig. 19b.

Figure 20a depicts the quite obvious difference of time cost among these clustering algorithms. Notably, the proposed PHD algorithm only took 219.281 seconds to perform clustering task using 575,000 data sets with Data set 6. The proposed PHD algorithm is faster than DBSCAN, IDBSCAN, and KIDBSCAN approaches in clustering comparison. Furthermore, an obvious difference of query numbers among DBSCAN, IDBSCAN, KIDBSCAN, and PHD is revealed in Fig. 20b. It is observed that the proposed PHD clustering algorithm has the lowest query number in carrying out clustering task. Figure 21 shows the system interface with the clustering result of the data set 4.

5.4 Experiment on real data

To establish practical applicability of the PHD algorithm, this simulation utilized two real data sets, iris data [8] and wine recognition data [15]. Iris data has three clusters that



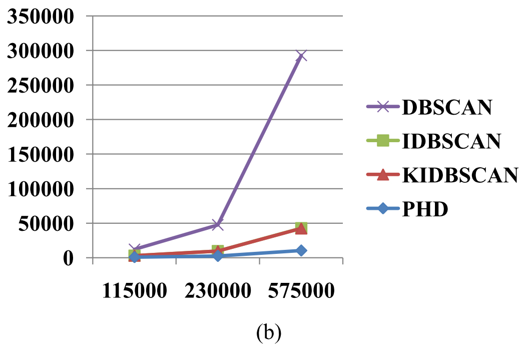
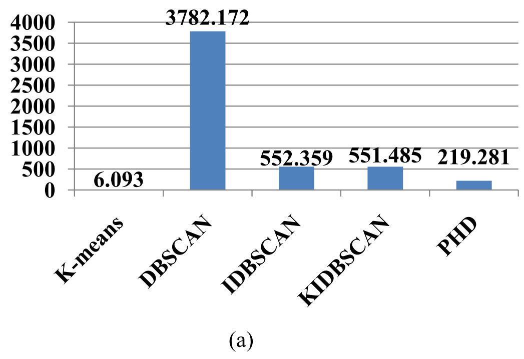


Fig. 19 (**a**) Performance comparison of K-means, DBSCAN, IDBSCAN, KIDBSCAN and PHD using data set 6 with various data sizes; x-axis indicates data sizes; y-axis represents the run-time cost (in seconds). (**b**) The comparison of number of queries of DBSCAN, IDBSCAN, KIDBSCAN and PHD using data set 6 with various data sizes; x-axis denotes data sizes; y-axis represents the number of queries (in millions)



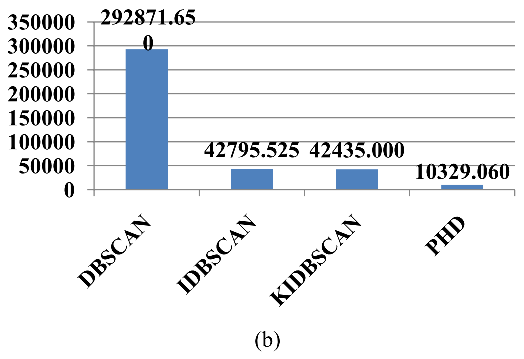


Fig. 20 (**a**) Performance comparison of K-means, DBSCAN, IDBSCAN, KIDBSCAN and PHD using 575,000 points with data set 6; x-axis indicates clustering algorithms; y-axis represents the run-time cost (in seconds). (**b**) The comparison of number of queries of DBSCAN, IDBSCAN, KIDBSCAN and PHD using 575,000 points with data set 6; x-axis denotes clustering algorithms; y-axis represents the number of queries (in millions)

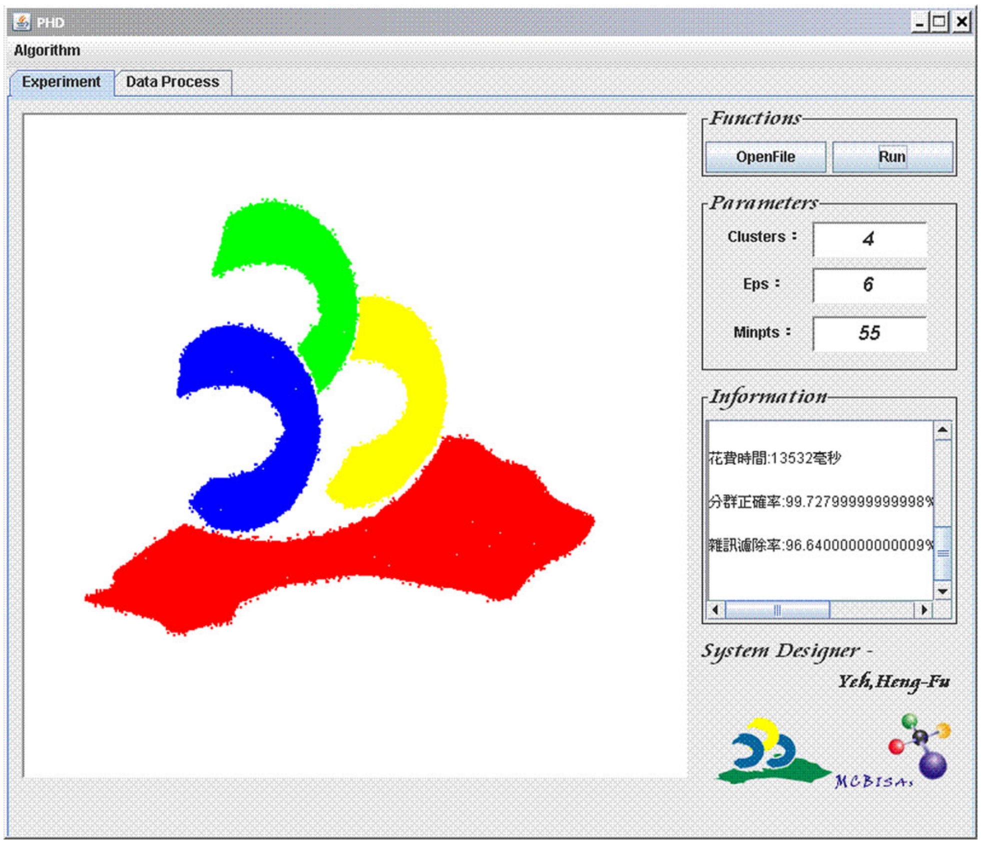
represents three different varieties of Iris flowers namely Iris setosa (I), Iris versicolor (II) and Iris virginica (III). Each cluster contains fifty instances, thus a total of 150 instances is available. Every instance is described by a set of four attributes, namely, sepal length, sepal width, petal length and petal width. Wine recognition data is obtained from UCI repository website [15]. This data set is the result of a chemical analysis of wines grown in the same region in Italy but obtained from three different cultivars. There are overall 178 samples, in which there are 59, 71 and 48 instances in cluster I, cluster II and cluster III respectively. Each sample exists in 13 dimensions. Before running clustering analysis, all attributes must be normalized to lie between 0 and 1, since different attributes are measured on different scales, when Euclidean distance formula is employed directly, the effect of numerous attributes might be completely dwarfed by others that have larger scales of measurement. The following equation reveals that the calculation of error rate depends on number of misclassified patterns (NMP) and the total number of patterns (TNP) in the data set. Figure 22 presents the system interface with the clustering result of Iris data.

$$Error(\text{in }\%) = \frac{NMP}{TNP} \times 100 \tag{3}$$

For iris data, the configuration of K, Eps and $Minpts$ are set to 3, 5 and 2. $K = 3$, $Eps = 10$ and $Minpts = 2$ are adopted for wine recognition data. The clustering results using the proposed PHD algorithm are presented from Tables 3–4. They indicate that the proposed PHD algorithm can also run on real data with the good clustering result.

6 Conclusion

This investigation presents an improved KIDBSCAN clustering algorithm named PHD algorithm for solving data clustering problem in large databases. The proposed PHD algorithm extends the existing KIDBSCAN by incorporating an improved partitioning technique to lower time taken



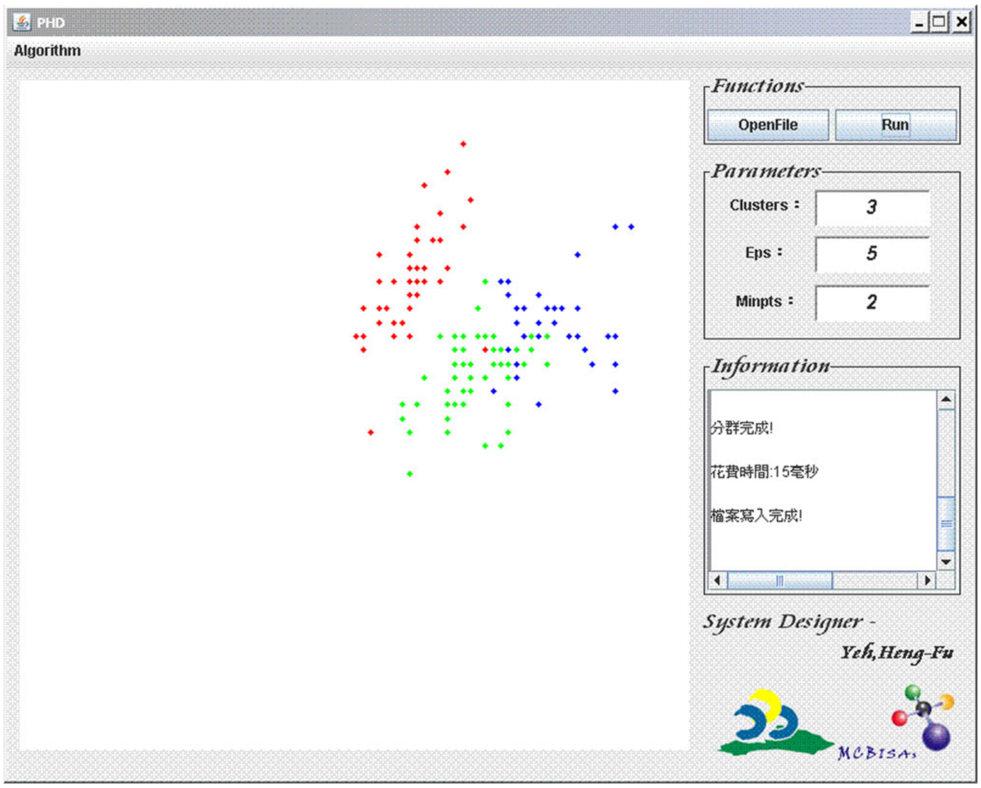
Clusters : 4
Eps : 6
Minpts : 55

花費時間:13532毫秒

分群正確率:99.72799999999998%

雜訊濾除率:96.64000000000009%

Fig. 21 The system interface with the clustering result of the data set 4



Clusters : 3
Eps : 5
Minpts : 2

分群完成!

花費時間:15毫秒

檔案寫入完成!

Fig. 22 The system interface with the clustering result of Iris data

Table 3 Iris data

Clusters found	points in cluster	Coming from I	II	III	Error (%)
C1	48	48	0	0	12.67%
C2	62	1	47	14	
C3	40	1	3	36	

Table 4 Wine recognition data

Clusters found	points in cluster	Coming from I	II	III	Error (%)
C1	59	57	2	0	5.06%
C2	67	1	65	1	
C3	52	1	4	47	

by neighborhood search. Experimental results have confirmed the partitioning method significantly shortens the run-time cost of clustering. PHD outperforms DBSCAN, IDBSCAN and KIDBSCAN in terms of execution time, without losing clustering quality. Although K-means performs clustering very quickly, it is not practical since it cannot filter out noise. The proposed PHD clustering algorithm is more efficient and effective than previous algorithms. The contribution of this work can be summarized as follows:

– The proposed PHD algorithm can efficiently be used for clustering large data sets.
– The proposed PHD algorithm is faster than KIDBSCAN with approximate three times in all simulations.
– The numbers of queries of the proposed PHD algorithm is less than KIDBSCAN with approximate 65% to 85% in all simulations.
– The proposed PHD algorithm is more scalable than other density-based algorithms as it conducts on partitions rather than the whole data set.

Acknowledgements The author would like to thank the National Science Council of Republic of China for financially supporting this research under contract no. NSC 98-2221-E-020-029 and NPUST 98-005.

References

1. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 94–105
2. Borah B, Bhattacharyya DK (2004) An improved sampling-based DBSCAN for large spatial databases. In: Proceedings of international conference on intelligent sensing and information processing, pp 92–96
3. Breitenbach M, Grudic GZ (2005) Clustering through ranking on manifolds. In: Proceedings of the 22nd international conference on machine learning, pp 73–80
4. Chen Y, Rege M, Dong M, Hua J (2008) Non-negative matrix factorization for semi-supervised data clustering. Knowl Inf Syst 17(3):355–379
5. Cheng H, Hua KA, Vu K (2008) Constrained locally weighted clustering. In: Proceedings of the VLDB endowment, pp 90–101
6. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd international conference on knowledge discovery and data mining, pp 226–231
7. Filippone M, Camastra F, Masulli F, Rovetta S (2008) A survey of kernel and spectral methods for clustering. Pattern Recogn 41:176–190
8. Fisher R (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7:179–188
9. Guha S, Rastogi R, Shim K (1998) CURE: An efficient clustering algorithm for large databases. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 73–84
10. Guha S, Rastogi R, Shim K (1999) ROCK: A robust clustering algorithm for categorical attributes. In: Proceedings of the 15th international conference on data engineering, pp 512–521
11. Karypis G, Han EH, Kumar V (1999) CHAMELEON: A hierarchical clustering using dynamic modeling. IEEE Comput 32(8):68–75
12. MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, vol 1, pp 281–297
13. Tsai C-F, Liu C-W (2006) KIDBSCAN: A new efficient data clustering algorithm for data mining in large databases. Lect Notes Comput Sci (LNCS) 4029:702–711
14. Tsai C-F, Yen C-C (2007) ANGEL: A new effective and efficient hybrid clustering technique for large databases. Lect Notes Comput Sci (LNCS) 4426:817–824
15. equation:UCI Repository. http://www.sgi.com/tech/mlc/db/
16. Wang T-P, Tsai C-F (2006) GDH: An effective and efficient approach to detect arbitrary patterns in clusters with noises in very large databases. Master thesis, National Pingtung University of Science and Technology, Taiwan
17. Ward JH Jr (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58(301):236–244
18. Zhang T, Ramakrishnan R (1996) BIRCH: An efficient data clustering method for very large databases. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 103–114

Cheng-Fa Tsai an IEEE member, is full professor of the Management Information Systems Department at National Pingtung University of Science and Technology (NPUST), Pingtung, Taiwan. His research interests are in the areas of data mining and knowledge management, database systems, mobile communication and intelligent systems, with emphasis on efficient data analysis and rapid prototyping. He has published over 160 well-known journal papers and conference papers and several books in the above fields. He holds, or has applied for, nine U.S. patents and thirty ROC patents in his research areas. Dr. Tsai is a member of IEEE, WSEAS and TAAI (Taiwanese Association for Artificial Intelligence).

Heng-Fu Yeh received the B.B.A. degree in Information Management from National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 2007. He is currently a graduate student in the Information Management Department, National Pingtung University of Science and Technology, Pingtung, Taiwan. His research interests include data mining, data clustering, and information hiding.

Jui-Fang Chang received the Ph.D. degree in Finance from United States International University, USA in 1997. Currently she holds Dean of General Education, National Kaohsiung University of Applied Sciences, Taiwan, ROC. She has a combined academic background and research experience in Finance, International Finance, International Marketing and International Business Management. Her current research concentrates on Genetic Algorithms, Particle swarm optimization (*PSO*), Fuzzy Logic, and Neural Networks Applied in Portfolio and Stock Market.

Ning-Han Liu received a PhD degree in computer science from National Tsing-Hua University, Taiwan, in 2005. He is an assistant professor of the Department of Management Information Systems, National Pingtung University of Science and Technology, Pingtung, Taiwan since 2006. His research interests include multimedia databases, audio processing in WSN, and artificial intelligence.