Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Using Greedy algorithm: DBSCAN revisited II*

YUE Shi-hong (岳士弘)†1, LI Ping (李 平)1, GUO Ji-dong (郭继东)2, ZHOU Shui-geng (周水庚)1

(1Institute of Industrial Process Control, Zhejiang University, Hangzhou 310027, China)

(2Yili Teacher's College, Yining 835000, China)

†E-mail: Shyue@iipc.zju.edu.cn

Received July 16, 2003; revision accepted Jan. 19, 2004

Abstract: The density-based clustering algorithm presented is different from the classical Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester *et al*., 1996), and has the following advantages: first, Greedy algorithm substitutes for R*-tree (Bechmann *et al*., 1990) in DBSCAN to index the clustering space so that the clustering time cost is decreased to great extent and I/O memory load is reduced as well; second, the merging condition to approach to arbitrary-shaped clusters is designed carefully so that a single threshold can distinguish correctly all clusters in a large spatial dataset though some density-skewed clusters live in it. Finally, authors investigate a robotic navigation and test two artificial datasets by the proposed algorithm to verify its effectiveness and efficiency.

Key words: DBSCAN algorithm, Greedy algorithm, Density-skewed cluster
doi:10.1631/jzus.2004.1405 **Document code:** A **CLC number:** TP393.9

INTRODUCTION

Clustering groups dataset data into meaningful subclasses in such a way that minimizes the intra-differences and maximizes the inter-differences of these subclasses; and is one of the most widely studied problems in data mining. There are many application areas for clustering techniques, such as statistical data analysis, pattern recognition, image processing, and other business processes etc. Many clustering algorithms have been proposed, in particular, DBSCAN (Ester *et al*., 1996) as a density-based clustering method attracts much attention due to its intuitive meaning and good performance.

However, a clustering algorithm that can do everything that DBSCAN can do is not yet available. Various new clustering algorithms appear occasionally. DBSCAN has been modified to great extent recently and used to derive a new procedure to calculate EPS which are most important parameters. This paper shall attack another problem of DBSCAN in large-scale spatial dataset, i.e., its inapplicability to datasets with density-skewed clusters; and its excessive consumption of I/O memory. This paper

1. Uses Greedy algorithm (Skieyca, 1990) to index the space in DBSCAN so that both time and space complexity are decreased to great extent;
2. Presents a merging condition to distinguish density-skewed clusters;
3. Shows relatively better performance in robotic navigation and in two artificial datasets.

The rest of this paper is organized as follows. Section 2 presents a summary on DBSCAN algorithm. Section 3 presents a based on Greedy algorithm and DBSCAN (GDBSCAN) clustering algorithm implementable in any dataset with den-

*Project (No. 2002AA2010) supported by the Hi-Tech Research and Development Program (863) of China

sity-skewed clusters, and compares its time cost and memory loan with other corresponding algorithms in Section 4. Three experiments were used to verify the efficiency and effectiveness of GDBSCAN in Section 5. Conclusions are presented in Section 6.

RELATED WORK

DBSCAN is designed to discover arbitrary-shaped clusters in any dataset *D* and at the same time can handle noise or outliers effectively. The core point in DBSCAN refers to such point that its neighborhood of a given radius (*EPS*) has to contain at least a minimum number (*Minpts*) of other points. The procedure for finding a cluster is based on the fact that a cluster can be determined uniquely by any of its core points. To find a cluster, DBSCAN starts from an arbitrary point *p* and retrieves all density-reachable points in *D* (i.e. there exists a chain of directly density-reachable points, which means a point is contained in the neighborhood of the core point sequentially) from *p* wrt *EPS* and *Minpts*. When *p* is a border point and none point is density-reachable from *p*, then *p* is assigned to noise temporarily. Then DBSCAN handles the next point in dataset *D*. Retrieval of density-reachable points is performed in a successive region query. A region query returns all points intersecting a specified threshold efficiently with R^*-trees. Before clustering a dataset, R^*-tree must be built (Ester *et al.*, 1996). However, the procedure has very high time cost in the whole clustering process and excessive consumption of memory of I/O to great extent although it reduces the time cost of DBSCAN to $O(n\log n)$, there is a huge amount of unnecessary search and waste of space resource. Furthermore, DBSCAN cannot correctly distinguish the density-skewed clusters in a dataset, which can be illustrated as follows. Let us see Fig.1 that shows a dataset holding 5 clusters with different densities and between-cluster distances. The two clusters on the left side are dense, and the other two clusters on the right side are sparse. If we choose the *EPS* value in terms of densities on the

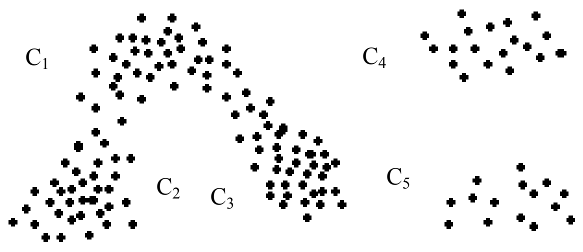

Fig.1 Five density-skewed clusters

right side, DBSCAN will merge the three clusters into one cluster. Otherwise, if the *EPS* value is chosen according to the densities of clusters on the left side, the two clusters on the right will be disrupted into multiple smaller clusters or outliers. Under such circumstances, there is no global EPS with which DBSCAN can aggregate the dataset correctly. However, if we partition the dataset into two parts such that the three dense clusters on the left side is held in the left partition and the two sparse clusters on the right are within the right partition, and cluster the two partitions separately, then a satisfactory clustering result can be obtained. Therefore, partitioning dataset is favorable as far as clustering efficiency is considered. Unfortunately, the partitioning technique is not easy exercise either. In fact, after partitioning the dataset, merging and synthesis of the partitioned sub-clusters with different EPS values must be done. The problem is that how and when to partition. In this paper, we shall give different treatments for different points to solve the above difficulty, and use Greedy algorithm and a careful merging condition instead of the corresponding components in DBSCAN.

CLUSTERING APPROACH BASED ON GDBSCAN

The GDBSCAN algorithm includes the following main steps

1. Retrieving core points based on the procedure of determining EPS in (Ester *et al.*, 1996);

2. Enhancing the effect of core points with higher densities by using larger EPS and reducing the effect of core points with lower densities by

using lower EPS;

3. Finding subclusters with the least number based on Greedy algorithm;

4. Merging all subclusters to get final clusters by designing carefully the merging condition. We shall detailedly illustrate the procedure in the following sections.

Retrieving core points with lower EPS

In order to obtain core points, we predetermine EPS by the existing procedure (Yue *et al.*, 2004) in any dataset D with their distance space S which consists of all different distances of arbitrary two point D; thus the EPS is the top $(\delta^2/(4c)\,C_n^2$ minimal distances in S, and let the set of all these distances be S_1; c is the number of clusters in D; n is number of points and δ is determined by a drift speed (Nakamura and Kehtarnavaz, 1998). After ordering all points in D, we retrieve points one by one according to their presence frequency in S_1 till no core point is found. However, considering that there are density-skewed clusters, i.e., the highest density of some clusters may only attain the density of border points of some other clusters, as shown in Fig.1, so that we must take lower EPS than the one of $[\delta^2/(4c)\,C_n^2]$ to ensure that a core point in any cluster can be found at least, which is necessary for finding a cluster in DBSCAN. In general, core points are located in the dense area in D instead of in the overlapped area of different clusters. It is clear that all clustering prototypes must be contained in core points. Intuitively, core points act as "skeleton structure" of each cluster because all core points are closer to the "interior" of each cluster than other points. When the core points are determined, then the belongingness of other points can easily be determined. For example, we can arrange any of them by the belongingness of the nearest core point from it.

Fig.2 shows intuitively "skeleton structure" consisting of core points in two clusters where those red points and connected line show the course of forming two arbitrary-shaped clusters. In the following, we let ρ_j be the set of core points in the neighborhood of any core point x_j with the *EPS* and (ρ_j) being the set of ρ_j, j=1,2,…,J.

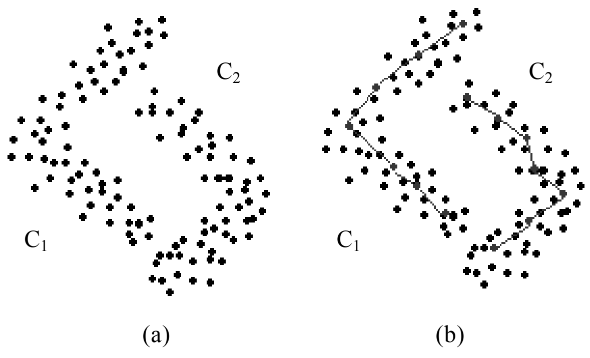

Fig.2 Function of core points of two clusters
(a) Source dataset; (b) Skeleton structure

Determine radii of core points by density

In order to distinguish different points with different densities, we redefine the neighborhood radius of the jth point x_j, j=1, 2, …, J in D instead of *EPS* by the point number in its neighborhood, obtaining

$$d_j = RD_j \cdot EPS,\ RD_j = |\rho_j| / \ddot{\rho},\ j = 1,2,...,J, \quad (1)$$

where RD_j is a relative density coefficient and $\ddot{\rho}$ is average value of $|\rho_j|$ for all $\rho_j \in \{\rho_j\}$. Clearly, when $\rho_j| > \ddot{\rho}$, $RD_j > 1$, it means this point locates in the larger density area in D. Inversely, when $|\rho_j| < \ddot{\rho}$, $RD_j < 1$, it means this point locates in the sparse area in D. Therefore, different points get different radii due to their different densities in D. This procedure means that the effects of the points with higher densities are enhanced and the points with lower densities are masked, and is consistent with the demand of classifying all core points as well. We renew constructing each $\rho_j \in \{\rho_j\}$ with d_j, j=1, 2, ..., J and turn to the next subsection.

Greedy algorithm in GDBSCAN

Now we use the core points of number as small as possible such that each point is contained in one member of $\{\rho_j\}$ at least. The condition aims at reducing the repeated search and waste of I/O memory in DBSCAN, and can be characterized by the following two conditions that

(1) find a group of ρ_j, j=1, 2, …, J with the least number of $\{\rho_j\}$ in D, and

(2) each core point is contained in at least one member of $\{\rho_j\}$.

Having gotten ρ_j, j=1, 2, …, J, the arbitrary-shaped clusters can be constructed by the linkage of neighborhood of these core points. Instead of the clustering procedure by R*-tree in DBSCAN which is time-consuming and repeated search, we use the classical Greedy algorithm for this purpose. It can be formulated by finding a group of finite members in $\{\rho_j\}$ with least number subject to overspread all the points in D, the group of members is called formally covers in this paper. We calculate the average sample number of the group of covers denoted by $\{\rho_j \mid j = 1,2,\ldots,m\}$,

$$mean_\rho = \frac{minpts_1^* + minpt_2^* + \ldots + minpt_m^*}{d_1 + d_2 + \ldots + d_m}, \quad (2)$$

where $mean_\rho$ is the average sample number in all covers; $minpt_k^*$ is the sample number in neighborhood ρ_k with d_k. The smaller m is, the larger is $mean_\rho$. Thus in order to approach those points with larger densities, we must find the smallest m. Namely, the idea is to find the minimal number of covers that overspread all samples, which is a classical cluster vertex cover (VC) problem (Thshihiro, 2000). Although this is an NP-hard problem, thus there are several Greedy algorithms that can effectively solve it (Skieyca, 1990).

The original formulation of the VC problem is

$$\min\{\sum_{j\in\Sigma} w_j \boldsymbol{s}_j \,\Big|\, \boldsymbol{A}s \geq \boldsymbol{b}, s \in \{0,1\}^\Sigma\}, \quad (3)$$

where Σ is the set of covers; and $\boldsymbol{s}_j$ is the characteristic vector; w_j is cost associated with Σ which is always 1.0 in our algorithm; $\boldsymbol{A}$ is 0-1 matrix and $\boldsymbol{b}$=**1**. The solution of the VC problem yields $\widetilde{c}$ points and let them be related one-to-one to $\widetilde{c}$ subclusters consisting of the points inside ρ_k, $k = 1,2,\ldots,\widetilde{c}$ each. Greedy algorithm is shown in Table 1.

Table 1 Greedy algorithm

```
Label the covers in $D$ with $\rho'$, for $j$=1, 2, …, $J$.
Mark $\rho'$ as un-visited. Construct
$\{\rho_j\}$ by $d_i$, $i$ = 1,2,…,$n$.
Initialize the characteristic vector $\boldsymbol{s}=\{0\}^k$,
Let $\{\rho'_{\text{anchor}}\}$ be maximal.
$S\{anchor\}$=1,
Mark $\rho'_{\text{anchor}}$ as visited.
While existing $\Phi$ is a set of un-visited covers that have
        the joint element with $\rho'_{\text{anchor}}$,
    FOR all elements in $\Phi$,
      IF $\rho'_{\text{proneer}} \in \Phi$ and $\rho'_{\text{anchor}} \cap \rho'_{\text{vector}}$ is minimal;
        $S[proneer]$=1;
        Mark $\rho'_{\text{proneer}}$ as visited;
        Replace the index of $\rho'_{\text{anchor}}$ as $\rho'_{\text{proneer}}$;
      END IF
    END FOR
END WHILE
```

Merging condition among subclusters

After the above steps, we obtain the least number of covers $\rho_1, \rho_2, \ldots, \rho_{\tilde{c}}$ to overspread all the samples. It is clear that every cover must be a part of a cluster; now the problem is which of them come from same cluster and which come from different clusters; in other words, which pair of them should (and which pair should not) be merged in the garden? As a fundamental principle, if two sets of aggregating points come from the same cluster, their intersection is no empty set and the density of points in their intersection must match the densities of two sets by themselves. On the other hand, we notice that the density distribution in D can be assumed as normal distribution with the mean of sample densities μ and variance δ being defined by

$$f(\mu,\delta) = \frac{1}{\sqrt{2\pi}\delta}\exp\{-\frac{(x-\mu)^2}{2\delta^2}\} \quad (4)$$

and shown in Fig.1. There are turning points at $x=\mu \pm\delta$, and $f(\mu, \delta)$ is a convex function when $x\in(\mu-\delta, \mu+\delta)$, otherwise is concave function so that the value of the function is rapidly decreasing. According to the notation, the merging condition is presented as follows. Let two sets of aggregating points move their positions until they intersect and finally their centers overlap. If the overlapping centers appear then two sets will be merged into a cluster without question, yet a puzzling problem appears in the case of intersecting. On what level two sets of aggregating points should intersect each other is just regarded as the boundary line for partitioning or merging, but this problem has no acceptable benchmark (Halkidi *et al*., 2002). Notice that if two sets of aggregating points come from the same cluster, then their intersection will not be an empty set and the density of their intersection must match themselves. Thus we first find the point with the largest density in the intersection (e.g., point P in Fig.3a in a 2-dimensional dataset); and find points A and B which have respectively the largest density shown by AC and BD in the two sets. Since the cluster distribution is assumed to be normal distribution in probability, then A, B, C, D are in the same plane. Let ζ be a crossing point of lines AD and CB, and compare the height of P with that of ζ. If the former is larger then two sets are merged, or else are partitioned. This can be shown in Fig.3b and 3c. Finally, the procedure of GDBDCAN based on the above several phases is shown in Table 2.

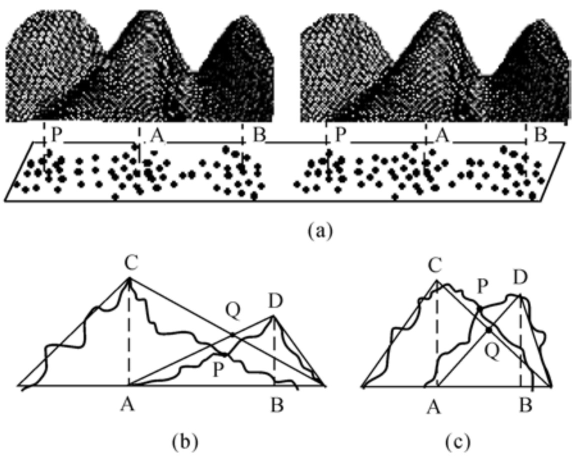

Fig.3 Illustration of merging condition
(a) Projection of density function; (b) Larger overlap density; (c) Smaller overlap density

Table 2 GDBSCAN algorithm

```
FOR Every neighborhood of core point $\rho_i$ in $\{\rho_i\}$.
    Find all subclusters or covers.
        Call the Greedy algorithm procedure for finding
            least number of covers.
        Call the merging condition to form final clusters.
END FOR
```

TIME COST OF GDBSCAN

The time cost of GDBSCAN algorithm includes mainly three parts: searching core points by EPS equation and calculating the radii of each core point; implementing Greedy algorithm; implementing merge condition. In the last two parts, the first part is very low. Greedy algorithm needs only finite steps when scanning the dataset, so its time cost is far less than that of $O(n\log n)$ in DBSCAN. On the other hand, the time cost of merging performance is far less than that of the $O(n\log n)$ since the subcluster number is least in Greedy algorithm. Furthermore, if the time-consumed in deleting the outlier is disregarded, we can observe that its time cost is far less than that of $O(n\log n)$ as well. In total, the time cost of GDBSCAN is less than that of DBSCAN.

EXPERIMENTS

Distinguishing density-skewed clusters

Let us use GDBSCAN for the example shown in Fig.1. We use lower EPS in order that a core point at least of each cluster can be found. According to the new merging condition, we can easily distinguish them intuitively because there exists almost equal density difference between C_1 and C_2 or C_1 and C_3 as that between C_4 and C_5. Therefore, all five clusters can be found correctly. This means that we can use a single EPS to distinguish them.

Test of time complexity and space complexity

This experiment is used to check the time cost and effectiveness of GDBSCAN. The source dataset is similar to the five clusters aforementioned in Fig. 1 yet exist in 8-dimensional space with 10000, 20000 and 30000 samples respectively, in which there exist not only density-skewed clusters but also different ratios of noises. They contain respectively 11, 22 and 30 clusters. Table 3 gives the clustering results from GDBSCAN and DBSCAN algorithm showing that DBSCAN cannot achieve usually correct results. On the contrary, satisfactory clustering result is obtained with GDBSCAN algorithm at less time cost and higher accuracy than GDBSCAN. This experiment was performed on PC-586 by C++ program.

Table 3 Clustering quality comparison: DBSCAN vs GDBSCAN

Name	GDBSCAN/DBSCAN		
Noise ratio	13%	23%	33%
Run-time (sec.)	13.4/78.3	21.7/106.5	43.36/178.5
Missing number	230/334	270/1022	397/2089
Maximal memory (M)	1.3/2.9	3.5/7.8	4.9/8.2

A robotic application

Robotic navigation (See web address) includes sensing scene interpretation and collision avoidance of stationary and moving points in the robot's neighborhood. A robot is often expected to navigate in unknown environment and hence must interpret and understand the environment from its sensor readings, when the robot employs range sensors for probing the environment. Fig.4a shows an environment with 5 points and a circular in shape navigating robot and Fig.4b is its point cloud representation obtained by projecting raw sensor data onto a global reference frame. Operations such as extracting the endpoints and centers of the points from the point cloud representation becomes mandatory in situations requiring point tracking, motion prediction and maneuvering past moving points. They also become essential for classifying points as stationary or moving from the characteristics of the point cloud cluster that represents the points. Evidently the nature of the point cloud represents points to a clustering based solution for extracting the point centers and its other attributes such as the border set. Since sensor data is in general error prone and outlier ridden due to various grounds, the need for robust clustering algorithm appears inevitable, so GDBSCAN algorithm is used in this paper. DBSCAN and PCM (Krishnapuram and Keller, 1993) which are often used for clustering in a dataset with outliers are employed simultaneously. The corresponding results are shown in the second column and third column. The calculation course and simulation results are as follows. Simulations are run on the sensory data obtained by navigating a robot in the Fig.4c environment containing 4 stationary points. Two transient points initially at positions 'a' and 'b' marked with arrows start moving away from the robot. Range data corresponding to these two transient points appear as outliers at location 'a' and 'b' in the point cloud representation

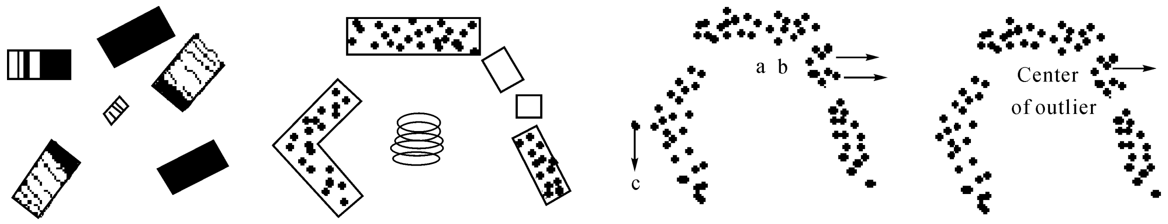

Fig.4 Simulation course of robotic model

Table 4 Comparison of maximal, minimal and average error of GDBCCAN , DBSCAN and PCM

Out.	GDBCCAN				DBSCAN				PCM			
	Min Err.	Max Err.	Ave. Err.	BD%	Min Err.	Max Err.	Ave. Err.	BD%	Min Err.	Max Err.	Ave. Err.	BD%
0	0.705	3.010	1.502	0%	1.232	3.186	1.576	0%	1.116	3.227	1.132	0%
15	9.13	10.17	10.15	35%	10.35	18.05	11.12	0%	10.10	28.97	18.86	8%
30	10.98	52.22	22.23	56%	12.65	53.12	21.69	14%	12.20	45.67	33.54	19%
45	15.01	53.54	24.67	46%	14.88	53.47	36.77	27%	15.17	55.67	30.98	52%
60	16.98	56.97	33.45	57%	20.55	54.33	37.82	54%	19.88	54.12	39.17	57%

shown in Fig.4d and can hinder accurate extraction of the points' features. Another source of noise is sensor misreading at the corner formed at the meeting point of the two points marked as 'c' in Fig.4d. At the corners, due to the phenomenon of multiple reflections, the range readings at times are higher than the usual value and appear as outliers at location 'c' in Fig.4. Thus the simulation environment of Fig.4 has three sources of outliers and DBSCAN algorithm is run over varying number of outliers. The results are tabulated for 5, 10 and 15 and 20 outliers at each of the three locations. The number of data points for each of the points was limited at 40. A number of runs were performed for each algorithm with random initializations for a given number of outliers. The minimum, maximum and average error over these runs measured through the square norm distance for a given number of outliers are reported in 3 columns for each version of DBSCAN and PCM in Table 4. The breakdown percentage (BP) (Krishnapuram and Keller, 1998) is in the fourth column. By breakdown we mean informally that the estimate of the prototype during that run has become completely unreliable. The table suggests that while the minimum error for the DBSCAN algorithm is marginally ahead of the other two algorithm versions; the percentage breakdown is far less for the GDBSCAN in the paper when compared with the standard version for 5 and 10 numbers of outliers at each location. Thereby the average error by GDBSCAN is lower than that of DBSCAN. However as the number of outliers increases, the fidelity of all the three versions decreases sharply and makes them unreliable. It is to be noted that the outliers are placed in such a position the algorithms can be tricked to form an erroneous cluster's center settling right in the middle between the two locations of outliers. This phenomenon is shown in Fig.4e with the erroneous cluster shown by a line and its center indicated by an arrow. The propensity for the algorithms to fall into this trap increases as the number of outliers at that location increases that to serve as attractor or sink for one of the four prototypes to be estimated. Ones can judge that the high average square error increase in noise of all these algorithms may be due to this factor.

CONCLUSION

In this paper, although GDBSCAN is mainly used in 2-dimensional space, it is easy to be implemented in higher 2−10 dimensional space as well. However, it has the same problem as DBSCAN, namely, they are difficult to be implemented in the above 10-dimensional space (Dash *et al*., 2001; Han, 2001)

The clustering method is of two types at present: one handles directly the data in the dataset, another indexes data space by B-tree, R*-tree, and so on. The latter has less time cost but more space cost than the former. We do not evaluate which of them takes first place since this problem should be decided by users through the resource that they possess and the clustering task they face.

GDBSCAN is more efficient since Greedy algorithm execute a scan in only finite steps. A robotic navigation and two artificial experiments have been verified from several viewpoints. How can we identify whether a dataset has density-skewed clusters? Our research showed that when there exist density-skewed clusters, GDBSCAN gives lower minimum of clustering validity index, and we think that the conclusion is inevitable. However, there do not exist a clustering method that can do everything, GDBSCAN not excepted. In fact, some assumptions in this paper do not hold for arbitrary circumstances. Establishing an adaptive and less assumptions clustering algorithm would be promising work for our future concern.

References

Bechmann, N., Kriegel, H.P., Schneider, R., Seeger, B., 1990. The R*-tree: An Efficient and Robust Access Method For Points and Rectangles. Proc. ACM SIGMOD Int. Conf. Alt. City, NJ, p.322-331.

Dash, M., Liu, H., Xu, X., 2001. '1+1>2': Merging Distance and Density Based Clustering. 7th Int. Conf. DASFAA' 01, HK, p.118-202.

Ester, M., Kriegel, H.P., Sander, H., Xu, X., 1996. A Density-Based Algorithm for Discovering Clusters in Large

Spatial Datasets with Noise. Proc. 2nd Int. Conf. KDDD. Portland, Oregon, p.1232-1239.

Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2002. Clustering validity checking methods: Part II. *SIGMOD Record*, **31**(4):51-62.

Han, J., 2001. Data Mining. Morgan Kaufmann Publishers, USA, p.242-266.

Krishnapuram, R., Keller, J.M., 1993. A possibilistic approach to clustering. *IEEE Trans. Fuzzy Syst.*, **1**(2): 98-106.

Krishnapuram, R., Keller, J.M., 1998. An unified views: fuzzy robust clustering. *IEEE Trans. Fuzzy Syst.*, **5**(2): 270-293.

Lozano, S., Dobado, D., Larraneta, J., 2002. Modified fuzzy *c*-means algorithm for cellular manufacturing. *Fuzzy Sets Syst.*, **126**:23-32.

Nakamura, E., Kehtarnavaz, N., 1998. Determining number of clusters and prototype locations via multi-scale clustering. *Pattern Recognition Letters*, **19**(3):1265-1283.

Skieyca, S., 1990. Minimum Vertex Cover, Implementing Discrete Mathematics: Combinatory and Graph Theory Wit Mathematic. Addison-Wesley, MA, p.234-245.

Thshihiro, F., 2000. Approximation algorithms for submodular set cover with applications. *IEICE Trans. Inf. & Syst.*, **18**(3):156-166.

Yue, S.H., Li, P., Zhou, S.G., Gu, Y.K., 2004. Using statistics: DBSCAN revisited I. http://www.cise.zju.edu.cn/iipc/shyue.