SCALABLE AND VISUALIZATION-ORIENTED CLUSTERING FOR EXPLORATORY SPATIAL ANALYSIS

J.H.Guan, F.B.Zhu, F.L.Bian

[a] School of Computer, Spatial Information & Digital Engineering Center, Wuhan University, Wuhan, 430079, China-jhguan@wtusm.edu.cn

TS, WG II/6

KEY WORDS: GIS, Analysis, Data Mining, Visualization, Algorithms, Dynamic, Multi-resolution, Spatial.

ABSTRACT:

Clustering can be applied to many fields including data mining, statistical data analysis, pattern recognition, image processing etc. In the past decade, a lot of efficient and effective new clustering algorithms have been proposed, in which famous algorithms contributed from the database community are CLARANS, BIRCH, DBSCAN, CURE, STING, CLIGUE and WaveCluster. All these algorithms try to challenge the problem of handling huge amount of data in large-scale databases. In this paper, we propose a scalable and visualization-oriented clustering algorithm for exploratory spatial analysis (CAESA). The context of our research is 2D spatial data analysis, but the method can be extended to higher dimensional space. Here, "Scalable" means our algorithm can run focus-changing clustering in an efficient way, and "Visualization-oriented" indicates that our algorithm is adaptable to the visualization situation, that is, choosing appropriate clustering granularity automatically according to current visualization resolution. Experimental results show that our algorithm is effective and efficient.

1. INTRODUCTION

Clustering, which is the task of grouping the data of a database into meaningful subclasses in such a way that minimizes the intra-differences and maximizes the inter-differences of these subclasses, is one of the most widely studied problems in data mining field. There are a lot of application areas for clustering techniques, such as spatial analysis, pattern recognition, image processing, and other business applications, to name a few. In the past decade, a lot of efficient and effective clustering algorithms have been proposed, in which famous algorithms contributed from the database community include CLARANS, BIRCH, DBSCAN, CURE, STING, CLIGUE and WaveCluster. All these algorithms try to challenge the clustering problem of handling huge amount of data in large-scale databases.

However, current clustering algorithms are designed to cluster a certain dataset in the fashion of once and for all. We can refer to this kind of clustering as global clustering. In reality, take exploratory data analysis for example, the user may first want to see the global view of the processed dataset, then her/his interest may shift to a smaller part of the dataset, and so on. This process implies a series of consecutive clustering operations: first on the whole dataset, then on a smaller part of the dataset, and so on. Certainly, this process can be directed in an inverse way, that is, user's focusing scope shifts from smaller area to larger area. We refer to this kind of clustering operation as focus-changing clustering. In implementation of focus-changing clustering, the naive approach is to cluster the focused data each time from scratch in the fashion of global clustering. Obviously, such approach is time-consuming and low efficient. The better solution is to design a clustering algorithm that carries out focus-changing clustering in an integrated framework.

On the other hand, although visualization has been recognized as an effective tool for exploratory data analysis and some visual clustering approaches were reported in the literature, these researches has focused on proposing new methods of visualizing clustering results so that the users can have a view of the processed dataset's internal structure more concretely and directly. However, seldom concern has been put on the impact of visualization on the clustering process. To make this idea clear, let us take 2-dimensional spatial data clustering for example. In fact, clustering can be seen as a process of data generalization on basis of certain lower data granularity. The lowest clustering granularity of a certain dataset is the individual data objects in the dataset. If we divide the dataset space into rectangular cells of similar size, then a larger clustering granularity is the data objects enclosed in the rectangular cells.

More reasonably, we define clustering granularity as the size of the divided rectangular cell in horizontal or vertical direction, and define *relative clustering granularity* as the ratio of clustering granularity over the scope of focused dataset in the same direction. Visualization of clustering results is also based on clustering granularity. However, visualization effect relies on the resolution of display device. For comparison, we define *relative visualization resolution* as the inversion of the size (taking pixel as measurement unit) of visualization window for clustering results in the same direction as the definition of clustering granularity. Obviously, a reasonable choice is that the relative clustering granularity is close to, but not lower than the relative visualization resolution of visualization window. Otherwise, the clustering results can't be visualized completely, which means we do a lot but only part of its effect is shown up in visualization.

In this paper, we propose a scalable and visualization-oriented clustering algorithm for exploratory spatial analysis (CAESA), in which we adopt the idea of combining DBSCAN and STING. We built a prototype to demonstrate the feasibility of the proposed algorithm. Experimental results indicate that our algorithm is effective and efficient. CAESA needs less region queries than DBSCAN does, and is more flexible than STING.

2. RELATED WORK

In recent years, a number of clustering algorithms for large databases or data warehouses have been proposed. Basically, there are four types of clustering algorithms: partitioning algorithms, hierarchical algorithms, density based algorithms and grid based algorithms. Partitioning algorithms construct a partition of a database *D* of *n* objects into a set of *k* clusters, where *k* is an input parameter for these algorithms. Hierarchical algorithms create a hierarchical decomposition is represented by a dendrogram, a tree that iteratively splits *D* into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of *D*. The dendrogram can either be created from the leaves up to the root (agglomerative approach) or from the root down to the leaves (divisive approach) by merging or dividing clusters at each step.

Ester et al. (1996) developed a clustering algorithm DBSCAN based on a density-based notion of clusters. It is designed to discover clusters of arbitrary shape. The key idea in DBSCAN is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN can effectively handle the noise points (outliers).

CLIQUE clustering algorithm (Agrawal et al., 1998) identifies dense clusters in subspace of maximum dimensionality. It partitions the data space into cells. To approximate the density of the data points, it counts the number of points in each cell. The clusters are unions of connected high-density cells with in a subspace. CLIQUE generates cluster description in the form of DNF expressions.

Sheikholeslami et al. (1998), using multi-resolution property of wavelets, proposed the WaveCluster method which partitions the data space into cells and app lies wavelet transform on them. Furthermore, WaveCluster can detect arbitrary shape clusters at different degrees of detail.

Wang et al. (1997, 1999) proposed a statistical information grid-based method (STING, STING+) for spatial data mining. It divides the spatial area into rectangular cells using a hierarchical structure and stores the statistical parameters of all numerical attributes of objects with in cells. STING uses a multi-resolution to perform cluster analysis, and the quality of STING clustering depends on the granularity of the lowest level of the grid structure. If the granularity is very fine, the cost of processing will increase substantially. However, if the bottom level of the grid structure is too coarse, it may reduce the quality of cluster analysis. Moreover, STING does not consider the spatial relationship between the children and their neighboring cells for construction of a parent cell. As a result, the shapes of the resulting clusters are isothetic, that is, all of the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected. This may lower the quality and accuracy of the clusters despite of the fast processing time of the technique.

However, all algorithms described above have the common drawback that they are all designed to cluster a certain dataset in the fashion of once and for all. Although methods has been proposed to focus on visualizing clustering results so that the users can have a view of the processed dataset's internal structure more concretely and directly, however, seldom concern has been put on the impact of visualization on the clustering process.

Different from the algorithms abovementioned, the proposed CAESA algorithm in this paper is a scalable and visualization-oriented clustering algorithm for exploratory spatial analysis. Here, "scalable" means our algorithm can run focus-changing clustering in an effective and efficient way, and "visualization-oriented" indicates that our algorithm is adaptable to visualization situation, that is, choosing appropriate relative clustering granularity automatically according to current relative visualization resolution.

3. SCALABLE AND VISUALIZATION-ORIENTED CLUSTERING ALGORITHM

3.1 CAESA OVERVIEW

Like STING, CAESA is also a grid-based multi-resolution approach in which the spatial area is divided into rectangular cells. There usually several levels of such rectangular cells corresponding to different levels of resolution, and these cells form a hierarchical structure: each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information associated with the attributes of each grid cell (such as the mean, standard deviation, distribution) is calculated and stored beforehand and is used to answer the user's queries.

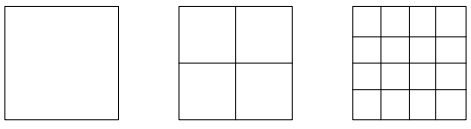

Figure 1: Hierarchical structure

Figure 1 shows a hierarchical structure for CAESA clustering. Statistical parameters of higher-level cells can easily be computed form the parameters of the lower-level cells. For each cell, there are attributed-dependent and attribute-independent parameters. The attribute-independent parameter is:

n—number of objects(points) in this cell

cellNo—number of this cell, it is calculated when the tree structure is establishing

layerNo—layer number of this cell

The attribute-dependent parameters are:

m—mean of all values in this cell, where value is the distance of two objects in this cell

s—standard deviation of all values of the attribute in this cell

d—density of all values in this cell

min—the minimum value of the attribute in this cell

max—the maximum value of the attribute in this cell

distribution—the type of distribution that the attribute value in this cell follows(such as normal, uniform, exponential, etc. NONE is assigned if the distribution type is unknown)

cellLoc—the location of this cell, it record the coordinate information associated to the objects in data set.

relevant—coefficient of this cell relevant to given query

In our algorithm, parameters of higher-level cells can be easily calculated from parameters of lower level cell. Let n_i, m_i, s_i, d_i, min_i, max_i, $cellLoc_i$, $layerNo_i$, and $dist_i$ be parameters of corresponding lower level cells, respectively. The parameters in current cell n_{i-1}, m_{i-1}, s_{i-1}, d_{i-1}, min_{i-1}, max_{i-1}, $cellLoc_{i-1}$, $layerNo_{i-1}$, and $dist_{i-1}$ can be calculated as follows.

$$n_{i-1} = \sum_i n_i \tag{1}$$

$$d_{i-1} = \frac{\sum_i d_i n_i}{n} \tag{2}$$

$$m_{i-1} = \frac{\sum_i m_i n_i}{n} \tag{3}$$

$$s_{i-1} = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) \times n_i}{n_{i-1}} - m_{i-1}^2} \tag{4}$$

$$\min_{i-1} = \min_i(\min_i) \tag{5}$$

$$\max_{i-1} = \max_i(\max_i) \tag{6}$$

$$layerNo_{i-1} = layerNo_i - 1 \tag{7}$$

$$cellLoc_{i-1} = \min_{cellNo_{i-1}} (cellLoc_i) \tag{8}$$

The calculation of $dist_{i-1}$ is much more complicated, here we adopt the calculating method designed by Wang et al.(1997) in their algorithm of STING.

i	1	2	3	4
n_i	80	10	90	60
m_i	18.3	17.8	18.1	18.5
$_{di}$	0.8	0.5	0.7	0.2
s_i	1.8	1.6	2.1	1.9
min_i	2.3	5.8	1.2	4.2
max_i	27.4	54.3	62.8	48.5
$layerNo_i$	5	5	5	5
$cellNo_i$	85	86	87	88
$cellLoc_i$	(80,60)	(90,60)	(90,60)	(90,60)
$dist_i$	NORMAL	NONE	NORMAL	NORAML

Table 1: Parameters of lower cells

According to the formula present above, we can easily to calculate the parameters in current cell:

n_{i-1} = 240

m_{i-1} = 18.254

s_{i-1} = 1.943

d_{i-1} = 0.6

min_{i-1} = 1.2

max_{i-1} = 62.8

$cellLoc_{i-1}$ = (80,60)

$layerNo_{i-1}$ = 4

$dist_{i-1}$ = NORMAL

The advantages of this approach are:

1. It is a query-independent approach since the statistical information stored in each cell represents the summary information of the data in the cell, which is independent of the query.

2. The computational complexity is O(k), where k is the number of cells at the lowest level. Usually, $k<<N$, where N is the number of objects.

3. The cell structure facilitates parallel processing and incremental updating.

3.2 Focus-changing Clustering

Focus-changing clustering means that CAESA can provide user the corresponding information when his interested dataset is changed. Take exploratory data analysis for example, the user may first want to see the global view of the processed dataset, then her/his interest may turn to a smaller part of the dataset to see some details, and so on. To fulfil focus-changing clustering, a simple method is to cluster focused data each time from scratch in the fashion of current clustering algorithm. Obviously, such approach is time-consuming and of low efficient. The better solution is to design a clustering algorithm that carries out focus-changing clustering in an integrated framework. Thus clustering time and I/O cost is reduced, and the clustering flexibility is enhanced.

For example, the user wants to select the maximal regions that have at least 100 houses per unit area and at least 70% of the house prices are above $200,000 and with total area at least 100 units with 90% confidence. We can describe this query using SQL like this:

```
SELECT    REGION
FROM      house-map
WHERE     DENSITY IN (100, ∞)
          AND price RANGE (200000, ∞)
          WITH PERCENT (0.7, 1)
          AND AREA (100, ∞)
          AND WITH CONFIDENCE 0.9;
```

After getting this information, perhaps she/he wants to see more detailed information, e.g., the maximal sub-regions that have at least 50 houses per unit area and at least 85% of the house prices are above $350,000 and with total area at least 80 units with 80% confidence. By the following SQL, we can get information we need from the original dataset. Here, we needn't scan all the original dataset again.

```
SELECT    SUB-REGION
FROM      house-map
WHERE     DENSITY IN (50, ∞)
          AND price RANGE (350000, ∞)
          WITH PERCENT (0.85, 1)
```

AND AREA (80, ∞)
AND WITH CONFIDENCE 0.85;

To complete this task, we should firstly recalculate the clustering granularity according to data objects distribution in the associated dataset, then divide the data space into rectangular cells of size equal to the minimum clustering granularity; count data objects density for each cell and store density value with each cell.

3.3 Visualization of Clustering Process

Although visualization has been recognized as an effective tool for exploratory data analysis and some visual clustering approaches were reported in the literature, these researches has focused on proposing new methods of visualizing clustering results so that the users can have a view of the processed dataset's internal structure more concretely and directly. However, seldom concern has been put on the impact of visualization on the clustering process. To make this idea clear, let us take 2-dimensional spatial data clustering for example. In fact, clustering can be seen as a process of data generalization on basis of certain lower data granularity. The lowest clustering granularity of a certain dataset is the individual data objects in the dataset. If we divide the dataset space into rectangular cells of similar size, then a larger clustering granularity is the data objects enclosed in the rectangular cells.

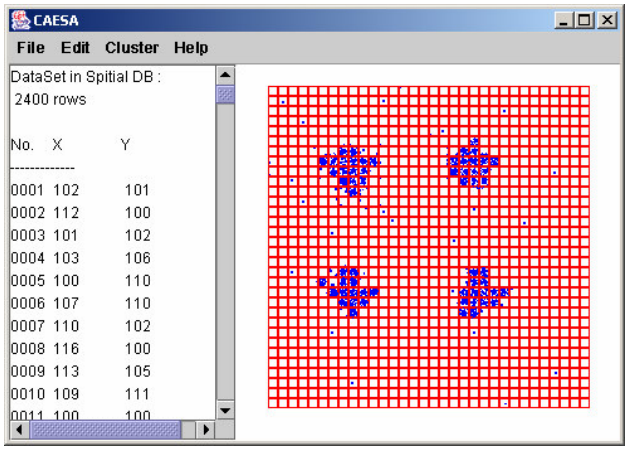

Figure 2: Expected result

More reasonably, we define clustering granularity as the size of the divided rectangular cell in horizontal or vertical direction, and define relative clustering granularity as the ratio of clustering granularity over the scope of focused dataset in the same direction. Visualization of clustering results is also based on clustering granularity. However, visualization effect relies on the resolution of display device. For comparison, we define relative visualization resolution as the inversion of the size (taking pixel as measurement unit) of visualization window for clustering results in the same direction as the definition of clustering granularity. Obviously, a reasonable choice is that the relative clustering granularity is close to, but not lower than the relative visualization resolution of visualization window. Otherwise, the clustering results can't be visualized completely, which means we do a lot but only part of its effect is shown up in visualization.

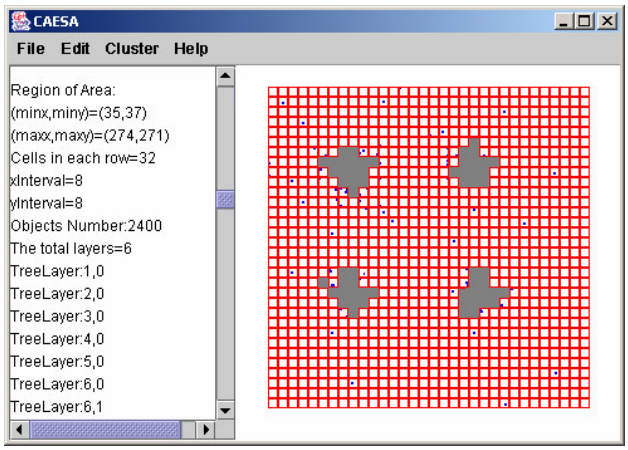

Figure 3: CAESA's result

Noises are random disturbance that reduces the clarity of clusters. In our algorithm, we can easily find noises and wipe off them by finding the cells with very low density and eliminate the data points in them precisely. This method can reduce the influence of the noises both on efficiency and on time. Unlike the noises, outliers are not well proportioned. Outliers are data points that are away from the clusters and have smaller scale compared to clusters. So outliers will not be merged to any cluster. When the algorithm finishes, the sub-clusters that have rather small scale are outliers.

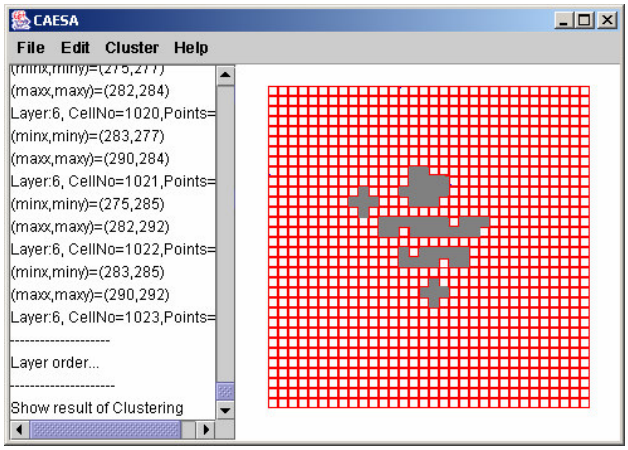

Figure 4: CAESA's result after user's focus changed

4. ALGORITHM

In this section we introduce the CAESA algorithm. First, we set the minimum clustering granularity according to data objects distribution in the whole concerned dataset; then divide the data space into rectangular cells of size equal to the minimum clustering granularity; after that, we count data objects density for each cell and store density value with each cell. Here, we adopt the cluster definition alike to grid-based clustering algorithms. It takes CASEA two steps to complete a given query, that is, calculate the statistical information and stores to the quad-tree, then get user's query and output the result.

First we should create a tree structure to keep the information of the dataset, there are two parameters must be input to setup the

tree. One parameter is dataset to be placed in the hierarchical structure; the other parameter is the number of desired cells at the lowest level. It needs two step to build the quad-tree: calculate the bottom layer cell statistical information directly by the given dataset, then calculate the higher layer cells statistical information according to the lower layer cells. We only need to go through the dataset once in order to calculate the parameters associated with the grid cells at the bottom level, the overall computation time is nearly proportional to the number of cells. That is, query processing complexity is O(*k*) rather than O(*n*), here *k* means the number of cells. We will analyze performance in more detail in later sections.

The process of creating the tree structure goes like this:

```
Input:
  D //Data to be placed in the hierarchical structure
  k //Number of desired cells at the lowest level
Output:
  T //Tree
CAESA BUILD Algorithm
  // Create the empty tree from top to down
  T = root node with data values initialized; //Initially
only root node
  i = 1;
  repeat
    for each node in level i do
      create 4 children nodes with initial values;
    end for
    i = i + 1;
  until 4^i = k;
  // Populate tree from bottom up
  for each item in D do
    determine leaf node j associated with the position of
D;
    update values of j based on attribute values in item;
  end for
  i = log4(k);
  repeat
    i = i - 1;
    for each node j in level i do
      update values of j based on attribute values in its 4
children;
  until i = 1;
End of CAESA BUILD Algorithm
```

Figure 5: CAESA BUILD algorithm

Once the user changes her/his focus scope, adjust clustering granularity based on the criterion that relative clustering granularity is close to, but not lower than the relative visualization resolution of visualization window. With the new clustering granularity, re-divide the focused data space; then resort to grid-based clustering algorithm to cluster the focused data. Note that new clustering granularity must be n times of the minimum clustering granularity where *n* is a positive nature number. Due to the sufficient information (such as location of coordinate, relevant coefficient etc.) , we can easily use the formal result to answer the new query.

The process of query answering is as follows:

```
Input:
  T //Tree
  Q //Query
Output:
  R //Regions of relevant cells
CAESA QUERY Algorithm
  QUEUE q;
  i = 1;
  repeat
    For each node in level i do
      if cell j is relevant to Q
        mark this cell as such;
        q.add( j );//cell j add to queue so as to calculate its
neighbor
      end if
    i = i + 1; //go on with the next level
  until all layers in the tree have been visited;
  for each neighbor cell i of cells in q AND i doesn't in q
do
    if (density( i ) > M_DENSITY)
      Identify neighboring cells of relevant cells to create
regions of cells;
    end if
  end for
  D1=R.data();
  calculate the new clustering granularity k1
  call CAESA BUILD Algorithm with parameter D1, k1
End of CAESA QUERY Algorithm
```

Figure 6: CAESA QUERY algorithm

To handle very large databases, the algorithm constructs units instead of original data objects. To obtain these units, first, partitioning is done to allocate data points into cells. Statistics information is also obtained for each cell. Then we test to see if a cell is qualified as a unit. The definition of unit is as follows:

The density of a unit is greater than a certain predefined threshold. Therefore, we determine if a cell is a unit by its density. If the density of a cell is *M* (*M*≥1) times of the average density of all data points, we regard such a cell as a unit. There may be some cells of low density but are in fact part of a final cluster, they may not be included into units. We treat such data points as separated sub-clusters. This method will greatly reduce the time complexity, as will be shown in the experiments.

The clusters identified in this algorithm are denoted by the representative points. Usually, user may want to know the detailed information about the clusters and the data points they include. For each data point, we need not test which cluster it belongs to. Instead, we need only find the cluster a cell belonging to, and all data points in the cell belong to such cluster. This process can also improve the speed of the whole clustering process.

5. PERFORMANCE EVALUATION

We conduct several experiments to evaluate the performance of the proposed algorithm. The following experiments are run on a PC machine with Win2000 operating system (256M memory). In CAESA, we only need to go through the data set once in order to calculate the parameters associated with the grid cells at the bottom level, the overhead of CAESA is linearly proportional to the number of objects with a small constant factor.

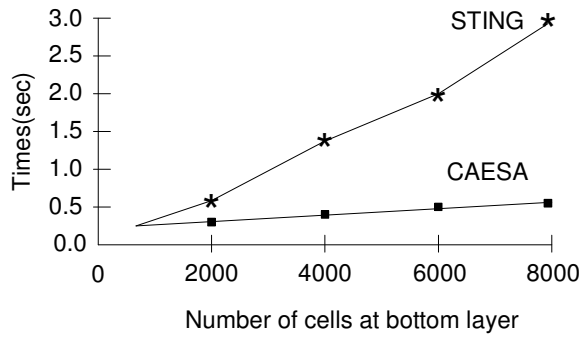

Figure 7: Overhead comparison between STING and CAESA when user change her/his focus

To obtain performance of CAESA, we implemented the house-price example discussed in Section 3.2. We generated 2400 data points(houses), the hierarchical structure has 6 layers in this test. Figure 2 shows the expected result, Figure 3 shows the result of our algorithm, and Figure 4 shows the result when user change her/his focus according to the last query. From the experimental result we can see apparently that our algorithm is valid and has a high performance.

6. CONCLUSION

In this paper, we proposed a scalable and visualization-oriented clustering algorithm for exploratory spatial analysis. It is of high performance and low computing cost, and it can run focus-changing clustering efficiently, can be adaptable to visualization situation, that is, choosing appropriate relative clustering granularity automatically according to current relative visualization resolution. We built a prototype to demonstrate the practicability of the proposed algorithm. Experimental results show our algorithm is effective and efficient.

REFERENCES

Agrawal R., Gehrke J., Gunopulos D. et al., 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of the ACM SIGMOD,* pp. 94–105.

David Hand, Heikki Mannila and Padhraic Smyth., 2001. Principles of Data Mining. Massachusetts Institute of Technolology Press.

Ester,M., Kriegel, H. P., Sander,J. et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc.of 2nd Int. Conf. Knowledge Discovery and Data Mining,* pp. 226-231.

Guha S., Rastogi R. and Shim K., 1998. CURE: an efficient clustering algorithm for large databases. In: *Proc. of the ACM SIGMOD*, pp. 73□84.

Halkidi, M., Batistakis, Y. and Vazirgiannis, M., 2001. Clustering algorithms and validity measures. In *Proc of 13th Intl. Conf. on Scientific and Statistical Database Management*, pp. 3 – 22.

Hinneburg, A., Keim, D. and Wawryniuk, M., 2003. Using projections to visually cluster high-dimensional data. *IEEE Computational Science and Engineering],* 5(2), pp. 14 – 25.

Hu, B. and Marek-Sadowska, M., 2004. Fine Granularity Clustering-Based Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 23(4), pp. 527 – 536.

Jiawei Han, Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

Matsushita, M. and Kato, T., 2001. Interactive visualization method for exploratory data analysis. In *Proc. of 15th Intl. Conf. on Information Visualization*, pp. 671–676.

Memarsadeghi, N. and O'Leary, D.P., 2003. Classified information: the data clustering problem. *IEEE Computational Science and Engineering*. 5(5), pp. 54 – 60.

Ng Ka Ka. E. and Ada Wai-chee Fu., 2002. Efficient algorithm for projected clustering. In *Proc.of ICDE*, pp. 273–273.

Sheikholeslami G., Chatterjee, S. and Zhang, A., 1998. WaveCluster: a multi-resolution clustering app roach for very large spatial databases. In *Proc. of 24th VLDB*, pp. 428□ 439.

Tin Kam Ho., 2002. Exploratory analysis of point proximity in subspaces. In *Proc. of 16th Intl. Conf. on Pattern Recognition*. Vol. 2, pp. 196 – 199

Wang Lian, Cheung, W.W., Mamoulis, N., et al., 2004. An efficient and scalable algorithm for clustering XML documents by structure. *IEEE Transactions on Knowledge and Data Engineering*, 16(1), pp. 82 – 96

Wang, W., Yang, J. and Muntz, R., 1997. Sting: a statistical information grid approach to spatial data mining. In *Proc. of VLDB*, pp. 186–195.

Wang, W., Yang, J. and Muntz, R., 1999. Sting+: An Approach to Active Spatial Data Miniing. In *Proc. of ICDE*, pp. 116–125.

ACKNOWLEDGEMENTS

The work is supported by Hi-Tech Research and Development Program of China under grant No. 2002AA135340, and Open Researches Fund Program of SKLSE under grant No. SKL(4)003, and IBM Research Award, and Open Researches Fund Program of LIESMARS under grant No. SKL(01)0303.