대용량 공간데이터베이스를 위한 확장된 밀도-격자 기반의 공간 클러스터링 알고리즘

Song Gao[†] · 김 호 석[††] · Ying Xia[†††] · 김 경 배[††††] · 배 해 영[†††††]

요 약

공간 데이터마이닝 분야에서 객체간의 거리, 연결성, 상대적인 밀도를 기반으로 비슷한 객체들을 하나의 그룹으로 묶는 공간 클러스터링은 중요한 컴포넌트이다. 공간 클러스터링 알고리즘은 밀도 기반 클러스터링과 격자 기반 클러스터링 알고리즘 등으로 나눌 수 있다. 밀도 기반 클러스터링 알고리즘은 다양한 모양과 크기의 클러스터를 구분할 수 있으며, 잡음을 제거할 수 있는 장점을 가지고 있는 반면에, 격자 기반 클러스터링 처리속도가 빠르다는 장점을 가지고 있다. 하지만, 대량의 공간 데이터 집합을 클러스터링 하는 것은 데이터 처리 비용이 급격하게 증가하기 때문에 클러스터링 처리 결과에 큰 영향을 준다. 본 논문은 대용량의 공간 데이터베이스에서 공간 객체간의 고밀도 영역을 식별하여 잡음을 제거하기 위한 수치데이터 값과 기본 격자간격 개수를 정의하는 확장된 밀도-격자 기반 클러스터링 알고리즘을 제안한다. 제안 알고리즘은 고밀도 영역 식별을 위하여 threashold(DT)를 정의하였으며, 격자 및 밀도 기반 기법의 장점을 이용하여 임의의 객체 클러스터링을 식별할 수 있는 성능을 향상시켰다. 성능평가에서 기존의 클러스터링 알고리즘과의 다양한 비교 평가 실험을 통하여, 제안 알고리즘이 빠르고 정확한 데이터 클러스터링 결과를 나타냄을 보인다.

키워드 : 클러스터링, 공간 데이터 마이닝, 공간 데이터베이스, Outlier

An Enhanced Density and Grid based Spatial Clustering Algorithm for Large Spatial Database

Song Gao[†] · Ho Seok Kim[††] · Ying Xia[†††] · Gyoung Bae Kim[††††] · Hae Young Bae[†††††]

ABSTRACT

Spatial clustering, which groups similar objects based on their distance, connectivity, or their relative density in space, is an important component of spatial data mining. Density-based and grid-based clustering are two main clustering approaches. The former is famous for its capability of discovering clusters of various shapes and eliminating noises, while the latter is well known for its high speed. Clustering large data sets has always been a serious challenge for clustering algorithms, because huge data set would make the clustering process extremely costly. In this paper, we propose an enhanced Density-Grid based Clustering algorithm for Large spatial database by setting a default number of intervals and removing the outliers effectively with the help of a proper measurement to identify areas of high density in the input data· space. We use a density threshold DT to recognize dense cells before neighbor dense cells are combined to form clusters. When proposed algorithm is performed on large dataset, a proper granularity of each dimension in data space and a density threshold for recognizing dense areas can improve the performance of this algorithm. We combine grid-based and density-based methods together to not only increase the efficiency but also find clusters with arbitrary shape. Synthetic datasets are used for experimental evaluation which shows that proposed mothod has high performance and accuracy in the experiments.

Key Words : Clustering, Spatial Data Mining, Spatial Database, Outlier

1. INTRODUCTION

Spatial data mining is the discovery of interesting characteristics and patterns that may exist in large spatial databases. Usually the spatial relationships are implicit in nature. Because of the huge amounts of spatial data that may be obtained from satellite images, medical equipments, Geographic Information System (GIS) etc, it's expensive and unrealistic for the users to examine spatial data in detail. Spatial data mining aims to automate the process of understanding spatial data by representing the data in a concise manner and reorganizing spatial databases to ac-

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구 센터 육성. 지원사업의 연구 결과로 수행되었음.

[†] 준 회 원 : Chongqing University of Posts and Telecommunications, School of Computer Science and Technology Master Course Student
[††] 준 회 원 : 인하대학교 대학원 컴퓨터 · 정보공학과 박사과정
[†††] 준 회 원 : Chongqing University of Posts and Telecommunications, School of Computer Science and Technology Professor
[††††] 종신회원 : 서원대학교 컴퓨터교육과 조교수
[†††††] 종신회원 : 인하대학교 컴퓨터공학부 교수

논문접수 : 2005년 12월 2일, 심사완료 : 2006년 8월 17일

commodate data semantics.

One of the main applications of clustering to spatial databases is to find clusters of points (where points represent the spatial objects) which are physically close together in geographic space. From this point, we propose DGCL to process dataset in spatial database. Usually spatial datasets are very large, such as the earth scientific datasets which include location and climate data of the earth during certain period. The disadvantage of shifting grid and GDILC is their low efficiency for large dataset, even though they can get correct clustering results. And that's what we want to improve in DGCL.

The aim of data clustering algorithms is to group the objects in spatial databases into meaningful subclasses. A good clustering algorithm should have the following characteristics. First, due to the huge amount of spatial data, an important challenge for clustering algorithm is to achieve good time efficiency. Second, a good clustering algorithm should be able to identify clusters irrespective of their shapes or relative position. Third, it should have better ability to handle noise or outliers. Outliers refer to spatial objects which are not contained in any cluster and should be removed during the mining process. Fourth, it should be order insensitive with respect to input data. Last, some applications may require clustering at hierarchical levels of coarseness which we call the multi-resolution property. To obtain results with different levels, the clustering algorithm may need input parameters, so fewer parameters are convenient for users.

None of the well-known clustering algorithms, such as shifting grid[1] and GDILC[2], satisfies the combination of these requirements, especially the efficiency for large dataset. This paper presents an efficient density-grid based clustering algorithm, DGCL, which can handle huge amount of spatial data with noise efficiently and find natural clusters correctly. When DGCL is performed on large dataset, a proper granularity of each dimension in data pace and a density threshold for recognizing dense areas can improve the performance of this algorithm.

The rest of this paper is organized as follows. Section 2 reviews related work. The detail algorithm is described in Section 3. Section 4 we analyze the time complexity. Section 5 shows the results of experiments. A conclusion is presented in the last section.

2. RELATED WORK

Density-based and grid-based clustering are two main clustering approaches. The former is famous for its capability of discovering clusters of various shapes and eliminating noises, while the latter is well known for its high speed. Shifting grid and GDILC are two kinds of clustering methods which are based on density and grid. Both of them have advantages and disadvantages.

Density-based[3] clustering algorithms regard clusters as dense regions of objects in the data space that are separated by regions of low density. A density-based cluster is a set of density-connected objects that is maximal with respect to density-reachability. Every object not contained in any cluster is considered to be noise. Typical example is DBSCAN. It grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise. It has two parameters(ε, MinPts). But the users usually don't know clearly about the suitable values of these two parameters for some data sets. Other drawback of this technique is the high computational complexity because of examining all the neighborhoods in checking the core condition for each object. Especially when the algorithm runs on very large datasets, this step is very expensive. Its time complexity is O(n log n), where n is the number of data objects [4]. Because of this point, it's also insensitive with the order of input data.

Grid-based[3] clustering algorithms use a multi-resolution grid data structure. It quantizes the space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.

A shifting grid[1] clustering algorithm does not require users inputting parameters. It divides each dimension of the data space into certain interval to form a grid structure in the data space. Based on the concept of sliding window, shifting of the whole grid structure is introduced to obtain a more descriptive density profile. It clusters data in a way of cells rather than in points [5]. But its processing is a recursive procedure. It may start at one cluster for all data points and become one data point per cluster eventually. If the difference between the result of previous iteration and that of current iteration is less than or equal to an acceptable error, *AcceptErr*, then the current result will be recognized as the final result. Therefore, for large dataset, this algorithm is time-consuming, even though it can enhance the accuracy of the results.

GDILC[2] is a grid-based density-isoline clustering algorithm. The idea of clustering using density-isoline comes from contour figures, density-based and grid-based clustering algorithms. It has two thresholds, namely distance threshold RT and density threshold DT. When calculating the density of a data sample, only the number of samples that are less than RT away from it are considered. It is the same with those data samples when combination for clusters. From this point, a grid-based method is involved to reduce the number of pairs of data samples to be considered. DT is used as a threshold to remove outliers. When clustering, density-isoline figure obtained from the distribution densities of data sample can be used to discover clusters. Because this algorithm needs to calculate the distance between each data sample and every data sample in its neighbor cells, the cost is also too high, especially for large data set.

In DGCL, we choose a grid-based method to reduce the number of data objects we have to process. We regard a grid cell as the minimum unit in data space, but not a data point. Compared with the number of data points, the number of grid cells is smaller especially in 2-dimension space. So it can increase efficiency, but to some extent it loses a little accuracy. The characteristic of density-based method is that it uses certain density threshold to distinguish dense areas with sparse areas, then it can find clusters with arbitrary shape. In DGCL, we use a density threshold DT to recognize dense cells before neighbor dense cells are combined to form clusters. We combine grid-based and density-based methods together to not only increase the efficiency but also find clusters with arbitrary shape.

3. THE DENSITY AND GRID BASED ALGORITHM

In DGCL, each data object is treated as a point in data space. We partition the 2 dimensions data space into non-overlapping rectangle units (grid cells), with each dimension being divided into equal number of intervals of equal length. When calculating the density of a cell, neighbor cells of each cell are also considered to strengthen the similarity of each pair of cells. A density threshold is used to distinguish the dense areas with sparse areas and outliers. Because a suitable granularity of each dimension is set, the clustering procedure need only one time without modifying the granularity again. Compared with shifting grid algorithm and GDILC, there is no iteration in DGCL and it doesn't need to calculate the distance of each pair of data points in data space. The time complexity is smaller than *O(n)*, which *n* is the number of data points in data space, yet depends on the number of grid cells.

3.1 Procedure of DGC

There are 7 steps in DGCL. In step 1, get the total number of the data points. According to the total number, construct the grid cell. In step 2, read the data set from the disk. If there is not sufficient memory to contain the whole data set, this algorithm divides them to several parts and read one part at a time to calculate the density of each cells until all parts have been read for one time. In step 3, calculate the density threshold DT by using the equation in GDILC[2]. The aim is to remove the outliers and empty cells as much as possible in step 4, and the fourth step can be regarded as a pre-clustering. The remainder cells will be regarded as useful cells. In the following steps, only the useful cells will be considered. So the number of useful cells is small compared with the number of data. In step 5, from the remainder cells, assign the adjacent cells to the same group which should be regarded as a cluster. But that's not the final result, because the fifth step can only find groups in the sub-area, and some groups are adjacent with each other. So, it's necessary to merge the adjacent groups together to become one group in step 6. There are still some outliers exiting in the groups, so in the last step, DGCL removes the outliers again to optimize the result. In the end, each group is a cluster. The sketch of DGCL is shown in (Figure 1).

```
1 : Construct grid cell according to the number of data points.
2 : Load data set.
3 : Distribute the data points into cells and calculate the density
    threshold DT.
4 : Remove outliers and empty cells.
5 : Group assignment.
6 : Group mergence procedure.
7 : Optimization.
```

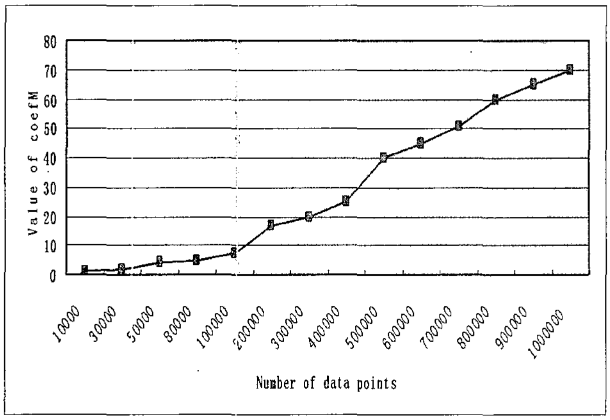
(Figure 1) The sketch of DGCL

3.2 Number of intervals

In this step, setting the number of intervals is a very critical procedure. If the size of the cells is too large, the algorithm will merge two or more clusters into one. Another drawback is that even though it can find the cluster at the right place, it still has so many blank spaces in the large size of the cell. So the result is not exact and satisfying. If the size of cell is too small, the algorithm may make the number of cells equal or close to the number of points. For large dataset, the cost of calculation is too expensive even though we regard the cell as the minimum unit for clustering. So it's necessary to find a method to set a suitable interval value to get both a better clustering result and a good efficiency. Here we adopt the method of GDILC[2]. Equation 1 is used to calculate the number of intervals *m*.

$$m = \sqrt{\frac{n}{coefM}} \qquad (1)$$

In equation 1, *n* is the number of data points. *coefM* is a coefficient to adjust the value of m. we propose it as an positive integer. In fact, it stands for the average number of data samples in a cell. But we don't want to regard it as a fixed value. Because in the experiments, we find the relationship between the number of cells and that of data points is not linear. So *coefM* also need to be adjusted to get a better result. In our experiments, *coefM* changes according to the following curve in (Figure 2), and it can get a better result as we expect.



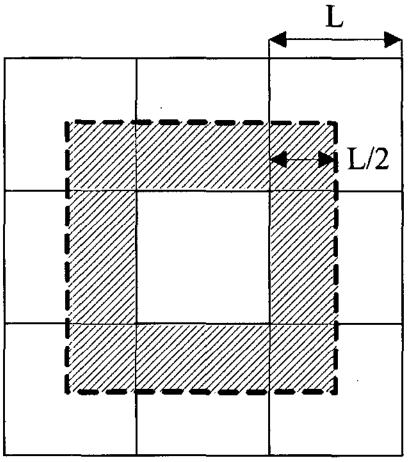
(Figure 2) *coefM* depends on the number of data

3.3 Density of the cell

For simply, we regard the density as the number of data points contained in the cell. The original grid-based clustering algorithm just considers about the density of the current cell itself[6]. The disadvantage is that it may decrease the relationship of neighbor cells which have similar data points. The attributes of a spatial object stored in a database may be affected by the attributes of the spatial neighbors of that object[5]. To improve this situation, we calculate the density of the considered cell with considering the data points in its neighbor cells such as what the shifting grid clustering algorithm does[1] as illustrated in (Figure 3). L means the width of each cell, and we never consider about the empty cells.

The definition of neighbor cells satisfies inequation 2. Assume that cell $C_{i_1 i_2}$ and $C_{j_1 j_2}$ are neighbor cells. m is the number of intervals.

$$|i_p - j_p| \le 1,\ (p = 1,2; 1 \le i,j \le m) \qquad (2)$$



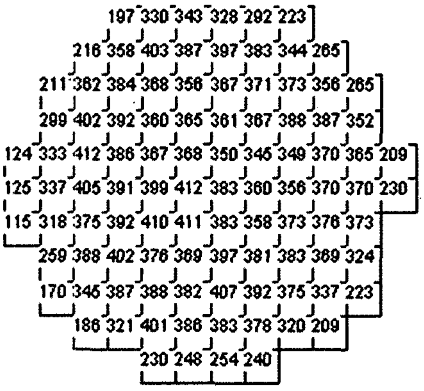
(Figure 3) The cell with gray color is the considered cell. The density of the considered cell is the sum of the density of the grey area and the line-shadowed part

3.4 Selection of the density threshold DT

The dense cells are surrounded by the sparse cells which are regarded as outliers. From the experiments, we find that the density of the dense cells usually decreases gradually from the core of the cluster to the boundary as illustrated in (Figure 4), and the density of sparse cells is obviously smaller than that of the boundary cells. So before clustering the cell set, we define a measurement to remove the sparse cells. The aim is to decrease the cost of calculation and increase the efficiency. The measurement, we call it density threshold (DT)[2], is defined using equation 3.

$$DT = \frac{mean(Density)}{log_{10}(m^2)} \times coefDT \qquad (3)$$

In equation 3, *mean(Density)* means the average density of all of the cells. m is the number of intervals. *coefDT* is an adjustable coefficient between 0.7 and 1. Lots of experi



197 330 343 328 292 223
216 358 403 387 397 383 344 265
211 362 384 368 356 367 371 373 356 265
299 402 392 360 365 361 367 388 387 352
124 333 412 386 367 368 350 346 349 370 365 209
125 337 405 391 399 412 383 360 356 370 370 230
115 318 375 392 410 411 383 358 373 376 373
259 388 402 376 369 397 381 383 369 324
170 346 387 388 382 407 392 375 337 223
186 321 401 386 383 378 320 209
230 248 254 240

(Figure 4) Varying grid density within a circle cluster ments show that when setting it to 0.95, good clustering result can be achieved in most conditions. All the cells of which the density is smaller than DT will be removed from the cell set. The remainder cells will be regarded as useful cells. In the following steps, the algorithm only considers about the useful cell, so compared with the number of data points, they're really few enough. The remainder cells also contain some unexpected cells. They will be further removed in step 7 (optimization).

3.5 Group assignment

This procedure starts from the first useful cell. If the considered cell *Ci* and all its *useful neighbors*, all belonging to set *Si*, haven't been assigned a group ID, the algorithm assigns a new group ID for all the cells in the set Si. If some cells in *Si* have had group IDs, the algorithm finds the ID of the cell with the maximum density in *Si* and assigns it to the other cells in *Si*. This procedure will not stop until all *useful cells* have been checked.(Figure 5) shows the sketch of group assignment.

```
Do{
  Select a useful cell Ci.
    If ( there is no group assigned to the cell Ci and its
         neighbors )
         Assign a new group to the cell and all its neighbors.
  Else
         Finding proper group ID and assign it to the other
         cells.
} While( there are still useful cells which haven't been checked )
```

(Figure 5) The sketch of group assignment

3.6 Group mergence

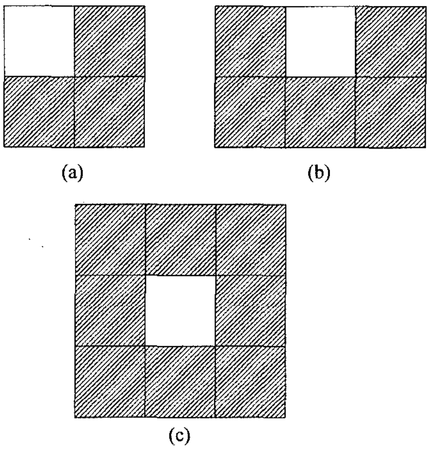
The procedure of group assignment only considers the current cell and its neighbors, so it clusters the similar cells into the same group in a sub-area. In the whole data space, the adjacent groups need to be merged together to form one group as we expected. The sketch of group mergence is shown in(Figure 6).

```
Do{
  Select one group gi.
  If( there are some groups adjacent with gi ){
      Merge them into one group gi'.
      Select the smaller group ID among them.
      Assign this ID to the new group gi'. }
}While( there are still groups which haven't been checked )
```

(Figure 6) The sketch of group mergence

3.7 Optimization

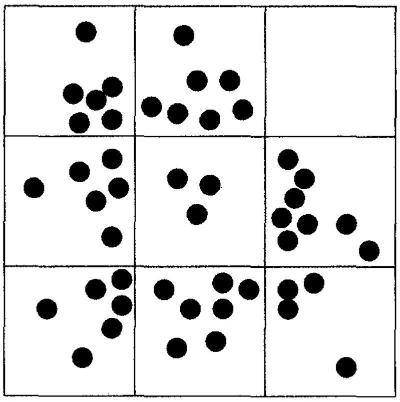
As we mentioned in the previous part, after step 4, the remainder cells also contain some cells which we have not expected.



(Figure 7) Situations of neighbors: (a) 3 neighbors in the corner and (b) 5 neighbors at the edge and (c) 8 neighbors inside

In DGCL, we calculate the cell density with considering about its neighbors. There are three kinds of situations of neighbors in 2-D space as illustrated in (Figure 7). The problem is that if the considered cell only contains a few data points, its useful neighbors contribute so much to calculate its density. It also includes into a group. But in fact, this cell should be regarded as a cell which contains outlier, and lots of experiments show that these situations often happen at the boundary of the group. (Figure 8) shows one situation of the problem. The fuscous cell is the considered cell. It only contains a relatively few data points compared with its neighbors when calculating the density. So this cell should be removed. The removed cells are regarded as *unuseful cells*, and the final result only contains the useful cells. We remove the cells which satisfy inequation 4.

$$RD = \frac{TD}{Num + 1} \tag{4}$$



(Figure 8) The fuscous cell should be removed

In equation 4, RD means the real number of data points which are contained in the considered cell. TD means the total density of the considered cell. *Num* is the number of its useful neighbors and 1 means the considered cell itself.

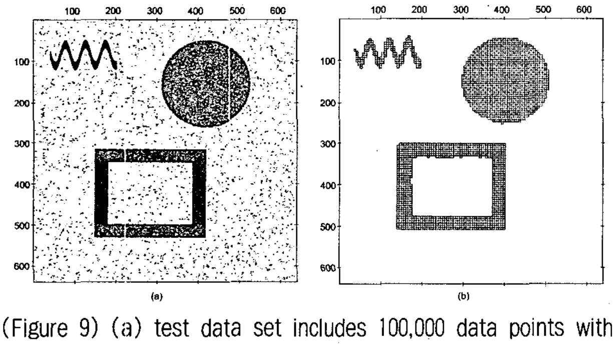
4. TIME COMPLEXITY

Because equation 1 is used to calculate the number of intervals, the total number of cells is m^2, namely n/coefM, in which n is the number of data points. After we get the data set, both the data distribution and removing outlier procedure need check or calculate all the cells. After that, only relatively few cells need to be checked for group assignment, merge procedure and optimization. So in most of the situations, the time complexity is smaller than $O(n)$. It much more depends on the number of grid cells.

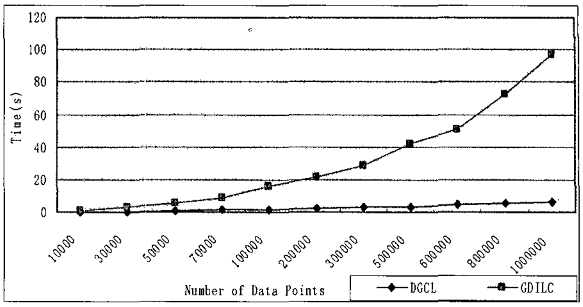
5. EXPERIMENTAL RESULTS

We performed experiments using a personal computer with 768MB of RAM and Pentium(R) 4 CPU 1.8GHz and running Windows XP Professional. The data sets are generated ourselves. In the data set, the data points are generated randomly according to certain kinds of distributions. Noises are randomly distributed. The clustering results are tagged by different colors. (Figure 9) (a) shows a data set with 100,000 data points, including 30,000 for circle, 40,000 for rectangle, 25,000 for sine curve and 5,000 noises.

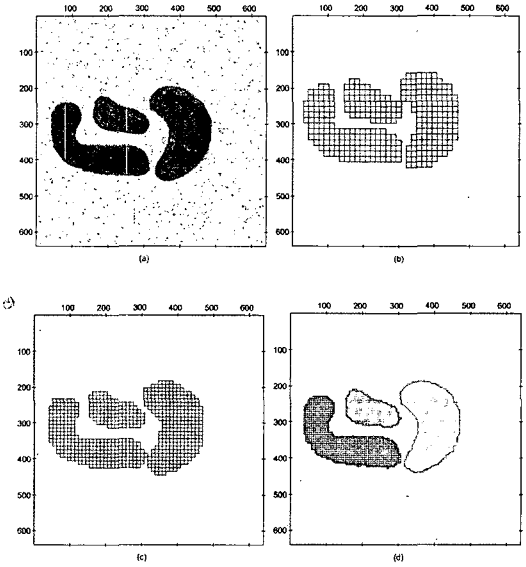
The result shows that DGCL can find the clusters correctly and eliminate outliers efficiently. Furthermore it also fast enough. If users want to get more exact result, they can adjust the coefficient *coefM*. The smaller the *coefM* is, the more exact result DGCL gets. But it need more time for calculation. Because GDILC has higher performance than shifting grid algorithm, we only show the comparison of DGCL and GDILC in (Figure 10). From that



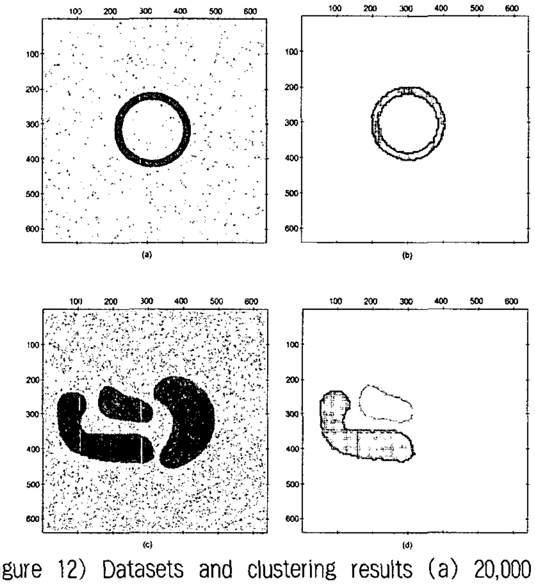
(Figure 9) (a) test data set includes 100,000 data points with 5,000 noises and (b) clustering result (3 clusters)



(Figure 10) Comparison between DGCL and GDILC



(Figure 11) clustering results in different levels. (a) data set, 55,000 data points with 2,500 noises and (b) *coefM* = 30, 1 cluster and (c) *coefM* = 15, 2 clusters and (d) *coefM* = 4, 3 clusters



(Figure 12) Datasets and clustering results (a) 20,000 data points with 1,000 noises and (b) 1 cluster and (c) 250,000 data points with 10,000 noises and (d) 3 clusters graph, we know that this algorithm has higher performance than GDILC.

(Figure 11) shows the results of one data set in different level by adjusting the coefficient *coefM*. Other data sets are also used to test this algorithm, the result is shown in (Figure 12).

6. CONCLUSION

In this paper, we present an efficient density and grid based clustering algorithm, DGCL. It applies a grid-based method to calculate the density of each celland can calculate automatically the granularity of each dimension according to the size of a given data set. In DGCL, a proper density threshold is calculated to distinguish dense cells with sparse cells. Adjacent dense cells are combined to form clusters. Compared with the shifting grid clustering algorithm and GDILC, it's much faster because it doesn't need to cluster recursively or calculate the distance between data points. The time complexity depends on the number of grid cells. Moreover, it is not affected by the outliers and can handle them properly. Drawback of this algorithm is that all the cluster boundaries are either horizontal or vertical. A faster method on I/O, such as an efficient indexing method or parallel control, will make the algorithm a whole lot faster. User can get different scale clustering results by adjusting the coefficient *coefM*. Experiment on synthetic datasets shows that DGCL is a stable and efficient clustering algorithm for large data set.

In future research, experiments will be performed both real and high dimensional dataset.

REFERENCES

[1] Ma, W.M., Eden, Chow, Tommy, W.S., "A new shifting grid clustering algorithm," Pattern Recognition 37(3), 503-514 (2004)

[2] ZHAO Yanchang, SONG Junde, "GDILC: A Grid-based Density-Isoline Clustering algorithm," IEEE (2001)

[3] Jiawei Han, Micheline Kamber, <<Data Minging: Concepts and Techniques>>, 2001 by Academic Press

[4] Yasser El-Sonbaty, M.A. Ismail, Mohamed Farouk, "An Efficient Density Based Clustering Algorithm for Large Databases," ICTAI (2004)

[5] A.H. Pilevar, M. Sukumar, "GCHL: A grid-clustering algorithm for high-dimensional very large spatial data bases," Elsevier B.V. (2004)

[6] Xiaowei Xu, Martin Ester, Hans-peter Kriegel, Jorg Sander, "Clustering and Knowledge Discovery in Spatial Databases," (1997)

[7] Yanchang Zhao, Chenqi Zhang, Yi-Dong Shen, "Clustering High-Dimensional Data with Low-Order Neighbors," Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '04)

[8] Shashi Shekhar, Sanjay Chawla, <<Spatial Databases: A Tour>>, 2003 by Pearson Eduction, Inc.

[9] Yu Qian, Kang Zhang, "GraphZip: A Fast and Automatic Compression Method for Spatial Data Clustering," SAC '04, March 14-17, 2004, Nicosia, Cyprus

[10] Yu Qian, Gang Zhang, Kang Zhang, "FACADE: A Fast and Effective Approach to the Discovery of Dense Clusters in Noise Spatial Data," SIGMOD 2004, June 13-18, 2004, Paris, France

[11] In-Soo Kang, Tae-wan Kim, and Ki-Joune Li, "A Spatial Data Mining Method by Delaunay Triangulation," GIS 97 LasVegas Nevada USA

[12] Eun-Jeong Son, In-Soo Kang, Tae-Wan Kim, Ki-Joune Li, "A Spatial Data Mining Method by Clustering Analysis," ACM GIS '98 11/98 Washington, D.C., USA

Song Gao

e-mail : gao_fly@hotmail.com

2004 Qingdao University, College of Information and Engineering, Computer Science and Technology, Bachelor

2004~ present Chongqing University of Posts and Telecommunications, School of Computer Science and Technology, Sino-Korea Chongqing GIS Research Center, Master Course Student

Interesting Field : Spatial data mining, LBS

김 호 석

e-mail : hskim@dblab.inha.ac.kr
2001년 인하대학교 전자계산공학과 (공학사)
2003년 인하대학교 대학원 컴퓨터공학과 (공학석사)
2003년~현재 인하대학교 대학원 컴퓨터·정보공학과(박사과정)
관심분야 : 다중레벨 데이터베이스, LBS, 유비쿼터스 컴퓨팅, RFID 미들웨어

Ying Xia

e-mail : xiaying@cqupt.edu.cn
1993 Chongqing University of Posts and Telecommunications, Department of Computer, Major in Computer and Telecommunication, Bachelor.
2001 Inha University in Korea, Department of Computer Science and Engineering, Master
2005~ present Chongqing University of Posts and Telecommunications, Sino-Korea Chongqing GIS Research Center, Associate Professor
Interesting Field : Database System and Application, GIS Middleware and Application, Mobile Computing, Spatial Database, LBS

김 경 배

e-mail : gbkim@seowon.ac.kr
1992년 인하대학교 전자계산공학과 (공학사)
1994년 인하대학교 대학원 전자계산공학과 (공학석사)
2000년 인하대학교 대학원 전자계산공학과 (공학박사)
2000년~2004년 한국전자통신연구원 선임연구원
2004년~현재 서원대학교 컴퓨터교육과 조교수
관심분야 : 이동 실시간 데이터베이스, 스토리지 시스템, GIS, VOD

배 해 영

e-mail : hybae@inha.ac.kr
1974년 인하대학교 응용물리학과(공학사)
1978년 연세대학교 대학원 전자계산학과 (공학석사)
1989년 숭실대학교 대학원 전자계산학과 (공학박사)
1985년 Univ. of Houston 객원교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학부 교수
1999년~현재 지능형GIS연구센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
2004년~2006년 인하대학교 정보통신대학원 원장
2006년~현재 인하대학교 대학원 원장
관심분야 : 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스 등