SUDEPHIC: Self-Tuning Density-Based Partitioning and Hierarchical Clustering*

Ding Zhou, Zunping Cheng, Chen Wang, Haofeng Zhou, Wei Wang, and Baile Shi

Dept. of Computing and Information Technology
Fudan University, Shanghai, China
0024133@fudan.edu.cn, c_zunping@yahoo.com.cn, chesterw@sohu.com
{haofzhou, weiwang1, bshi}@fudan.edu.cn

Abstract. Clustering is one of the primary techniques in data mining, for which to find the user expecting result is a major issue. However, to dynamically specify the parameters for clustering algorithms presents an obstacle for users. This paper firstly introduces a novel density-based partitioning and hierarchical algorithm, which makes it easy to employ synthetic feedback mechanism in clustering. Additionally, by investigating into the relation between parameters and the clustering result, we propose a self-tuning technique for the setting of parameters. Meanwhile, the density distribution within a cluster can be expressed in the result for the user to specify the cluster's feature. The algorithm is both evaluated in theory and practice. It outperforms many existing algorithms both in efficiency and quality.

1 Introduction

Data mining has been a heated topic for the last decade and still is. As a primary technique in the extracting of useful information and knowledge, a number of algorithms in cluster analysis have been proposed and have evolved into a large family. According to [3], existing clustering algorithms can be generally categorized into 4 classes: partitioning [17,16,18],hierarchical [10,9,27], density-based [6, 20,2,25] and grid-based [24,21,1]. There are also many algorithms combining two or more techniques in different categories to gain better efficiency and quality, such as [12,14,28,5].

In the process of cluster analysis, one important phase is the interpretation of clustering results [11]. Users expect clustering results to be interpretable and comprehensible. Therefore it is a major issue to describe the clustering results precisely as well as to provide easy ways for users to adjust the clustering for comprehensive results. Two problems are therefore involved: 1) how to include as much as useful information in the clustering results, e.g. the information of density; 2) how to find the user expecting results without loss of efficiency.

* This paper was supported by the Key Program of National Natural Science Foundation of China (No. 69933010) and China National 863 High-Tech Projects (No. 2002AA4Z3430 and 2002AA231041)

Y. Lee et al. (Eds.): DASFAA 2004, LNCS 2973, pp. 554–567, 2004.

As for the first problem, for example, *DBSCAN* [6] finds all the points that are density-connected for a given density threshold. But it neglects the inner density information of clusters. Meanwhile, the parameter of global threshold for *DBSCAN* makes it insensitive to the information of inner density variances in clusters. Aggregated data with the density above the threshold are grouped, but data in area with lower density data are treated as noises even if they are within a relatively aggregate area.

An obstacle for the second problem roots in that most existing algorithms require two or more parameters. And many of the clustering algorithms are very sensitive to their parameters. The large combination number of parameters make it impossible for users to dynamically specify the parameters for expecting results efficiently.

In this paper we propose an innovative density-based partitioning and hierarchical algorithm. The algorithm is a "blend" algorithm like *PHC* [28]. But it is based on a new notion of density-weight that makes it easy to adopt synthetic feedback mechanism in clustering. With the feedback capability, users are able to group data according to their understanding of data even if the data points are far apart in space. Meanwhile, by investigating into the relation between parameters and the clustering results, we introduce a self-tuning technique for parameter setting, which saves the costly time users spend in the setting of optimal parameters.

The rest of the paper is organized as follows. The full algorithm is narrated in section 2. Then experiments and performances compared with other algorithms are shown in section 3. Related algorithms on clustering are briefly discussed in section 4. Section 5 concludes the paper with a summary and make a short discussion of future research.

2 DEnsity-Based Partitioning and HIerarchical Clustering (DEPHIC)

2.1 Motivation

One important property of clusters is that the densities of aggregated data are not unique. The central density of a cluster may be times of the peripheral density (intra-differences). Also the differences of densities between clusters (inter-difference) can be obvious. Consequently, many real-data sets cannot be characterized by global similarity threshold. For example, in the dataset illustrated in Figure 1, cluster A, B and C may be considered as a large cluster if the global threshold that is suitable for cluster 2 is employed. Therefore, in the decision of similarity threshold, the density-variance between clusters should be taken into consideration.

Several alternatives were proposed to detect or analyze clusters with different densities. A straightforward way is to employ different parameter settings. However, since the number of possible combinations of parameters is very large, the time spent in seeking for the optimal combination of parameters is costly.

Our alternative way is to involve differentiations in points with different density in the computation of similarity. The new density-based partitioning and hierarchical clustering algorithm *DEPHIC* packages each point with an attribute that represents the point's density-weight. And this attribute is adopted in the hierarchical merge to achieve the discrimination of clusters with various densities.

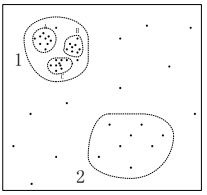


Fig. 1. Data Density Distribution

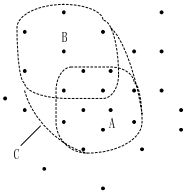


Fig. 2. When Merge

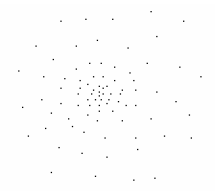


Fig. 3. Original Data

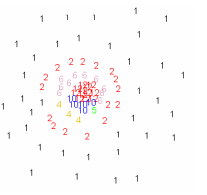


Fig. 4. $D-Weight$

2.2 DEnsity-Based Partitioning and HIerarchical Clustering (DEPHIC)

In this section, we focus on a new density-based partitioning and hierarchical clustering algorithm in principle like such an extended *PHC* [28]. The idea of *PHC* is to merge small *Merging Clusters* hierarchically. *PHC* contains two phases, in the first, *Base Clusters* are generated by grouping the $Radius_D$ nearest neighbors of points, which are selected randomly from the database; In the second phase, *Merging Clusters(MC)* are merged to achieve the *Maximum Merging Clusters(MMC)*. For details, refer to [28].

Our new algorithm *DEPHIC* also includes two main steps: density-based hexagonal partition (*DEP*) and density-based hierarchy merge (*DEH*). In *DEP*, we partition the data space into equal-sized grids and sort them by their densities. Then *Base Clusters* are created by firstly selecting the points from denser area. In the second phase *DEH*, hierarchical merge based on density is performed. It merges the *Base Clusters* generated in *DEP* to MCs and finally stops when all the MCs become $MMCs$.

2.2.1 The Density Weight: D-Weight

The $D-Weight$ of a point P can be regarded as an attribute that reflects the density of region which P is in and the approximate scale of the cluster that P belongs to. For simplicity, let's begin with formal definitions:

Definition 1 (A point's D-Weight). *Given a point P, there are two steps to determine its $D-Weight$: the initial value of $P.D-Weight$ is given by the number of points that fall into the same grid as P. And P.D-Weight is dynamically modified when the cluster containing P is merged with another cluster. Specifically, on merging two clusters X and Y:* **if P is in the overlap of X and Y,**

$$P.D{-}Weight = \begin{cases} \dfrac{P.D - Weight + P'.D - Weight}{2} & D - Weight(X) \geq (1+\varepsilon)D - Weight(Y) \\ & \textbf{or } D{-}Weight(X) \leq (1{-}\varepsilon)D - Weight(Y) \\ P.D - Weight & (1-\varepsilon)D - Weight(Y) < D - Weight(X) \\ & < (1+\varepsilon)D - Weight(Y) \end{cases}$$

where $P \in X$ and $P' \in Y$ but representing the same point, **else if P is not in the overlap,**

$$P.D - Weight = \begin{cases} P.D - Weight + \triangle & D - Weight(Y) \geq (1+\varepsilon)D - Weight(X) \\ P.D - Weight & Otherwise \end{cases}$$

where ε is maximum tolerable error in determining the equivalence of $D - Weight$. The Δ is the tuning factor that indicate the scale(S) of cluster which was added to cluster X while merging. It is determined by the S multiplied by a shrinking factor. In sparse distribution of data, larger Δ is preferred.

Definition 2 (A cluster's D-Weight). *Given a cluster C, its D-Weight is the maximum D-Weight of its points. Formally, $C.D - Weight = max(x.D - Weight)$, where $x \in C$.*

Intuitively, the $D - Weight$ distribution over a cluster reflects the density distribution of this cluster. Note that the $D - Weight$ is dynamically adjusted in the hierarchical merge. One situation of the merge is illustrated in Figure 2, suppose that cluster C is created by merging cluster A and B. A is near the center of C where the density is larger. According to Definition 1, the points in A have larger $D - Weight$. The points in B, because of its loose density, have lower D-Weight. After the merge of A and B, the $D - Weight$ of points in A is incremented by Δ. Therefore the points in region A now have bigger $D{-}Weight$. Meanwhile the $D - Weight$ in B remains unchanged.

In a cluster, a point P's $D - Weight$ reflects the density of the region P is in. A high $D - Weight$ can be generated in either of the following situation: 1) the point P is in a very dense region. Therefore P's $D - Weight$ is initiated high because of the large scale that P is in; 2) P was in a region that has larger $D - Weight$ than the $Merging\ Clusters$ around it. And P's $D - Weight$ is incremented gradually in its merge with its neighbor $Merging\ Cluster$. In the second situation, a high $D - Weight$ of P indicates that P is near the center of its final MMC.

Figure 3 and 4 illustrates a real dataset cluster with intra-density variance and its $D - Weight$ distribution. We can see that in the center part where the density of points is high, the $D - Weight$ of points can be relatively large. The points in the margin area of the cluster, on the other hand, have lower $D - Weight$. The property of $D - Weight$ is of value for the recognition of clusters in certain shapes because of its reflection of density information of a region. Meanwhile, the density distribution within a cluster can be recognized by the values of points' $D - Weight$.

2.2.2 Density-Based Hexagonal Partition (DEP)

Remember that in the first phase of *PHC*, points are selected randomly for the creation of *Base Cluster*s. Thus the selected point may be noise or biased. In experiments, we show that *PHC* may generate different clustering results on the same dataset with the same parameters, which we name as instable. The instability of *PHC* roots in that the density features of clusters is overlooked in its first phase of generating *Base Clusters*. Consequently, we first partition the space into equal-sized grids, then sort them by their densities descendingly and generate the base clusters with from larger $D - Weight$ points to smaller. The method of *Base Cluster* creation is introduced in [28].

Input:	the set of data points $SetOfPoints$, average distance $Radius_D$ between two points in data space D
Output:	null
Method:	DEP($SetOfPoints$,$Radius_D$) (1) Partition the data space with grid diameter of $Radius_D$ (2) Stamp the $D - Weight$s of points with the scales of grids they belong to. (3) Sort the grids descendingly by their scales. (4) Create *Base Clusters* with the points in sorted grids.

Fig. 5. Algorithm DEP

2.2.3 Density-Based Hierarchical Merge (DEH)

In hierarchical clustering, the type of the linkage metric significantly affects the algorithms. Many existing algorithms adopt similarity between clusters for the inter-cluster linkage metrics, however, some of them ignore the property of density in clusters. In *DEH*, the second phase of density-based partitioning and hierarchical clustering (*DEPHIC*), we determine the similarity between two clusters by the sum of the $D-Weight$ of their overlap points. Figure 6 illustrates the algorithm *DEH*. Merge(MC_i,MC_j) merges MC_i and MC_j by removing them from MCS and adds a merge cluster with D-Weightes modified, and finally stop when no merge can be made. The procedure CanMerge(MC_i,MC_j,r) first computes the sum (S) of the D-Weight in overlapped points, and decides not to merge the two clusters when S is less than γ of both the sums of MC_i and MC_j.

3 Self-tUning DEnsity-Based Partition and HIerarchy Clustering (SUDEPHIC)

An important criterion in evaluating a clustering algorithm is its reliance on parameters and whether the parameters are meaningful to users. Currently, most clustering algorithms need two parameter or more to adjust the performance of the clustering. It becomes almost impossible for users to determine the optimal

Input: the set of $MergingClustersMCS$, the similarity threshold γ
Output: null
Method: DEH(MCS,γ)
(1) do
(2) change ← false;
(3) for every two MC_i and MC_j in MCS do
(4) if CanMerge(MC_i,MC_j,r)= true do
(5) change←true;
(6) Merge(MC_i,MC_j);
(7) while(change);

Fig. 6. Algorithm DEH

parameters easily due to the large number of combinations of parameters. Another problem is the tuning of parameters. First of all, the optimal parameters of a clustering algorithm are determined by the distribution of data points. Thus, there is often no straightforward way to determine the initial value of the parameters; On the other hand, the step in the adjustment is hard to determine. Besides, many clustering algorithms are oversensitive to their parameters, often producing sheer different results of clustering even if a slight change in parameters. In this section, we propose two methods in tuning of the only parameter of $DEPHIC$.

3.1 SUDEPHIC[1]: Tuning When the Desired Cluster Number Is Given

Given desired number of cluster, a self-tuning strategy is employed in $SUDEPHIC^1$. We tune the parameter γ based on the historical γ and their corresponding cluster numbers. A histogram with fixed number m of records is firstly built at the beginning of the iterations of clustering. During each iteration of the clustering, the γ for current iteration and the cluster number in this iteration is added as a record to the histogram. Historical records further than m are discarded to keep the regression curve smooth. Then the histogram is used to generate a m-rank polynomial *P(c)* in each iteration. Thus the new γ to be used in the next iteration is predicted by *P(K)* taking into the desired number of clusters K. Following is the algorithm $SUDEPHIC^1$:

Figure 8 illustrates how the polynomial *P(K)* is used in NextRatio(K) to predict the γ in next iteration of clustering. As illustrated in the figure, before the next iteration of clustering, there have been 5 iterations. To predict the next parameter for 6^{th} iteration, procedure NextRatio(K) first make a polynomial regression P(c) for the recent 5 points, where the c is the number of clusters and P(c) is the historical similarity thresholds.NextRatio(K) then calculate the P(c') to get the γ' and use $(\gamma' + \gamma)/2 = \gamma^6$ as the new similarity threshold for the next iteration.

Input: the set of points $SetOfPoints$,the desired number of clusters K, similarity threshold γ
Output: null
Method:
(1) SUDEPHIC[1](SetOfPoints, K,γ)
(2) **while** the difference between current cluster number N and K is large **do**
(3) DEPHIC(D,γ);
(4) add a record of $[\gamma, N]$ to the histogram H;
(5) new_$\gamma \leftarrow$ (NextRatio(K,H)+ γ)/2;

Fig. 7. Algorithm SUDEPHIC[1]

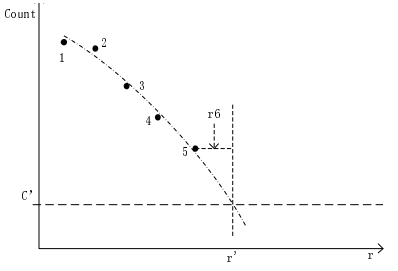


Fig. 8. Determining the next ratio

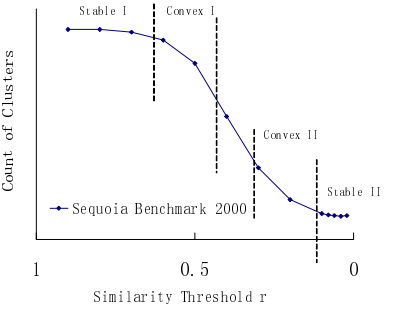


Fig. 9. γ-count relation

3.2 SUDEPHIC[2]: Tuning Parameter Automatically

In this section, we try to eliminate the last parameter in $DEPHIC$. We start by introducing an interesting relationship between the similarity threshold γ and the number of clusters *count* under γ. And then we discuss how the feature between γ and *count* is employed in the non-parameter clustering algorithm SUDEPHIC[2].

3.2.1 The Relation between γ and *count* in DEPHIC

In general, the smaller γ is, the more clusters will be merged in the hierarchical merge of $DEPHIC$ and there will be fewer and larger clusters in the result. Moreover, the decline in the number of clusters *count* usually doesn't follow straight line. It is proved that the clustering on sequence data represents interesting property[26]: there are often convex parts in the $\gamma - count$ relation curve. And after a large amount of experiments on real datasets and synthetic data sets, we discover that similar property also exits in the $\gamma - count$ relation in $DEPHIC$. Figure 9 illustrates the relation between γ and *count* in the clustering on *Sequoia Benchmark 2000* data. We will also show the γ-count relation in the experiments 4.1.4.

3.2.2 Tuning Parameter Automatically

When we examine Figure 9 more closely, we can see that there are two convex parts in the histogram curve: $ConvexI$ and $ConvexII$. Between these two parts, there is a sharp fall in the count of clusters, which we heuristically regard as the instable part. Obviously, in the instable part, the clustering performance is very sensitive to the parameter γ, a slight change in γ may lead to sharp difference in the number of clusters. Therefore a relatively small step in the adjustment of γ is preferred if clustering performance within this range is aimed. On the other hand, before $ConvexI$ and after $ConvexII$, algorithm is less sensitive to the similarity threshold γ. The *count* changes slightly even if there is a great difference in γ. Intuitively, we are mostly interested in the stable parts[1], especially the performance of clustering at $StableII$. Therefore the γ after $ConvexII$ can be regarded as the reasonable similarity threshold.

To determine whether a point reaches $StableII$, the *Sharpness* and *Convexity* at a point is defined.

Definition 3 (Sharpness). *Given a point* $X(\gamma_i, \text{count}_i)$*, let the slope of points before it is* $\boldsymbol{SLOPE}_{before}$ *[13], and* $\boldsymbol{SLOPE}_{after}$ *is the slope of points after it. The* Sharpness *of the curve at this point can be measured by the difference between* $\boldsymbol{SLOPE}_{before}$ *and* $\boldsymbol{SLOPE}_{after}$*. Formally:*

$$Sharpness = \boldsymbol{SLOPE}_{before} - \boldsymbol{SLOPE}_{after}$$

Definition 4 (Convexity). *A curve is* Convex Down *at a point* $X(\gamma_i, \text{count}_i)$ *if the* Sharpness *at this point is positive, and the curve is* Convex Up *if the* Sharpness *at X is negative.*

According to Definition 4, we can see that convex like $ConvexI$ is *Convex Up* and the $ConvexII$ is *Convex Down*.

Due to the special property of the $\gamma - count$ curve in $DEPHIC$, the problem of clustering without user-specified similarity threshold can be degenerated to the problem of adjusting according to the trend of the curve. In $SUDPEPHIC$[2], we begin the iteration of clustering by an initial γ, whose value doesn't affect the quality of clustering. In each iteration of clustering, a point that represents the $[\gamma_i, count_i]$ in this iteration is added to the histogram. And we monitor the *Sharpness* and *Convexity* of the current point. Recall that the $\gamma - count$ curve is dynamically built. Then if the point falls into the *Convex Down* part, we begin to check the change in the count of clusters. If the change in count becomes small enough, or rather, $\frac{|count_{k-1} - count_k|}{count_{k-1}} \leq \varepsilon$, the iteration will be suspended. And the $\gamma - count$ point now is at the $StableII$ part of the $\gamma - count$ curve.

[1] Admittedly, there may be some cases in which people care about the clustering performance within this "instable" range, but it is not very common.

4 Experiments

In this section we evaluate the quality of *SUDEPHIC* as well as the quality and efficiency of *DEPHIC*. We compare them with *K-Means*, *DBSCAN* and *PHC*.

All experiments were performed on Pentium IV with 2.2 GHz containing 256MB DDR of main memory and Windows XP as operating system. All algorithms were implemented in Java and were executed on the Java Virtual Machine Version 1.4.1 from Sun. We use the synthetic data sets as well as the database of *Sequoia Benchmark 2000* in the experiments.

(a) DS1 (b) DS2 (c) DS3 (d) DS4

Fig. 10. DataSets

4.1 Quality Evaluation

4.1.1 Stability: Density-Based Partition and Random Select

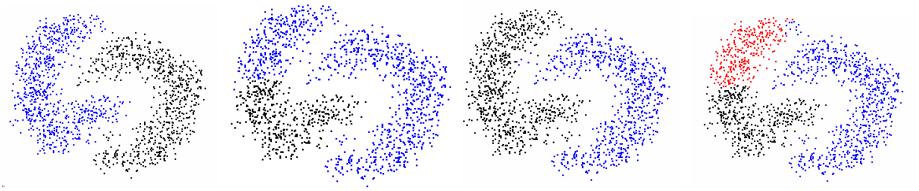
Recall that in the first phase of *PHC*, points are selected randomly for the creation of Base Clusters. Thus the selected point may be noise or biased. When the noise points are selected, the *PHC* may generate different clustering results on the same dataset with the same parameters. On the other hand, the results of *DEPHIC* with certain parameter remain unique. We can see from Figure 11 that *PHC* may be instable on DS1.

4.1.2 Clustering on Synthetic Data Set

Synthetic data sets reported in *DBSCAN*, *PHC* and *CHAMELEON* are used to test the clustering quality of *DEPHIC* in this section. *DEPHIC* performs well on these data sets with proper parameter. Since the algorithm *SUDEPHIC* is just the *DEPHIC* with the optimal similarity threshold. We present the performance of *DEPHIC* on some data sets in Figure 12

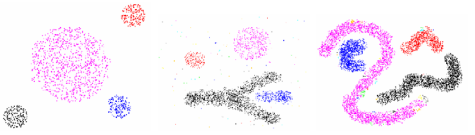
4.1.3 Clustering on Special Cluster Shape

One representative shape that *PHC* isn't able to cluster well is illustrated in Figure 13. X is a Merging Cluster with sparse density and Y is a MC with dense



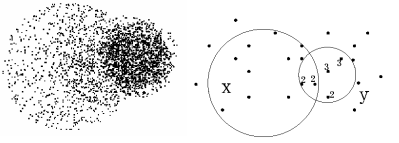
(a) DEPHIC $\gamma = 0.03$ (b) PHC$^{(1)}$ $\gamma = 0.1$ (c) PHC$^{(2)}$ $\gamma = 0.1$ (d) PHC$^{(3)}$ $\gamma = 0.1$

Fig. 11. Instability of PHC



(a) DEPHIC on DS2 (b) DEPHIC on DS4 (c) DEPHIC on DS3

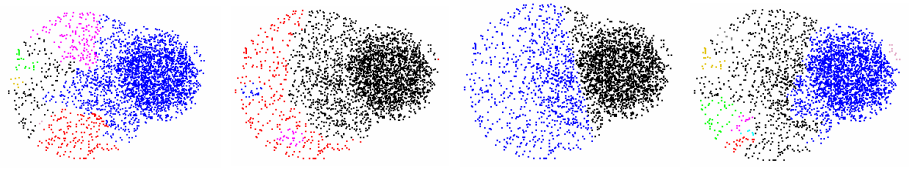
Fig. 12. Clustering On Synthetic Data Set



(a) DS6 (b) Details about DS6

Fig. 13. DS6

points. Let γ be 0.38 for *PHC*, we see from the picture that though more than γ of Y's points are contained in cluster X, cluster Y should rather be separated, because it is more acceptable that Y be extended to include the points on its right than be merged with X on its left. In *PHC*, as the number of common points in Y is 2 and Y has a size 5, which make the similarity in Y be $\frac{2}{5} > 0.38 = \gamma$, the two MCs are merged. However, after importing D-Weight, the similarity in Y will be $\frac{2+2}{3+3+2+2+2} = 0.33 < 0.38 = \gamma$. Thus, the density-based hierarchical merge guarantees that X and Y be separated, which coincides with what we presumed. Figure 14(a) 14(d) 14(c) illustrate the clustering results of *PHC*, *DEPHIC* and K-Means on this shape of data distribution.



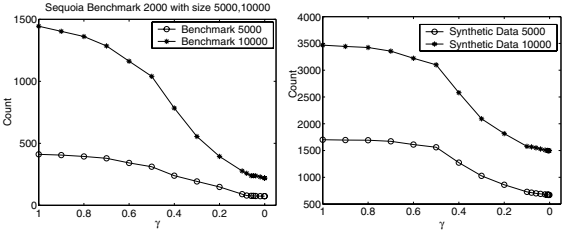
(a) PHC on DS6 (b) DBSCAN on DS6 (c) k-means on DS6 (d) DEPHIC on DS6

Fig. 14. Performances on DS6

Though given the best performances, we can see in Figure 14(b) that *DBSCAN* recognized a large part of the cluster on the left to be noise (denoted by color red). Meanwhile, *PHC* didn't separate the two clusters in Figure 14(a). Due to the limited space, other shapes of data distribution on which *DBSCAN* and *PHC* fail to do well are not listed here (e.g. dumb ring like shape).

4.1.4 γ-Count Relation in DEPHIC

As mentioned in Section 3.2, the larger γ is, the more clusters there will be in the clustering. However, we discover that the decline in the number of clusters count usually doesn't follow straight line. Intuitively, the fall in the cluster number count won't continue to be sharp and it tends to achieve a region of values in which the fall in count becomes negligible. We conduct lots of experiments on real datasets as well as synthetic datasets. And it is interesting to discover that there are often two convex parts in the curve of the decreasing count. Figure 15 are the γ-count relation in real data set and synthetic data set. We can see that both follow the property described in section 3.2. Moreover, our experiments show that the qualities of clustering by $SUDEPHIC^{1,2}$ are independent of the initial value of γ.



(a) (b)

Fig. 15. γ-count relation in DEPHIC

4.2 Runtime Analysis

4.2.1 Runtime and Speed-Up w.r.t. Desired Cluster Number

Figure 16 shows the runtime of $SUDEPHIC^1$ with different desired number of clusters. We can see that *SUDEPHIC* approximately has five or six times as the runtime of *DEPHIC* on the same data set. That is to say, *SUDEPHIC* performs only six iterations of *DEPHIC* to achieve the optimal quality of clustering, which is a great superiority to manual adjustment.From Figure 16, we can see that the runtime of $SUDEPHIC^1$ remains stable to all desired numbers of clusters and is therefore predicatble.

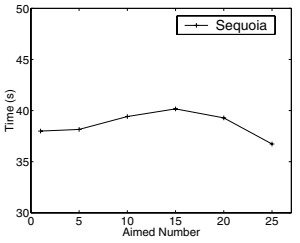


Fig. 16. Runtime and Speed-Up w.r.t. Desired Cluster Number

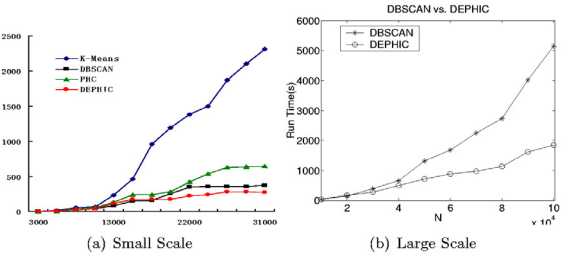


Fig. 17. Runtime and Speed-Up w.r.t. Database Size

4.2.2 Runtime and Speed-Up w.r.t. Database Size

The complexity of *DEPHIC* relies on two parts: *DEP* and *DEH*. We can see from Algorithm *DEP* that the time consumption of *DEP* consists of two parts, the first part of which is used in partitioning the data space. By checking all the points, the partition is performed in time $O(n)$. The other part of time consumption is in the stamping of points. In this part, the time is within $O(\sum_{i=1}^{k} n_i)$, where n_i is the size of grid G_i. Because the points in all the grids equal to the size of the dataset n, the time complexity in this part is also $O(n)$. Therefore, the total time complexity of *DEP* is $O(n + n) = O(2n)$, which is bounded by $O(n)$. The second phase *DEH* in *DEPHIC* is similar to the second phase of *PHC*, so according to [28] the time complexity of *DEH* is still bounded within $O(mn)$,where the m is the number of *Base Clusters*.

Figure 17(a) compares of runtime of *K-Means*, *DBSCAN*, *PHC* and DEPHIC on a series of the subsets of *Sequoia Benchmark 2000* point data. Figure 17(b) represents the performances of four algorithms on database with larger scale.

5 Related Work

Pure partitioning, hierarchical and density-based clustering algorithms have obvious merits and demerits: partitioning method only performs well on convex shapes; hierarchical method is able to handle arbitrary shapes but most algorithms in this method require a time complexity of $O(n^2)$ and density-based algorithms strongly depend on their parameters.

Partitioning algorithms can be further categorized into probabilistic clustering [22,4,7], *k-medoids* methods [16,18], and *k-means* [17] methods. Such methods concentrate on how well points fit into their clusters and tend to build clusters of proper convex shapes. Another way of clustering is based on data grid, which include *BANG*, *STING*, *WaveCluster* [19,24,21]. They are fast and handle outliers favorably. Grid-based methodology is also used as an intermediate step in many algorithms([1], [8]). The algorithm *STING* works with numerical attributes and is designed to facilitate "region oriented" queries. Density-based algorithm was

firstly proposed in [6]. It groups regions with sufficiently high density into clusters, and discovers clusters of arbitrary shape in spatial databases with noise. [2] is an extension of the *DBSCAN*. It computes an augmented cluster ordering for automatic and interactive cluster analysis. Other extension of density-based algorithms includes [15] and [25]. One representative of the algorithms that synthesize the features of partition and hierarchy is *CHAMELEON*[14]. Considering that *CURE*[10] ignores information about the aggregate cohesion of objects in two different clusters, *CHAMELEON* first uses a graph partitioning algorithm to cluster the data objects into a large number of relatively small sub clusters. It then aggregates sub clusters by the cohesion and the closeness among clusters. *CHAMELEON* requests two input parameters in setting the thresholds of the two measures. Another similar algorithm is *PHC*[28]. It utilizes *K-Means* to build sub clusters in its first phase and then uses a relative simple function to measure the closeness between clusters. *PHC* is more efficient than *CHAMELEON*.

6 Conclusion

In this paper, we propose an efficient clustering algorithm *DEPHIC* which relies on the notion of density-based partitioning and hierarchy.It introduces the density-weight of a point to distinguish clusters with inner density variance. In the setting of the parameter, two self-tuning strategy are introduced in $SUDEPHIC^1$ and $SUDEPHIC^2$. Experimental results demonstrate that $SUDEPHIC^{1,2}$ is significantly more efficient in discovering clusters of arbitrary shape than the well-known previous work.

References

[1] AGRAWAL R. etc. CLIQUE: Automatic subspace clustering of high dimensional data for data mining applications. *In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, 94-105.

[2] ANKERST M. etc. OPTICS: Ordering Points to Identify the Clustering Structure. *In Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)*, 49-60.

[3] BERKHIN P. Survey of Clustering Data Mining Techniques. In: *Accrue Sotware*, 2002

[4] CHEESEMAN P. etc. Bayesian Classification (AutoClass): Theory and Results. *In Fayyad, U.M., etc. (Eds.) Advances in Knowledge Discovery and Data Mining, AAAI Press/MIT Press.*, 1996.

[5] DASH M. etc. $1 + 1 > 2'$: Merging Distance and Density Based Clustering. In: *the 7th International Conference on Database Systems for Advanced Applications, DASFAA 2001*, 32-39.

[6] ESTER M. etc. A density-based algorithm for discovering clusters in large spatial databases with noise. *In Proc.of the Second Intl Conference on Knowledge Discovery and Data Mining*, 1996.

[7] FRALEY C. etc. MCLUST: Software for model-based cluster and discriminant analysis. *Tech Report 342, Dept. Statistics, Univ. of Washington*, 1999.

[8] GOIL S. etc. MAFIA: Efficient and scalable subspace clustering for very large data sets. In: *Technical Report CPDC-TR-9906-010, Northwestern University*, 1999.

[9] GUHA S. etc. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *In Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, 512-521.

[10] GUHA S. etc. CURE: An Efficient Clustering Algorithm for Large Databases. In: *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data(SIGMOD'98)*, 73-84.

[11] HAN J.W. etc. Data Mining: Concepts and Techniques. In: *Morgan Kaufmann Press*, June, 2000.

[12] HINNEBURG A. etc. An efficient approach to clustering in large multimedia databases with noise. *In Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, 58-65.

[13] JAIN R. The Art of Computer Systems Performance Analysis. In: *John Wiley & Sons, Inc.*, 1991

[14] KARYPIS G. etc. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. In: *Computer*, 32: 68-75, 1999.

[15] KRIEGEL H.P. etc. Incremental OPTICS: Efficient Computation of Updates in a Hierarchical Cluster Ordering. In: *Proc. 5th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'03)* 2003.

[16] KAUFMAN L. etc. PAM & CLARA: Finding Groups in Data: An Introduction to Cluster Analysis. In: *New York: John Wiley & Sons*, 1990.

[17] MACQUEEN J. k-means:Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley Symp. Math. Statist, Prob.*, 1:281-297.

[18] NG R. etc. CLARANS: Efficient and effective clustering method for spatial data mining. *In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, 144-155.

[19] SCHIKUTA E. etc. The BANG-clustering system: grid-based data analysis. *In Proceeding of Advances in Intelligent Data Analysis, Reasoning about Data, 2nd International Symposium*, 513-524.

[20] SANDER J. etc. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. *In Data Mining and Knowledge Discovery* 2, 2, 169-194.

[21] SHEIKHOLESLAMI G. etc. WaveCluster: A multiresolution clustering approach for very large spatial databases. *In Proceedings of the 24th Conference on VLDB*, 428-439.

[22] WALLACE C. etc. Intrinsic classification by MML-the Snob program. *In the Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, 37-44.

[23] WANG W. etc. STING+: An approach to active spatial data mining. *In Proceedings 15th ICDE*, 116-125.

[24] WANG W. etc: a statistical information grid approach to spatialdata mining. *In Proceedings of the 23rd Conference on VLDB*, 186-195.

[25] XU X. etc. DBCLASD: A distribution-based clustering algorithm for mining large spatial datasets. *In Proceedings of the 14th ICDE,*, 324-331.

[26] YANG J. etc. CLUSEQ: efficient and effective sequence clustering. *In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE), 2003.*

[27] ZHANG T. etc. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data(SIGMOD'96)*, 103-114.

[28] ZHOU H.F. etc. PHC: A Fast Partition and Hierarchy-Based Clustering Algorithm. In: *Journal of Computer Science and Technology*, Vol.18, No.3, 407-411.