sciendo

JAISCR, 2021, Vol. 11, No. 4, pp. 319 – 330
10.2478/jaiscr-2021-0019

A NOVEL GRID-BASED CLUSTERING ALGORITHM

Artur Starczewski[1,*], Magdalena M. Scherer[2], Wojciech Książek[3], Maciej Dębski[4,5], Lipo Wang[6]

[1]*Department of Intelligent Computer Systems, Czestochowa University of Technology al. Armii Krajowej 36, 42-200 Czestochowa, Poland*

[2]*Faculty of Management, Czestochowa University of Technology, Poland*

[3]*Faculty of Computer Science and Telecommunications, Cracow University of Technology, Warszawska 24, 31-155 Kraków, Poland*

[4]*Management Department, University of Social Sciences, 90-113 Łódź, Poland*

[5]*Clark University, Worcester, MA 01610, USA*

[6]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*

[*]*E-mail: artur.starczewski@pcz.pl*

Submitted: 24th January 2021; Accepted: 23rd September 2021

Abstract

Data clustering is an important method used to discover naturally occurring structures in datasets. One of the most popular approaches is the grid-based concept of clustering algorithms. This kind of method is characterized by a fast processing time and it can also discover clusters of arbitrary shapes in datasets. These properties allow these methods to be used in many different applications. Researchers have created many versions of the clustering method using the grid-based approach. However, the key issue is the right choice of the number of grid cells. This paper proposes a novel grid-based algorithm which uses a method for an automatic determining of the number of grid cells. This method is based on the k_{dist} function which computes the distance between each element of a dataset and its kth nearest neighbor. Experimental results have been obtained for several different datasets and they confirm a very good performance of the newly proposed method.

Keywords: data mining, grid-based clustering, grid structure

1 Introduction

Clustering refers to grouping objects into meaningful clusters so that the elements of a cluster are similar, whereas they are dissimilar in different clusters. Data clustering is a very useful technique used in a number of fields, such as data mining, pattern recognition, spatial data analysis, and others. A wide of large collections of data presents a great challenge to clustering algorithms, so a lot of new different clustering algorithms and their configurations are being intensively developed, e.g. [8, 10, 11]. However, there is no clustering algorithm which creates the right clusters for all datasets. Moreover, the same algorithm can also produce different results depending on the input parameters applied. Therefore, cluster validation is also used to

assess the results of data clustering. Up to now, a number of authors have proposed different cluster validity indices or modifications of existing ones, e.g., [7, 20, 23, 24]. Generally, clustering algorithms can be divided into four categories including partitioning, hierarchical, density-based, and grid-based clustering. The first group includes partitioning algorithms such as e.g. *K-means*, *Partitioning Around Medoids* (*PAM*) [3, 28] or *Expectation Maximization* (*EM*) [16]. The original *K-means* algorithm is often used, but it cannot process datasets which include *noise* elements. In the next category, two approaches are described, i.e. agglomerative and divisive. The simple agglomerative clustering algorithms are e.g. the *Single-linkage* or *Complete-linkage*, and the *DIvisive ANAlysis Clustering* algorithm represents the divisive approach [17, 19]. On the other hand, the density-based algorithms can discover clusters of arbitrary shapes occurring in datasets and also can remove *noise* elements. In this category of algorithms, the *Density-Based Spatial Clustering of Application with Noise* (*DBSCAN*) is the most well-known algorithm [6]. This original algorithm has many various modifications and extensions, e.g. [2, 4, 5, 15, 21, 27]. The *DBSCAN* algorithm requires two input parameters, i.e. the *eps* and *MinPts*. The determination of these parameters is difficult and their right choice is a fundamental issue. In literature, some methods have already been proposed, e.g. [12]. The last category, i.e. the grid-based approach includes algorithms such as e.g. the *Statistical Information Grid-based* (*STING*) or *Wavelet-based Clustering* (*WaveCluster*) [18, 22, 26]. This kind of algorithms is very effective, because it can handle different types of datasets and also remove *noise* elements. However, the clustering results depend on the appropriate division of the data space into a certain number of cells. Then the algorithm performs required operations on the cells.

In this paper, a new grid-based clustering algorithm is presented. This algorithm makes it possible to discover clusters of arbitrary shapes in datasets and it also removes *noise* elements. Moreover, the number of grid cells is calculated based on an analysis of changes in the distances between each element of the dataset and its k-th nearest neighbor. This paper is organized as follows: Section 2 presents related papers. In Section 3 the new clustering algorithm is described in detail while Section 4 illustrates experimental results obtained on datasets. Finally, Section 5 presents conclusions.

2 Related works

In grid-based clustering algorithms, the data space is divided into a number of cells that form a grid, and then they perform clustering on the grid structure. These kinds of algorithms have an advantage over other approaches because they process grid cells and not all data elements. Thus, the time complexity is low and mostly depends on the number of grid cells. The quality of results of the grid-based clustering method depends on the right choice of the number of grid cells. A well-known grid-based algorithm is STING (*STatistical INformation Grid*)[26]. In this algorithm rectangular cells and a hierarchical structure are used. There are several levels of cells corresponding to different levels of resolution. Moreover, some statistical parameters such as mean, maximum, minimum and data distribution are computed. The next important grid-based algorithm is Wave Cluster [22], which is used to find clusters in large spatial datasets. First, a multidimensional grid is imposed on to the data space. The original features are transformed by the wavelet transform and next dense regions are found. The wavelet transform is a signal processing technique used to decompose a signal into a different frequency. In turn, the CLIQUE (*Clustering In QUEst*) recursive joins cells when the density of cells exceeds a given threshold [1]. Another clustering technique that combines the grid-based and density approaches to clustering datasets is the ASGC (*Axis Shifted Grid Clustering Algorithm*). This method uses two grid structures and density cells . The second grid structure is formed by shifting the coordinate axis and if the density of the cells exceeds the threshold, all the nearest cells are grouped to form clusters. Moreover, in the literature there are also described modifications of the DBSCAN algorithm based on a grid-based approach [14, 9, 13], which usually require a smaller time complexity than the original DBSCAN algorithm. However, this kind of algorithms often suffers from a few problems, i.e. neighbor explosion and merging redundancies. The GDCF (*Grid-based DBSCAN with Cluster Forest*) algorithm is proposed to address these problems [2]. This algorithm can

Algorithm 1: The new grid based clustering algorithm (GBCN)

Input : *gMatrix* - this matrix contains the same number of *cells* as the grid and the number of data items belonging to a cell of the grid is written to the respective *cell* of the matrix, *cellVector* - this vector contains the numbers of the non-empty cells of *gMatrix*

Output: a set of clusters saved in the *clustersList*

```
1  Make a copy of gMatrix : cMatrix ← gMatrix ;
2  Create empty vectors : tVector, t, cellsAround ;
3  Create an empty list of clusters : clustersList ← ∅ ;
4  Initialize the number of clusters : c ← 0 ;
5  while length of cellVector > 0 do
6      Increase the number of clusters : c ← c + 1 ;
7      Add the number of the non-empty cell of gMatrix to clustersList [c] from cellVector [1] :
         clustersList[ c] ← cellVector[1] ;
8      Delete the content of the cell of gMatrix indicated by cellVector [1] :
         gMatrix[ cellVector[1]] ← 0 ;
9      Add to tVector the numbers of the non-empty gMatrix cells located around the cell indicated
         by cellVector[1] ;
10     if length of tVector > 0 then
11         Add the numbers of the gMatrix cells from tVector to clustersList [c]:
             clustersList[c] ← clustersList[c] + tVector ;
12     Remove the first element from cellVector: cellVector [-1] ;
13     while length of tVector > 0 do
14         Delete the content of the gMatrix cell with the numbers given by tVector :
             gMatrix[tVector] ← 0 ;
15         foreach poz ∈ tVector do
16             Find the numbers of the non-empty gMatrix cells around the cell indicated by the poz,
                 the numbers are saved in the cellsAround vector ;
17             if length of cellsAround > 0 then
18                 Add the numbers of the gMatrix cells from the cellsAround to the t vector :
                     t ← t + cellsAround ;
19                 Delete the content of the gMatrix cell with the numbers given by cellsAround :
                     gMatrix[cellsAround] ← 0 ;
20                 Delete the numbers from the cellsAround vector : cellsAround ← ∅ ;
21             Remove the element which includes the number of poz from the cellVector;
22         Update the content of tVector : tVector ← t ;
23         Delete the numbers from t vector : t ← ∅ ;
24         if length of tVector > 0 then
25             Add the numbers of the gMatrix cells from tVector to clustersList [c] :
                 clustersList[c] ← clustersList[c] + tVector ;
```

reduce an excessive amount of redundant merging computation.

3 The novel grid based clustering algorithm

In this Section, a detailed description of the novel grid-based clustering algorithm called *Grid-Based Clustering with Noise* (GBCN) is presented. This new algorithm applies some concepts taken from the DBSCAN algorithm, e.g. it uses the radius of the neighborhood, which is also very important in the DBSCAN algorithm. Usually, this parameter is determined by the distance function denoted as the k_{dist}. This function calculates distances between each element of a dataset and its k-th nearest neighbor. The number of the k-th nearest neighbors is an input parameter of the k_{dist} function and it is marked by k.

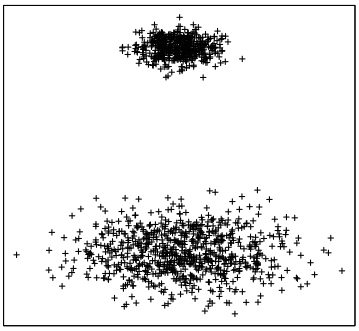


Figure 1. An example of a 2-dimensional dataset consisting of two clusters.

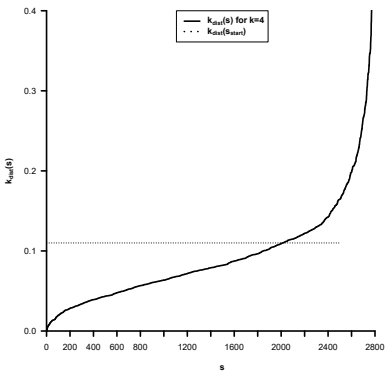


Figure 2. Sorted values of the k_{dist} function ($k = 4$) for the example dataset.

As mentioned above, in the grid-based algorithms, the grid structure is created, i.e. the data space is partitioned into a finite number of cells. The size of a grid is the key parameter of this kind of algorithms. Sometimes the grid structure is defined arbitrarily, e.g. the number of cells equals 50x50 or 60x60 can be the right choice for some datasets, but in the new proposed approach the k_{dist} function is used to determine the size of the grid. Moreover, similar to the DBSCAN algorithm, a minimal number of elements that create clusters is defined and it is equal to 5. Such value is often used in various experiments where the DBSCAN algorithm is applied. The clusters with the number of elements smaller than 5 are regarded as *noise*. Moreover, the k parameter, i.e. the number of the nearest neighbors for the k_{dist} function is equal to 4 if the minimal number of elements in a cluster is 5. Let us denote the 2-dimensional dataset by X and $x_m \in X$, where $m = 1, ..., n$. The number of elements n of X is also specified by $|X|$.

For a grid-based algorithm the key issue is the choice of the size of a grid, i.e. the number of cells in the grid structure. In this new proposed approach, first the minimal and maximal values are calculated in each dimension of the X and they are marked as x_{min1}, x_{min2}, x_{max1} and x_{max2}, respectively. Next, the two ranges are defined as follows:

$$\begin{aligned} range_1 &= x_{\max 1} - x_{\min 1} \\ range_2 &= x_{\max 2} - x_{\min 2} \end{aligned} \tag{1}$$

The number of cells in the grid is calculated based on the $range = \max\{range_1, range_2\}$ parameter and the k_{dist} function. It equals $v_c * v_c$, where v_c is the number of *intervals* and it is defined as follows:

$$v_c = \frac{range}{k_{dist}(s_{start})} \tag{2}$$

where the value of the $k_{dist}(s_{start})$ function is calculated based on the sorted values of the k_{dist} function. Let us use an example artificial dataset to explain in detail the way of determining of s_{start}. For instance, Figure 1 shows an example of a 2-dimensional data set consisting of two clusters which contain 200 and 500 elements, respectively. Moreover, elements of the data set are scaled (by the scale() function from the R package). The k_{dist} function (for the k=4) is used to determine all the distances between each element of the dataset and its k-th nearest neighbors. Then, the results are sorted in an ascending order,

and Figure 2 presents these results. It can be observed that there is a place called the "*knee*", where the start value of the "*knee*" is marked by the dotted line. This value is defined by $k_{dist}(s_{start})$, and s_{start} is expressed as follows [25]:

$$s_{start} = |S_{dist}| - n \tag{3}$$

where the set S_{dist} includes all the calculation distances by the the k_{dist} function, $|S_{dist}|$ is the number of the elements of S_{dist}, and n is the number of the elements of the X dataset.

Thus, in this new method the data space is divided into the number of cells equal to $v_c * v_c$. For instance, the v_c value is 55 for the example dataset presented in Figure 1, and the number of all the cells equals 3025 (55x55). Then, each element of the dataset is mapped into the grid.

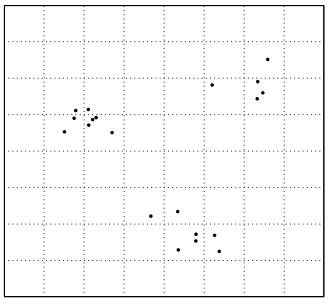


Figure 3. An example of a data space divided into grid.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	**1**	0
0	**1**	**1**	0	0	**1**	**3**	0
0	**2**	**4**	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	0	0	**3**	**2**	0	0
0	0	0	0	0	0	0	0

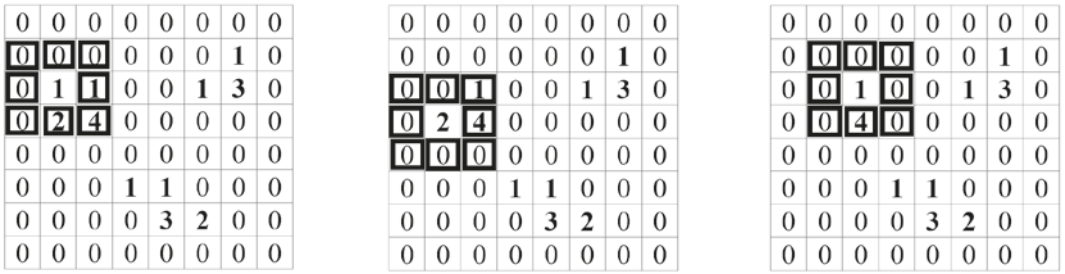
Figure 4. An example of *gMatrix*.

It is worth noting that sometimes the distances between clusters can be very small (see Figure 6(g)). In such cases, a grid-based clustering algorithm can put all the data elements in one cluster. This problem can be solved by increasing the number of grid cells. Thus, if the new proposed grid-based clustering algorithm creates one cluster, the number of cells is increased, i.e. v_c is doubled ($v_c = v_c * 2$) and the algorithm is run again.

The workflow of the new grid-based algorithm (GBCN) is presented by **Algorithm 1**. First, the input parameters are defined, i.e. *gMatrix* - this matrix contains the same number of the cells as the grid and the number of data items belonging to a cell of the grid is written to the respective cell of the matrix, *cellVector* - this vector contains the numbers of the non-empty cells of *gMatrix*.

Figure 3 presents an example of a dataset, where the data space is divided into a grid. On the other hand, Figure 4 shows *gMatrix*, which has 8 rows, 8 columns and eleven non-empty cells. This matrix maps data items from the grid (Figure 3). The cells of the matrix can be marked by m_{ij}, where i-row and j-column are from 1 to 8 and, e.g. the $m_{3,2}, m_{4,2}, m_{3,3}$ and $m_{4,3}$ cells are non-empty and their values are equal to 1,2,1 and 4, respectively. The next input parameter, i.e. *cellVector* includes the indices of non-empty cells of *gMatrix*. For example, the {(3,2), (4,2), (3,3), (4,3), (6,4), (6,5), (7,5), (3,6), (7,6), (2,7), (3,7)} indices make it possible to indicate the non-empty cells of *gMatrix* (see Figure 4). However, *cellVector* may use another way of the indication of the cells,. e.g. it can be base on cell numbering. The cells of the successive columns of *gMatrix* can be numbered in the following way: the cells indicated by $m_{1,1}, m_{2,1}, ..., m_{8,1}$ have numbers $1, 2, ..., 8$, the cells indicated by $m_{1,2}, m_{2,2}, ..., m_{8,2}$ have numbers $9, 10, ..., 16$, and so on. Such method of cell numbering is used in the new algorithm.

When the new algorithm starts, the *cMatrix* copy of *gMatrix* must be made, because it is used to determine the location of data elements in the grid after the algorithm is finished. Next, the empty *tVector*, *t*, *cellsAround* vectors and the empty *clustersList* list are created (see lines 1,2 and 3 in the workflow). The *clustersList* list saves the cell numbers belonging to the clusters, the temporary *tVector*, *t* and *cellAround* vectors are necessary during the algorithm operation. Initially the number of clusters c equals 0 (line 4). The *while* loop (line 5) loops through a block of code as long as the length of the *cellVector* is greater than 0. In this block the number of cluster c is increased by one



(A) (B) (C)

Figure 5. Modification of the content of the *gMatrix* example by the new algorithm.

(line 6) and next the number of the cell indicated by $cellVector[1]$ is added to $clustersList[c]$ (line 7). In line 8, the content of the cell of *gMatrix* indicated by $cellVector[1]$ is deleted, i.e. it equals 0. Then, the numbers of the non-empty cells of *gMatrix* located around the cell indicated by $cellVector[1]$ are added to *tVector* (line 9).

Figure 5 shows the cells of *gMatrix* located around a few selected cells. They are marked with bold squares, e.g. in Figure 5(A), the numbers of the cells located around cell 11 (it includes 1) are 2,3,4,10,12,18,19, and 20. However, only cells 12, 19 and 20 are non-empty and numbers 12, 19 and 20 can be added to *tVector*.

If the length of $tVector > 0$, than the numbers of cells from this vector are added to $clustersList[c]$ and then the content of $cellVector[1]$ is deleted (lines 11-12). The next *while* loop (line 13) loops through a block of code as long as the length of *tVector* is greater than 0. And again, the content of the *gMatrix* cells is deleted (it equals 0) for the numbers of the cells indicated by *tVector* (line 14). It is necessary, because these cells cannot be used later. In the *foreach* loop, the numbers of the non-empty *gMatrix* cells around the cell indicated by *poz* are found (line 16) and they are saved in the *cellsAround* vector. Next, if the length of *cellsAround* is greater than 0, it is added to the *t* vector (line 18). The content of the *gMatrix* cells is deleted (it equals 0) for the numbers of cells indicated by *cellsAroundd* and then the numbers from *cellsAround* are deleted too (lines 19-20) . In line 21, the element which includes the number of *poz* is removed from *cellVector*. Moreover, the content of *tVector* is updated by *t* (lines 22) and the numbers of cells are deleted from *t* (line 23). If the length of *tVector* is greater than 0, the cell numbers from *tVector* are added to $clustersList[c]$ (line 25). The *while* loop loops ends when *tVector* equals 0. When the length of *cellVector* is greater than 0, the new cluster is created, i.e. c increases by 1, and all the process is repeated until $cellVector = 0$ and then *gMatrix* has only empty cells. When the algorithm is finished, all the clusters are written into *clustersList* and the number of clusters equals the length of the list. As mentioned above, *gMatrix* doesn't include any values, but for each cluster the numbers of the cells can be calculated based on *cMatrix* and *clustersList*. Clusters including fewer than 5 members are regarded as *noise*.

Figure 5 shows the modification of the content of the example matrix by the new algorithm. At the start, the first non-empty cell is chosen, i.e. the cell of number 11 (includes 1) and the numbers of the adjacent non-empty cells are determined, i.e. 12, 19 and 20 (see A). Next, the content of cell 11 is removed (it equals 0). Then, for the number of non-empty cells, i.e. cell 12 (it includes 2), the numbers of the adjacent non-empty cells are determined, i.e. cells 19 and 20 (see B). And again, the content of cell 12 is removed (it equals 0). Next, the number of the non-empty cell is 19 (it includes 1) and the adjacent non-empty cell is cell 20 (see C). The content of cell 19 is removed (it equals 0). This process is continued until the example matrix is empty, and all the clusters are saved.

In the next Section, the results of the experimental study is presented to confirm the effectiveness of this new algorithm.

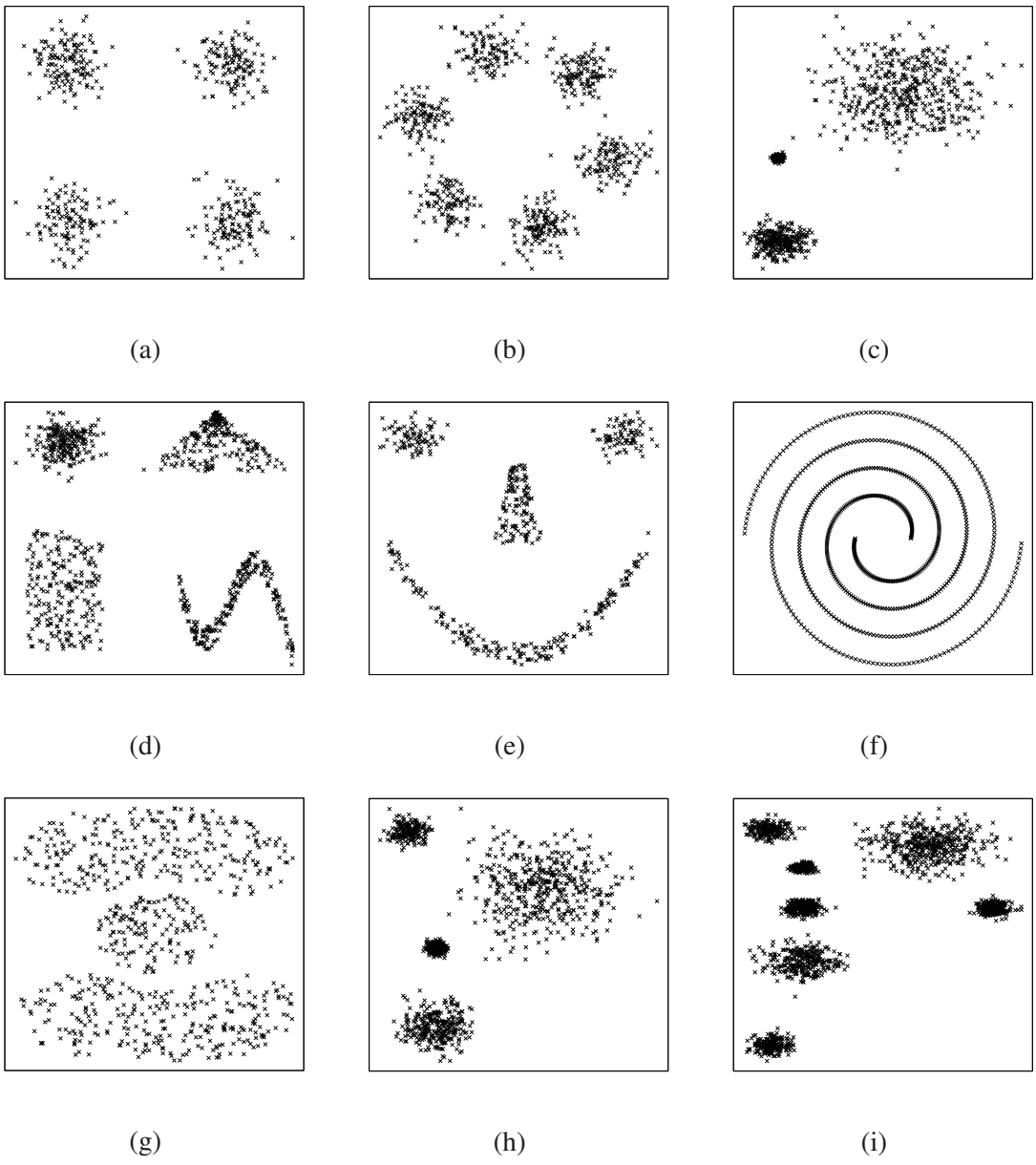


Figure 6. Examples of 2-dimensional artificial datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9.

4 Experimental study

In this Section, several experiments are described which were conducted on 2-dimensional artificial datasets using the *GBCN* algorithm. It is worth noting that this algorithm can recognize clusters with arbitrary shapes and different sizes.

Table 1. A detailed description of the artificial datasets.

Datasets	No. of elements	Clusters
Data 1	500	4
Data 2	700	6
Data 3	1000	3
Data 4	900	4
Data 5	500	4
Data 6	700	2
Data 7	700	3
Data 8	1200	4
Data 9	1950	7

Table 2. Results of clustering the artificial dataset by the *GBCN* algorithm.

datasets	number of *intervals*	number of clusters	number of *noise elements*
Data 1	37	4	5
Data 2	44	6	6
Data 3	64	3	17
Data 4	59	4	2
Data 5	51	4	2
Data 6	44	2	0
Data 7	64	3	9
Data 8	71	4	14
Data 9	114	7	25

Moreover, the number of grid cells is determined automatically. In the experiments are used various artificial datasets, and visual inspection is also used for the evaluation of the accuracy of the *GBCN* algorithm. Additionally, this algorithm is compared to the well-known *CLIQUE* algorithm.

Table 3. Results of clustering the artificial dataset by the *CLIQUE* algorithm

datasets	number of *intervals*	density threshold	number of clusters
Data 1	37	0.02	4
Data 2	40	0.01	6
Data 3	40	0.1	3
Data 4	30	0.05	4
Data 5	15	0.1	4
Data 6	15	0.03	2
Data 7	35	0.01	3
Data 8	26	0.09	4
Data 9	36	0.05	7

4.1 Datasets

Table 1 shows a detailed description of nine 2-dimensional datasets. They are called *Data* 1, *Data* 2, *Data* 3, *Data* 4, *Data* 5, *Data* 6, *Data* 7, *Data* 8 and *Data* 9, respectively. As mentioned above, they contain varied numbers of clusters (from 2 to 7 clusters) and data elements. These datasets are presented in Figure 6. It can be observed that the shapes and distances between the clusters are very different, i.e. some of the clusters are very close and far others are from each other. For instance, in *Data* 3 the elements create three clusters with different sizes, *Data* 4 contains elements that create Gaussian, square, triangle, and wave shapes, and *Data* 6 is the so-called spirals problem, where the points are on two entangled spirals.

4.2 Experiments

In this Section, the evaluation of the performance of the new algorithm *GBCN* is shown.

As mentioned above, the choice of the number of cells is very important in grid-based algorithms. In this new method, this parameter is determined automatically and it is described in Section 3. It is worth noting that some elements of a dataset create *noise* and should not be considered. For instance, in the *DBSCAN* algorithm, the *MinPts* parameter specifies the minimum number of elements that can form a cluster. It is usually 5 because such a choice of the number of elements guarantees creation of clusters with different shapes and sizes. In the *GBCN* algorithm, the minimal number of elements is also equal to 5. In conducted experiments the 2-dimensional artificial datasets are used, i.e.: *Data* 1,

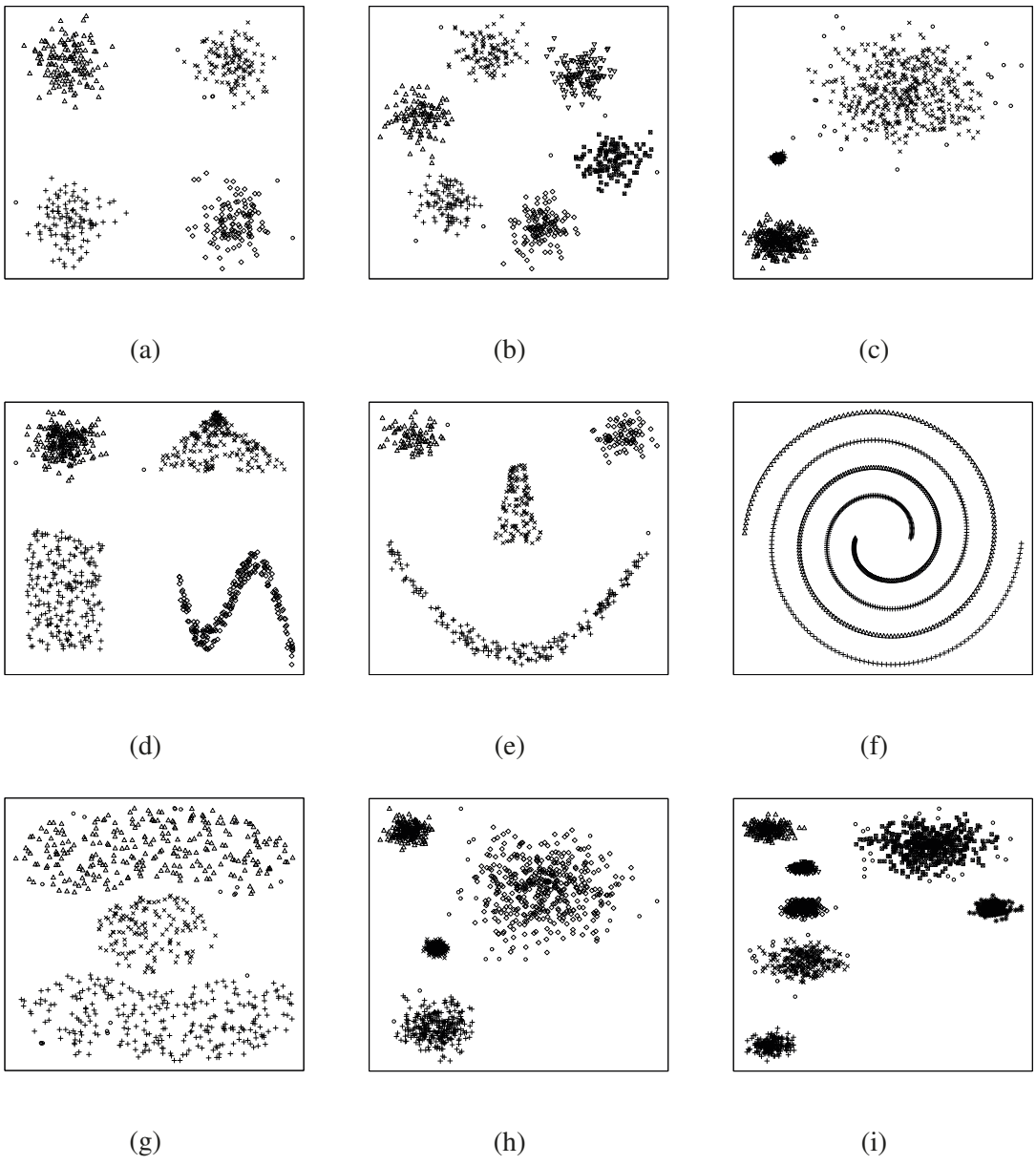


Figure 7. Results of the *GBCN* clustering algorithm for 2-dimensional datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9.

Data 2, *Data* 3, *Data* 4, *Data* 5, *Data* 6, *Data* 7, *Data* 8 and *Data* 9 sets. When the number of grid cells is specified by the new method, the *GBCN* algorithm is used to cluster the artificial datasets. Figure 7 shows the results of the algorithm, where each cluster is marked with different signs. It should be noted that despite the fact that the differences of distances and shapes between clusters are significant, all the datasets are clustered correctly by the clustering algorithm. Moreover, the data elements classified as the *noise* are marked with a circle, and their number is small in all the datasets. Table 2 presents detailed information about the results of experiments conducted with the use of the *GBCN* algorithm. The table shows the number of used *intervals*, the number of received clusters, and the number of the *noise elements* for individual data sets. Generally, only one parameter must be calculated for this algorithm, i.e. the number of *intervals* and it is calculated automatically. On the other hand, the well-known CLIQUE algorithm was used in experiments on these artificial datasets and it required two input parameters, i.e. the number of *intervals* and the density threshold. In this case, the calculation of the parameters is not easy. Table 3 presents the results of clustering for the *CLIQUE* algorithm, and the parameters which were determined experimentally.

5 Conclusions

In this paper, a new grid-based algorithm *GBCN* is proposed. This algorithm uses input parameters., i.e. the number of *intervals* and the minimal number of elements in clusters. The first parameter is determined automatically, i.e by the k_{dist} function, which computes the distance between each element of a dataset and its kth nearest neighbor. The distances are used to calculate the size of the intervals for each dimensional data and then make it possible to determine the right number of grid cells. The next parameter defines the minimal number of elements in clusters and it is equal to 5. It should be noted that such value is very often used by the DBSCAN algorithm. In the conducted experiments, several 2-dimensional datasets were used in which the number of clusters, sizes, and shapes varied within a wide range. From the perspective of the conducted experiments, this new grid-based algorithm and the method of determining the number of *intervals* are very useful. All the presented results confirm the high efficiency of the newly proposed approach.

Acknowledgements

The paper is financed under the program of the Polish Minister of Science and Higher Education under the name ”Regional Initiative of Excellence” in the years 2019-2022; project number 020/RID/2018/19; the amount of financing PLN 12,000,000.00.

References

[1] Agrawal R., Gehrke J., Gunopulos D., Raghavan P.: Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec., vol. 27, pp. 94-105 (1998).

[2] Boonchoo T., Ao X., Liu Y., Zhao W., He Q.: Grid-based DBSCAN: Indexing and inference. Pattern Recognition, Vol. 90, pp.271-284 (2019).

[3] Bradley P., Fayyad U.: Refining initial points for k-means clustering. In Proceedings of the fifteenth international conference on knowledge discovery and data mining, New York, AAAI Press, pp. 9-15 (1998).

[4] Chen Y., Tang S., Bouguila N., Wanga C., Du J., Li H.: A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data. Pattern Recognition, Vol.83, pp.375-387 (2018).

[5] Darong H., Peng W.: Grid-based dbscan algorithm with referential parameters. Physics Procedia, 24, Part B, pp.1166-1170 (2012).

[6] Ester M., Kriegel H.P, Sander J., Xu X.: A density-based algorithm for discovering clusters in large spatial databases with noise, In Proceeding of 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226-231 (1996).

[7] Fränti P., Rezaei M., Zhao Q.: Centroid index: Cluster level similarity measure. Pattern Recognition, Vol. 47, Issue 9, pp. 3034-3045 (2014).

[8] Gabryel M.: Data Analysis Algorithm for Click Fraud Recognition. Communications in Computer and Information Science, Vol 920, pp.437-446 (2018).

[9] Gan J. , Tao Y.: Dbscan revisited: mis-claim, unfixability, and approximation. SIGMOD (2015).

[10] Grycuk R., Najgebauer P., Kordos M., Scherer M., Marchlewska A.: Fast Image Index for Database Management Engines. Journal of Artificial Intelligence and Soft Computing Research, Vol. 10, Issue 2, pp.113 - 123 (2020)

[11] Hruschka E.R., de Castro L.N., Campello R.J.: Evolutionary algorithms for clustering gene-expression data, In: Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on. pp. 403-406, IEEE (2004).

[12] Karami A., Johansson R.: Choosing DBSCAN Parameters Automatically using Differential Evolution. International Journal of Computer Applications, Vol. 91, pp.1-11 (2014)

[13] Kumar K.M., Reddy A.R.M.: A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method. Pattern Recognition, vol 58, pp.39-48 (2016).

[14] Liu F., Wen P. and Zhu E.: Efficient Grid-based Clustering Algorithm with Leaping Search and Merge Neighbors Method. IOP Conf. Series: Materials Science and Engineering, vol. 242 (2017)

[15] Luchi D., Rodrigues A.L., Varejao F.M.: Sampling approaches for applying DBSCAN to large datasets. Pattern Recognition Letters, Vol.117, pp.90-96 (2019).

[16] Meng X., van Dyk D.: The EM algorithm - An old folk-song sung to a fast new tune. Journal of the Royal Statistical Society, Series B (Methodological) Vol. 59, Issue 3, pp. 511-567 (1997).

[17] Murtagh F.: A survey of recent advances in hierarchical clustering algorithms. Computer Journal, Vol. 26, Issue 4, pp. 354-359 (1983).

[18] Patrikainen A., Meila M.: Comparing Subspace Clusterings, IEEE Transactions on Knowledge and Data Engineering, Vol.18, Issue 7, pp.902-916 (2006).

[19] Rohlf F.: Single-link clustering algorithms. In: P.R Krishnaiah and L.N. Kanal (Eds.), Handbook of Statistics, Vol. 2, pp. 267-284 (1982).

[20] Sameh A.S., Asoke K.N.: Development of assessment criteria for clustering algorithms. Pattern Analysis and Applications, Vol. 12, Issue 1, pp. 79-98 (2009).

[21] Shah G.H.: An improved dbscan, a density based clustering algorithm with parameter selection for high dimensional data sets. In Nirma University International Engineering,(NUiCONE) pp. 1-6 (2012).

[22] Sheikholeslam G., Chatterjee S., Zhang A.: WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. The International Journal on Very Large Data Bases, Vol.8 Issue 3-4, pp.289-304 (2000).

[23] Shieh H-L.: Robust validity index for a modified subtractive clustering algorithm. Applied Soft Computing, Vol. 22, pp. 47-59 (2014).

[24] Starczewski A.: A new validity index for crisp clusters. Pattern Analysis and Applications, Vol.20, Issue 3, pp. 687-700 (2017).

[25] Starczewski A., Cader A.: Determining the Eps Parameter of the DBSCAN Algorithm Lecture Notes in Computer Science, Vol. 11509, pp. 420-430 (2019).

[26] Wang W., Yang J., Muntz R.: STING: A Statistical Information Grid Approach to Spatial Data Mining. VLDB '97 Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 186-195 (1997).

[27] Viswanath P., Suresh Babu V.S.: Rough-dbscan: A fast hybrid density based clustering method for large data sets. Pattern Recognition Letters, Vol. 30 Issue 16, pp.1477-1488 (2009).

[28] Zalik K.R.: An efficient k-means clustering algorithm. Pattern Recognition Letters, Vol.29, Issue 9, pp.1385-1391 (2008).

Artur Starczewski received the M.Sc. degree in electrical engineering from Częstochowa University of Technology, Poland. In 2000, he received his Ph.D. degree in computer science from the AGH University of Science and Technology, Cracow, Poland. He is an Assistant Professor in the Department of Computer Engineering, Częstochowa University of Technology. His research interests include data clustering, data mining, and pattern recognition. He has authored many research papers on fuzzy systems and clustering algorithms.

Magdalena Scherer received her M.Sc. degree in computer science from the Czestochowa University of Technology, Poland, in 2008 and her Ph.D. in management in 2016 from the same university. Currently, she is an assistant professor at Czestochowa University of Technology. Her present research interests include machine learning for prediction and classification.

Wojciech Książek received his B.Eng. and M.Sc. degrees in computer science at the Cracow University of Technology, Kraków, Poland, in 2019. Since then, he has been working as a research and teaching assistant at the Department of Computer Science, Faculty of Computer Science and Telecommunications of the Cracow University of Technology. His research interests include machine learning, deep learning, genetic algorithms, classification, ensemble learning, data clustering, and their applications in medicine.

Maciej Dębski, Master in Management and Master in International Relations (SGH Warsaw School of Economics) Ph.D. in management (SGH Warsaw School of Economics). His research interests and practical aspects are focused on the use of marketing tools, especially in tourism and applications of artificial intelligence methods in marketing. Professionally working at University of Social Sciences as a Head of Tourism and Marketing Department.

Dr. **Lipo Wang** received the Bachelor degree from National University of Defense Technology (China) and Ph.D. from Louisiana State University (USA). His research interest is artificial intelligence/machine learning with applications to communications, image/video processing, biomedical engineering, and data mining. He has authored 320 papers, of which 110 are in journals. He has authored 2 monographs and edited 20 books. His work has been cited 7,800 times in Google Scholar. He was/will be keynote speaker for 40 international conferences. He was President of the Asia-Pacific Neural Network Assembly (APNNA) and received the APNNA Excellent Service Award.