UDC 004.421
Original Scientific Paper

PNMBG: Point Neighborhood Merging with Border Grids

Renxia Wan *wrx1022@mail.dhu.edu.cn*
College of Information Science and Technology
Donghua University, Shanghai 201620, P.R. China

Jingchao Chen *jc_chen@dhu.edu.cn*
College of Information Science and Technology
Donghua University, Shanghai 201620, P.R. China

Lixin Wang *wlx@mail.dhu.edu.cn*
College of Information Science and Technology
Donghua University, Shanghai 201620, P.R. China

Xiaoke Su *suxiaoke@mail.dhu.edu.cn*
College of Information Science and Technology
Donghua University, Shanghai 201620, P.R. China

Abstract

The special clustering algorithm is attractive for the task of grouping arbitrary shaped database into several proper classes. Up to now, a wide variety of clustering algorithms designed for this task have been proposed, the majority of these algorithms is density-based. But the effectivity and efficiency still is the great challenges for these algorithms as far as the clustering quality of such task is concerned. In this paper, we propose an arbitrary shaped clustering method with border grids (PNMBG), PNMBG is a crisp partition method. It groups objects to point neighborhoods firstly, and then iteratively merges these point neighborhoods into clusters via grids, only bordering grids are considered during the merging stage. Experiments show that PNMBG has a good efficiency especially on the database with high dimension. In general, PNMBG outperforms DBSCAN in the term of efficiency and has an almost same effectivity with the later.

Keywords: Clustering, Grid clique, Point neighborhood, Border grids, Merging

1. Introduction

Clustering is the unsupervised classification of objects into homogeneous meaningful groups called clusters such that objects in the same cluster are more similar to each other than objects in different clusters according to some defined criteria [1]. It is useful in a number of tasks, for example, by partitioning objects into clusters, interesting object groups may be discovered, such as the groups of clients in a banking database having a heavy investment in real estate.

Cluster analysis has become the subject of active research in several fields such as statistics, pattern recognition, machine learning and data mining. Up to now, a large number of clustering algorithms have been proposed, and also received a lot of attention in the last few years (c.f. section 2). In these algorithms, discovery of arbitrary shaped clusters is often to be a real obstacle for their applications. [2][3] imply that some typical clustering algorithms such as k-means, CURE, ClARANS and so on will get some poor results if there are some nonconvex shape data sets or some ball-shaped data sets of significantly differing sizes in the database. To get the arbitrary shaped clusters, algorithms based on density are designed (DBSCAN is a typical one), but these algorithms also face challenges from the efficiency and the effectivity.

In this paper, we present the new clustering algorithm PNMBG. The new algorithm requires only one input parameter, can discover arbitrary size and shaped clusters, is efficient even for large data sets especially data with high dimension. The rest of this paper is organized as follows: In section 2, we survey related work. In section 3, we give the definition of cluster grids. The new algorithm PNMBG is presented in section 4. The experimental results are reported to illustrate the new algorithm in section 5. Finally, we draw our conclusions in section 6.

2. Related work

There are many clustering algorithms proposed, these algorithms may be classified into partitioning, hierarchical, density and grid based methods [4]. Partitioning methods determine a partition of the points into clusters, such that the points in a cluster are more similar to each other than to points in different clusters. They start with some arbitrary initial clusters and iteratively reallocate points to clusters until a stopping criterion is met. They tend to find clusters with hyperspherical shapes. Examples of partitioning algorithms include k-means [5], k-prototypes [1], PAM [6], and EM [7]. Hierarchical clustering methods can be either agglomerative or divisive. An agglomerative method starts with each point as a separate cluster, and successively performs merging until a stopping criterion is met. A divisive method begins with all points in a single cluster and performs splitting until a stopping criterion is met. The result of a hierarchical clustering method is a tree of clusters called a dendogram. Examples of hierarchical clustering methods include BIRCH [8] and CURE [9]. Density-based clustering methods try to find clusters based on the density of points in regions. Dense regions that are reachable from each other are merged to formed clusters. Density-based clustering methods excel at finding clusters of arbitrary shapes. Examples of density-based clustering methods include DBSCAN [3] and DENCLUE [10]. Grid-based clustering methods quantize the clustering space into a finite number of cells and then perform the required operations on the quantized space. Cells containing more than a certain number of points are considered to be dense. Contiguous dense cells are connected to form clusters. Examples of grid-based clustering methods include STING [11] and CLIQUE [12].

In our discussion, we will focus our interests on clustering algorithms which are reported to work reasonably on arbitrary shaped databases. ClARANS is introduced in [13], which is an improved k-medoids method. An experimental evaluation indicates that CLARANS runs efficiently on database of thousands of objects. [3] points out that CLARANS will get a poor clustering result if there are some nonconvex shape data sets or some ball-shaped data sets of significantly differing sizes in the database. Furthermore, CLARANS has a $O(n^2)$ computational complexity, where n is the number of objects. Thus for large databases, CLARANS is prohibitive due to its run time. The classical density-based spatial clustering algorithm is DBSCAN [3]. To discover clusters, DBSCAN checks the ε-neighborhood of each point in the database. If the ε-neighborhood of a point P contains more than MinPts, a new cluster with P as a core object is created. Objects in the ε-neighborhood of P are then added to this new cluster. Then, DBSCAN iteratively aggregates points that are directly density-reachable from these core points. Merging clusters will happened when a core point belongs to these clusters. The process terminates when no new point can be added to any cluster. For spatial database, the average computational complexity of DBSCAN is $O(n \log n)$, otherwise, the average computational complexity is $O(n^2)$. OPTICS [14] is an extension of DBSCAN. Like DBSCAN, OPTICS requires the input of the two parameters, ε and MinPts. However, instead of producing the clustering result for one pair of parameter values, OPTICS produces an ordering of the data points such that clustering result for any lower value of ε and similar value of MinPts can be visualized and computed easily. Due to the structural equivalence, OPTICS has the same computational complexity of DBSCAN. DENCLUE [10] is a clustering method based on a set of density distribution functions. Its basic idea is to model the overall point density analytically as the sum of influence functions of the data points, clusters can be identified by determining density-attractors and clusters of arbitrary shape can be easily described by a simple equation based on the overall density function.

We summarize other algorithms that are reported to be feasible on clustering arbitrary shapes on some special data sets as follows:

[15] proposes an algorithm for clustering arbitrary shapes in data stream with an index structure CDS-tree, and data skew factor is used to adjust automatically the partition granularity according to the change of data streams. [16] presents a density-based cluster method which is designed to discover clusters in an evolving data stream. During the clustering process, the “dense” micro-cluster (named core-micro-cluster) is introduced to summarize the clusters with arbitrary shape, while the potential core-micro-cluster and outlier micro-cluster structures are proposed to maintain and distinguish the potential clusters and outliers. A pruning strategy is designed to guarantee the precision of the weights of the micro-clusters with limited memory. An approach for distributed density-based clustering is presented in [17], the approach is based on two main concepts: the extension of local models created by DBSCAN at each node of the system and the aggregation of these local models by using tree based topologies to construct global models. [18] proposes an algorithm called as D-Stream. The algorithm is also a density-based approach, it uses an online component which maps each input data record into a grid and an offline component which computes the grid density and clusters the grids based on the density. A density decaying technique is adopted to capture the dynamic changes of a data stream. [19] introduces the notion of local scaling in density based clustering, which determines the density threshold based on the local statistics of the data. The local maxima of density are discovered using a k-nearest-neighbor density estimation and used as centers of potential clusters. Each cluster is grown until the density falls below a pre-specified ratio of the center point’s density. [20] proposes a fully distributed clustering algorithm, called PENS (peer density-based clustering). PENS is hierarchical cluster assembly, which enables peers to collaborate in forming a global clustering model without requiring a central control or message flooding. In [21], a similarity measure based on spatial overlapping relation is proposed, which calculates the similarity between a pair of data points by using the mutual overlapping relation between them in a multi-dimensional space, and a spatial overlapping based hierarchical clustering method is also developed and implemented to justify the effectiveness of the proposed similarity measure.

3. Cluster grids

In this section, we will define some conceptions about cluster grids which are used in the new proposed algorithm.

We assume that the input data has *d* dimensions, and each input data record is defined within the space $S = S_1 \times S_2 \times \ldots \times S_d$, where S_i is the definition space for the *i*th dimension.

We partition the *d*-dimensional space *S* into grids. Suppose for each dimension, its space S_i , $\mathrm{i} = 1, \cdots, \mathrm{d}$ is divided into p_i partitions as $S_i = S_{i,1} \cup S_{i,2} \cup \ldots \cup S_{i,p_i}$, then the data space *S* is partitioned into $N = \prod_{i=1}^{d} p_i$ grids. For a grid *g* that is composed of $S_{1,j_1} \times S_{2,j_2} \times \ldots \times S_{d,j_d}$, $\mathrm{j}_i = 1, \cdots, \mathrm{p_i}$, we denote it as $g = (j_1, \mathrm{j}_2 \cdots, \mathrm{j_d})$.

A data record $x = (x_1, \mathrm{x}_2, \cdots, \mathrm{x_d})$ can be mapped to a grid *g*(*x*) as follows: $g(x) = (j_1, \mathrm{j}_2 \cdots, \mathrm{j_d})$ where $x_i \in S_{i,j_i}$

Definition 1: (neighboring grids) Consider two grids $g_1 = (j_1^1, j_2^1, \cdots, j_d^1)$ and $g_2 = (j_1^2, j_2^2, \cdots, j_d^2)$, if there exists *k*, $1 \le k \le d$, such that:

1) $| j_k^1 - j_k^2 | = 1$;

2) $| j_l^1 - j_l^2 | = 1$ or $| j_l^1 - j_l^2 | = 0$, $1 \le l \le d$ and $l \ne k$.

Then g_1 and g_2 are called neighboring grids, denoted as $g_1 \sim g_2$.

In order to facilitate our discussion, we generalize the unit metrics in all dimensions to uniform symbol as *e* while ignoring their differences.

Definition 2: (grid clique) A set of grids $G=(g_1,\ldots,g_m)$ is a grid clique if for any two grids $g_i, g_j \in G$, there exist a sequence of grids $g_{k_1},\cdots,g_{k_l}$ such that $g_{k_1}=g_i, g_{k_l}=g_j$, and $g_{k_1}\sim g_{k_2}, g_{k_2}\sim g_{k_3},\ldots, g_{k_{l-1}}\sim g_{k_l}$.

Definition 3: (cluster grid cover) Consider an object cluster *C* and a grid clique *G*, if each grid of *G* at least has one object of *C*, and each object of C is in a grid of *G*, then *G* is called the grid cover of the cluster *C*, and every grid in *G* is called cluster grid.

Definition 4: (internal and border grids) Consider a cluster *C* and its grid cover gCov, For a grid *g* in *G*, if there exists a neighboring grid of *g* that is not in cluster *C*, then *g* is called a border grid of *C*. Otherwise *g* is called an internal grid of *C*.

Figure 2 shows the internal grids and the border grids of cluster C1 and C2 which are described in figure 1.

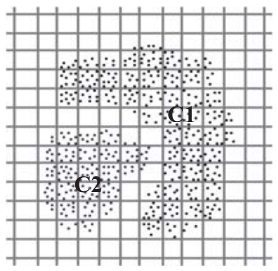


Figure 1: two clusters with arbitrary shapes

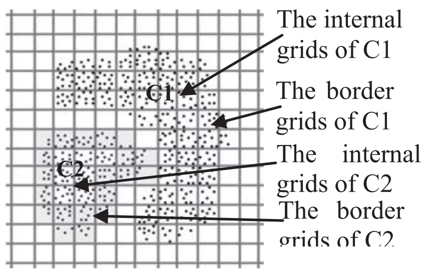


Figure 2: the internal grids and the border grids of cluster

4. The proposed algorithm

In this section, we present the algorithm PNMBG (Point Neighborhood Merging with Border Grids) based on the previous discussions of cluster grids.

The basic idea of the algorithm is to partition the object set into disjoint object neighborhoods first, and then iteratively merges these object neighborhoods into clusters via grids.

At the beginning of the clustering process, we run a *normalization process* in order to get a interval with [0,1] on every dimension.

```
Algorithm: PNMBG (Object set: S, Threshold of D: $\delta$)
-------------------------------------------------------
 1: partition the normalized object space into grids;
 2: for each x in S do;
 3:     let $NH_x = \{y \mid distance(x,y) \le \delta, y \in S\}$;
 4:     ClusterSet= ClusterSet + $NH_x$;
 5:     S=S-$NH_x$;
 6: end for
 7: repeat
 8:     numClu=| ClusterSet |;
 9:     ClusterSet = Merging(ClusterSet, $\delta/2$);
10: until | ClusterSet |= =numClu
11: return ClusterSet
```

Figure 3: PNMBG: point neighborhood merging with grids

NH_x is the subset of *S,* contained in the $\delta -$neighborhood of object *x*, the for loop on *S* (namely step 2 to step 6) is point-neighborhood partition process, **Merging** is used to combine these subsets into a bigger cluster. The **Merging** process is performed repeatedly until *ClusterSet* cannot be merged anymore. The final *ClusterSet* (in step 11) is the set of clusters we require. Algorithm **Merging** is shown in Figure 4.

```
Algorithm: Merging(Set of Clusters: ClusterSet, Threshold of D: $\delta'$ )
--------------------------------------------------
1.let $bgSet=\{bgC_x | \forall C_x \in ClusterSet$, $bgCx$ is the set of border grids of $C_x$ };
2. for each $bgC_x$ in $bgSet$ do
3.     for each $g_x$ in $bgC_x$
4.     let $BSg_x=\{g_y| distance(g_x,g_y)\le 2\sqrt{d}e+\delta'$, $g_y\in bgC_y$, $bgC_y\in bgSet$ and $bgC_y\neq bgC_x$ };
5.         for each $g_y$ in $BSg_x$
6.             if $D(g_x, g_y)\le\delta'$ then
7.                 $C_x=C_x\cup C_y$;
8.                 $bgC_x= bgC_x\cup bgC_y$ - {g| g is internal grid in the new $C_x$, belonged to $bgC_x$ or $bgC_y$ };
9.                 $BSg_x=BSg_x - bgC_y$;
10.                $ClusterSet= ClusterSet - \{C_y\}$;
11.            end if
12.        end for
13.    end for
14. end for
15. return ClusterSet;
```

Figure 4: Algorithm Merging

BSg_x is set of the border grids which have a no more than $2\sqrt{d}e+\delta'$ distance from the grid g_x, where e is the unit metric value of grid in our discussion, *d* is the number of dimensions, we let the grid unit metric *e* satisfy $e\le\frac{\delta}{k\sqrt{d}}$ where *k* is a constant positive integer, namely, $k=1,2,\cdots$, in our discussion, we let *k* be 1and set δ' to be an experiential value $\delta/2$.

PNMBG only retrieve those objects in border grids of grid cliques on point neighborhoods or interim clusters. **Merging** will happen iteratively when the distance measure value of some border grids coming from different object sets (point neighborhoods or interim clusters) is not more than $\delta/2$, and larger interim clusters will be produced till the final clusters come into being. The border grids of every new interim cluster is the union of border grids of its previous interim clusters which have been merged into the new one, it does not include the grids that being border grids before **Merging** while being internal grids after **Merging**.

Since the point-neighborhood partition process (step 2 to step 6 in PNMBG) needs to scan the object space *S* only once, it will take a $O(n)$ computational cost. Due to the dimensional unit metric *e* satisfying: $e\le\frac{\delta}{k\sqrt{d}}$, we can affirm directly that objects in such grid belong to one cluster. Then, in such situation, we just count objects, it will take a $O(n)$ computational consumption. Since merging only happens between the two object sets (point neighborhoods or interim clusters) which have a no more than $2\sqrt{d}e+\frac{\delta}{2}$ distance measure value between them. Every object set resides in a finite regional grid clique whose border grids are finally applied to merge. PNMBG only retrieves the objects in the border grids of the grid cover on these interim clusters with dictionary sort according by the subscript order. **Merging** will happen iteratively when the compatible relation measure value of the border grids of some interim clusters is not more than $\delta/2$, and larger interim clusters will be produced till the final clusters come into being. The border grids of every new interim cluster is the union of border grids of its previous interim clusters which have been merged into the new one, it does not include the grids that being border grids before **Merging** while being internal grids after **Merging**.

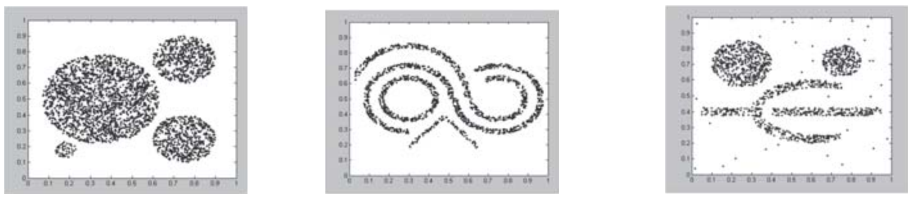
Only those border grids which have a no more than $2\sqrt{d}e+\frac{\delta}{2}$ distance measure value from the considering one are concerned in the **Merging** step, and every border grid has a no more than $\left\lceil\frac{\delta}{\sqrt{d}e}\right\rceil\times\left\lceil\frac{\delta}{\sqrt{d}e}\right\rceil$ grids in it, where $\left\lceil\frac{\delta}{\sqrt{d}e}\right\rceil$ is the round up number of $\frac{\delta}{\sqrt{d}e}$, thus retrieving all these grid cliques for **Merging** will take a $O(n\log n)$ computational cost. Thus, the average computational complexity of PNMBG with grids is $O(n\log n)$.

Furthermore, due to the objects in internal grids are not computed in the merging step, merging large interim clusters under this condition will cut down a dramatic computational consumption.

5. Performance evaluation

In this section, we evaluate the performance of PNMBG. We compare it with the performance of DBSCAN because this is the first arbitrary shape clustering algorithm and original density based clustering algorithm. All of our experiments are performed on a PC with 2.2GHz CPU and 512MB memory, running on Windows XP professional. We have implemented our algorithm in VC++6.0, which is hooked up with Matlab 7.0 to visualize the results.

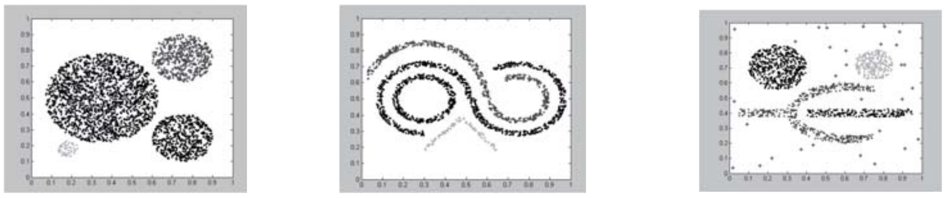
Three synthetic sample databases SD1, SD2, SD3, which are depicted in figure 5, were used in the effectivity (accuracy) test. SD1 has four ball-shaped clusters of significantly differing sizes. SD2 contains four clusters of nonconvex shape. SD3 has four clusters of different shape and size with additional noise.



SD1 SD2 SD3

Figure 5: sample databases

To get ideal cluster results, we set different parameters of δ for SD1, SD2 and SD3. To show the results of both clustering algorithms, we visualize each cluster by a different color. In SD3, we use bright red asterisks to figure the outliers which are singletons or members of the clusters whose size is not more than 2 after the final merging has been done. The cluster results are presented in the figure 6.



SD1 ($\delta=0.035$) **SD2** ($\delta=0.046$) **SD3** ($\delta=0.039$)

Figure 6: clusters discovered by PNMBG

From figure 6, we can see that PNMBG can detect correctly arbitrary shapes in the database. In this term, it has the same performance as DBSCAN [3].

In the efficiency test, a real data set and a synthetic data set were used. The real data set is from the Forest CoverType data set which is obtained from the UCI machine learning repository website (i.e. http://kdd.ics.uci.edu/databases/covertype/covertype.html). This data set contains totally 581012 observations and each observation consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. In this set, there are 7 forest cover type classes. In our testing, we used all the 10 quantitative variables. Because the synthetic datasets can be generated by controlling the number of data points, the dimensionality, and the number of clusters, with different distribution, in a way similar to [22], we produced one high dimensional synthetic data set hSD to test the abilities of these two algorithms in clustering high dimensional data sets. The data set satisfies a series of Gaussian distributions, contains 10k data with 200 dimensions and belongs to 10 different classes. Like [3], we fixed the parameter MinPts of DBSCAN to 4, and adopt cluster purity (c.f. appendix A) as the measure of the clustering quality. The comparison of PNMBG and DBSCAN on these data sets is shown in table 1.

Data sets	Parameter		Run time(sec.)		Cluster purity	
	ε	δ	**PNMBG**	**DBSCAN**	**PNMBG**	**DBSCAN**
Forest CoverType	56.584	79.355	582.164	1421.228	82.4%	79.7%
hSD	0.0154	0.0294	457.769	1055.426	90.7%	69.4%

Table 1: results of PNMBG and DBSCAN (ε is the radius parameter of DBSCAN, δ is threshold parameter of PNMBG)

From this table, we can see that the run time of PNMBG is significantly shorter than that of DBSCAN. And the cluster purity of PNMBG is higher than that of DBSCAN, especially for high dimensional data sets. All the experimental results show that PNMBG has a better clustering quality than DBSCAN.

6. Conclusions

In this paper, we present the algorithm PNMBG which is designed for arbitrary shaped clustering. PNMBG is a crisp partition method. It groups objects to point neighborhoods first, and then classes these point neighborhoods into different clusters by merging method. The application of cluster grids during the merging step significantly reduces the computational cost of clustering. Experiments also show that PNMBG has a good efficiency especially on the database with high dimension. In general, PNMBG outperforms DBSCAN in the terms of effectivity and efficiency.

7. Appendix A: purity

Cluster purity is one of the ways of measuring the quality of a clustering solution. Let there be *k* clusters of the dataset *D* and size of cluster C_j be $|C_j|$. Let$|C_j|_{class=i}$ denote number of items of class *i* assigned to cluster *j*. Purity of this cluster is given by

$$purity(C_j) = \frac{1}{|C_j|}\max_i(|C_j|_{class=i}) \tag{1}$$

The overall purity of a clustering solution could be expressed as a weighted sum of individual cluster purities

$$purity = \sum_{j=1}^{k}\frac{|C_j|}{|D|}purity(C_j) \tag{2}$$

In general, the lager the value of purity is, the better the solution is.

References

[1] Huang, Z.: Extensions to the k-Means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*.1998:283-304.

[2] Halkidi, M., Vazirgiannis, M.: Clustering validity assessment: Finding the optimal partitioning of a data set. Proceedings of ICDM. 2001.

[3] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial database with noise. Proceedings of 2nd *International Conference on Konwledge Discovery and Data Mining (KDD'96)*.1996:226-23

[4] Han, J., Kanber, M., Tung, K.H.: Spatial clustering methods in data mining: A survey. *Geographic Data Mining and Knowledge Discovery*. Taylor and Francis. 2001:1-29.

[5] MacQueen, J: Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*. 1967:281-297.

[6] Kaufma, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. *John Wiley& Sons*. 1990.

[7] Bradley, P.S., Fayyad, U.M., Reina, C.A.: Scaling EM (Expectation Maximization) clustering to large database. *Microsoft Research Report, MSR-TR-98-35*.1998.

[8] Zhang, T., Ramakrishnam, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96)*. 1996: 103-114.

[9] Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. *Proceedings of the ACM SIGMOD conference on Management of Data (SIGMOD'98)*.1998:73-84.

[10] Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large Multimedia Database with noise. *Proceedings of 4th International Conference on Konwledge Discovery and Data Mining (KDD'98)*.1998:58-65.

[11] Wang, W., Yang, J., Muntz, R.: STING: A statistical information grid approach to spatial data mining. *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*.1997:186-195.

[12] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. *Proceedings of the ACM SIGMOD conference on Management of Data (SIGMOD'98)*.1998:94-105

[13] Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*. 1994:144-155

[14] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. *Proceedings of the ACM SIGMOD conference on Management of Data (SIGMOD'99)*.1999:49-60.

[15] Sun, H., Yu, G., Bao, Y., Zhao, F., Wang, D.: CDS-Tree: An effective index for clustering arbitrary shapes in data streams. *Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications(RIDE-SDMA'05)*.2005.

[16] Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. *Proceedings of the 2006 SIAM conference on data mining (SDM'2006)*.2006:326-337.

[17] Le-Khac, N., Aouad, L.M., Kechadi, M.: A new approach for distributed density based clustering on grid platform. *Proceedings of the 24th British national conference on databases (BNCOD'07)*.2007:247-258.

[18] Chen, Y., Tu, L.: Density-based clustering for real-time stream data. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.2007:133-142.

[19] Biçici, E., Yuret, D.: Locally scaled density based clustering. *Proceedings of the 8th international conference on adaptive and natural computing algorithms (ICANNGA'07)*. 2007:739-748.

[20] Li, M., Lee, G., Lee, W., Sivasubramaniam, A.: PENS: an algorithm for density-based clustering in peer-to-peer systems. *Proceedings of the 1st international conference on scalable information systems*.2006

[21] Chen, H., Guo, G., Huang, Y., Huang, T.: A spatial overlapping based similarity measure applied to hierarchical clustering. *Proceedings of the 5th international conference on fuzzy systems and knowledge discovery(FSKD'08)*. 2008: 371-375.

[22] Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. *Proceedings of 29th international conference on very large data bases*. 2003, 81–92.