# A Half-Split Grid Clustering Algorithm by Simulating Cell Division

Wenxiang Dou Graduate School of Information, Product and Systems, Waseda University 2-7 Hibikino, Wakamatsu, Kitakyushu-shi, Fukuoka, 808-0135, Japan william@ruri.waseda.jp

Abstract—Clustering, one of the important data mining techniques, has two main processing methods on data-based similarity clustering and space-based density grid clustering. The latter has more advantage than the former on larger and multiple shape and density dataset. However, due to a global partition of existing grid-based methods, they will perform worse when there is a big difference on the density of clusters. In this paper, we propose a novel algorithm that can produces appropriate grid space in different density regions by simulating cell division process. The time complexity of the algorithm is O(n) in which n is number of points in dataset. The proposed algorithm will be applied on popular chameleon datasets and our synthetic datasets with big density difference. The results show our algorithm is effective on any multi-density situation and has scalability on space optimization problems.

Keywords—data clustering; grid clustering; unsupervised learningt

## I. INTRODUCTION

Clustering is an important data analysis method with unsupervised learning process. The objective of clustering is to divide a given dataset into several clusters based on data similarity or density distribution. So, it can be applied in many fields such as machine learning, biology, image processing and pattern recognition.

In [1], there are four major categories classified on clustering patterns as following: partition-based, hierarchical-based, density-based and grid-based methods. K-means [2], a typical partition-based method and hierarchical clustering (HC) have been popular in clustering analysis area. Many research efforts have been done on them such as Genetic K-means [3], CURE [4] and ROCK [5]. However, these algorithms require the number of clusters in advance, which is considered to be one of the biggest drawbacks and are weakness on dealing with multiple shape dataset.

For avoiding the above problems, DBSCAN [6] and SNN [7], density-based methods, have been proposed. These density-based algorithms assume that two neighbor points with same cluster share similar number of points in a given area and find clusters automatically by expanding points adjacent to each other. Although they can easily find clusters with different

## Jinglu Hu

Graduate School of Information, Product and Systems, Waseda University 2-7 Hibikino, Wakamatsu, Kitakyushu-shi, Fukuoka, 808-0135, Japan jinglu@waseda.jp



Fig. 1. A dataset with three clusters with big difference of density.

sizes and shapes, there is evident drawback on processing dataset with variant densities. Moreover, the time complexity is high on large dataset because they need to calculate the similarity of all pairs of points. There are many variants trying to improve effect and speed such as DD-DBSCAN [8] and ST-DBSCAN [9].

In order to process large dataset more efficiently, the gridbased clustering algorithm [10] [11], has been proposed. It partitions data space into a given number cells and all operations of clustering are on these cells instead of data objects. The feature of the high processing rate has attracted many attentions. In [12-16], these researches attempt to overcome the drawback of the gird structure caused by the global density parameter so that the algorithms can accurately process the multiple density dataset. GMDBSCAN [12] and GRPDBSACN [14] use suitable local parameters by combining DBSCAN algorithm in different density regions to generate clusters. GDCLU [16] identifies local density based on the density of neighbor grids. Edla and Jana [15] try to find an optimal gird size using the cluster and local outlier factor. But the existing grid-based algorithms still cannot break the limitation on processing dataset with big density difference, because they do the clustering on the global size of the gird structure. For example, Fig. 1 shows a dataset with two small



Fig. 2. (a) Division process on cell meiosis. (b) Half-split process on a grid with satisfying the split state SplitCon(D(g) = 3).

clusters and one big cluster. If the global size is set well for the big cluster, two small clusters will be identified as a same group. If it is set for the small cluster, all of points in the big cluster will be regarded as noise. So, there is no a suitable global size for this kind of dataset. Existing grid-based algorithms cannot deal with this problem.

In this paper, we propose a new half-split grid clustering algorithm which partitions different density areas into suitable gird structures respectively by simulating cell division.

Our algorithm uses the binary tree to construct the half-split grid structure. Initially, the whole data space is regarded as the root node of the tree. When the number of data points contained by some leaf node exceeds a given threshold, the space of the node will be split into two child nodes with the same size of the space. The split process will continue until the space of all leaf nodes of the binary tree contains the data points whose number is less than the threshold. So, our gird structure consists of space grids contained by all leaf nodes whose density distribution of data points is similar in the same level of the binary tree. Finally, clusters will be produced in our half-split gird structure.

Therefore, the proposed algorithm can process arbitrary multi-density dataset and the time complexity is the same low as grid-based algorithm.

The organization of the rest of paper is as follows. In Section II, we describe the idea and implementation of our half-split grid clustering algorithm in data space. Section III gives the experimental results. Section IV is the conclusions and the future work.

## II. THE PROPOSED HALF-SPLIT GRID ALGORITHM

### A. A Behavior Simulation on Cell Division

Existing grid-based algorithms regard data space as a collection of cells with the same size. Clustering on the same cells has a great limitation on processing multi-density dataset. In our research, we find we can break the limitation and let cells multiply their generations with a suitable size in different density areas by simulating the division behavior of cells. In

Fig.2, the left shows a Meiosis process in which DNA is reduced to half the original number when some condition is satisfied. The right of the Fig.2 shows our simulation on cell division. But the space of the parent gird will be reduced to half the original size rather than points (like DNA). In our half-spilt method, a split condition needs to be set in advance. When some gird reaches the condition, the half-split will be caused on this grid. For example, in Fig.2, SplitCon(D (g)= 3) means that a grid containing 3 points will be split to half iteratively until no any descendant grid contains more than 2 points.

So, our method initially regards whole data space as a cell. We put points into their corresponding cells one by one. If some cell reaches the split condition, we do the half-split operation on this cell until no descendant of it satisfies the split condition. Then we repeat the procedure until all points are put in the cells they belong to. Finally, we generate clusters on the half-split gird structure which is built by using binary tree. Fig.3 shows a half-split result of the dataset in Fig.1 with the SplitCon(D(g) = 4).



## B. Formal Definitions

**Definition 1:** Let  $S = S_1 \times S_2 \times \cdots \times S_d$  be the given data space with *d* dimensions where  $S_i = [x_i, y_i]$   $(1 \le i \le d)$   $(x_i, y_i \in \mathbb{R} \text{ and } x_i < y_i)$ . The dataset in the *S* containing *n* data points can be represented



Fig. 4. Neighbor relation construction of a grid being spilt

as  $P = (p_1, p_2, ..., p_n)$  where  $p_i = (p_{i1}, p_{i2}, ..., p_{id})(1 \le i \le n), p_{ij}$  $(p_{ij} \in S_j)$  represents the value in the  $j^{ih}$  dimension of the  $i^{ih}$  data points.

**Definition 2:** Half-Split Tree (HS-Tree) is a structure of the Binary Tree generated by half split of the data space as follows:

(1) The initial data space *S* is the root of the HS-Tree. All other notes are the sub space *s* of the *S* where  $s = s_1 \times s_2 \times \ldots \times s_d$   $(s_i \subseteq S_i \text{ and } s \subseteq S, 1 \le i \le d)$ .

(2) Each non-leaf note is the space split to two sub space with the same size and contains the position information of the split.

(3) All leaf notes constitute our half-split grid structure. Each data point will be put in its corresponding leaf note by indexing HS-Tree from top to down. So, each leaf note is a grid which can be represented as  $Gi = (g_{i1}, g_{i2}, ..., g_{id})(g_{ij} \subseteq S_j, 1 \le j \le d)$  (*i*=1,2, ..., *n*, *n* is the number of leaf notes) and  $G_1 \cup G_2 \cup ... \cup G_n = S$ .

**Definition 3:** Let  $G_x = (g_{x1}, g_{x2}, ..., g_{xd})$  and  $G_y = (g_{y1}, g_{y2}, ..., g_{yd})$  be any two grids. If  $\{\exists g_{xi} \in G_x, \exists g_{yi} \in G_y | g_{xi} \cap g_{yi} = \emptyset$  and  $g_{xi} \cup g_{yi}$  is a continuous space  $g_{xyi}$  defined as  $\{\forall k \in [Min(g_{xyi}), Max(g_{xyi})] | \neg \exists (k \notin g_{xi} \text{ and } k \notin g_{yi})\} (1 \le i \le d)\}$ , then we call  $G_x$  and  $G_y$  are neighbor each other defined as Neighbor( $G_x, G_y, con$ ) where *con* means a connection between two neighbor grids. When a gird is split, it is necessary to evaluate each neighbor of the parent gird whether it is also neighbor for two child grids. If it is judged neighbor to a child grid by the above definition, then a connection will be created between them.

Fig.4 shows an example to illustrate the production of neighbor connections on the split process. In the left graph, the 2D space  $X \times Y$  has been split into three grids with G1, G2 and G3. G3 has two connections C2 and C3 which connect two neighbors G1 and G2 respectively. In the right graph, G3 is split into G4 and G5, and a connection C4 is created firstly between G4 and G5 because they must be neighbor each other. Then the neighbors of the parent G3 are assigned to the child grids.

For the child G4, (G1, G4) in X dimension and (G2, G4) in Y dimension satisfy the definition 3. So, G1 and G2 also are neighbors of G4 and two connections C2 and C3 are produced to connect two neighbors labeled as Neighbor(G1, G4, C2) and Neighbor(G2, G4, C3). For the child G5, it does not satisfy the

definition 3 with G1 in Both X and Y. So G5 only has two neighbors G2 and G4.

**Definition 4:** Grid density  $D(G_i)$  is defined as number of points contained by the grid  $G_i$ .

 $N(G_i)$  is defined as a neighbor gird set of the grid  $G_i$ .

 $L(G_i)$  represents a value about which lever the grid  $G_i$  located at on the HS-tree.

So, a grid  $G_i$  can be called a core grid if only if it satisfies two conditions as follows:

- 1.  $D(G_i)$  is equal or larger than the input parameter MinPts
- 2. (2)  $\sigma_{L(G_i) < \gamma}$ .

 $\sigma_{L(G_i)}$  is the level deviation degree defined as:

$$\sigma_{L(G_i)} = \frac{\sum_{j=1}^{n} (L(G_j) - L(G_i))^2}{n}$$
(1)

Where  $\gamma$  is a threshold,  $G_j \in N(G_i)$  and n is the number of neighbor grids of the grid  $G_i$ .

**Definition 5:** SplitCon(*condition*) is defined as a trigger condition. The half-spilt will take place when the *condition* is satisfied. In this paper, we set the *condition* as  $D(G_i) = 2*MinPts$ . That means a grid will be split when the number of points in the grid reaches 2\*MinPts. However, it is our future work that the *condition* can be set appropriately to solve other problems.

In the following, we will introduce the detailed implementation about the half-spilt grid algorithm and the clustering on the gird structure.

## C. Grid Structure Production by Half-Split Algorithm

Before we do the clustering, the data space should be changed into the grid structure by our half-split algorithm.

Firstly, the HS-Tree will be initialized and the root of the tree is the whole data space *S*. So, all notes under the root are the sub space of the S. In this paper, they are regarded as gird space  $Gi = (g_{i1}, g_{i2}, ..., g_{id})(g_{ij} \subseteq S_j, 1 \le j \le d)$ .

Secondly, we put data point into its corresponding leaf note one by one by indexing HS-Tree from top to bottom. If some leaf note contains 2\*MinPts points (the split condition of the definition 5), this note *Gi* will be split into two child grids from the center of the MAX( $g_{ij}$ )( $g_{ij} \in Gi$ ,  $1 \le j \le d$ ). Then, the neighbors of the parent note will be assigned to its corresponding child grids by the definition 3. The split process also will be run on child grids iteratively until no any grid satisfies the split condition. The split procedure ends when all data points are put into the HS-Tree and all leaf nodes do not reach the split condition.

Finally, for optimizing the border grids of clusters, we will split the leaf node again whose level deviation degree is equal and larger than the threshold.

The algorithm procedure is described as follows:

```
Initialize HS – Tree with whole dataspace as the root
GridStructureBuild(HSTree hs, Dataset d, MinPts){
1. For each point p in d {
      find corresponding leaf note g for p on hs;
2.
      put p into g;
3.
      If SplitCon(D(g) = 2 * MinPts) is true{
4
5
          HalfSplit(g);
6
     }
7
  }
8
   For each leaf note g on hs{
9
      If \quad \sigma_{_{L(G_i)}} \geq \gamma \{
          H a lfS p lit(g);
10.
11.
12. }
13. Return hs;
}
HalfSplit(HSTree g){
1. Select a dimension s with the largest size in all dimensions of g;
2. Split g into g1 and g2 from the center of s;
```

```
    Split g into gr and g2 from the center of s,
    Create neighbor grids for g1 and g2;
    if SplitCon(D(g1)=2*MinPts) is true{
    HalfSplit(g1);
    }
    if SplitCon(D(g2)=2*MinPts) is true{
    HalfSplit(g2);
    }
```

So, all leaf nodes and their connections on the HS-tree form our grid structure where the clustering will be done after the half-split algorithm.

## D. Clustering on Half-Split Grid Structure

The clustering pattern on our gird structure is similar with the basic grid-based method which generates clusters by finding their core girds and expanding them from their neighbors. But, the definitions of the core gird and neighbor grid is different with the existing grid-based methods.

For our gird structure, there could be two close or overlapped clusters with different grid size. So, it is not enough to define a dense grid as a core grid. Our core girds are not only the dense grids whose density is equal or larger than the MinPts, but also have the similar size with the neighbor girds. The similarity of the size can be indicated by the level deviation degree with the neighbor girds because the girds in the same level of the HS-Tree have the same size. The definition 5 shows the definition of our core grids.

Our neighbor grids are also different with the existing gird structure which has the uniform gird size so that the neighbor grids can be found by the serial number of the gird in ever dimension. The neighbor connections should be generated in the split process of the gird for locating the neighbor girds. The definition 3 shows the process about the production of neighbor grids.

So, our clustering algorithm is showed as follows:

Step 1: Find a core grid as an initial core area of a cluster.

Step 2: Expand the core area by merging neighbor core grids continuously until no core grid can be found around the core area.

Step 3: Generate a cluster by merging boundary girds around the core area.

Step 4: Repeat Step 1 to Step 3 until no core gird can be found. Step 5: The data points in the rest grids will be labeled as noises.

## III. EXPERIMENTS

In this section, we evaluate the performance of our half-split grid clustering algorithm on three datasets. Two datasets are frequently used by the existing grid-based clustering algorithms and we want to prove we also can do well in their datasets. The other one is our complex synthetic dataset which cannot be processed by other grid-based algorithms.

In experiment 1, we use two Chameleon datasets [17] t4.8k and t7.10k showed by Fig.5(a) and (b). The t4.8k has 8000 data points and 6 clusters with many noises. T7.10k has 10000 data points and 9 clusters with many noises. Each cluster has arbitrary shape and similar density in these datasets. Many existing gird-based algorithms can process this datasets well.

TABLE I PERFORMANCE COMPARISON OF GDCLU AND HALF-SPLIT GRID IN MILLISECOND

	t4.8k	t7.10k
Our algorithm	68	72
GDCLU	63	36

Fig.5(c) and (e) shows the space partition by using our algorithm. Fig.5(d) and (f) shows the results of our half-spilt grid clustering algorithm on the two datasets.

Table I shows the comparison with GDCLU on the speed.

So, from the above results, we can see our algorithm has the similar effect and speed on the datasets experimented by other grid-based algorithms. But, our half-spilt gird structure has more advantages than the uniform grid structure because the



Fig. 5. Experiments on dataset t4.8k and t7.10k with MinPts = 4 and  $\gamma$ =0.8

sparse space will be partitioned as one big grid rather than many empty girds partitioned by existing grid-based algorithm. The advantage of the structure could be reflected in sparse high dimension space because the data space is partitioned optimally. So, the next step of our research is going to improve our algorithm to process high dimension datasets efficiently.

In experiment 2, we create a complex synthetic dataset which has the embedded clusters structure with big difference

on the density. The dataset showed in Fig.6(a) consists of 5 clusters and has 984 data points. Two middle clusters and two small clusters are embedded in a big cluster. The whole space of each embedded cluster is smaller than the interval space among data points of the big cluster. Fig.6(b) shows a uniform gird structure with a critical size on the synthetic dataset. If the uniform size is smaller than the critical size, then all the points of the big cluster will be regarded as noises. Fig.6(c) shows the above situation. On the contrary, if the uniform size is



Fig. 6. Experiments on our synthetic dataset

equal or larger than the critical size, then two middle clusters and two smaller clusters will be regarded by existing gridbased clustering algorithms as two clusters. Our half-split gird structure which generates appropriate grid structure in different density space are showed in Fig.6(d) and the clustering results are showed in Fig.6(e). From the experiments, we see our half-split gird clustering algorithm can process arbitrary density and shape dataset. Moreover, the algorithm can utilize the data space optimally according to the density and it could be extend to process space optimization and segmentation problems in future our work.

However, the high dimension space is still the weakness of our algorithm. When producing our gird structure on a high dimension space, our algorithm will take much time on splitting a sparse grid space. We call this split shocking problems. Solving the above problem is also our next work.

## IV. CONCLUSIONS

In this paper, we introduce a new half-split grid clustering algorithm on multiple density dataset. The algorithm partitions data space into the gird structure which has the corresponding suitable size in different density areas by simulating the behavior of the cell division. So, the half-split gird algorithm can generate clusters efficiently on arbitrary multi-density dataset. The experiments show our algorithm can deal with the synthetic dataset which existing grid-based methods cannot deal with.

But our algorithm still cannot process high dimension space well. Especially, it will take much time on space split process when data are discrete in high dimension space.

In the future research, we will improve our algorithm for processing high dimension dataset. We also want to apply our space division ideas in other fields such as image processing and classification by extracting optimal space.

#### REFERENCES

- J. Han, M. Kamber and J. Pei, "Data Mining: Concepts and Techniques," 2<sup>nd</sup> ed. Diane Cerra, 2006
- [2] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. 5th Berkeley Symp. Math. Stat. Probab.,L. M. L. Cam and J. Neyman, Eds. Berkeley, CA: Univ. California Press, 1967, vol. I.
- [3] K. Krishna and M. Narasimha Murty, "Genetic K-means algorithm," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 29, no. 3, pp. 433– 439,Jun. 1999.
- [4] S. Guha, R. Rastogi and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. of .the ACM SIGMOD Conf.*, 1998, pp.73-84
- [5] S. Guha, R. Rastogi and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," in *Information Systems 25(5), 2000,* pp.345-366.

- [6] M. Ester, H. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc.* of .the 2<sup>nd</sup> KDD Conf., 1996, pp.226-231
- [7] L. Ertoz, M. Steinbach and V. Kumar, "A new shared nearest neighbor clustering algorithm and its applications" in *Proc. Workshop Clustering High Dimensional Data Appl.*, 2002, pp. 105–115
- [8] B. Borah and D. Bhattacharyya, "A clustering technique using density difference," in signal proc. of the Communications and Networking, 2007, pp585-588
- [9] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatialtemporal data," in *Data and Knowledge Engineering*, 60(1), 2007, pp208-221
- [10] W. Wang, J. Yang and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proc. of .the 23<sup>th</sup> VLDB Conf.*, 1997, pp.186-195
- [11] E. Schikuta, "Grid-Clustering: A hierarchical clustering method for very large data sets," in Technical Report TR-CRPC No. 93358, Center for Research on Parallel Computation, Rice University, Houston, 1993

- [12] X. Chen, Y. Min, Y. Zhao and P. Wang, "GMDBSCAN: Multi-density DBSCAN clustering based on grid," in *Proc. of .the δ<sup>th</sup> ICEBE Conf.*, 2008, pp.780-783
- [13] J. Ren, L. Meng and C.Hu, "CABGD: An Improved Clustering Algorithm Based on Grid-Density," in Proc. of .the 4<sup>th</sup> ICIC Conf., 2009, pp.381-384
- [14] H. Darong and W. Peng, "Grid-based DBSCAN algorithm with referential parameters," in *Proc. of the ICAPIE Conf.*, 2012, pp.1166-1170
- [15] DR. Edla and PK. Jana, "A grid clustering algorithm using cluster boundaries," in Proc. of the WICT Conf., 2012, pp.254-259
- [16] G. Esfandani, M. Sayyadi and A. Namadchian, "GDCLU: a new griddensity based CLUstring algorithm," in Proc. of .the 13<sup>th</sup> ACIS Conf., 2012, pp.102-107
- [17] Chameleon Data Sets. http://glaros.dtc.umn.edu/gkhome/cluto/cluto/ download.