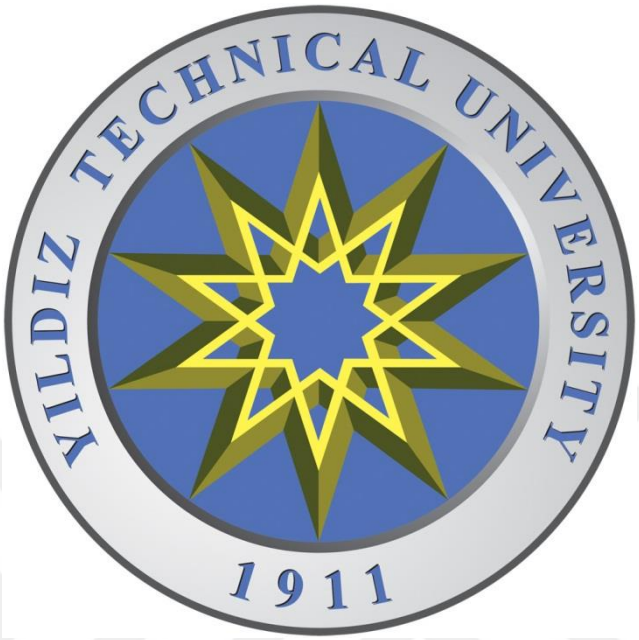


GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MSc. THESIS

NOVEMBER, 2016

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FUZZY NEIGHBORHOOD GMDBSCAN USING REPRESENTATIVE POINTS ALGORITHM

ABDALLAH R.S. MEKKY

MSc. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING

ADVISER
ASSIST. PROF. DR. OĞUZ ALTUN

İSTANBUL, 2016

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FUZZY NEIGHBORHOOD GMDBSCAN USING REPRESENTATIVE POINTS ALGORITHM

A thesis submitted by Abdallah R.S MEKKY in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 28.10.2016 in Department of Computer Engineering, Computer Engineering Program.

Thesis Adviser

Assist. Prof. Dr. Oğuz ALTUN

Yıldız Technical University

Approved By the Examining Committee

Assist. Prof. Dr. Oğuz ALTUN

Yıldız Technical University _____

Prof. Dr. Nizamettin AYDIN

Yıldız Technical University _____

Prof. Dr. Selim AKYOKUŞ

Doğuş University _____

ACKNOWLEDGEMENTS

All thanks and gratitude to Allah, that by his entire well I have done with Master degree. And have this work finished. Also, I am so thankful to our road instructor Assistant Prof. Dr. Oğuz Altun for being the advisor for this work and hold the responsibility during the period of the work in this thesis to reach these results. Best gratitude to my family who can wait for three years without me between them to achieve my goal to hold the master degree in computer engineering. All thanks to the Turkish government and Turkish people to have me as a guest between them.

November, 2016

Abdallah R.S Mekky

TABLE OF CONTENTS

LIST OF SYMBOLS

ε	The radius of a number of objects
C	Cluster
q	Group of point
p	Any point p
k-dist.	The distance in the neighbor of node
M	Set of data objects
k-th	Nearest neighbor for a point
n	Distance Volume
k	Number of elements in the distance
m	The fuzziness exponent
Jm	Objective function
xk	Feature vector
u_{ik}	Optimal membership values
v_i	Optimal cluster centers
X	Dataset
x_i	The i^{th}data object
D_{ik}	The squared distance between the k^{th}data object and i^{th}cluster center
β	The termination criterion between [0, 1]
R^s	Feature space of the dataset
$\rho()$	Dissimilarity between the i^{th}data object and j^{th}cluster center
V_{old}	Old set of medoids
$N_x(y)$	Neighborhood function
$d(x, y)$	Distance between x and y
$\mathrm{FN}(\mathrm{x}, \varepsilon)$	Fuzzy neighborhood set
$\mathrm{CardFN}()$	Fuzzy core point
ε_2	Threshould of density
ε_1	Radius of the grid
w^{max}	Maximum summation membership degree of points
S	Group of nodes
$Z_1(x_{ij}^*)$	Z-score Standardization method
α	Correction factor

LIST OF ABBREVIATIONS

CURE	Clustering Using Representatives
CD	Cell Density
calcMaxChange	Calculate Maximum difference in cluster membership
DBSCAN	Density-Based Clustering Method Based on Connected Regions with Sufficiently High Density.
DENCLUE	Clustering Based on Density Distribution Functions
Eps	The radius of a number of objects
FNDBSCAN	Fuzzy Neighborhood DBSCAN
FNGMDBSCAN-UR	Fuzzy Neighborhood Grid-Based Multi-Density DBSCAN
FCM	Fuzzy C-Mean algorithm
FCMdd	Fuzzy C-Medoids algorithm
GMDBSCAN	Grid-Based MDBSCAN
GMDBSCANUR	Grid-based MDBSCAN Using Representatives
GD	Grid Density
ITER	Iteration
MDBSCAN	Multi-Density DBSCAN
MinPts	The Minimum number of objects in the specific radius
MaxChange	Maximum Change in dataset
MinCard	Minimum Cardinality
NEps (q)	Neighbors of node q in its specified radius
OPTICS	Ordering Points to Identify the Clustering Structure "DENsity-based CLUstEring"
SP-Tree	Spanning Tree
VGrid	The Grid Neighborhood Volume
VEps	The Data Point’s Neighborhood volume
VCell	The Cell Volume

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABSTRACT

FUZZY NEİGHBORHOOD GRİD-BASED MULTİ-DENSİTY DBSCAN USİNG REPRESENTATİVE POİNTS

Abdallah R.S. MEKKY

Department of Computer Engineering

MSc. Thesis

Adviser: Assist. Prof. Dr. Oğuz ALTUN

Clustering process is considered one of the most important parts in data mining. One of the famous algorithms is Density-Based spatial Clustering of Application with Noise (DBSCAN). It is a density-based clustering algorithm that uses a crisp neighborhood function to calculate the neighbor sets. Its behavior depends on distance function between two data objects. Fuzzy clustering uses a fuzzy neighborhood function that allows a node in the dataset to have a membership degree at each point in the dataset. The neighborset for a node in fuzzy clustering depends on the contribution of the data objects around it.

In this work, we propose a new algorithm that combines the speed of DBSCAN with the accuracy of fuzzy clustering. The algorithm is a fuzzy neighborhood grid-based multi-density DBSCAN using representative points.It uses a grid-based neighborhood to separate the dataset into small nets. It uses fuzzy neighborhood function to create neighborhood sets and core points for clusters. Also to speed up the algorithm we use representative points from the data set. We divide the dataset into two parts: We use the first part for pre-clustering. It must be well scattered and represent the shape of the dataset. The second part is used for preprocessing and merging the clusters by the end of the clustering process. It is noticeable that FNGMDBSCAN-UR is more accurate than crisp DBSCAN with nested shapes and multi-dense datasets.

Key words: Clustering; Fuzzy neighborhood function; DBSCAN; FN-DBSCAN; GMDBSCAN-UR; Density-based clustering; Grid-based Clustering.

ÖZET

TEMSİLCİ NOKTA KULLANIMI İLE BULANIK KOMŞULUK IZGARA TABANLI ÇOK-YOĞUNLUKLU DBSCAN

Abdallah R.S. MEKKY

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: YRD. DOÇ. Dr. Oğuz ALTUN

Kümeleme işleme veri madenciliğinde önemli işlemlerden biridir. Çok bilinen algoritmalardan biri Gürültülü Uygulamanın Yoğunluk Temelli Uzamsal Kümelemesi (DBSCAN) algoritmasıdır. Bu, yoğunluk bazlı komşu kümelerini belirlemek için komşuluk fonksiyonu kullanan bir algoritmadır. Davranışı iki veri nesnesi arasındaki uzaklığı ölçmeye yerayan uzaklık fonksiyonuna bağlıdır. Bu algoritma bulanık kümeleme verisetindeki bir noktanın diğer tüm noktalar ile bir üyelik derecesi tutmasına izin veren bir bulanık komşuluk fonksiyonu kullanır. Bulanık kümelemede komşuluk kümesi çevresindeki veri nesnelerinin katkısına bağlıdır.

Bu çalışmada DBSCAN algoritmasının hızı ile bulanık kümelemenin doğruluğunu birleştiren bir algoritma öneriyoruz. FNGMDBSCAN-UR algoritması temsilci noktalar kullanan bulanık komşuluk ızgara bazlı çok yoğunluklu bir DBSCAN algoritmasıdır. Veri setini küçük networklere ayırmak için ızgara bazlı bir komşuluk kullanır. Kümeler için çekirdek noktalar ve komşuluk kümeleri oluşturmak için bulanık komşuluk fonksiyonu kullanır. Algoritmayı hızlandırmak için veri setinden temsilci noktalar kullanır. Veri seti ikiye bölünür: Birinci kısım önkümeleme için kullanılır. Bu kısmın veri setini temsil edebilmesi ve iyi yayılmış olması gerekir. İkinci kısım kümeleme işlemi sonunda önişleme ve kümeleri birleştirme için kullanılır. FNGMDBSCAN-UR algoritmasının iç içe şekiller ve çok yoğunluklu verileri kümelemede DBSCAN algoritmasından daha başarılı olduğu gözlemlenmiştir.

Anahtar Kelimeler: Kümeleme, bulanık komşuluk fonksiyonu, DBSCAN, FB_DBSCAN, GMDBSCAN-UR, yoğunluk bazlı kümeleme, ızgara bazlı kümeleme

CHAPTER 1

INTRODUCTION

1.1 Literature Review

A cluster is a group of data similar to each other in a data set, and dissimilar to the other objects in the same data set [1], [2], [3]. The process and techniques which are used to assign objects to their clusters is called “Clustering analysis” or “Clustering”. Basically, clustering is the task of grouping the objects that seem more similar to each other in one group (Cluster) [4]. Dissimilarity can be estimated based on the attribute values that describe the objects. Often a distance measurement is used to achieve this task [1].

The vast field of using clustering algorithms in data mining, machine learning, bioinformatics, and data compression makes it a point of interest for many developers. In the last years, the industry of information increased because of the processed information by clustering algorithms. Examples of these industries are products ranking, market analysis and understanding, fraud detection, and customer retentions [1]. The clustering algorithm is a black box that accepts the datasets as input and produces a set of clusters as an output [5].

There is often confusion between data classification and data clustering. Classification has well-known types of data, that we need to sort in the dataset to its type (the pre-defined classes), based on learning by examples. Clustering is different since the data is unsupervised so we don’t know the type of data labels in advance [6], [4]. Clustering analysis is a cornerstone of the data mining and there are many methods and algorithms developed in Clustering analysis. The problem with clustering methods is that interpreting the clusters may be difficult. In addition, the algorithms will assign data to clusters even if there are no clusters in the data [6]. So that, if the point of our work is to

find the clusters of the used datasets, we then have to check if the data set exhibits a clustering tendency. Workers in this field must know that the data set that are collected in the real life has an error ratio (noise) because of the measurements methods or due to the missing values in the dataset [1]. That's why we need a pre-processing function. Pre-processing process predicts the missing data in the dataset by checking the similar data in the data set. Pre-processing and post-processing techniques are used to eliminate the noise [1]. In post-processing technique, we try to fix up the clusters we produced. For example, two clusters that are close to each other can be merged to make one big cluster.

Thousands of clustering algorithms are developed. Partitioning algorithms learn clusters directly. Since the prototypes which have common features of certain classes are formed, then the elements are taken by these classes according to their degree of similarity degree. The clusters try to discover the cluster of nodes in a repeated way, getting about nodes among subsets, and sometimes working to Figure out the group of nodes which is full of data objects [1], [6], [2].

Hierarchal algorithm clusters gradually [6], [2]. The nearest objects are assigned to the same cluster and object a bit far away from the previous ones are put into the same cluster and so forth [7].

Usually, the algorithms that are considered as density based look upon clusters as a heavy area with data nodes and detached from the non-heavy area. The key way of dense clustering is figuring out the heavy area and non-heavy area thus, the algorithm can find the spot clusters.

On the other side when the data space split into net and all cells are equal, and after that the algorithm operates the work to this fixed area, we call it Grid based clustering algorithm. Cells which contain a number of objects equal or greater than a particular amount of nodes, are considered by the algorithm as heavy "dense" nodes. The cluster is constructed by matching all these heavy nodes.

There are many models-based algorithms. Such algorithms launch with unmethodical parameters and increasingly set them trying to discover the highest probability of these parameters. These factors show the contribution of the cluster, in each situation of the data objects. There are many other algorithms which are used outside of the data mining

community or being just theoretically applicable. In the following the summary of the main clustering methods and the cub-classes of these families:

- Hierarchal methods [2]:
    - Agglomerative algorithms [6], [2]: work as a bottom-up fashion that forms the cluster until all the objects in the cluster .
- Divisive algorithms [6], [2] : It devides the data set into small clusters then smaller and smaller untill reach the points in the clusters. [6]
- ClusterclusterPartitioning methods [6], [2]:
    - Centroid algorithms [6], [2]: use the gravity center of the object to represent the cluster. example: K-means.
    - Medoids algorithms [1] : example on it is K-medoids methods
- Algorithms that depend on their density [6], [2] .
- Methods that depend on Cell distribution [6], [2] .

In the field of data mining, especially in the process of clustering data sets many of requirments must be taken into account as follow follow [1], [6], [4]:

- Which elements the algorithm can deal with.
- Can we deal with big data sets?
- Can the algorithm work with multi-dimensional datasets?
- Can algorithm find the clusters even if the shapes are not clear?
- What about dealing with noise?
- Does it take a lot of time to give results?
- Does it depend on the order of data?
- Should researcher give the parameters or it is self-working?
- Do results make any sense?

For a better understanding of the thesis, here are some definitions that we used inside of it.

To Cluster [1], [4]: put a cluster of data objects collected together because of having a common attribute.

Hard Clustering algorithms [1], [5]: The way of attaching a data object to one and only one cluster of data directly.

Fuzzy Clustering algorithms [5], [6], [7]: Method of putting a data node to many clusteres of data depending on how much this node contribute in construction this cluster.

Hierarchical Divisive Clustering Algorithms [1], [6], [2], [4]: This method working as a splitter. The algorithm takes the cluster of data and starts splits it to smaller cluster until reach to just one object.

Hierarchical Agglomerative Clustering [1], [6], [2], [4]: It's the opposite of Hierarchical Divisive Clustering. Since the algorithm takes each data object and starts combine it with another one for sake of creating a cluster.

Partitional Clustering Algorithms [1], [6], [2], [4]: This clustering way create a cluster of data without having overlap between objects in the cluster.

Distance measure [1], [6], [4], [8]: One of the ways that we can measure and check if the node belongs a specific group of nodes that has similar attribute or not.

CURE [1], [9], [6]: Taking representative points from the data set, but must be drawing the same shape of the dataset, and use this object to start attaching the data to each other creating the cluster.

DBSCAN [1], [2], [4]: Mainly it looks at the data space and figuring out the heavy areas and the non-heavy areas. For all heavy areas, the algorithm connects the nodes that have the same similarity.

OPTICS [1], [6], [3]: So similar to DBSCAN by taking two values Eps and MinPts and working specially to detect and Figure clusters which depend on density.

DENCLUE [1], [6], [3]: Density Clustering method where the algorithm clusters the data node relaying on its formula of density.

MDBSCAN [1], [6]: This algorithm works with different heavy areas and set a different parameter for each of this area according to its density.

GMDBSCAN [1]: Same as the previous algorithm, but since we have a different value of heavying for the data set. So the algorithm splits it to the net and has some specific parameters for each cell of this net.

Cell Density [1], [7]: is the amount of data in a cell.

GMDBSCANUR [1]: The algorithm here is a combination of the main clustering algorithm. It uses the Grid system to specify the parameters for each cell, also it uses the representative points approach that CURE algorithm uses.

FNDBSCAN [5], [7]: is the algorithm that uses fuzzy neighborhood contribution and Fuzzy core points to start clustering using the same steps of DBSCAN.

FNGMDBSCAN-UR is the algorithm proposed in this study. Algorithm works same as GMDBSCAN-UR. But instead of clustering based on the distance between nodes, it clusters based on the neighborhood contribution of each node in the cell, and the Fuzzy core points.

1.2 Objective of the Thesis

Our study is working on many algorithms. The thesis explains the results of many tests as well for different data sets. Our contribution in this thesis is to provide a new clustering algorithm called “FN-GMDBSCAN-UR”, Fuzzy neighborhood grid based multi-density DBSCAN using representative points. The algorithm first constructs a spanning tree from the data set based on grids, since the algorithm will split data set to a known number of grids. And then it calculates the membership degree for each object in the dataset to construct a Fuzzy neighborhood set and define the Fuzzy core data points. After which starts the clustering according to the original DBSCAN, but with the new parameters that we have; Following this merging and labeling the points to the clusters starts. We tested the algorithm experimentally and the results show efficient of detection noise and good results in cluster shapes.

1.3 Hypothesis

We organized the report to be sequent as follow: the second chapter will handle with the previous work of researchers and the needed background to understand our work. Chapter 3 will talk about all the techniques and methods to reach the resulted clustering

algorithm. Chapter 4 will be specific for explaining our idea and the approach that we came out with. Chapter 5 will be for the experimental results of each clustering algorithm and the used datasets as well. As coming to the end of the report we have the conclusion of our work, the notes on the study, and the needed future work to improve the algorithm.

CHAPTER 2

RELATED WORK

2.1 Density Based Clustering Algorithms

Clustering is the main key to data mining. Extensive study of clustering started more than 50 years ago. Several clustering algorithms are proposed in this field, but there is no one clustering algorithm that can fit all kind of work in this industry. In our study, we focus on few clustering algorithms which are studied in detail to reach our new algorithm. One can categorize the choices by considering a different kind of clustering algorithms. In the basic clustering algorithms, categories could be Partitional or hierarchical. When the resulted clusters are nested in each phase of the process and they are different in manner to the amount then the algorithm used is a hierarchical algorithm. When having a single segmentation of data object that contains a constant cluster of data, then the algorithm is a Partitional algorithm. The Partitional algorithms can be very clear since it aims to minimize the objective function. In the following sections, we will discuss three Clustering algorithms related to our work. These are density-based (DBSCAN), multi-density DBSCAN (MDBSCAN), and GMDBSCAN Clustering algorithms.

2.1.1 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is one of the main references in clustering. As mentioned before the algorithm operates a distinguished method to discover the dense area. A heavy area is full of data points. Non-heavy areas exist between the heavy areas. One of the advantages of this algorithm is handling clusters that have different shapes and construct clusters from it. [1]. The key Function of DBSCAN is to put the dataset in two types of regions. The dense region that will be the later part of the cluster and the sparse

region which is not in the cluster and algorithm is considered as noise. The data points that are in the same neighborhood or that are in the dense neighborhood area are allocated to the same cluster. The algorithm here must take a radius (Ɛ) for objects and a minimum number of points in the radius (MinPts). If the radius of the object contains at least the minimum number of points, then a new cluster is constructed and the point will be the core point in this cluster "C". For any point in C that is supposed to be a core point, the algorithm looks at its neighborhood, which is in the area of radius Eps. If there is any point that didn't enter into the clustering process, the algorithm takes them under the process and checks the points in the neighborhood; if the number is greater than MinPts, then the point will be part of the data cluster. Algorithm keeps iterating until the dataset runs out of data points and all the candidate data points are added to the cluster C [1].

Let us talk about some definitions to explain the clustering processes as follows:

- The circular area around node with specific radius Ɛ is addressed as "Ɛ-neighborhood.
- A core point is the object that contains neighbors more than or equal to the MinPts. In other words, it is the object that its neighborhood degree is more than or equal to the MinPts [5].
- Border point: It is not a core point by itself and the neighborhood degree is less that MinPts, but it is in the other core point's neighborhood.
- Directly density-reachable [1], [10]: the point that can be directly reached from p if point q is one of node p's neighbors and p is a core point.

Density-reachable [10]: node q is density-reachable object if and only if there is a connected series of directly density-reachable core points connected to them.

Algorithm 1: Basic DBSCAN Algorithm

```
1:  Begin
2:  Inputs: (D, MinPts, Eps)    // D is the dataset
3:  For each o ∈ D              // o is data object
4:      If o is not clustered, then
5:          If o is core point, then
```

Collect the objects density-reachable from *o* and assign them to a new cluster

6: **else** assign *o* to noise

7: Done

DBSCAN Algorithm Problems:

One well known density based clusteing algoritms is DBSCAN A point in favor of DBSCAN, is that there is no need to suffer while knowing the number of clusters also no matter what is the shape of the dataset it can deal with it and extract clusters from it. Unlike the main DBSCAN problem is extremly sensitive to the parameters. Therefore the number of clusters will change and the Error rate will be high. If the user does not enter the radius then the algorithm will be forced to find it so that, it will start to count it for every node with its neighbors which will take a long time. In addition, we have one value for the radius that doesn't make any sense when we have to go over all the points in the dataset. Some of the points are so close to each other, but in another area, nodes are a little bit far. So we don't have to expect DBSCAN to work at the optimum performance when the dataset is multi-density.

2.1.2 Multi-Density-Based Spatial Clustering of Applications with Noise (MDBSCAN)

First of all, we have to mention some new definitions which are related to the algorithm.

Must-link approach: If two nodes are in the same group of nodes and this group represents the linked nodes, then these nodes are in the same cluster.

K-nearest neighbor list is the closest group of nodes to a specific node.

To make the algorithm dynamic we must use the link approach. In this way it finds the parameters for each different dense area. This results in many parameters. The algorithm then filters on the parameters to take the best one, such as radius, and operates the clustering process based on it.

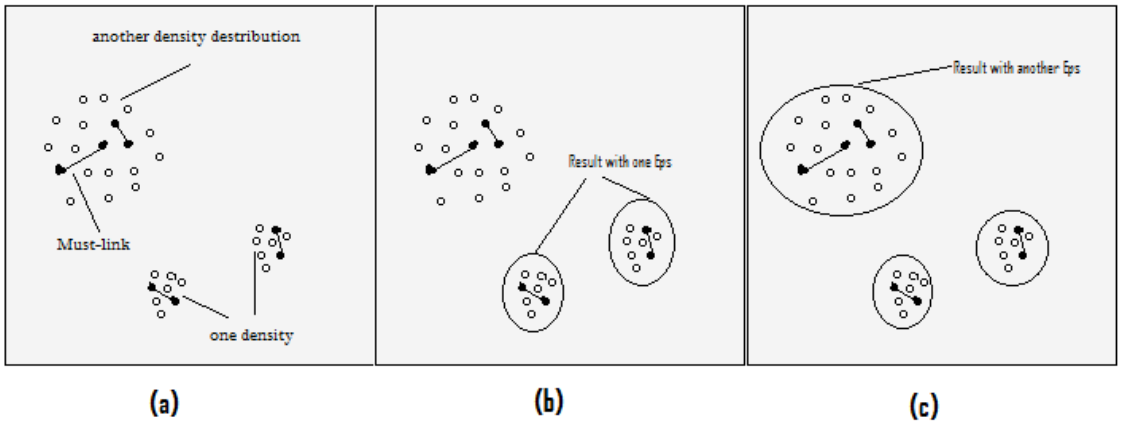


Figure 2.1: Example on MDBSCAN algorithm and Multi-density datasets [1]

Inspecting Figure 2.1(a) we can see the problem of DBSCAN. It is going to use just one Eps for the dataset. We can see two density distributions with four must-link points that represented with black solid objects with the line. The result of basic DBSCAN can be obvious in Figure 2.1(b) So the number of clusters will be just in the area of radius parameter Eps. MDBSCAN will give much better results because it will take into account the dense areas to find out Eps parameters that are fit with each area. So as we can see in Figure 2.1(c) the algorithm constructs two clusters according to the parameters and the dense areas. As mentioned before, according to the distance between the node and its closest neighbor, the distance of the density allocation changes. If two nodes are connected by the approach of "must link" then we can say both of them are in the same cluster. In this way, it is easy to find out the radius of each dense area around nodes since the algorithm knows the distances between neighbors.

Issues with MDBSCAN Algorithm:

MDBSCAN is still considered a time-consuming clustering algorithm. Moreover, the algorithm doesn't work very well in a large dataset so that the result is not so accurate.

2.1.3 GMDBSCAN-UR Clustering Algorithm

Because MDBSCAN is working solely with a global MinPts parameter for the entire dataset, it poses one of the main problems of this algorithm so the results that we get are inaccurate. Moreover, the algorithm takes a long time to provide results. To solve these

problems Grid-based Multi Density-Based Spatial Clustering of Application with Noise using Representative points algorithm (GMDBSCAN-UR) is proposed.

Phases of GMDBSCAN-UR:

1. data input and calibrate it for clustering process.
2. Start to split data into grids, so that each cell in the grid has the same valueof Eps.
3. Calculate the density for each cell and set the data in spanning tree.
4. Save the data that we got from the previous step in a bitmap to be used in the clustering process.
5. For each grid start to operate the clustering process locally. If algorithm finds two clusters and their distance is less than Eps, algorithm combines them to one cluster, and iterates itself.
6. The nodes that are next to borders and don't connect to other nodes or connected with a small number of nodes are assumed to be noise.

We are going to talk about each step in details in this section.

Partition into grids:

It splits the data space into a specified number of cells. Each cell is similar to the neighbor next to it. The radius of cells will be clear to the algorithm so it will use it to start the clustering as local clustering.

Construct SP-Tree:

For each non-empty grid, a value of points is taken in the grid (gNum). Then it goes through the dimensions until it reaches d+1-th dimension and in each dimension. When the algorithm reaches the last leaf it starts to make a leaf that contains the data between the node in a diminution and the other nodes at another diminution. The following Figure illustrates; an example of constructing a SP-Tree:

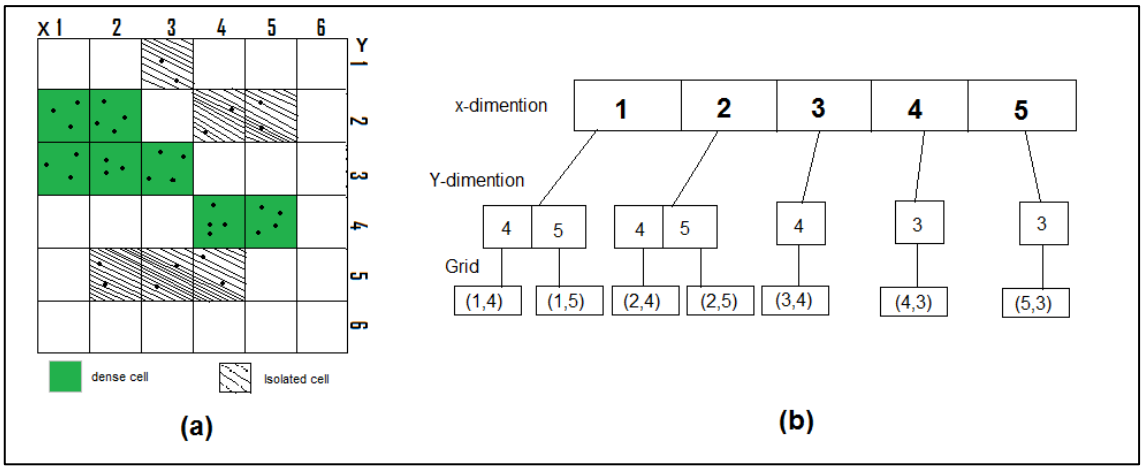


Figure 2.2: Framework of Constructing a SP-Tree [1]

As shown in the Figure the data space is divided into 36 cells, but it has only 7 dense grids Figure 2.2(a). So these grids are the only used grids to create the SP-Tree Figure 2.2(b).

Bitmap Forming:

GMDBSCAN is similar to DBSCAN in calculating the distance and deciding whether the data nodes are neighbors or not. GMDBSCAN-UR algorithm only calculates the distance between any two data objects which exist in the same or adjacent grids. If the algorithm finds out two nodes are in the range of each other that means they are in the same neighborhood as each other. The information about distance is stored in a bitmap.

Selecting Parameters (MinPts, Eps):

The algorithm takes the density of the cells as a regional minimum number of points for the cell. If the grid volume doesn't match with a number of nodes in that grid, we use a correction factor. The Factor value of a node is a number of neighbors over the grid volume. This factor shows the relation between MinPts and the cell density. Therefore, we can use it to find the local MinPts. Since we have the length of the cell; so usually Eps equals half of the grid distance.

Clustering:

Basically, GMDBSCAN-UR works in this step locally since we calculate the local MinPts for each grid and we start clustering the dataset. A group of distributed clusters

is created in this process. From these small clusters, if the algorithm finds two clusters of data that are close to each other in a way the parameters for them match with conditions of clustering, the algorithm combines them and create one cluster. For each time the algorithm merges a cluster, new values for parameters come up.

Handling border nodes and Noise:

Normally in the dataset, the amounts of nodes that are considered as noise are not many. The algorithm checks the points that are close to borders; if the point is not qualified to be inside a cluster, which means it has not enough neighbors match the threshold or far away from cluster center, then algorithm assigns it as noise.

Shortcomings of GMDBSCAN-UR algorithm:

For each time the algorithm calculates the radius and neighbor set, it takes time. When the algorithm merges clusters and starts to find new values of parameters it takes even more time. It is the reason GMDBSCAN-UR is considered time-consuming.

2.2 Fuzzy sets Based Algorithms:

In the previous section, the clustering algorithms provide a crisp way of clustering. The data object belongs to only one cluster so that the crisp partitioning can be presented as a binary matrix. The membership of each object in the dataset to the cluster is either 0 if the object does not belong to the cluster C or 1 if the data objects belongs to that cluster. On the other hand, Fuzzy clustering methods allow the data object to belong to multi-clusters at the same time. That means for each node in the data set we can create a matrix that represents the membership degree of this node. The Scholars successfully could integrate fuzzy clustering approaches in many algorithms [5]. In the following sections, we will discuss three of these applications.

In the previous section, the clustering algorithms provide a crisp way of clustering. The data object belongs to only one cluster so that the crisp partitioning can be presented as a binary matrix. The membership of each object in the dataset to the cluster either to be 0 if the object is not belonging to the cluster C or 1 if the data objects belongs to that cluster. On the other hand, Fuzzy clustering methods allow the data object to belong to multi-clusters at the same time. That means for each node in the data set we can create a matrix that represents the membership degree of this node. The Scholars successfully

could integrate fuzzy clustering approaches in many algorithms [5]. In the following sections, we will discuss three of these applications.

2.2.1 Fuzzy C-Mean (FCM)

This clustering algorithm is a general form of K-means [9]. As with any Fuzzy clustering algorithm, each point can belong to multiple clusters to a different degree, using a distance-based membership function to assign fuzzy membership to the cluster center. Before Fuzzy cluster means (FCM) works it must have three parameters [1], [9]:

- The number of clusters, C
- The fuzziness exponent, m
- $Epsilon$ which controls the stopping point.

The algorithm deals with cluster centers when it wants to start to give values to its nodes. It sums the distances between the cluster center and the node, and accordingly decreases the objective function for the method. In the case of FCM, the algorithm can work in two ways, we can give it the membership degree and let it find the cluster center, or give it the cluster center and let it discover the membership degrees [1], [11]. The objective function for FCM is as in (2.1):

$$\mathrm{Jm(U,V)} = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{m} D_{ik}(x_k, v_i) \tag{2.1}$$

In the previous equations, the dataset is presented by X and n is the absolute value of. c Stands for the cluster number. We use U so for the element (x_k) that exist in cluster (i) the membership degree will be presented asu_{ik}, cluster center's set is presented with V. While $D_{ik}(x_k, v_i)$ is the distance between cluster center and data node multiplied by itself.

From the previous formula we can extract the formulas that handle the mentioned two cases for algorithm as shown in 2.2 and 2.3:

$$u_{ik} = \frac{D_{ik}(x_k, v_i)^{\frac{1}{1-m}}}{\sum_{j=1}^{c} D_{jk}(x_k, v_j)^{\frac{1}{1-m}}} \tag{2.2}$$

$$v_i = \frac{\sum_{j=1}^{n}(u_{ij})^m x_j}{\sum_{j=1}^{n}(u_{ij})^m} \quad (2.3)$$

The cluster centers are calculated using the degree of membership of all data points, and the algorithm terminates when the change in distance of the cluster centers falls below epsilon. The following Algorithm 2 shows the process:

Algorithm 2: Fuzzy C-Means Algorithm

1: Input: X, c, m, Eps
2: initiate cluster values
3: for all $o \in X$ // o is data object
4: The maximum change is always equal to the radius plus 1 and we use it to terminate the loop. Maximum change = Eps + 1
5: While maximum change < Eps
6: membership degree = new membership degree.
7: using equation 2.3: for all $o \in m$ // m is cluster center set calculate membership degree and update it.
8: Based on formula 2.2 calculate membership degree for all the cluster and attach its nodes
9: New Maximum change value = maximum change value from maximum membership degree matrix
10: if maximum change = Eps
End while
11: Return the values of membership matrix and value of cluster centers.

Advantages of Fuzzy C-Means:

Since the algorithm is a soft clustering algorithm, it can deal with data that sometimes is mixed and overlapped.

Disadvantages of Fuzzy C-Means:

As we noticed in the clustering process, the algorithm must have a pre-given number of clusters to start working. Moreover, the algorithm iterates itself several times, which makes it a time-consuming algorithm.

2.2.2 Fuzzy C Medoids (FCMdd)

The FCM algorithm assumes that the data points are represented by numeric feature vectors [1]. So that the algorithm produces cluster centers located in R^s, the feature space of the dataset. However, finding the nodes while the datasets are represented in numeric feature vector is considered difficult. As mentioned in the previous section DBSCAN is related to the distance between data objects; and the Eps parameter needs to be appropriately set. On the other hand, to make the FCM algorithm accepts and deal with relational data modification is required. From the previous chapter, we know that the FCM algorithm seeks to minimize an objective function. The algorithm can achieve the minimization value by using relational data. As the FCM algorithm chooses representative nodes to be the initial cluster center. The FCMdd clustering algorithm is one of a "Hard c-medoid" algorithms set that developed mainly to cluster textual data by Krishnapuram [2]. It seeks to minimize the objective function (2.4):

$$J_m(V, X) = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^{m}\rho(x_i, v_j) \tag{2.4}$$

In the previous equation, the dataset is presented by X and n is the absolute value of . c stands for the cluster number, and for the matrix of membership degree for nodes we use U so for the element (x_i) that exist in cluster j. The membership degree will be presented as u_{ij}, cluster center's set is presented with V, while $\rho(x_k, v_i)$ is the dissimilarity between cluster center and data node.

As with the FCM algorithm, the membership function is used with a modification by replacing the squared distance with the dissimilarity. The following formula could be an alternative to the previous one.

$$u_{ij} = \frac{\rho(x_i, v_j)^{\frac{1}{1-m}}}{\sum_{k=1}^{c}\rho(x_i, v_k)^{\frac{1}{1-m}}} \tag{2.5}$$

FCMdd is similar to FCM in defining the cluster centers before starting clustering; In addition, the algorithm is using the two techniques that the previous algorithm uses. These are either to have the cluster centers and find the cluster membership degree or have the membership degree and find the cluster centers. FCMdd is not an alternative optimization algorithm. In order for FCMdd to initialize cluster centers, two techniques

can be used. The first technique is to select nodes randomly, but this technique suffers from limitations in terms of time consumption. The second technique is to pick the nodes randomly and put them in V as cluster centers; then find out the biggest dissimilarity among the data nodes and replace the previous cluster centers with them. This technique is much better than other techniques produced by random selection in the manner of the quality and it is known as "Initialization III [5]". The algorithm can be stopped in two ways: if the cluster center set is fixed and no change on it happens; meaning there won't be any change on membership matrix and also if a number of repetition reaches the limit. A formal description of FCMdd is listed in Algorithm 3.

Algorithm 3: Fuzzy C-medoids Algorithm

1: Input: D, V, c // D: dataset, V: random cluster center dataset, c number of clusters
2: Iter= 0 // iter: iterations
3: While ($V = V^{old}$ or iter= maximum iteration)
4: For I = 0 to c
5: Cluster center = current cluster center
6: Using Equation 2.5 calculate the membership degree matrix
7: For all data, nodes do the following
8: Calculate the maximum dissimilarity between objects
9: Make the local value of cluster center equal to x_p
10: End for
11: iter= iter++
12: End while
13: Return clusters

2.2.3 Fuzzy Neighborhood DBSCAN (FN-DBSCAN)

DBSCAN uses the distance between two data objects in the specific radius to define the density of the set [5] [7] [10]. FN-DBSCAN and uses the fuzzy membership function for all points within the distance radius. For a better understanding of how applying fuzzy membership function is much accurate than applying the crisp distance function, let us investigate about points x_1 and x_2 in Figure2.3.

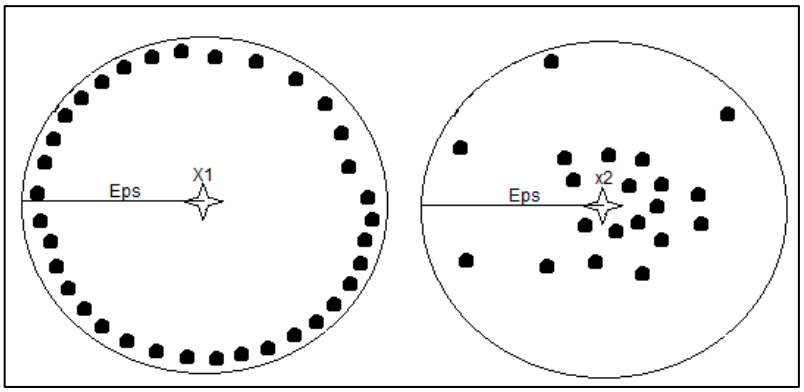


Figure 2.3: Similarity of points x_1 and x_2 comparing to Dissimilarity for the same two points when we use fuzzy methods [5].

Both nodes x_1 and x_2 have the same number of neighbors within the radius Eps ≤ d^{max} radius [7]. According to crisp neighborhood relation that used in DBSCAN both points are the same, but in fuzzy neighborhood function point x_1 will have a higher membership degree of being a core point than that of point x_2 [7]. The neighborhood function is as follows:

$$N_x(y) = \begin{cases} 1 - \dfrac{d(x,y)}{d^{max}} & if\ d(x,y) \leq\ \varepsilon \\ 0 & otherwise \end{cases} \tag{2.6}$$

Where d^{max} is maximum distance between nodes $d(x,y) \leq 1$, and $d(x,y)$ is the distance between x and y points.

Neighborhood relation is determined by the formula above. Neighborhood degrees of points with varying distances to the core point will be different from each other as shown in Figure 2.4:

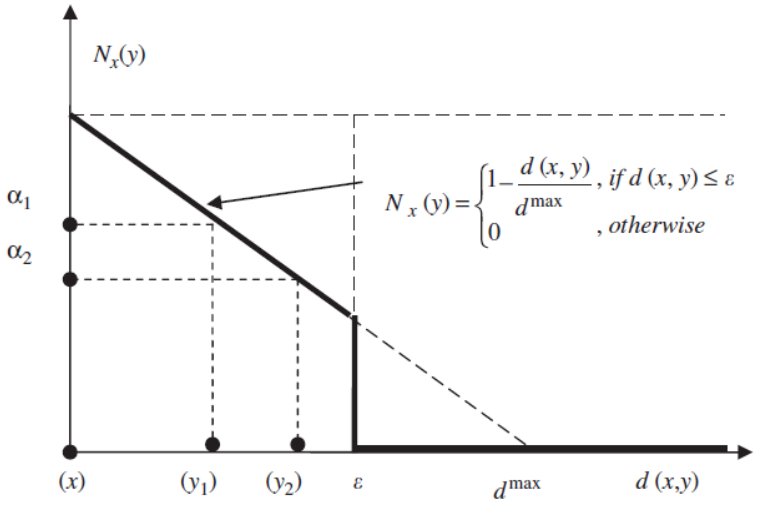


Figure 2.4: Fuzzy neighborhood relation for different points [7].

Points y_1 and y_2 have different neighborhood membership degree to the point X. Hence, the membership degree of y_1, i.e. α_1, is higher than the membership degree of y_2, i.e. α_2 [1]. In this section we will redefine some approaches that used in DBSCAN to illustrate the way of FN-DBSCAN works.

Definition 2.2.3.1: The general formula to find the fuzzy neighborhood set of point x that belongs to the data set with radius ε is:

$$\mathrm{FN}(x, \varepsilon) = (\langle y, N_x(y) \rangle | \, y \in X, N_x(y) \geq \varepsilon \,\} \tag{2.7}$$

N_x is membership function between nodes. Also ε is a parameter that determines the minimal threshold of the neighborhood membership degree.

Definition 2.2.3.2: point x is called a fuzzy core point with parameters ε_1 and ε_2 If

$$\mathrm{CardFN}(x, \varepsilon_1, \varepsilon_2) \equiv \sum_{y \in N(x, \varepsilon_1)} N_x \geq \varepsilon_2 \text{ [7]}$$

Holds for any point x∈ X where

$$N(x, \varepsilon_1) = \{y \in X | N_x \geq \varepsilon_1\}$$

And $\varepsilon_2 = MinPts/w^{max}$

Determines the ε_1-level set of the fuzzy neighborhood set of the point x.

Those two definitions are used in FN-DBSCAN instead of the core point and point neighborhood set in DBSCAN. In FN-DBSCAN algorithm, the directly density-reachable points are the same as the definition in DBSCAN algorithm. The following algorithm of FN-DBSCAN shows how it is similar to DBSCAN.

Algorithm 4: FN-DBSCAN Algorithm

1: inputs: dataset, ε_1 and ε_2

2: The initial status for points is unclassified and t=1

3: using definitions 2.3.3.1 and 2.3.3.2 assign fuzzy core points

4: The status of point p will have changed to be classified and a new cluster will be created and contains node p.

5: Create an empty set of seeds S. Find all the unclassified points in the set N (p, ε_1) and put all these points into the set S

6: For all $o \in S$

7: if o is classified, we remove o from the seed set and make its status "classified".

8: Algorithm checks for fuzzy core points based on parameters ε_1 and ε_2 and add all the unclassified points in the set N (q, ε_1) to the set S.

9: End for

10: Find a new fuzzy core point p with parameters ε_1 and ε_2 and repeat steps 4-7

10: Mark all the points, which do not belong to any cluster as noise.

11: End

FN-DBSCAN algorithm always gives better results than DBSCAN algorithm does by adjusting appropriate neighborhood membership functions.

CHAPTER 3

Used Techniques and Background

Our study is working over many clustering algorithms, that are mixed between being density-based and fuzzy based. In this chapter we will focus on some approaches and definitions that present the idea of the research. The previous algorithms which mentioned in the last chapter have some problems that we reduced by our presented algorithm. This algorithm is primarily concerned with merging between hard and soft clustering algorithms to solve the problem of accuracy.

3.1 Types of Points in DBSCAN

The following Figure 3.1 is showing the types of points in the dataset. Let's assume the number of minimum points (MinPts) is 3, here we illustrate the Definitions that we mentioned in the DBSCAN analysis in section 2.1.1:

- Labeled point A and the other red points are core points because each one is in the Ɛ-neighborhood containing at least three points.
- Points B and C are density-reachable from A or any other core point.
- Since B and C points are not directly density-reachable from point A Then A isn't density-reachable from neither B nor C since they are not core points [12].
- Red points are called density-connected

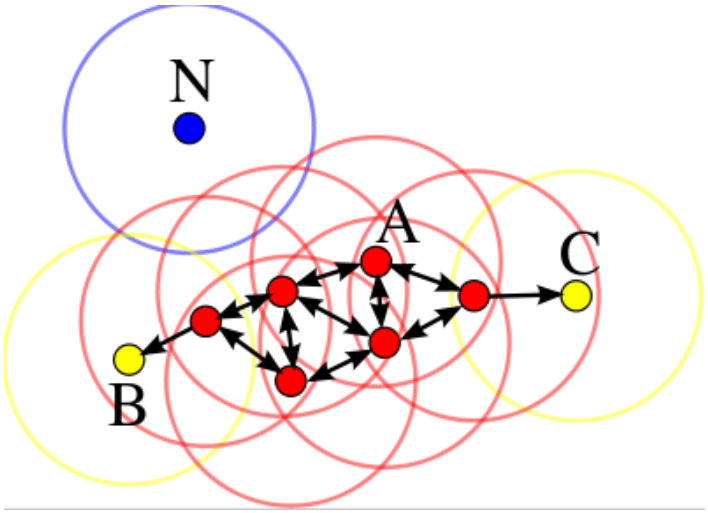


Figure 3.1: Illustration of point's type in density based algorithms

When we cluster the data sets based on the density, then the nodes will be connected to each other according to their density. In the clustering process, if a node is not connected to any other node then the algorithm marks it as noise as the blue point in Figure 3.1. Density-based algorithms like DBSCAN, operate the clustering process according to two conditions, the first is taking a radius Ɛ; the second is to check the neighborhood in this radius. Check if the number of points is equal to MinPts or greater than the number of nodes. The cluster will be constructed around this point. Then the DBSCAN will start to repeat itself and check the points that are connected to the core point "in its radius", and sometimes clusters will be combined as well. The algorithm finishes when the dataset runs out of data nodes that are not clustered.

3.2 Clustering and Data categories

Choosing the clustering algorithm depends on many factors, one of them is the type of dataset that we will test. Datasets have varied types of data. We mention some of them below:

Binary [1]**:** this type has two values either 0 when the variable is absent and 1 when the variable is present.

Categorical or nominal [1]**:** this type refers to the string or the names, such as the car models and company names or branches; it is more general than binary and has many values.

Ordinal [1]**:** This type is the type of data that can be ordered. It is used a lot we sort something in order, example: “first”, “second”.

Interval-scaled [1]**:** This type is used for the records of the measurement degrees, such as body mass, length and so on.

Ratio-scaled [1]**:** This type is the scale of measured value that is nonlinear; means there are variable affecting the scale.

The data types don’t stop here, many other data types are available, so the researcher must use the algorithm based on the data type that he has.

3.3 Criteria for DBSCAN based Clustering

To say that two clusters or two data points are similar, similarity measures, similarity coefficient, dissimilarity measures or distance are used. Euclidean distance between two points is a common distance/dissimilarity measure. Its formula is as follows:

$$d(x, y) = \left(\sum_{j=1}^{d} \left(\boldsymbol{x_j} - \boldsymbol{y_j}\right)^2\right)^{1/2} \quad (3.1)$$

Distance in clustering is a critical value in terms of how small the distance is and how much the nodes will be in the same cluster. As we mentioned before similarity in DBSCAN is extremely important and according to it, the element will be sorted to the cluster.

3.4 FN-GMDBSCAN-UR Must Be Known Definitions

Cell [1] is the leaf in spanning tree and it is the smallest part of the net.

Cell-Density (CD) [1] is the result of dividing the number of nodes in the cell by the volume of that cell.

Dense Cell [1] [5] is the cell that will be clustered because it contains number of nodes more that or at least equal to the MinPts parameter.

Cell-Neighboring: when the cells that have a node in common e.g. a “mutual node” then we call them neighbor nodes.

VCell [1] is the cell volume

VEps [1] is the data point's neighborhood volume

Fuzzy core point [5]: is the point that has a neighborhood membership degree that is greater than or equal to the threshold. In other words, when the fuzzy set cardinality FSCard, is greater than MinCard for the data object, we call it a fuzzy core point [5].

Minimum Cardinality (MinCard) [5]: in fuzzy clustering, it is used instead of MinPts parameter and its summation reflect the "density" of the cell and based on it the clustering will operate.

CHAPTER 4

Proposed Cluster FN-GMDBSCAN-UR

This chapter focuses on the idea that we propose and the algorithm steps that the dataset will go through to achieve the expected results. In this study, we are interested in the density-based clustering algorithms. The proposed algorithm is a density-based clustering algorithms. We modify grid multi-density DBSCAN to use a fuzzy membership degree. The membership degree tells us the neighbor sets and the core points to start the clustering process. In addition, we use representative points to increase the speed of the clustering algorithm. FN-GMDBSCAN-UR is a modified algorithm from GMDBSCAN-UR and we will illustrate these steps in the following sections.

4.1 Our Main Idea

In starting the process of clustering, the input data pass through many steps and procedures to achieve the final aim by making all the dataset clustered. So looking at the clustering algorithm as a black box it will be similar to a factory with input and output. Diving more deeply into this black box we have a multi-density dataset as input. And we use the grid-based clustering for reducing the time complexity and divide the data space into grid cells. Each cell has a number of nodes that must represent the shape of the data in the grid. In our research, the algorithm will start dealing with the cell as one object. It calculates the neighborhood membership degree for the point in the radius of each point and starts constructing the neighborhood sets of each point. In addition, it assigns the core points in the cell to start clustering process.

The process of calculating the membership degree has two options, the linear way, and the exponential way. In the first way calculating the neighborhood membership relies on

a linear equation that is shown in Figure 4.1. The other way is using the exponential equation to calculate the neighborhood membership degree and using a constant factor to increase the sensitivity of the membership degree; as shown in Figure 4.2.

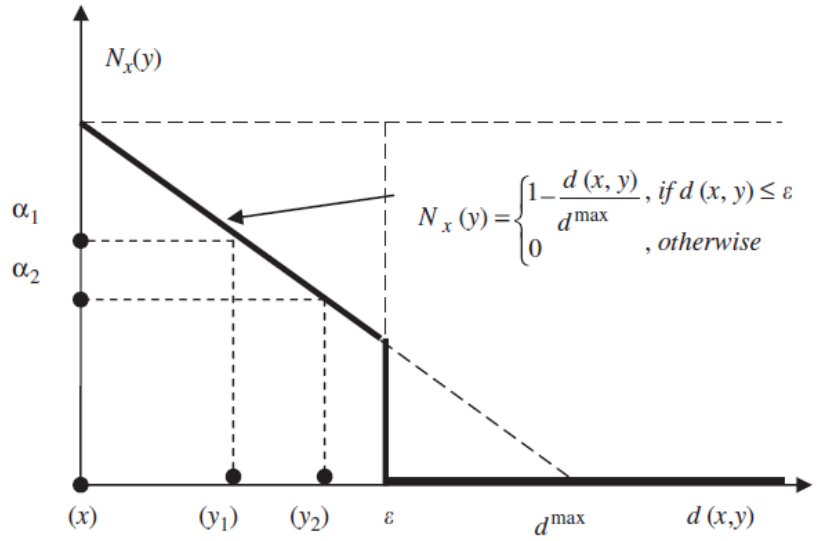


Figure 4.1: Fuzzy neighborhood relation using linear equation [5]

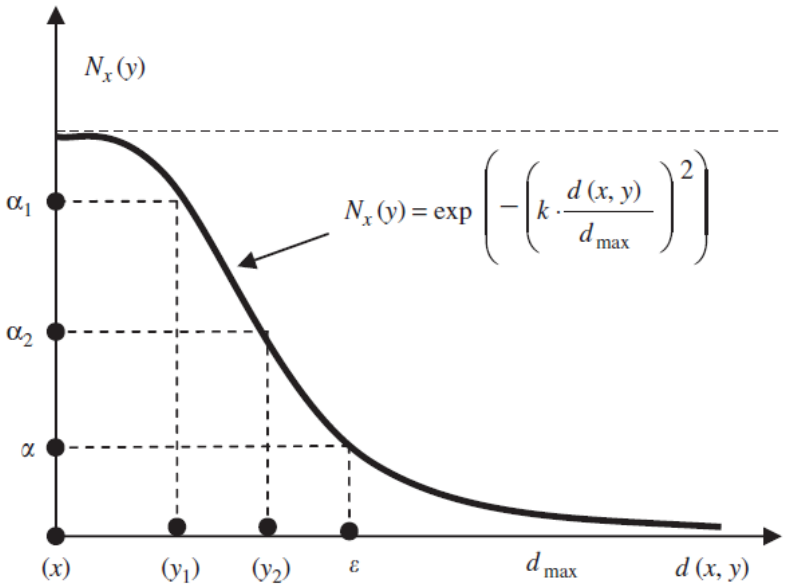


Figure 4.2: exponential neighborhood relation [5]

Sin this instance, here core points and density of grids will depend on the contribution of each point in the neighborhood membership relation. Following this, we use DBSCAN to check if the requirements of clustering match. If so, the algorithm will merge the nodes at the cell level. In addition, the algorithm combines clusters that

match the condition for clustering between them to each other. Latterly, the labeling process starts labeling the points that we didn't choose as delegated nodes to the clusters that came out from the previous process. Finishing the clustering process is achieved by the post-processing phase. At the end, one of the core steps after clustering is to eliminate noise and outliers.

4.2 FNGMDBSCAN-UR Phases

The proposed clustering algorithm, FN-GMDBSCAN-UR, working phases:

1. Data input and calibrate it for clustering process
2. Start to split data to the net, that each cell in the net has the same volume.
3. For each cell in the net, the algorithm chooses nodes that form the dataset, and use them for pre-clustering operation.
4. Calculate and assign the parameters "Eps,$\varepsilon_2, Minpts$", which are needed to start the clustering operation; by matching the nodes and the conditions of clustering.
5. Save the data that we got from the previous step in a bitmap to be used in the clustering process.
6. For each grid start to operate the clustering process locally, then if algorithm finds two clusters whose distance is less than Eps, algorithm combines them together; and iterates itself.
7. For the rest of the data that didn't participate in clustering operation, we enter it and reiterate the clustering process until finding the final result.
8. For the nodes that are next to borders and don't connect to other nodes or connected with a small number of nodes, algorithm deals with them as noise.

In next subsections we will go over each step to explain in more detail our new multi-density clustering algorithm. It relays on the Fuzzy neighborhood membership relation function and using representative point.

4.2.1 Entering Data and Data Calibration

The aim of Data Standardization is to make the data dimensionless. The location and the scale of the data originally taken by the algorithm will not stay the same and most

probably we will lose it. But it is necessary to standardize variables because we are going to calculate the similarity and dissimilarity using Euclidean distance. The preparation of the dataset goes in two ways: the first is called global calibration, which handles all the nodes in the dataset. The second is the Within-cluster Standardization method that refers to the standardization process that occurs within clusters on each variable. The Z-score Standardization method is one of the famous techniques that transform the normal variants to standard score form. Z-Score equation:

$$x_{ij} = Z_1(x_{ij}^*) = \frac{x_{ij}^* - \bar{x_{ij}^*}}{\sigma_j^*} \tag{4.1}$$

Where $\bar{x_{ij}^*}$ and σ_j^* represent the mean and standard deviation of j^{th} column.

After this step, we will lose the location and the original data's information.

4.2.2 Create a grid of Data Objects

In order to make the operation of clustering run fast, the algorithm splits the data space to a net, where all cells sizes are equal. Figure 4.3 shows a dataset that is a multi-dense dataset. With regular DBSCAN, it will work on it as a one data space, but using Grid-based approach makes it easier. Cells are going to be two types, the one with a density that matches the clustering conditions, and the other that will be reconsidered as noise after the clustering process finishes.

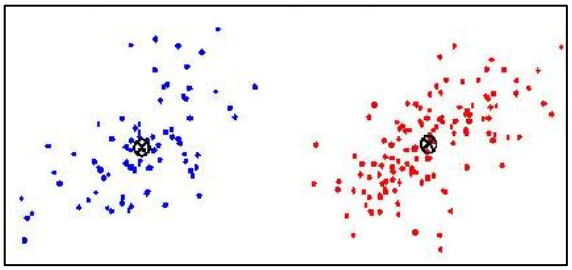


Figure 4.3: Multi-density Dataset [1]

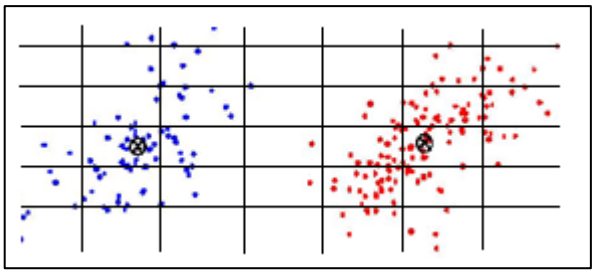


Figure 4.4: Divided the dataset to smaller cells [1]

4.2.3 Take a Delegated Dataset from the Original Dataset

After dividing the data space to grids, we choose a number of points to represent the data in the clustering process. The data points that the algorithm will choose as a representative must satisfy an important condition. These chosen nodes must be formed in the data space very well so no part of the dataset's shape is lost in the clustering process. The benefit of this step is to give the proposed algorithm" FN-GMDBSCAN-UR" a good improvement in time consumption.

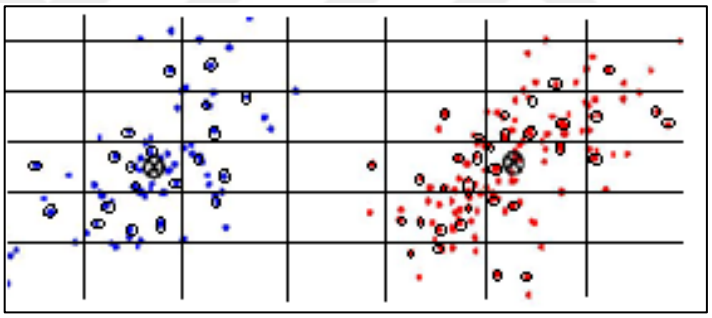
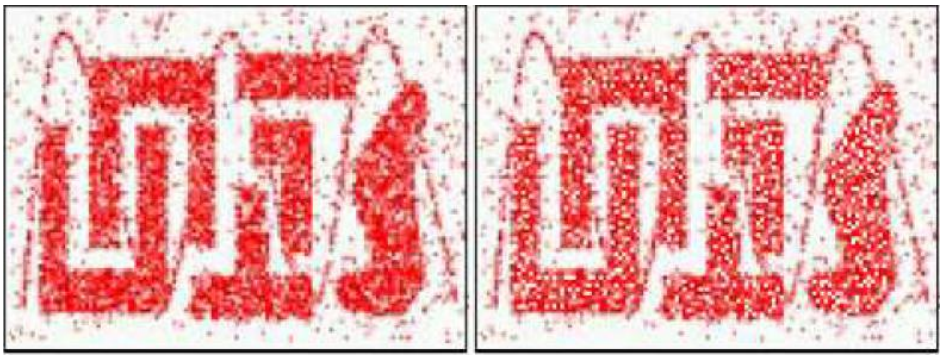


Figure 4.5: well scattered representative points in each cell illustration [1]

Our algorithm goes over the data space and enters all the cells. It takes a group of delegated points and inserts them in a new dataset for a representative point. The new data set will be used in the first clustering operations so that it must have the same shape as the original data space. The rest of dataset will stay as it is in the group that will be used in the last two phases to give the final result of the process. The Figure below shows the original dataset and the representative points.



(a) (b)

Figure 4.6: dataset after and before choosing the representative points [1]

4.2.4 Select the Main Parameters for the Algorithm

Two approaches are available to apply for selecting the minimum number of points and the radius. The first is to put the radius as constant and use varying MinPts from cell to cell. The second method is to make the MinPts as a constant and a varying radius that depends on the point's presentation in the data set. For each cell in the dataset, we calculate Ɛ2 parameter which represents the density of the point. If the summation of the Fuzzy cardinality of the point is greater than this parameter; then the algorithm will consider the point as a Fuzzy core point. For calculating this parameter, we have to calculate the maximum cardinality of the point in each cell with a certain Eps. The equation of Ɛ2 is as (4.2):

$$Ɛ2 = \alpha * \frac{MinPts}{w^{max}} \tag{4.2}$$

Where $w^{max} = max_{i=1,2,\ldots,n} w_i$ and w_i is the summation membership degree of points in the radius Eps to the neighborhood set.

α : is a constant real number that is used as a correction factor and it changes from dataset to another depending on the density of the data set. If the dataset is dense dataset, then the factor will be big. The value of this factor comes from experiment.

4.2.5 Construct the Bitmap

In this step, most of the statistical results that are needed in the next step will be ready and saved in a bitmap. The fuzzy neighborhood membership calculation between two nodes which exists in the same or adjacent cells are done. The algorithm will compare

the result with the radius Eps and the information stored in a bitmap. If the distance between nodes is less than the radius and the fuzzy membership degree of the point $N_x(y)$is equal or bigger than Minimum cardinality, then the point will be added to the fuzzy neighborhood set and we store it in a bitmap. The same thing happens to the fuzzy core points; where the fuzzy core point is the point that its fuzzy membership degree is bigger than or equal to Ɛ2 for each cell.

Where:

$$N_x(y) = \begin{cases} 1 - \frac{d(x,y)}{d^{max}} & if\ d(x,y) \leq Eps \\ 0 & otherwise \end{cases} \quad (4.3)$$

One of the important parameters that help in calculating the minimum number of points MinPts is Cell Density "CD". This is the number of nodes in one cell. As mentioned before the algorithm uses an initial value of MinPts to start the clustering; and here it is the density of each cell as a local minimum number of points. Sometimes the value is not so accurate so we divide the number of nodes in cell over the volume of the cell as shown in the next formula:

$$\text{factor} = \text{MinPts}/\ \text{CD} = \ VEps/VCell \quad (4.4)$$

$$MinPts = factor * CD \quad (4.5)$$

4.2.6 Start the Pre-Clustering Between the Similar Nodes and Similar Clusters

We use the approach of DBSCAN algorithm locally in each cell using the computed MinPts, Eps, Fuzzy neighboring degree, and Ɛ2. We use the parameters to operate the clustering process and make the first clusters of data nodes. We have many phases to reach the final shape of cluster:

1. Algorithm checks the density of cell in the data space, which is un-clustered cell yet. And we deal with borders. The cell is considered as dense cell if its density is bigger than or equal to the predefined threshold.
2. The cells that are close to the dense cells but are not dense cells by themselves are called sparse cells. This kind of cell has two classifications: a border or noise. Isolated cells are the cells that are not dense and don't belong to any cluster or are not next to any core point so they are isolated from the points in

data space. Our algorithm “FN-GMDBSCAN-UR”, searches for the closest cluster to that isolated node and allocate it to that cluster. Then algorithm recalculates the minimum number of nodes again according to formula 4.5.

3. Our algorithm uses the basic approach of crisp DBSCAN algorithm. The points that weren’t in the clustering process, the algorithm comes on them. Let us assume there is an un-clustered node *P*; algorithm marks *P* as visited and computes the fuzzy neighborhood membership value of node *P*. We check the density value and compare it with fuzzy neighborhood threshold Ɛ2. If it is less than the threshold then it is a noise, otherwise, it is a fuzzy core node.
4. Up to this point, the algorithm created clusters of data that are separated from each other. If a node comes in between two clusters or belongs to two clusters at the same time, the algorithm combines the clusters together. If a data node is not in any of the clusters the algorithm labels it to the nearest representative node.

4.2.7 Labeling all points to the best Cluster

The Labeling and post-processing step starts with getting the entire dataset and allocating all the points to the nearest cluster. The resulted clusters came from a subset of the data so the algorithm takes the rest of the dataset that wasn’t clustered and labels it. The algorithm attaches each data object to the cluster that contains the representative points close to it. Now we have all the data clustered; if two clusters are close to each other, the algorithm combines the clusters together and so on. After labeling, all the remaining data objects to the nearest data representative objects and merging all the nearest clusters to each other. In doing this, will have almost the final number of clusters for the dataset.

4.2.8 Noise Abstraction

Datasets contain outliers that come from wrong measurement techniques or due to the problems in entering data. The outliers’ distance from the nearest clusters and the neighborhood around them is usually high. Each clustering approach has its own

method to catch the outliers and remove them. In our algorithm "FN-GMDBSCAN-UR" because the outlier points have large distances form core points and the rate of merging between them is slower than the normal clusters, the algorithm can detect them and mark them as outliers. For outliers, the density of the object collection is small, and the distances are big between each point and the nearest cluster so that the algorithm can detect them easily.

4.3 FN-GMDBSCAN-UR Algorithm Basic Code

Algorithm 5: FNGMDBSCAN-UR Algorithm

1: Divide data space to equal grids

2: Take representative points that represent the density and shape of the original dataset

3: Specify parameters ε_1 and ε_2 and MinPts // explain what those parameters are

4: Mark all the points in the dataset as unclassified. Put the initial value of t to 1.

5: Discover the fuzzy core points that match the conditions of parameters ε_1 and ε_2

6: If point matches the conditions of clustering, then we mark it as clustered point and algorithm starts a new cluster and it is the core point of it.

7: Create an empty set of seeds S. Find all the unclassified points in the set $N(p, \varepsilon_1)$ and put all these points into the set S

8: Get a point q in the set S, algorithm assign q to be a clustered node, when it belongs to a cluster, then it removes it from the seeds set.

9: if q matches with the parameters ε_1 and ε_2, it is a core point. if so, add all the unclassified points in the set N (q, ε_1) to the set S.

10: Repeat steps 6 and 7 until the set of seeds is empty

11: Find a new fuzzy core point p with parameters ε_1 and ε_2 and repeat steps 4-7

12: Label all the non-representative points to its nearest representative points

13: Start post processing and remerging clusters

14: Mark all the points, which do not belong to any cluster as noise.

15: Done

CHAPTER 5

RESULTS AND DISCUSSION

We made a number of experiments over many clustering algorithms and types of datasets. Also, we will compare the results of three clustering algorithms: DBSCAN, GMDBSCAN-UR, and FNGMDBSCAN-UR. The following sections will contain some Figures and tables that show the number of clusters, the run time and the accuracy of each clustering algorithm.

5.1 Artificial Dataset

Chameleon dataset [1] is an artificial dataset, it called chameleon because it helped in improving the chameleon algorithm, and we use it to test our proposed algorithm. The dataset has 8000 data object. In our test, we take the real value attributes. The challenge in clustering such a dataset is that it has six multi-densities clusters. The clusters are nested and are not simple shaped. Each cluster of those has an arbitrary shape and number of noise around it. The Figure 5.1 below shows the shape of the chameleon dataset.

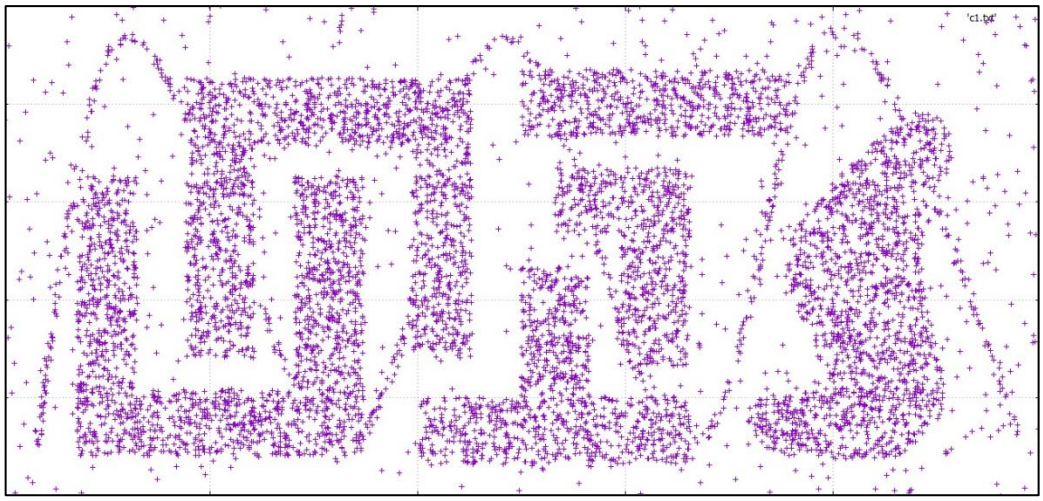


Figure 5.1: Artificial “chameleon” dataset.

5.2 Real Dataset

We use the well-known Irıs flower dataset [1] that is used in many tests in data mining fields. The dataset contains 150 elements and three classes; 50 elements for each class. We will use the dataset to show the result of the algorithms in the light and small datasets. Figure 5.2 shows the two columns of the iris dataset.

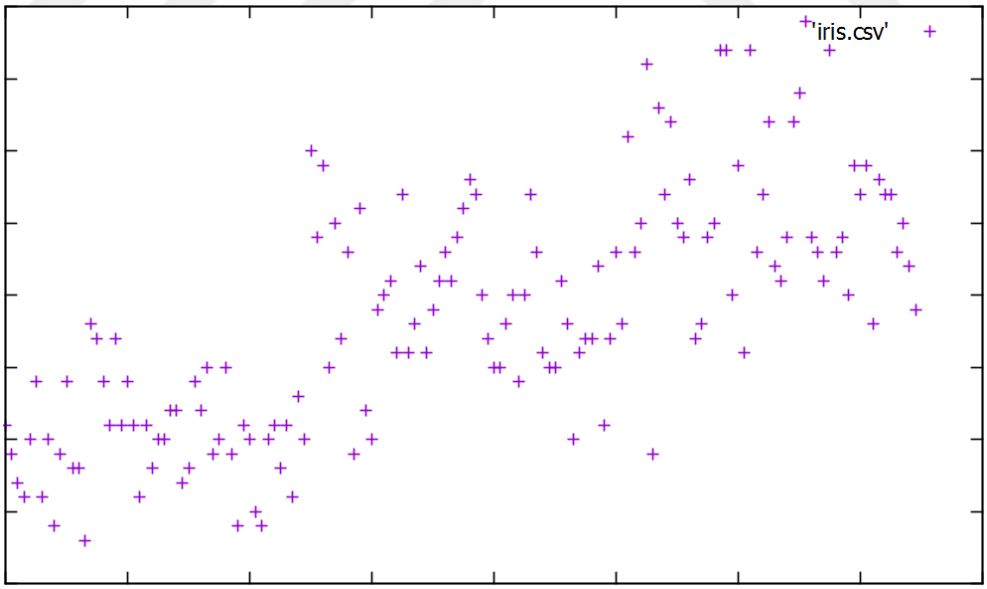


Figure 5.2: Real “iris” dataset

5.3 Crisp DBSCAN Experimental Results

In order to test our algorithm FNGMDBSCAN-UR; which is based on fuzzy neighborhood analysis; with GMDBSCAN-UR and the Crisp DBSCAN, we use the artificial chameleon dataset and the real iris dataset as mentioned in the earlier section. The algorithms are written in java and the experiments were carried on a computer with cpu Intel core i5, 2.50GHZ, 6GB RAM. The results of crisp DBSCAN on the datasets are as in Figure 5.3. When we run DBSCAN on chameleon dataset the algorithm fails in discovering the clusters. In our experiment, the algorithm could just discover one cluster and assign all the data points to that cluster as shown in Figure 5.3. DBSCAN merges between different clusters that hard to be clustered. That's why DBSCAN is considered as an inappropriate clustering algorithm for multi-density datasets.

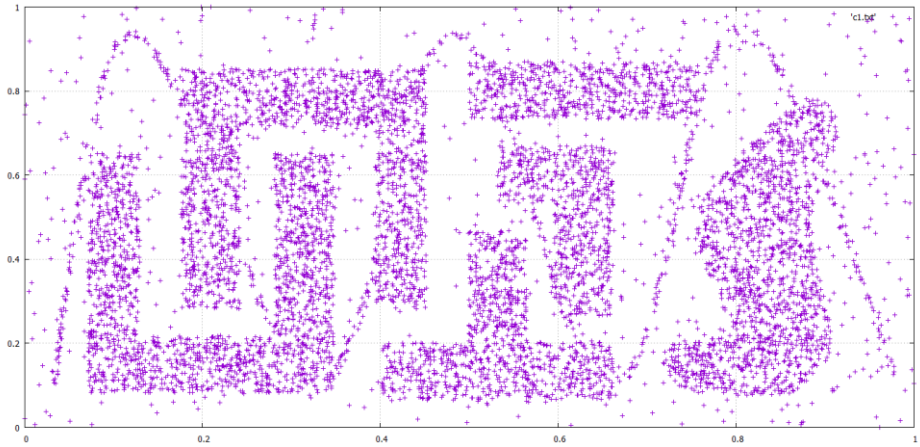


Figure 5.3: The results of DBSCAN on chameleon Dataset. The algorithm puts all the data to a single cluster.

Also running the algorithm on a simple dataset like iris comes out with inaccurate results as shown in Figure 5.4.

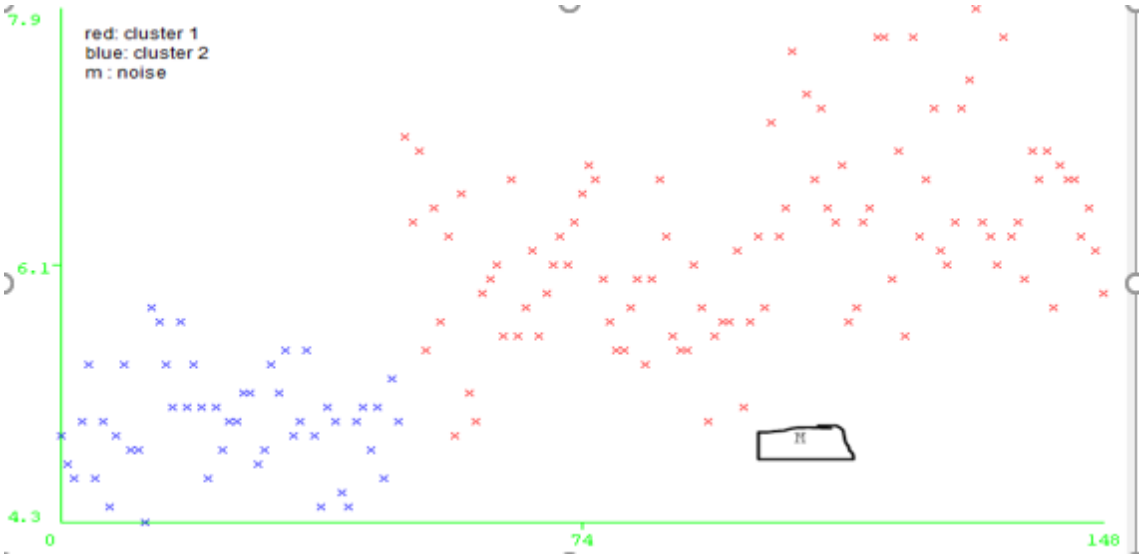


Figure 5.4: The results of DBSCAN on Iris Dataset shows 4 clusters and un-clustered nodes

Table 5.1 shows the result of DBSCAN on iris and chameleon dataset. The first column in the table shows the name of the dataset. The second column shows the runtime in a millisecond. The third column shows a number of clusters. The fourth shows the clustered points and finally the fifth shows the noise points.

Table 5.1: iris and chameleon datasets clustering results summary using DBSCAN

Dataset	**Run Time (ms)**	**No. of clusters**	**No. of cluster points**	**No. of noise points**
iris	7	2	140	1
chameleon	5640	1	8000	0

It is obvious that DBSCAN is inaccurate. For example, in chameleon, it must have six clusters while what we find is just one cluster. The same for iris it must contain two clusters with no noise; while it has unshaped 4 clusters with some noise in DBSCAN algorithm.

5.4 GMDBSCAN-UR Experimental Results

GMDBSCAN-UR solves the problems of crisp DBSCAN. Since the algorithm divides the data space to grids and starts the local clustering, then start merging between the

sub-clusters. The algorithms show good results for the chameleon dataset. The algorithm can catch the noise in a much better way and also specifies the six clusters in the dataset as shown in Figure 5.5.

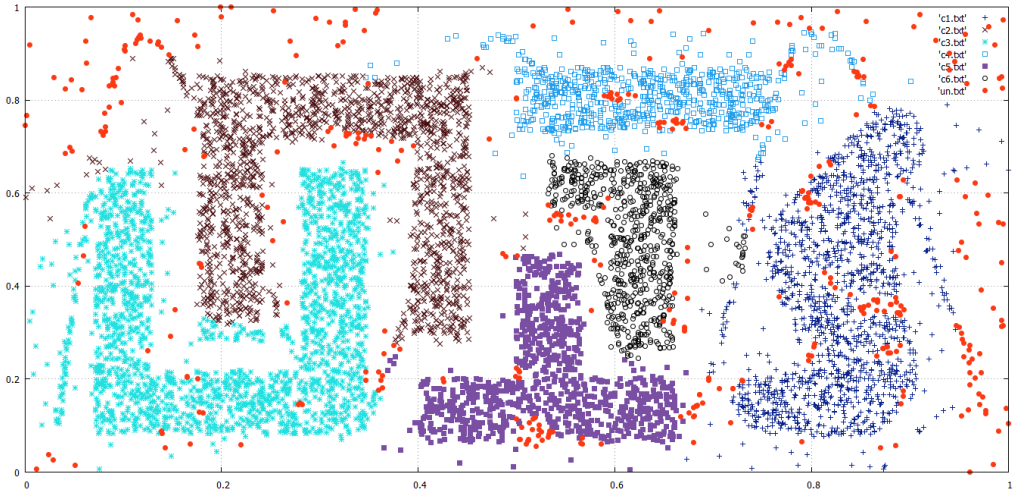


Figure 5.5: The results of GMDBSCAN-UR on chameleon Dataset.

The Figure above is the result of 8000 points in the dataset. The algorithm gave six clusters each one with arbitrary shape and much noise. In this algorithm, if the data points of dataset increase, the runtime complexity will increase as will in a linear way. However, the algorithm still has some problems with the accuracy as we can see in the Figure. There are some clusters have points that must be belong to other clusters, as we can see in cluster two and cluster three. Also, noise detection is not so accurate. Applying the algorithm on iris dataset will give as very good results as shown in Figure 5.6. The run time and accuracy are good, but still, we can notice cluster one has points that do not belong to it. Moreover, there are some points that the algorithm catches as noise while they are belonging to another cluster.

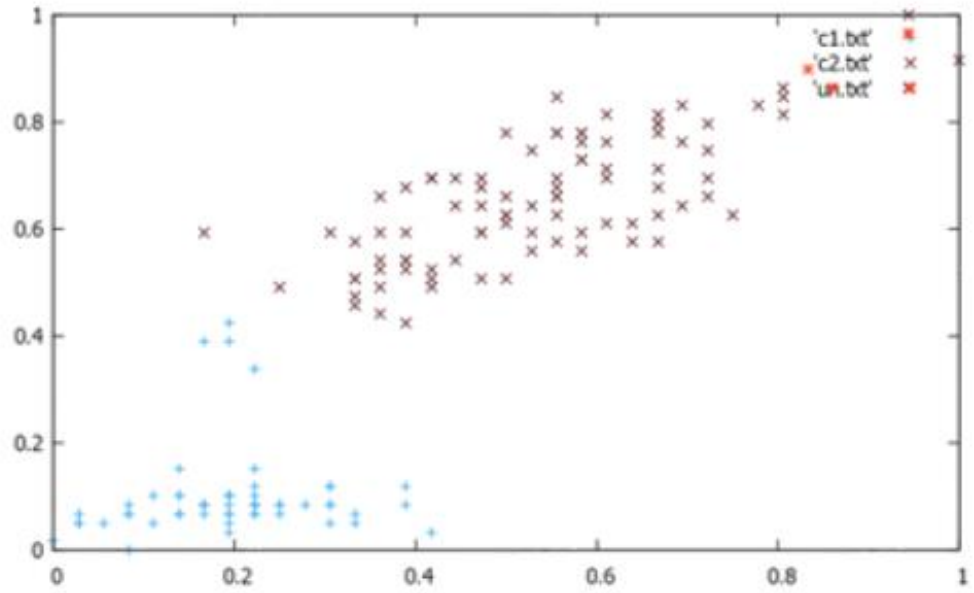


Figure 5.6: The results of GMDBSCAN-UR on Iris Dataset.

The following table 5.2 and table 5.3 show the clustering results of chameleon and IRIS datasets. We calculate clusters' accuracy and the total accuracy of the algorithm over the datasets respectively.

Table 5.2: the clustering result for chameleon dataset using GMDBSCAN-UR

Clusters	**No. of points**	**No. of Error points**	**Accuracy rate %**
C1	1607	130	91.91
C2	1646	54	96.71
C3	1902	96	94.95
C4	696	94	86.49
C5	938	33	96.48
C6	643	40	93.77
Noise	412		
Runtime (ms)	8059		
Total accuracy	94.10%		

The result shown in table 5.2 shows the accuracy of the algorithm. We can see that the algorithm is fast and good in somehow in giving the result, but the accuracy of it is not good enough. And it is similar to the next table 5.3 with iris dataset.

Table 5.3: the clustering results of iris dataset using GMDBSCAN-UR

Clusters	No. of points	No. of Error points	Accuracy rate %
C1	54	4	92.59
C2	91	3	96.70
Noise	5		
Runtime (ms)	81		
Total accuracy	95.17%		

If we compare the results in table 5.3 and table 5.1 the runtime is a little bit slow, but the results that we have in table 5.3 and table 5.2 is much better than the result of table 5.1 as we use the crisp DBSCAN. We notice in Figure 5.5 and Figure 5.6 the accuracy of detecting the noise and the right point to each cluster has a problem as illustrated in Figure 5.7 and 5.8 for chameleon and iris datasets respectively.

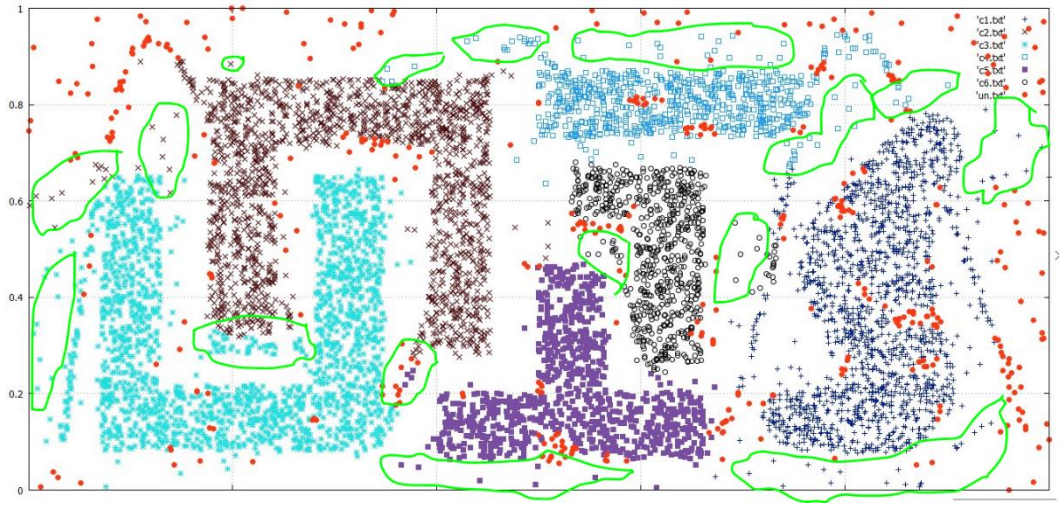


Figure 5.7: clustering errors in chameleon dataset in GMDBSCAN-UR

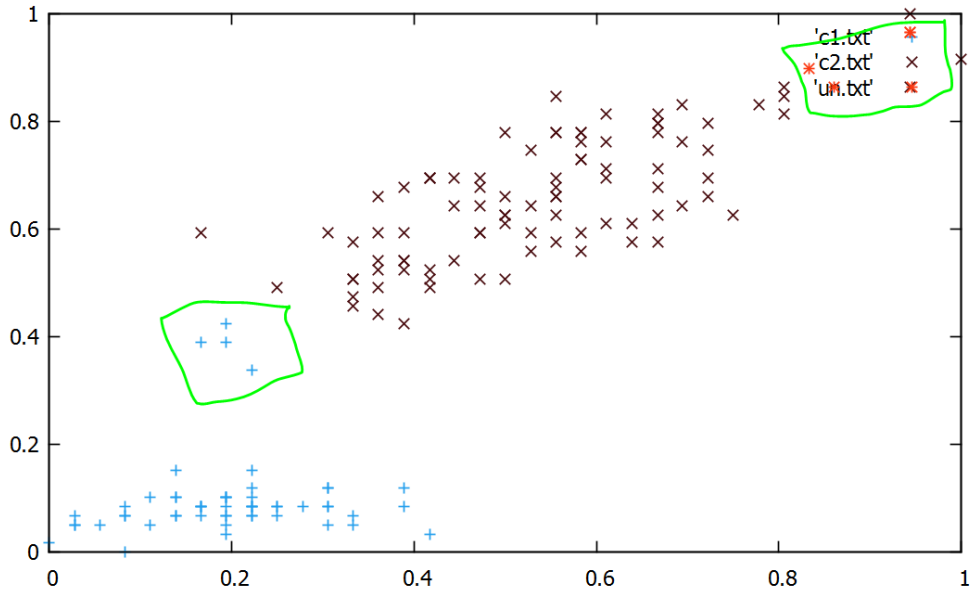


Figure 5.8: error area in iris dataset clustering result using GMDBSCAN-UR

The green circled areas show the error in the clusters since the points supposed to either belong to other cluster or to be assigned as a noise. The result of this problem in accuracy maybe because of the hard clustering method that GMDBSCAN-UR using in clustering. Since each node must belong to one cluster and the similarity method that the algorithm follows here is distance-based measurement. The algorithm we propose in this study will go over these problems. It uses the fuzzy neighborhood function to define the neighborhood sets and the fuzzy core points as mentioned in the previous chapters.

5.5 FNGMDBSCAN-UR Experimental Results

FNGMDBSCAN-UR stands for Fuzzy neighborhood grid multi-density DBSCAN which is our proposed clustering algorithm. The algorithm first, instead of just calculate the distance between points in the dataset and assign each point to one cluster according to crisp DBSCAN, it calculates the neighborhood membership degree. Then allocate the point to many clusters as it has many membership degrees, the algorithm will assign it to the best cluster. Also, the algorithm split the dataset to cells to make the clustering process easy and more accurate. Moreover, it takes some representative points to increase the speed of clustering as GMDBSCAN-UR does. Same as the previous two chapters we have tested the algorithm on chameleon and iris datasets. The results that we got are impressive since the algorithm gave more accurate results and a little bit fast speed than the previous two algorithms. The most important thing that

FNGMDBSCAN-UR gives to us is the accuracy in detecting the nose in an efficient way and allocates each data object to its right cluster. Figure 5.9 shows the result of chameleon dataset after take it as the input of FNGMDBSCAN-UR.

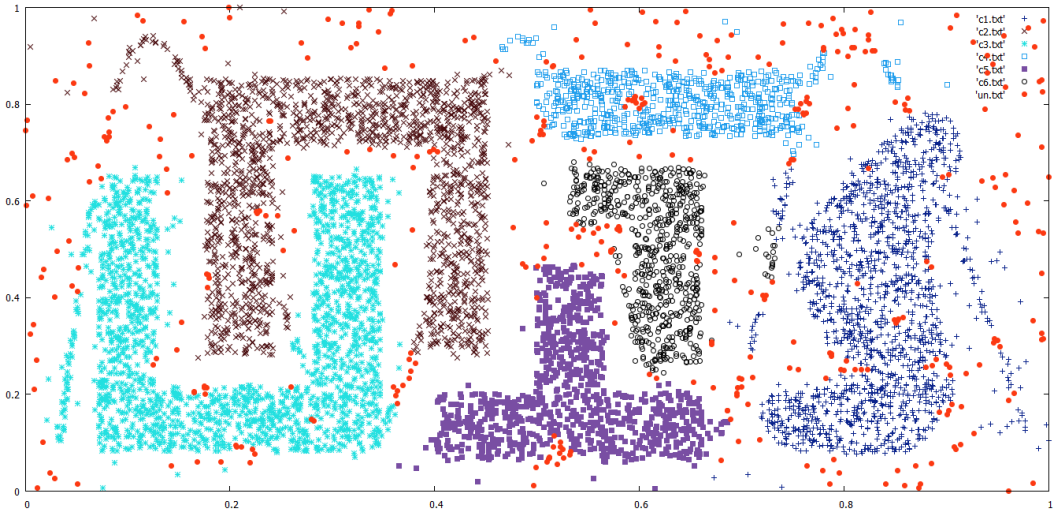


Figure 5.9: clustering result of chameleon dataset using FNGMDBSCAN-UR clustering algorithm

It is obvious how the algorithm reduces the noise and can detect the noise points in a much better way. The algorithm is not so perfect, but the results that it gives is so much better than the rest of studied algorithms in this field. This performance can be obvious more in iris dataset since the algorithm detect just two clusters and no noise and all the data objects are allocated to the right clusters as shown in Figure 5.10.

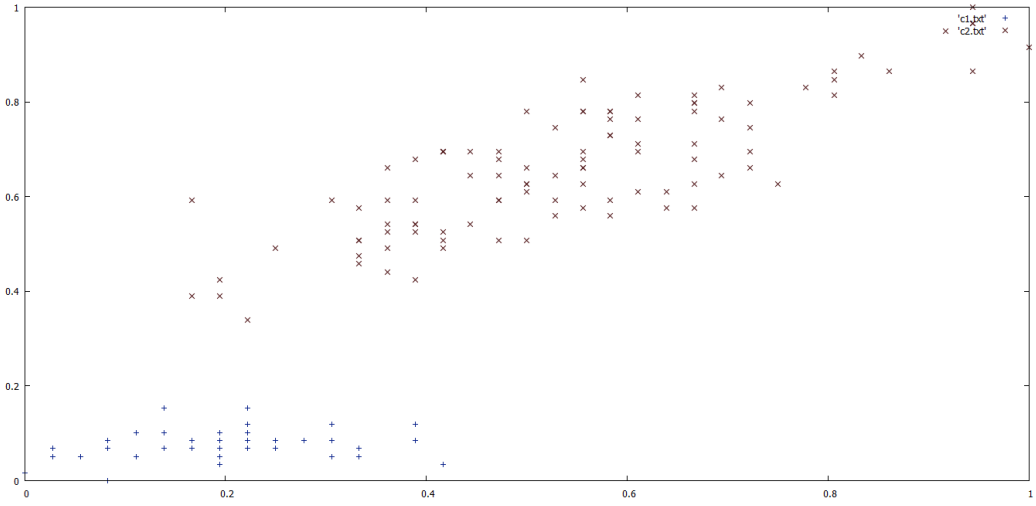


Figure 5.10 clustering result of iris dataset using FNGMDBSCAN-UR clustering algorithm

To explain the results more next table 5.4 and 5.5 show the results cluster by cluster and the accuracy of the algorithm. Table 5.4 shows the number of clusters and the number of points in each cluster, also the number of error points in each cluster and the accuracy present for each cluster for the chameleon dataset. And the same thing for table 5.5 for iris dataset as follow.

Table 5.4: clustering algorithm result for chameleon dataset using FNGMDBSCAN-UR

Clusters	No. of points	No. of Error points	Accuracy rate %
C1	1646	96	94.16
C2	1711	30	98.24
C3	1846	41	97.77
C4	650	42	93.53
C5	962	25	97.40
C6	642	33	94.85
Noise	398		
Runtime (ms)	7604		
Total accuracy	96.60%		

Table 5.5: clustering algorithm result for iris dataset using FNGMDBSCAN-UR

Clusters	No. of points	No. of Error points	Accuracy rate %
C1	50	0	100
C2	100	50	50
Noise	0		
Runtime (Ms)	68		

5. 6 Results Summary

5. 6.1 Artificial “Chameleon” Dataset

In this section, we will start a comparison between clustering algorithms that we used on the chameleon dataset to Figure out which clustering algorithm gives the most accurate result between all. In table 5.6 each row illustrates the cluster number and the quality of each cluster in the data set after applying the three clustering algorithms: DBSCAN, GMDBSCAN-UR, and FNGMDBSCAN-UR respectively. Looking deeply at the table we will notice the difference in the percentage of quality for each algorithm. The results tell how the proposed algorithm is strong and accurate in detecting the clusters and noise in the dataset.

Table 5.6: the quality results of three clustering algorithms on chameleon dataset

Cluster number	**GMDBSCAN-UR accuracy %**	**FNGMDBSCAN accuracy %**
C1	91.91	94.16
C2	96.71	98.24
C3	94.95	97.77
C4	86.49	93.53
C5	96.48	97.40
C6	93.77	94.85
Total	94.10	96.60

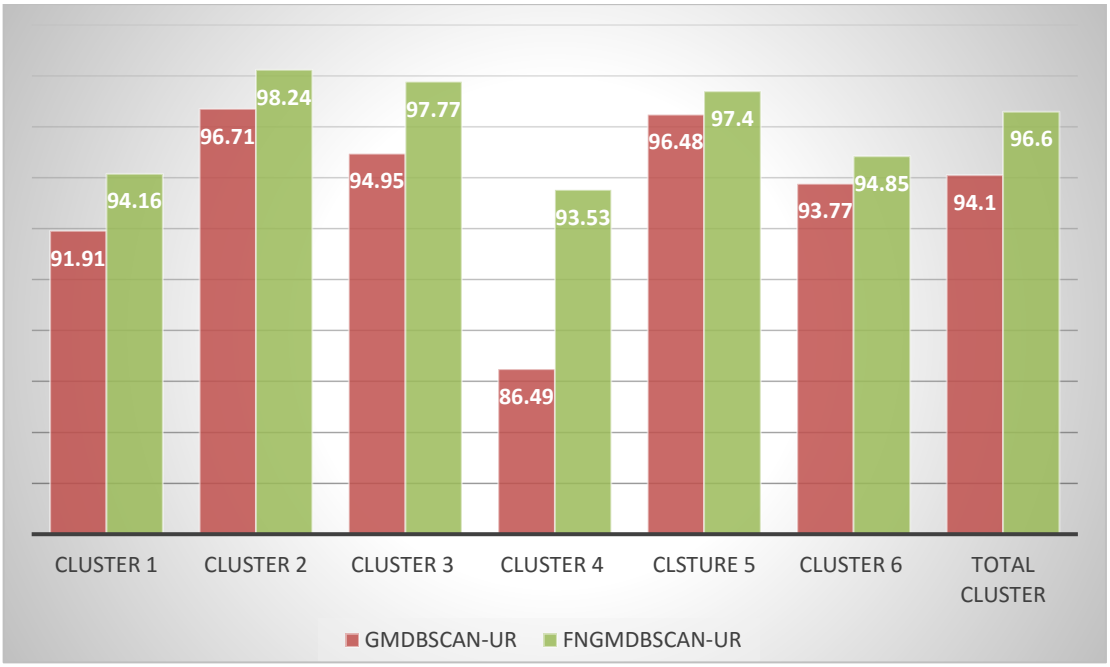


Figure 5.11: chameleon dataset clustering results curves using two clustering algorithms

As we can see always our algorithm FNGMDBSCAN-UR is much better in quality and it always gives a high percent more than the rest two clusters.

5. 6.2 Real "IRIS" Dataset

Iris dataset is considered as a small dataset because of the small number of elements inside it. Table 5.7 shows the different results that we have from running the clustering algorithms GMDBSCAN-UR, and FNGMDBSCAN-UR on iris dataset and comparison in accuracy between them.

Table 5.7: the quality of clustering algorithms over iris dataset

Cluster number	GMDBSCAN-UR accuracy %	FNGMDBSCAN accuracy %
C1	92.59	100
C2	96.70	100
Total	95.17	100

The following chart is the comparison between GMDBSCAN-UR and FNGMDBSCAN-UR. The chart shows the quality for each of them and how our

proposed algorithm is working very well comparing with other clustering algorithms which are used in this study.

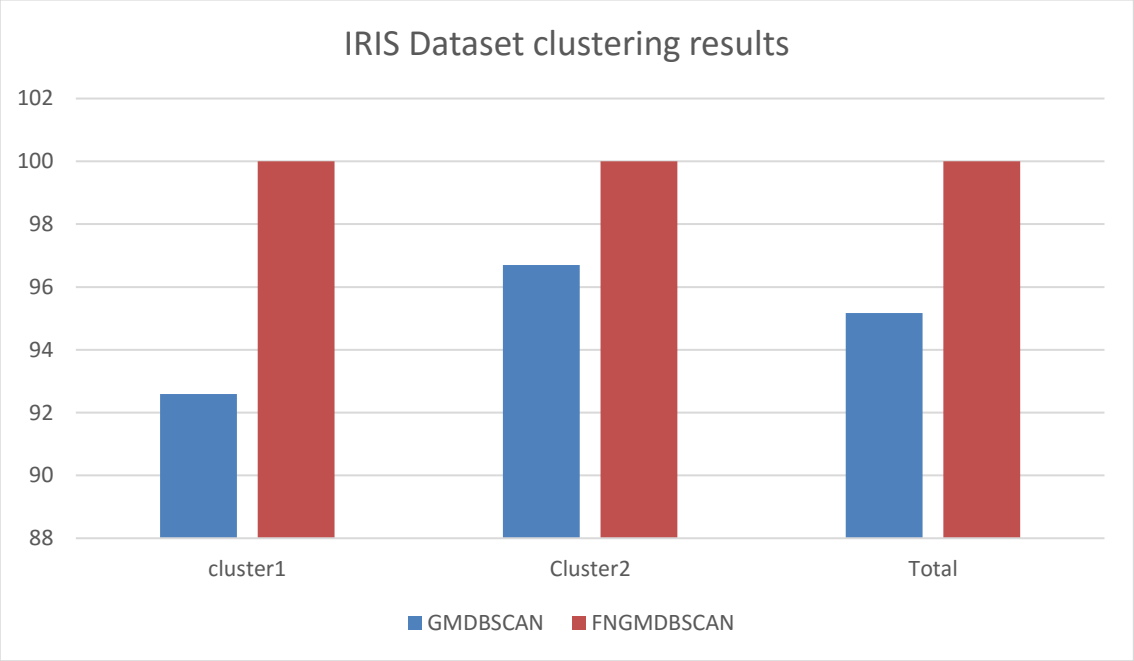


Figure 5.12: iris dataset clustering results in curves using three clustering algorithms

CHAPTER 6

Conclusion and Future Work

In this study, we proposed a new clustering algorithm that use the fuzzy neighborhood function and is based on DBSCAN clustering algorithm. Also, our algorithm works over a very important approach, that is, the representative points, and splitting to grids approach. We used two well-known datasets in the research field to test our algorithm "FN-GMDBSCAN-UR" and its performance. Algorithm gives a very good result in the performance and speed ranges.

Also instead of using the crisp function to calculate the distance between two points we use fuzzy neighborhood function to calculate the membership degree of the point. We can assign the fuzzy neighborhood sets and the fuzzy core points using this technique. The approaches that our algorithm based on are: split the data space to a net that contains a number of cells. Take two datasets one for the delegated nodes, and the other for the rest of the nodes in the dataset. Operate a local clustering process to find, but using the fuzzy theory sets to calculate the contribution of nodes in each cluster. Combine the similar clusters in the dataset if there are. Bring the rest of data and attach them to the nearest fuzzy core point. Algorithm handles with noise and border nodes. FN-GMDBSCAN-UR algorithm gave great results with multi-dense datasets and also with the normal dataset as well. Algorithm shows that it can deal with a dataset that contains complex shapes and give very good accurate results as well.

Our study can be improved in many phases that are mentioned below:

In the previous sections we instead of using the linear way to calculate the fuzzy neighborhood membership degree, we can use the exponential function. This is better than the linear function in efficiency. The node representation method can be changed

as well. Since we take some random representative nodes, we can take representative points that are the core points and in this way, the time consumption of the algorithm will be less. Because the processing time for taking representative points and assign the core points in the datasets will be saved in just one step.

REFERENCES

[1] Alhanjouri, M., and Ahmed, R., (2013). " New Density-Based Clustering Technique: GMDBSCAN-UR", International Journal of Advanced Research in Computer Science, 1: 976-5697.

[2] Berkhin, P., (2006). "Survey of Clustering Data Mining Techniques", Springer Berlin Heidelberg, 2: 25-71.

[3] Abudalfa, S., and Mikki, M., (2013). "A Dynamic Linkage Clustering Using KD-Tree", The International Arab Journal of Information Technology, 10 (3): 25-55.

[4] Chormunge, S., Jena, S. and Sanjay, C., (2014). "Performance Evaluation of Clustering Methods for Low and High Dimensional Data", International Journal of Advanced Research in Computer Science, 2: 976-5697.

[5] Parker, J., (2013). Accelerated Fuzzy Clustering, Ph.D. Thesis, University of South Florida, USA.

[6] Kotsıantıs, S. and Pıntelas, P., (2013). "Recent Advances in Clustering: A Brief Survey", International Journal of Conceptions on Computing and Information Technology, 1(1): 2345 – 9808.

[7] Nasibov, E. and Ulutagay, G., (2009). " Robustness of Density-based Clustering Methods with Various Neighborhood Relations", Science Direct, 160: 3601–3615.

[8] Babu, B., Chandra, N. and Gopal, T., (2016). "Clustering Algorithms For High Dimensional Data – A Survey Of Issues And Existing Approaches", 2(1): 2231–5292.

[9] Parker, J., Hall, L. and Kandel, A., (2013). "Scalable Fuzzy Neighborhood DBSCAN", Iranian Journal of Fuzzy Systems, 10(3): 1-20.

[10] Ester, M., . Kriegel, H., Sander, J. and Xu, X., (1998). "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", 2nd International Conference on Knowledge Discovery and Data Mining, 2(2): 169-194.

[11] Parker, J., Hall, L. and Bezdek, J.,(2012). "Comparison of Scalable Fuzzy Clustering Methods", IEEE International Conference on Fuzzy Systems, 10-15 June 2012, Brisbane, Australia.

[12] Singh, V. and Sood, M., (2013). “Krill Herd Clustering Algorithmusing DBSCAN Technique”, International Journal of Computer Science & Engineering Technology (IJCSET), 4(3): 2229-3345.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname : Abdallah R.S MEKKY

Date of birth and place : 24/3/1990 - Palestine

Foreign Languages : Arabic, English and Turkish

E-mail : Abdallah_makki@outlook.com

EDUCATION

Degree	Department	University	Date of Graduation
Master Degree	Computer Engineering	Yildiz technical University	2016
Undergraduate	Computer Engineering	Islamic University of Gaza	2013
High School	Normal school	Alkarmel High school	2008

WORK EXPERIENCE

Year	Corporation/Institute	Enrollment
2016	Sajidat for E-Trading	WEB CONTENT MANAGER
2012	Controid company	Co-founder and programmer
2011	Municipality of GAZA	VOLUNTEER

PUBLISHMENTS

Conference Papers

1 Altun, O. and Mekky, A., (2016). “Fuzzy Neighborhood Grid-Based DBSCAN Using Representative Points”, The Third International Conference on Data Mining, Internet Computing, and Big Data, 22-24 Julay 2016, Konya, Turkey .