Incremental Spatial Clustering in Data Mining Using Genetic Algorithm and R-Tree

Nam Nguyen Vinh[1] and Bac Le[2]

[1] Vietnam Informatics and Mapping Corporation
nguyenvinhnam@vietbando.vn
[2] Faculty of Information Technology – University of Science / National University of Ho Chi Minh City, Vietnam
lhbac@fit.hcmus.edu.vn

Abstract. In this article, we present an algorithm based on genetic algorithm (GA) and R-tree structure to solve a clustering task in spatial data mining. The algorithm is applied to find a cluster for a new spatial object. Spatial objects that represent for each cluster computed dynamically and quickly according to a clustering object in the clustering process. This improves the speed and accuracy of the algorithm. The experimental results show that our algorithm yields the same result as any other algorithm and is accommodated to the clustering task in spatial data warehouses.

Keywords: Spatial data mining, Clustering, Genetic Algorithm.

1 Introduction

Due to advanced data collection techniques such as remote sensing, census data acquiring, weather and climate monitoring etc. contemporary geographical datasets contain an enormous amount of data of various types and attributes. Analyzing this data is challenging for traditional data analysis methods which are mainly based on extensive statistical operations. Since classical data mining methods enable us to detect valuable information from extensive relational databases, spatial data mining (SDM) can be an appropriate technique for detecting possible interesting patterns in geographical datasets. SDM is a knowledge discovery process of extracting implicit interesting knowledge, spatial relations, or other patterns not explicitly stored in databases [13, 14].

Clustering is one of the tasks of spatial data mining. It is a typical unsupervised learning technique for grouping similar data points. A clustering algorithm assigns a large number of data points to a smaller number of groups such that data points in the same group share the same properties while, in different groups, they are dissimilar. Clustering has many applications, including part family formation for group technology, image segmentation, information retrieval, web pages grouping, market segmentation, and scientific and engineering analysis [3].

Genetic algorithms belong to a class of directed search methods that are be used for both solving optimization problems and modeling the core of evolutionary systems.

L.T. Bui et al. (Eds.): SEAL 2012, LNCS 7673, pp. 270–279, 2012.

They use a heuristic rather than analytical approach, and thus their solutions are not always exact and their ability to find a solution often depends on a proper and sometimes fragile specification of the problem representation and the parameters that drive the genetic algorithm [1]. There have been proposals and approaches for the application of genetic algorithms for the clustering problems. These algorithms often encode chromosomes based on one or more cluster centers. The centers were chosen at random or determined by K-mean when initializing population [3, 6, 7, 8, 9]. Number of clusters is one of the input parameters, and a method used to determine these cluster centers affect to accuracy and speed of these algorithms.

In this paper, we propose an efficient incremental spatial clustering algorithm, which determines arbitrary shaped clusters. The algorithm based on Genetic Algorithms and R-tree structure. The proposed algorithm can work on very large database and yield the same result as any other algorithm.

2 Preliminaries

2.1 Spatial Clustering

Spatial clustering groups spatial objects such that objects in the same group are similar and objects in different groups are unlike each other. This generates a small set of implicit classes that describe the data. Clustering can be based on combinations of non-spatial attributes, spatial attributes (e.g., shape), and proximity of the objects or events in space, time, and space–time.

In general, the major clustering methods can be classified into the following categories [2]:

- **Partitioning methods:** Given a database of n objects or data tuples, a partitioning method constructs k ($\leq$ n) partitions of the data, where each partition represents a cluster. That is, it classifies the data into k groups, which together satisfy the following requirements: (1) each group must contain at least one object, and (2) each object must belong to exactly one group. Representative algorithms include k-means, k-medoids, CLARANS, and the EM algorithm.
- **Hierarchical methods:** A hierarchical method creates a hierarchical decomposition of a given set of data objects. Representative algorithms include BIRCH and Chameleon.
- **Density-based methods:** Most partitioning methods cluster objects based on the distance between objects. Their general idea is to continue growing a given cluster as long as the density (the number of objects or data points) in the "neighborhood" exceeds a threshold. Representative algorithms include DBSCAN, OPTICS, and DENCLUE.
- **Grid-based methods:** Grid-based methods quantize the object space into a finite number of cells that form a grid structure. Representative algorithms include STING, WaveCluster, and CLIQUE.

2.2 Genetic Algorithm

Genetic Algorithms (GAs) was invented by John Holland and developed this idea in his book "Adaptation in natural and artificial systems" in the year 1975. Holland proposed GA as a heuristic method based on "Survival of the fittest". GA was discovered as a useful tool for search and optimization problems.

The basic genetic algorithm is as follows [4]:

a. **[Start]** Genetic random population of n chromosomes (suitable solutions for the problem)
b. **[Fitness]** Evaluate the fitness f(x) of each chromosome x in the population
c. **[New population]** Create a new population by repeating following steps until the New population is complete
   - **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
   - **[Crossover]** With a crossover probability, cross over the parents to form new offspring. If no crossover was performed, offspring is the exact copy of parents.
   - **[Mutation]** With a mutation probability, mutate new offspring at each position in chromosome
   - **[Accepting]** Place new offspring in the new population.
d. **[Replace]** Use new generated population for a further sum of the algorithm.
e. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population.
f. **[Loop]** Go to step b for fitness evaluation.

2.3 R-tree

An R-tree is a height-balanced tree similar to a B-tree with index records in its leaf nodes containing pointers to data objects. Nodes correspond to disk pages if the index is the disk-resident, and the structure is designed so that a spatial search requires visiting only a small number of nodes. The index is completely dynamic: inserts and deletes can be inter-mixed with searches, and no periodic reorganization is required [5].

3 Algorithm

Our clustering method is density-based. In our experiments, Euclidian distance has been applied. The proposed algorithm for clustering objects in the 2D plane is depicted as following:

Input: A set of spatial objects P and the configure K
Output: A set of clusters C
$C \leftarrow \emptyset$
$foreach\ p\ in\ P$
  $Cluster\_GA(p, K, C)$

Where, $\boldsymbol{P}$ is a set of clustering objects. $\boldsymbol{K}$ is a configuration for the algorithm that includes a threshold for clustering and algorithm settings for GA – The GA operators selection, crossover, crossover probability, mutation and mutation probability. The function $\boldsymbol{Cluster\_GA}$ finds a cluster in $\boldsymbol{C}$or create a new cluster for object $\boldsymbol{p}$. We implement the algorithm by defining the steps of a genetic algorithm.

3.1 Encoding

A chromosome represents a cluster, and is encoded as shown in Fig. 1.

Valid_Flag	ClusterId	SObjects	Reps

Fig. 1. Encoding a cluster

Valid_Flag indicates whether this cluster is valid. A cluster is valid if it contains at least one spatial object. $ClusterId$ is a numeric value indicating cluster identity. $SObjects$ contains objects that belong to this cluster. The R-tree structure is used to store these objects in $SObjects$ to accelerate computing the fitness. $Reps$ is a subset of $SObjects$ that represents this cluster, and is determined after applying the fitness function defined in section 3.2. The number of objects in $Reps$ changes correspondingly to the position of a clustering object. An example demonstrating this encoding way shown in Fig. 2.

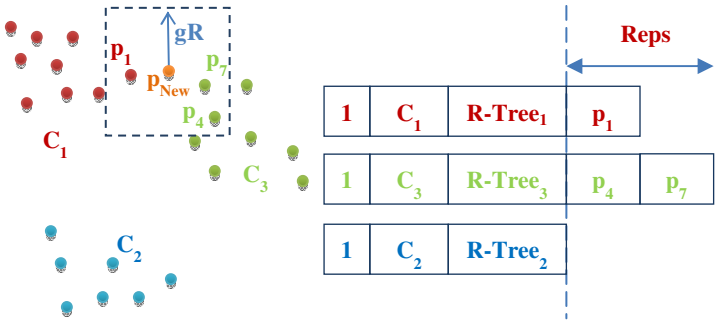


Fig. 2. An example of the cluster encoding

3.2 Fitness Function

The fitness function is applied for each valid cluster in C. It takes the R-tree structure of a cluster to find spatial objects to represent this cluster and to store them in the $Reps$ part. The fitness function acts in the following way:

- **Find candidates represent the cluster**
  We compute the minimum bounding rectangle MBR^p for clustering object p and expand this MBR^p with threshold gR through the following expressions:

$$MBR^{query}.left = MBR^{p}.left - gR$$
$$MBR^{query}.top = MBR^{p}.top - gR$$
$$MBR^{query}.right = MBR^{p}.right + gR$$
$$MBR^{query}.bottom = MBR^{p}.bottom + gR$$

This MBR^{query} is used to search in the $SObject$ field of each cluster. The selected objects are the searching result. As shown in Fig. 2, $p_1$is a representative object for cluster C_1 and $p_4$and p_7 for cluster C_3. Cluster C_2 has none.

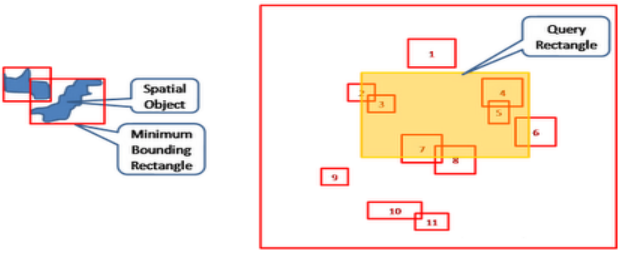


Fig. 3. Query spatial objects by R-tree

- **Returned value**
  The searching result determines the returned value of this function. The result returned by the function is 1 if the number of the representative objects ≠ ∅, otherwise ∞.

Clusters having fitness value ∞ can not be applied genetic operators to accelerate algorithm performance.

3.3 Crossover

The algorithm randomly selects a position in the $Reps$ part of each parent cluster to test merging condition. If expression (1) below is satisfied, two parent clusters are merged into one single cluster.

$$Distance(Reps_i^m, p) \leq gR \;\&\&\; Distance(Reps_j^n, p) \leq gR \quad (1)$$

$Distance$ is the Euclidian distance between two spatial objects. $Reps_i^m$and $Reps_j^n$ are two objects in $Reps$ of clusters m and n respectively. The example in Fig. 4 indicates the merging case (clusters C_1 and C_3 are going to be merged).

3.4 Mutation

This operator is used to insert clustering object p into a mutated cluster if the following condition (2) is satisfied:

$$Distance(Reps_k^m, p) \leq gR \quad (2)$$

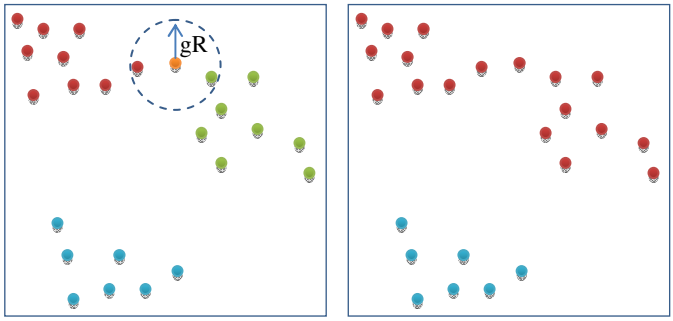


Fig. 4. Merging two clusters

3.5 Algorithm Cluster_GA

```
        void Cluster_GA(p, K, C)
        {
[1]         P ← ∅
            foreach cluster in C
            {
                [f, Reps] ← Fitness(p, cluster)
                if(f ≠ ∞)
                    P.push(cluster)
            }
            if(P ≠ ∅)
            {
                Crossover(p, P, K)
                Mutation(p, P, K)
            }
            else
            {
[2]             c_new ← CreateNewCluster(p)
                C.insert(c_new)
                return
            }
            if(no.of generations < K.MAX_GENERATIONS)
                goto [1]
            else
                if p not belongs to any c in C
                    goto [2]
            return
        }
```

4 Experiments

The experiments were done on a 2.40 GHz Intel® Core™ i5 machine with 2GB main memory. The program was compiled by the Visual Studio C++ 2010 compiler using level 3 optimization.

To evaluate our system, we use *Shape* datasets available from [12]. These are relatively small but quite diverse in shape, so they are suitable for checking the correctness of the algorithm for different distributions. Their shapes are shown in Fig. 5. The clustering performance of the proposed algorithm is demonstrated on two different datasets. Dataset *HitPosition_10631* includes 10.631 points and *clus100000* has 100.000 points. These datasets are illustrated in Fig. 6a, b.

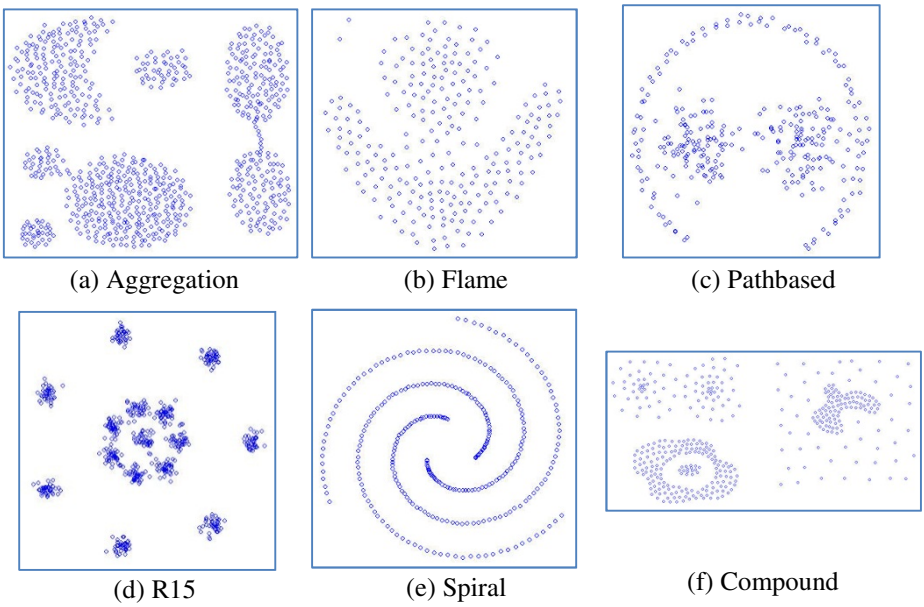
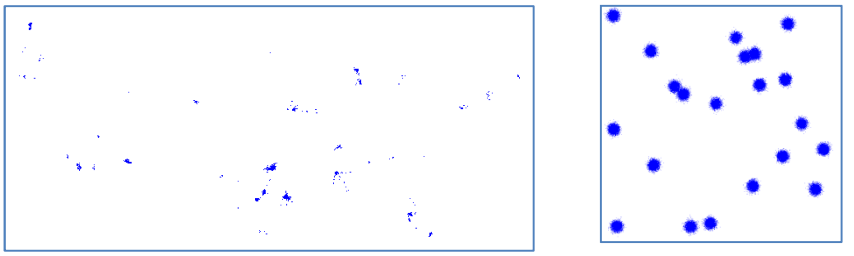


Fig. 5. Shape Datasets

Clustering radius gR is dynamically computed for each dataset by expression (3) below. With *Shape* datasets, we adjusted the gR parameter by multiplying with a coefficient corresponding to each dataset to obtain desired results. These coefficients are listed in Table 1. The parameters settings for GA are shown in Table 2.

$$gR = \min(x_{max} - x_{min}, y_{max} - y_{min}) / 30 \quad (3)$$

With visualizing clustering results, we can easily check the algorithm's correctness on the *Shape* datasets. These results are shown in Fig. 7. It can be seen that the proposed algorithm could discover clusters of arbitrary shape.



(a) Dataset HitPosition_10631 (b) Dataset clus100000

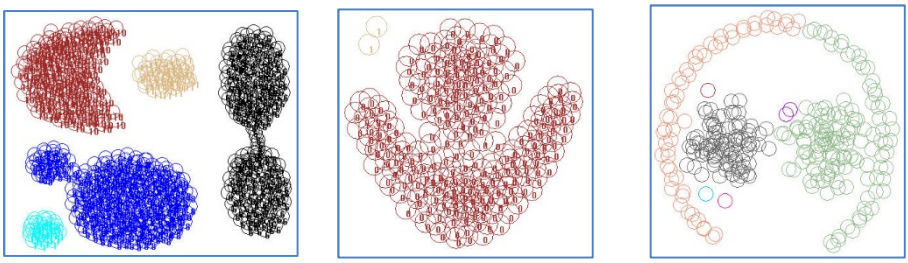
Fig. 6. Datasets for testing the scalability of our algorithm

Table 1. Adjusting coefficients for parameters gR

Dataset	Adjusting coefficient
Aggregation	2.00
Flame	3.00
Pathbased	2.05
R15	1.50
Spiral	1.50
Compound	2.50

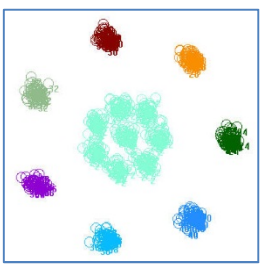
Table 2. The parameters settings for GA

Settings Type	Value
Population size	250
Selection	Elitist
Crossover	One point
Mutation	Uniform
Crossover Probability	0.6
Mutation Probability	0.01
Maximum Generations	50

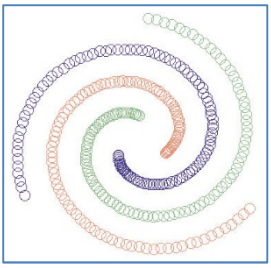


(a) Aggregation (b) Flame (c) Pathbased

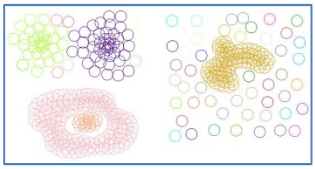
Fig. 7. Clustering results of the Shape datasets



(d) R15



(e) Spiral



(f) Compound

Fig. 7. *(Continued)*

We used sweep-line algorithm [10] to verify the accuracy of our algorithm on larger datasets because it is easy to implement and gives more accurate results. The sweep-line algorithm with the same parameter gR of our algorithm found 36 clusters for *HitPosition_10631*and 17 clusters for *clus100000*. Our clustering results are shown in Fig. 8a, b and Table 3. It can be seen that the proposed algorithm could yield the same result as any other algorithm.

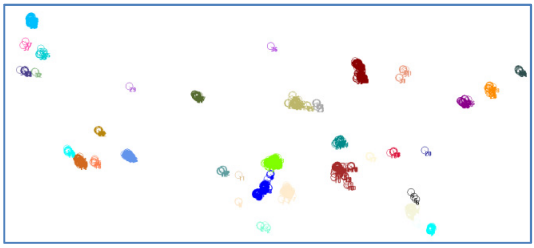


Fig. 8a. HitPosition_10631 clustered

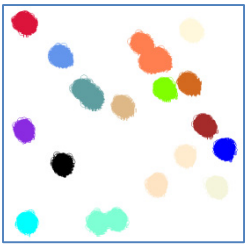


Fig. 8b. Clus100000 clustered

Table 3. The result of the experiment

No.	HitPosition_10631		Clus100000	
	Clusters	Time (ms)	Clusters	Time (ms)
1	36	343	17	15.865
2	36	328	17	15.865
3	36	343	17	15.538
4	36	343	17	15.928
5	36	327	17	15.600
6	36	358	17	15.568
7	36	327	17	15.600
8	36	328	17	15.538
9	36	328	17	15.881
10	36	328	17	15.600

R-tree structure is built based on the smallest rectangles that contain geometry objects without regard to their type. Moreover, the distance between two objects can be determined by the mathematical formula. Therefore, our algorithm can be applied for clustering different kinds of objects such as points, lines and polygons.

5 Conclusions

This paper introduces a new spatial clustering algorithm based on GA and R-tree structure. The proposed algorithm can detect clusters of arbitrary shape and yield the same result as any other algorithms. Furthermore, our algorithm can be applied to cluster any spatial object type, such as point, line and polygon. Spatial databases generally support the R-tree structure, so the algorithm is highly applicable.

References

1. Data Mining – Know It All. Morgan Kaufmann Publishers (2009)
2. Geographic Data Mining and Knowledge Discovery, 2nd edn. CRC Press (2009)
3. Pham, D.T., Afify, A.A.: Clustering techniques and their applications in engineering. Submitted to Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science (2006)
4. Introduction to Genetic Algorithms. Springer (2008)
5. Guttman, A.: R-tree: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, vol. 14(2) (June 1984)
6. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.: FGKA: A Fast Genetic K-Means Clustering Algorithm. In: ACM Symposium on Applied Computing (2004)
7. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.: Incremental Genetic K-Means Algorithm and its Application in Gene Expression Data Analysis. BMC Bioinformatics (2004)
8. Ding, Q., Gasvoda, J.: A Genetic Algorithm for Clustering on Image Data. International Journal of Information and Mathematical Sciences (2005)
9. Al-Shboul, B., Myaeng, S.-H.: Initializing K-Means using Genetic Algorithms. World Academy of Science, Engineering and Technology (2009)
10. Zălik, K.R., Zălik, B.: A sweep-line algorithm for spatial clustering. Journal of Advances in Engineering Software 40(6) (2009)
11. Lin, H.-J., Yang, F.-W., Kao, Y.-T.: An Efficient GA-based Clustering Technique. Tamkang Journal of Science and Engineering 8(2), 113–122 (2005)
12. http://cs.joensuu.fi/sipu/datasets/
13. Koperski, K., Adhikary, J., Han, J.: Knowledge discovery in spatial databases: Progress and Challenges. In: Proceedings of the SIGMID Workshop on Research Issue in Data Mining and Knowledge Discovery, Technical report 96-08. University of British Columbia, Vancouver, Canada (1996)
14. Koperski, K., Han, J.: Discovery of Spatial Association Rules in Geographic Information Databases. In: Proc. 4th Int. Symp. on Large Spatial Databases, pp. 47–66. Springer, Berlin (1995)