

Computer Engineering Department  
Faculty of Engineering  
Deanery of Higher Studies  
The Islamic University-Gaza  
Palestine



# NEW DENSITY-BASED CLUSTERING TECHNIQUE

Rwand D. Ahmed

Supervisor:  
Dr. Mohammed A. Alhanjouri

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of  
Master of Science in Computer Engineering

**GAZA, PALESTINE**  
**(SEPTEMBER, 2011) 1432 H**

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



الجامعة الإسلامية - غزة  
The Islamic University - Gaza

هاتف داخلي: 1150

عمادة الدراسات العليا

ج س غ/35/  
الرقم Ref ..... 2011/09/25




التاريخ Date .....

## نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحثة/ روندة داود عبد المجيد أحمد لنيل درجة الماجستير في كلية الهندسة قسم هندسة الحاسوب وموضوعها:

### NEW DENSITY BASED CLUSTERING TECHNIQUE

وبعد المناقشة التي تمت اليوم الأحد 27 شوال 1432هـ، الموافق 2011/09/25م الساعة الحادية عشرة صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

	مشرفاً ورئيساً	د. محمد أحمد الحنجوري
	مناقشاً داخلياً	أ.د. إبراهيم سليمان أبو هيبه
	مناقشاً داخلياً	د. وسام محمود عاشور

وبعد المداولة أوصت اللجنة بمنح الباحثة درجة الماجستير في كلية الهندسة / قسم هندسة الحاسوب.

واللجنة إذ تمنحها هذه الدرجة فإنها توصيها بتقوى الله ولزوم طاعته وأن تسخر علمها في خدمة دينها ووطنها.

والله ولي التوفيق،،،

عميد الدراسات العليا



أ.د. فؤاد علي العاجز

## *Dedication*

*I would like to dedicate this work with my deep love to:*

*To my beloved parents ...*

*To my brothers and sisters ...*

*To my friends ...*

The Researcher  
Rwand D. Ahmed

# *Acknowledgements*

Firstly, I thank Almighty ALLAH for making this work possible. Then, I would like to thank Dr. Mohammed Alhanjouri, my supervisor, for his support and guidance throughout the thesis. Also I would like to thank my instructors in the field of computer engineering, particularly Prof. Ibrahim Abuhaiba, for supporting my research and fostering my interest in the subject. Special thanks for Dr. Wesam Ashour for his valuable guide and comments.

# Contents

Dedication.....	ii
Acknowledgements .....	iii
Contents .....	iv
List of Figures .....	vii
List of Tables .....	ix
List of Abbreviations.....	x
Abstract in Arabic .....	xi
Abstract in English .....	xii
<b>Chapter 1 : Introduction .....</b>	<b>1</b>
1.1 Clustering Definition .....	1
1.2 Definitions and Preliminaries .....	3
1.3 Clustering Algorithms .....	4
1.4 Our Contribution .....	6
1.5 Thesis Structure .....	7
<b>Chapter 2 : Related Work.....</b>	<b>8</b>
2.1 DBSCAN: A Density-Based Clustering Method Based on Connected Regions with Sufficiently High Density.....	10
2.2 OPTICS: Ordering Points to Identify the Clustering Structure.....	12
2.3 DENCLUE: Clustering Based on Density Distribution Functions .....	13
2.4 MDBSCAN Clustering Algorithm.....	14
2.5 GMDBSCAN Clustering Algorithm.....	16
2.6 Cure Clustering Algorithm .....	19

2.7 Grid- Based Clustering Algorithms .....	21
<b>Chapter 3: Techniques and Background .....</b>	<b>23</b>
3.1 Density-Reachability and Density Connectivity in DBSCAN Algorithm .....	23
3.2 Data Types in Clustering Analysis .....	24
3.3 Similarity and Dissimilarity.....	25
3.4 Scale Conversion.....	25
3.5 GMDBSCAN-UR Related Definitions .....	26
<b>Chapter 4: Proposed Algorithm "GMDBSCAN-UR" .....</b>	<b>27</b>
4.1 Our Adopted Idea .....	27
4.2 GMDBSCAN-UR Steps .....	28
4.2.1 Datasets Input and Data Standardization.....	28
4.2.2 Dividing Dataset into Smaller Cells.....	29
4.2.3 Chosen Representative Points.....	30
4.2.4 Selecting MinPts and Eps Parameters.....	31
4.2.5 Bitmap Forming .....	32
4.2.6 Local-Clustering and Merging the Similar Sub-clusters using DBSCAN Algorithm .....	32
4.2.7 Labeling and Post Processing.....	33
4.2.8 Noise Elimination.....	34
4.3 GMDBSCAN-UR Algorithm Properties.....	34
4.4 GMDBSCAN-UR Algorithm Pseudo-Code .....	36
<b>Chapter 5 : Experimental Results.....</b>	<b>37</b>
5.1 Adult Dataset.....	37
5.2 Chameleon Dataset .....	38

5.3 DBSCAN Results.....	39
5.4 MDBSCAN Results .....	43
5.5 GMDBSCAN Results .....	46
5.6 GMDBSCAN-UR Results .....	49
5.7 The Results Summary .....	55
5.7.1 Adult Final Results .....	55
5.7.2 Chameleon Final Results .....	57
<b>Chapter 6 : Conclusion And Future Work.....</b>	<b>59</b>
6.1 Conclusion .....	59
6.2 Future Work .....	59
<b>References.....</b>	<b>61</b>

# *List of Figures*

Figure 1.1: Diagram of clustering algorithms.....	2
Figure 2.1: Clustering with K-MEANS and PAM algorithms.....	9
Figure 2.2: Clustering an artificial data set with CURE algorithm.....	10
Figure 2.3: Framework of MDBSCAN algorithm .....	15
Figure 2.4: Framework of MDBSCAN algorithm .....	17
Figure 2.5: Clusters generated by hierarchical algorithms.....	20
Figure 3.1: Density reachability and density connectivity in density-based clustering.....	24
Figure 4.1: Multi-density dataset .....	29
Figure 4.2: Multi-density Dataset. ....	30
Figure 4.3: Dividing the dataset to smaller cells .....	30
Figure 4.4: Taking a well scattered representative points in each cell .....	30
Figure 4.5: Dataset along with its representative points.....	31
Figure 4.6: Using same MinPts with varying Eps.....	31
Figure 4.7: Using same <i>Eps</i> with varying <i>MinPts</i> .....	32
Figure 5.1: Adult dataset.....	37
Figure 5.2: Chameleon dataset.....	39
Figure 5.3: Adult dataset clustering results using DBSCAN algorithm.....	40
Figure 5.4: Chameleon dataset clustering results using DBSCAN algorithm.....	41
Figure 5.5: Adult dataset clustering results using MDBSCAN algorithm.....	44
Figure 5.6: Chameleon dataset clustering results using MDBSCAN algorithm.....	45
Figure 5.7: Adult dataset clustering results using GMDBSCAN algorithm.....	47
Figure 5.8: Chameleon dataset clustering results using GMDBSCAN algorithm.....	48
Figure 5.9: Adult dataset clustering results using GMDBSCAN-UR algorithm.....	50
Figure 5.10: Chameleon dataset clustering results using G MDBSCAN-UR algorithm.....	51
Figure 5.11: The result of choosing representative points from "chameleon" with vvvvvvvvv GDBSCAN-UR algorithm.....	53
Figure 5.12: The result of choosing representative points from chameleon dataset with vvvvvvvvv GDBSCAN-UR algorithm and labeling the remainder points of the dataset....	53
Figure 5.13: The final result of choosing representative points from "chameleon", labeling vvvvvvvvv the remainder points of the dataset and remerging steps with GDBSCAN-UR vvvvvvvvv algorithm .....	54



Figure 5.14: The result of clustering adult dataset with GDBSCAN-UR algorithm.....	55
Figure 5.16: Adult dataset clustering results curves using the four algorithms.....	56
Figure 5.17: Chameleon dataset clustering results curves using the four algorithms.....	58

# List of Tables

Table 5.1: Adult dataset specification.....	38
Table 5.2: Adult and chameleon datasets clustering results summary using DBSCAN algorithm.....	42
Table 5.3: Adult and chameleon datasets clustering results summary using MDBSCAN algorithm..	46
Table 5.4: Adult and chameleon datasets clustering results summary using G MDBSCAN algorithm	49
Table 5.5: Adult and chameleon datasets clustering results summary using G MDBSCAN-UR nnnnnnnnn algorithm.....	52
Table 5.6: Adult dataset comparative clustering results summary using the four algorithms.....	56
Table 5.7: Chameleon dataset comparative clustering results summary using the four algorithms....	57

## *List of Abbreviations*

<b>CD</b>	Cell Density.
<b>CNum</b>	The Cell number (ID).
<b>CURE</b>	Clustering Using REpresentatives.
<b>DBSCAN</b>	A Density-Based Clustering Method Based on Connected Regions with Sufficiently High Density.
<b>DENCLUE</b>	Clustering Based on Density Distribution Functions "DENsity-based CLUstEring".
<b>Eps</b>	The radius of a number of objects.
<b>GMDBSCAN</b>	Grid-Based MDBSCAN.
<b>GMDBSCAN-UR</b>	Grid-based MDBSCAN Using Representatives.
<b>GNum</b>	The Number of points in the Grid.
<b>MDBSCAN</b>	A Multi-Density DBSCAN.
<b>MinPts</b>	The Minimum number of objects (Points) each object of a cluster the neighborhood of a given radius ( <i>Eps</i> ) has to contain.
<b>NEps(p)</b>	The Number of points in Eps of the point p.
<b>NextLay</b>	A pointer which points to a leaf cell in layer d.
<b>ND</b>	The Node-Density.
<b>OPTICS</b>	Ordering Points to Identify the Clustering Structure.
<b>P</b>	Some point, P.
<b>VEps</b>	The Data point's Neighborhood volume.
<b>VCell</b>	The Cell Volume.

# طريقة جديدة للعقدة اعتمادا على الكثافة

روندة داود أحمد

## الملخص

خوارزمية الاعتماد على الكثافة المكانية للتطبيقات في العقدة هي أحد الخوارزميات المعتمدة على الكثافة وأشهرها لتحليل التجمعات ويمكنها اكتشاف تجمعات بأشكال تعسفية مختلطة بالوضوءاء. ولكن هذه الخوارزمية لا تختار التجمعات وفقا لتوزيع البيانات المعتمد على الكثافة، وهي ببساطة تستخدم معامل لعدد النقاط الأدنى اللازم و يتم اختياره على مستوى جميع البيانات باختلاف كثافتها وبالتالي نتيجة عقدة قاعدة البيانات متعددة الكثافات تكون غير دقيقة، بالإضافة إلى حالات استخدام قواعد البيانات العنقودية الكبيرة، هذا سوف يستغرق الكثير من الوقت للعقدة ، و لحل المشاكل باقتراح خوارزمية جديدة، تعتمد على تقسيم البيانات إلى شبكة من الخلايا بالإضافة لاستخدام النقاط الممثلة للبيانات، خوارزمية الاعتماد على الكثافة المتعددة المكانية باستخدام النقاط الممثلة. في هذا البحث، نحن نطبق آلية جديدة غير خاضعة للرقابة في التعلم تعتمد على خوارزمية الكثافة المكانية الديسكان، و نحن نقترح الاعتماد على تقنية تقسيم البيانات إلى شبكة من الخلايا لتقليل الوقت المستغرق في عملية العقدة ، يتم اختيار عدد من النقاط الموزعة بشكل جيد من كل خلية، هذه النقاط يجب أن تأخذ شكل وحجم البيانات الكلي. و بذلك عملنا في هذا البحث يتبنى حلا وسطا بين اختيار نقطة واحدة متوسطة و استخدام جميع النقاط لتمثيل البيانات. بعد ذلك نُعامل جميع البيانات داخل الخلية الواحدة ككائن واحد ونطبق جميع العمليات داخل تلك الخلية. نطبق العقدة في كل خلية و ندمج بين التجمعات الناتجة. استخدام الحد الأدنى من النقاط لمعامل محلي في كل خلية في الشبكة يجعلنا نتغلب على مشكلة عدم القدرة على تحديد التجمعات في البيانات المتعددة الكثافة في حالة العقدة باستخدام خوارزمية الديسكان. هذا بدوره سوف يحسن في الوقت. و بعد ذلك نضع علامات للنقاط التي لم يتم اختيارها كنقاط ممثلة لتصبح تنتمي للتجمعات الناتجة. و أخيرا، تأتي خطوة التخلص من الضوضاء.

# NEW DENSITY-BASED CLUSTERING TECHNIQUE

RWAND D. AHMED

## ABSTRACT

Density Based Spatial Clustering of Applications of Noise (DBSCAN) is one of the most popular algorithms for cluster analysis. It can discover clusters with arbitrary shape and separate noises. But this algorithm cannot choose its parameter according to distribution of dataset. It simply uses the global minimum number of points (*MinPts*) parameter, so that the clustering result of multi-density database is inaccurate. In addition, when it used to cluster large databases, it will cost too much time. We try to solve these problems by integrated the grid-based in addition to using representative points in our new proposed density-based GMDBSCAN-UR clustering algorithm. In this research, we apply an unsupervised machine learning approach based on DBSCAN algorithm. We propose a grid-based cluster technique to reduce the time complexity. Grid-based technique divides the data space into cells. A number of well scattered points in each cell in the grid are chosen. These scattered points must capture the shape and extent of the dataset as all. Thus, our work in this research adopts a middle ground between the centroid-based and the all-point extremes. Next we treat all data in the same cell as an object, and all the operations of clustering are done on this cell. We make local clustering in each cell and merge between the resulted clusters. We use local *MinPts* for every cell in the grid to overcome the problem of undetermined clusters in multi-density datasets in clustering with DBSCAN clustering algorithm case. This will enhance the time complexity. Next step is labeling the not chosen points to the resulted clusters. Finally, we make post processing and noise elimination.

Keywords: *Dbscan* , *Multi-density*, *Grid-based*, *Representative*.

# Chapter 1

## Introduction

### 1.1 Clustering Definition

Clustering is the process of grouping the data into classes or clusters, so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters. Dissimilarities are assessed based on the attribute values describing the objects. Often, distance measures are used [1]. The field of clustering has undergone major revolution over the last few decades; it has its roots in many areas, including data mining, statistics, biology, and machine learning. Clustering is characterized by advances in approximation and randomized algorithms, novel formulations of the clustering problem, algorithms for clustering massively large data sets, algorithms for clustering data streams, and dimension reduction techniques [2].

We study the requirements of clustering methods for large amounts of data and explain how to compute dissimilarities between objects represented by various attribute or variable types. Several studies examine a lot of clustering techniques, organized into the following categories: partitioning methods, hierarchical methods, density-based methods, grid-based methods, model-based methods, methods for high-dimensional data (such as frequent pattern-based methods), and constraint-based clustering [1].

Data mining has attracted a great deal of attention in the information industry and in society as a whole in recent years, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge which can be used for applications ranging from market analysis, fraud detection, and customer retention, to production control and science exploration. Data mining can be viewed as a result of the natural evolution of information technology in a lot of functionalities such as *data collection and database creation*, *data and advanced data analysis* (involving data warehousing and data mining) [1]. Clustering, which divides the data to disparate clusters, is a crucial part of data mining. The objects within a cluster are "similar," whereas the objects of different clusters are "dissimilar" [3]. Clustering is one of the most useful tasks in data mining process. Data clustering, also called cluster analysis, segmentation analysis, taxonomy analysis, or unsupervised classification, is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and

objects in different clusters are quite distinct. Data clustering is often confused with classification, in which objects are assigned to predefined classes. In data clustering, the classes are also to be defined [1]. There are many algorithms used for clustering such that: hierarchical clustering techniques, fuzzy clustering algorithms, center-based clustering algorithms, search-based clustering algorithms, graph-based clustering algorithm, grid-based clustering algorithms, density-based clustering algorithms, model-based clustering algorithms, subspace clustering [1] as shown in Figure 1.1.

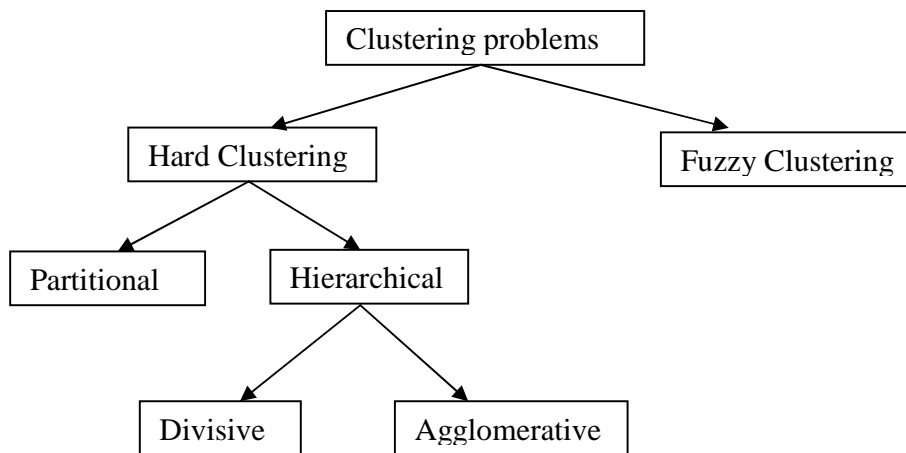


Figure 1.1: Diagram of clustering algorithms.

There are many algorithms that deal with the problem of clustering large number of objects. The different algorithms can be classified regarding different aspects. These methods can be categorized into partitioning methods [4, 5, 6], hierarchical methods [4, 7, 8], density based methods [9, 10, 11], grid based methods [12, 13, 14], and model based methods [15, 16].

Here in this research, we concentrate around the topic of DBSCAN algorithm, (Density-Based Spatial Clustering of Applications with Noise), and enhance it at all, time and space complexity, support multi-density grid based clustering in effective and efficient way.

DBSCAN checks the Eps-neighborhood of each point in database. If Eps- neighborhood of a point  $p$  contains more than  $MinPts$ , a new cluster with  $p$  as a core object is created. It then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters. The process terminates when no new point can be added to any cluster [17]. The conventional DBSCAN and its improved algorithms presented in

papers [18, 19, 20, 21] can only process the numerical data. They are incapable of processing data with categorical attributes. Usually, the densities of dataset used in cluster analyses are different, however, until now there is no a very effective algorithm to get the accurate density of the dataset with multi-density. DBSCAN [18], density-based clustering not only availably avoids noises but also effectually clusters various datasets, whereas; for the multi-density dataset, DBSCAN is not a good algorithm for which the runtime complexity is high [1]. In order to reduce the time complexity, the academia has presented a grid-based cluster technique [22, 23], which divides the data space into disjunctive grid. The data in the same grid can be treated as a unitary object, and all the operations of clustering are on the grid [22].

## 1.2 Definitions and Preliminaries

The following terms are used throughout the thesis:

**Definition 1.1:** A *Cluster*: is a well defined collection of similar patterns and patterns from two different clusters must be dissimilar.

**Definition 1.2:** A *Hard (or crisp) clustering algorithm* : is a clustering algorithm that assigns each pattern to one and only one cluster.

**Definition 1.3:** A *Fuzzy clustering algorithm*: is an algorithm that assigns each pattern to each cluster with a certain degree of membership.

**Definition 1.4:** *Hierarchical Divisive Clustering Algorithm*: the algorithm proceeds from the top to the bottom, i.e., the algorithm starts with one large cluster containing all the data points in the data set and continues splitting clusters

**Definition 1.5:** *Hierarchical Agglomerative Clustering Algorithm*: the algorithm proceeds from the bottom to the top, i.e., the algorithm starts with clusters each containing one data point and continues merging the clusters

**Definition 1.6:** *Partitional Clustering Algorithms*: the algorithms those create a one-level non overlapping partitioning of the data points.

**Definition 1.7:** A *Distance Measure*: is a metric based on which the dissimilarity of the patterns are evaluated.

**Definition 1.8:** *CURE*: is an algorithm that identifies clusters having non-spherical shapes and wide variances in size by representing each cluster by a certain fixed number of points that are



generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction.

**Definition 1.9:** *ROCK*: is an algorithm that measures the similarity of two clusters by comparing the aggregate inter-connectivity of two clusters against a user-specified static inter-connectivity model.

**Definition 1.10:** *DBSCAN*: is a density based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise.

**Definition 1.11:** *OPTICS*: is an algorithm that computes an augmented cluster ordering for automatic and interactive cluster analysis and produces a data set clustering explicitly. It creates an ordering of the objects in a database, additionally storing the core-distance and a suitable reachability distance for each object. An algorithm was proposed to extract clusters based on the ordering information

**Definition 1.12:** *DENCLUE*: is a method that clusters objects based on the analysis of the value distributions of density functions.

**Definition 1.13:** *MDBSCAN*: is an algorithm that uses must link constraints in order to calculate parameters to ascertain Eps for each density distribution automatically, which used to deal with multi-density data sets.

**Definition 1.14:** *GMDBSCAN*: is an algorithm that based on spatial index and grid technique. It is used to cluster large databases.

**Definition 1.15:** *GMDBSCANUR*: the proposed algorithm in this study. It is a multi density clustering algorithm based on grid and uses representative points.

**Definition 1.16:** *Cell Density*: is the amount of data in a cell.

### 1.3 Clustering Algorithms

There are thousands of clustering techniques one can encounter in the literature. Most of the existing data clustering algorithms can be classified as Hierarchical or Partitional. Within each class, there exists a wealth of sub-class which includes different algorithms for finding the clusters.

While hierarchical algorithms [24] build clusters gradually (as crystals are grown), partitioning algorithms [7] learn clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets, or try to identify clusters as areas highly populated with data.

Density based algorithms [25] typically regard clusters as dense regions of objects in the data space that are separated by regions of low density. The main idea of density-based approach is to find regions of high density and low density, with high-density regions being separated from low-density regions. These approaches can make it easy to discover arbitrary clusters. Recently, a number of clustering algorithms have been presented for spatial data, known as grid-based algorithms. They perform space segmentation and then aggregate appropriate segments [26].

Many other clustering techniques are developed, primarily in machine learning, that either have theoretical significance, are used traditionally outside the data mining community, or do not fit in previously outlined categories. So we can summarize the clustering algorithms as follows [27]:

- Hierarchical Methods
  - Agglomerative Algorithms
  - Divisive Algorithms
- Partitioning Methods
  - Relocation Algorithms
  - Probabilistic Clustering
  - K-medoids Methods
  - K-means Methods
  - Density-Based Algorithms
    - Density-Based Connectivity Clustering
    - Density Functions Clustering
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Constraint-Based Clustering
- Clustering Algorithms Used in Machine Learning

- Gradient Descent and Artificial Neural Networks
- Evolutionary Methods
- Scalable Clustering Algorithms
- Algorithms For High Dimensional Data
  - Subspace Clustering
  - Projection Techniques
  - Co-Clustering Techniques

Clustering is a challenging field of research in which its potential applications pose their own special requirements. The following are typical requirements of clustering in data mining:

- Type of attributes algorithm can handle.
- Scalability to large data sets.
- Ability to work with high dimensional data [28, 29].
- Ability to find clusters of irregular shape.
- Handling outliers (noise).
- Time complexity.
- Data order dependency.
- Labeling or assignment (hard or strict vs. soft or fuzzy [30, 31, 32]).
- Reliance on a priori knowledge and user defined parameters.
- Interpretability of results.

However, clustering is a difficult problem combinatorial, and differences in assumptions and contexts in different communities have made the transfer of useful generic concepts and methodologies slow to occur.

## **1.4 Our Contribution**

This research is principally concerned with the theoretical and experimental study of a set of multi-density clustering algorithms. And then make improvements on these clustering algorithms results in both quality and time.

The contribution of this thesis is that we developed a new clustering algorithm named "GMDBSCAN-UR", Grid-based Multi-density DBSCAN Using Representative, by using sp-tree

for clustering complicated and complex shaped datasets in a fast and accurate fashion based on grid and uses representative points that represent the dataset which leads to give the clustering result in an early time compared with using all points in the datasets which leads to a time consuming. Then the remaining points are labeled to the clusters based on that each non representative point to which cluster is the corresponding nearest representative point belongs. Experimental results are shown in this thesis to demonstrate the effectiveness of the proposed algorithms. We compared our proposed algorithm results with other famous related algorithms results. And we present that our new proposed algorithm is the best one in both quality and time.

## **1.5 Thesis Structure**

The rest of the report is organized as follows: Chapter 2 talks about related work which discusses the clustering problem and background, Chapter 3 summarizes the methodology of the new proposed clustering algorithm and a number of concepts related to the techniques used in our proposed algorithm; then Chapter 4 illustrates the idea of the new proposed algorithm with more details and shows our contribution for improving efficiency of GMDBSCAN-UR to cluster complex data sets, Chapter 5 shows the experimental results which compare the our new proposed algorithm with the previous density-based algorithms; and finally Chapter 6 concludes the thesis and presents suggestions for future work.

# Chapter 2

## Related Work

Clustering has been studied extensively for more than 40 years and across many disciplines due to its broad applications. Most books on pattern classification and machine learning contain chapters on cluster analysis or unsupervised learning. Several textbooks are dedicated to the methods of cluster analysis. Clustering algorithms group the data points according to various criteria, as discussed by Jain and Dubes (1988) [33], Fukunaga (1990) [34], Clustering may proceed according to some parametric model, as in the k-means algorithm of MacQueen (1967)[35], or by grouping points according to some distance or similarity measure as in hierarchical clustering algorithms. Other approaches include graph theoretic methods, such as Shamir and Sharan (2000)[36], physically motivated algorithms, as in and algorithms based on density estimation as in Fukunaga (1990)[34]. First we will talk about Semi-supervised clustering, and then we will talk about the famous density-based algorithms. Semi-supervised clustering, which uses class labels or pairwise constraints on some examples to aid unsupervised clustering, has been the focus of several recent projects [37]. Existing methods for semi-supervised clustering fall into two general approaches: constraint-based and distance based methods. At present, many scholars incorporated pairwise constraints into state-of-art clustering algorithms. Kiri Wagstaff et al. [38] incorporated pairwise constraints into k-means algorithm so as to satisfy these constraints in the process of clustering; Sugato Basu et al. [37] proposed the PCK-Means algorithm which modifies the objective function of clustering so that these constraints can be satisfied in some degree, however it must rely on parameters and a large number of constraints; Nizar Grira et al. [39] proposed the PCCA algorithm which was used to image database categorization; Davidson et al. [40] enhanced the hierarchical clustering with pairwise constraints, and presented intractability results for some constraint combinations [41]. Wei Tang et al. [42] proposed a feature projection method with pair wise constraints ,which can handle the high-dimension sparse data effectively. There are a lot number of clustering algorithms like K-means, PAM (Pattern Analysis and Machine Intelligence), Cure (Clustering Using REpresentative) , Rock (Robust Clustering using linKs) and DBSCAN clustering

algorithms, all have advantages and disadvantages over other. Here we shall summarize some of them quickly in order to compare the number of them with the proposed algorithm and highlight the effectiveness of the new proposed algorithm over the old ones.

Partitioning techniques like K-MEANS and PAM clustering algorithms assume clusters are globular and are of similar sizes. Both fail in large variation in cluster sizes and when cluster shapes are convex as in Figure 2.1 below. The dataset in Figure 2.1 below contains two convex clusters. K-MEANS and PAM clustering algorithms fail to find the correct clusters, so that the right cluster take points from the left one and the vice versa, and it is wrong result.

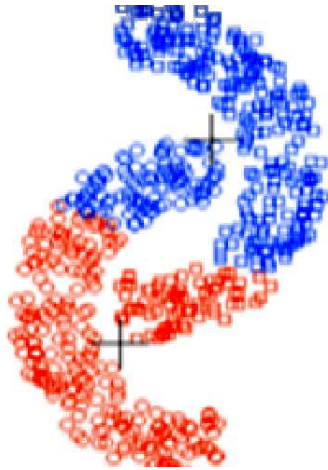


Figure 2.1: Clustering with K-MEANS and PAM algorithms.

Hierarchical Techniques like CURE and ROCK clustering algorithms use a static models to determine the most similar cluster to merge in the hierarchical clustering. CURE measures the similarity of two clusters based on the similarity of the closest pair of the representative points belonging to different clusters, without considering the internal closeness (i.e., density or homogeneity) of the two clusters involved. It fails to take into account special characteristics and shapes as in Figure 2.2 below, we get a wrong clustering result . ROCK measures the similarity of two clusters by comparing the aggregate inter-connectivity of two clusters against a user-specified static inter-connectivity model, and thus it ignores the potential variations in the inter-connectivity of different clusters within the same dataset.



Figure 2.2: Clustering an artificial data set with CURE algorithm.

This chapter is mainly concerned with presenting the density-based algorithms: DBSCAN, OPTICS (Ordering Points to Identify the Clustering Structure), DENCLUE (DENSITY-based CLUSTERing), MDBSCAN (Multi-density DBSCAN), GMDBSCAN (Grid-based Multi-density DBSCAN), Grid-Based algorithm and CURE clustering algorithm. In our research we will talk in more specific and deep about the density-based algorithms which we interested in. The present study is particularly based on developing DBSCAN accompanied by the grid in addition to using representative points.

## 2.1 DBSCAN: A Density-Based Clustering Method Based on Connected Regions with Sufficiently High Density

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise. It defines a cluster as a maximal set of density-connected points[1].

The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius (Eps) has to contain at least a minimum number of objects (*MinPts*), i.e. the cardinality of the neighborhood has to exceed a threshold[43]. DBSCAN checks the Eps-neighborhood of each point  $p$  in the database. If the  $NEps(p)$  has points more than *MinPts*, a new cluster  $C$  containing the points in  $NEps(p)$  is created. Then, the Eps-neighborhood of all points  $q$  in  $C$  which has not yet been processed is checked. If  $NEps(q)$  contains points more than *MinPts*, the neighborhood of  $q$  which is not contained in  $C$  are added to the cluster and their Eps-

neighborhood is checked in the next step. This procedure is repeated until no new point can be added to current cluster  $C$  [10].

The basic ideas of density-based clustering involve a number of new definitions that are intuitively presented here.

- The neighborhood within a radius  $\epsilon$  of a given object is called the  $\epsilon$ -neighborhood of the object.
- If the  $\epsilon$ -neighborhood of an object contains at least a minimum number,  $MinPts$ , of objects, then the object is called a core object.
- Given a set of objects,  $D$ , we say that an object  $p$  is directly density-reachable from object  $q$  if  $p$  is within the  $\epsilon$ -neighborhood of  $q$ , and  $q$  is a core object.
- An object  $p$  is density-reachable from object  $q$  with respect to  $\epsilon$  and  $MinPts$  in a set of objects,  $D$ , if there is a chain of objects  $p_1, \dots, p_n$ , where  $p_1 = q$  and  $p_n = p$  such that  $p_{i+1}$  is directly density-reachable from  $p_i$  with respect to  $\epsilon$  and  $MinPts$ , for  $1 \leq i \leq n$ ,  $p_i$  belongs to  $D$ .
- An object  $p$  is density-connected to object  $q$  with respect to  $\epsilon$  and  $MinPts$  in a set of objects,  $D$ , if there is an object  $o \in D$  such that both  $p$  and  $q$  are density-reachable from  $o$  with respect to  $\epsilon$  and  $MinPts$ .
- Density reachability is the transitive closure of direct density reachability, and this relationship is asymmetric. Only core objects are mutually density reachable. Density connectivity, however, is a symmetric relation[1].

### **DBSCAN Algorithm Problems:**

DBSCAN [45] is a famous density-based clustering method, which can discover the clusters with arbitrary shapes and does not need to know the number of clusters initially in its algorithm. However, DBSCAN needs to know two parameters:  $Eps$  and  $MinPts$  and the value of parameter  $Eps$  is important for DBSCAN algorithm, but the calculation of  $Eps$  is time-consuming, it must draw a sorted k-dist graph for dataset and user determines the first valley as the threshold  $Eps$  in the graphical representation. What's more, due to a single global parameter  $Eps$ , it is impossible to detect some clusters using one global- $MinPts$ . It does not perform well on multi-density data sets. In the multi-density data set, DBSCAN may merge between different clusters and may also



neglect other clusters that assign them as noise. In DBSCAN, the user can specify the values of parameters  $\epsilon$ , but it is difficult.  $\epsilon$  can be calculated by k-dist map, but drawing k-dist map spends a great deal of time. Also the runtime complexity of constructing R\*-tree and implementation of DBSCAN are not linearly [45].

## 2.2 OPTICS: Ordering Points to Identify the Clustering Structure

Although DBSCAN can cluster objects given input parameters such as  $\epsilon$  and MinPts, it still leaves the user with the responsibility of selecting parameter values that will lead to the discovery of acceptable clusters. Actually, this is a problem associated with many other clustering algorithms. Such parameter settings are usually empirically set and difficult to determine, especially for real-world, high-dimensional data sets. Most algorithms are very sensitive to such parameter values: slightly different settings may lead to very different clustering of the data. Moreover, high-dimensional real data sets often have very skewed distributions, such that their intrinsic clustering structure may not be characterized by global density parameters. To help overcome this difficulty, a cluster analysis method called OPTICS was proposed. Rather than produce a data set clustering explicitly, OPTICS computes an augmented cluster ordering for automatic and interactive cluster analysis. This ordering represents the density-based clustering structure of the data. It contains information that is equivalent to density-based clustering obtained from a wide range of parameter settings. The cluster ordering can be used to extract basic clustering information (such as cluster centers or arbitrary-shaped clusters) as well as provide the intrinsic clustering structure. By examining DBSCAN, we can easily see that for a constant MinPts value, density based clusters with respect to a higher density (i.e., a lower value for  $\epsilon$ ) are completely contained in density-connected sets obtained with respect to a lower density. Recall that the parameter  $\epsilon$  is a distance—it is the neighborhood radius. Therefore, in order to produce a set or ordering of density-based clusters, we can extend the DBSCAN algorithm to process a set of distance parameter values at the same time. To construct the different clustering simultaneously, the objects should be processed in a specific order. This order selects an object that is density-reachable with respect to the lowest  $\epsilon$  value so that clusters with higher density (lower  $\epsilon$ ) will be finished first. Based on this idea, two values need to be stored for each object core-distance and reachability-distance:

- The core-distance of an object  $p$  is the smallest  $\epsilon'$  value that makes  $\{p\}$  a core object. If  $p$  is not a core object, the core-distance of  $p$  is undefined.
- The reachability-distance of an object  $q$  with respect to another object  $p$  is the greater value of the core-distance of  $p$  and the Euclidean distance between  $p$  and  $q$ . If  $p$  is not a core object, the reachability-distance between  $p$  and  $q$  is undefined[1].

## 2.3 DENCLUE: Clustering Based on Density Distribution Functions

DENCLUE (DENSity-based CLUstEring) [1] is a clustering method based on a set of density distribution functions. The method is built on the following ideas: (1) the influence of each data point can be formally modeled using a mathematical function, called an influence function, which describes the impact of a data point within its neighborhood; (2) the overall density of the data space can be modeled analytically as the sum of the influence function applied to all data points; and (3) clusters can then be determined mathematically by identifying density attractors, where density attractors are local maxima of the overall density function. Let  $x$  and  $y$  be objects or points in a  $d$ -dimensional input space. The influence function of data object  $y$  on  $x$  is a function

$$f_B^y : F^d \longrightarrow R_0^+ \quad 2.1$$

which is defined in terms of a basic influence function:

$$f_B^y(x) = f_B(x, y) \quad 2.2$$

This reflects the impact of  $y$  on  $x$ . In principle, the influence function can be an arbitrary function that can be determined by the distance between two objects in a neighborhood. The distance function,  $d(x, y)$ , should be reflexive and symmetric, such as the Euclidean distance function. It can be used to compute a square wave influence function,

$$f_{Square}^y(x, y) = \begin{cases} 0 \dots\dots & \text{if } d(x, y) > \sigma \\ 1 \dots\dots\dots & \text{otherwise} \end{cases} \quad 2.3$$

or a Gaussian influence function,

$$f_{Gauss}(x, y) = e^{-\frac{d(x, y)^2}{2\sigma^2}} \quad 2.4$$

## 2.4 MDBSCAN Clustering Algorithm

This algorithm uses must link constraints in order to calculate parameters to ascertain Eps for each density distribution automatically, which used to deal with multi-density data sets for DBSCAN algorithm. MDBSCAN algorithm can reckon the parameter of DBSCAN for multi-density data sets with constraints. MDBSCAN is a method incorporate pairwise constraints (must-link) proposed in some semisupervised clustering algorithms in order to calculate parameters effectively and automatically which will be used to deal with multi-density data sets for traditional DBSCAN algorithm. MDBSCAN can find the clusters of different sizes, shapes and densities in multi-density data sets given pairwise constraints[17].

### Semi-supervised clustering with constraints

Semi-supervised clustering, which uses class labels or pairwise constraints on some examples to aid unsupervised clustering, has been the focus of several recent projects [37]. Existing methods for semi-supervised clustering fall into two general approaches: constraint-based and distance-based methods. At present, many scholars incorporated pairwise constraints into state-of-art clustering algorithms. Kiri Wagstaff et al. [38] incorporated pairwise constraints into kmeans algorithm so as to satisfy these constraints in the process of clustering; Sugato Basu et al. [37] proposed the PCK-Means algorithm which modifies the objective function of clustering so that these constraints can be satisfied in some degree, however it must rely on parameters and a large number of constraints; Nizar Grira et al. [39] proposed the PCCA algorithm which was used to image database categorization; Davidson et al. [40] enhanced the hierarchical clustering with pairwise constraints, and presented intractability results for some constraint combinations [41]; Wei Tang et al. [42] proposed a feature projection method with pairwise constraints ,which can handle the high-dimension sparse data effectively. [17]

### MDBSCAN Related Definitions:

**Definition 2.1:**( Must-link constraints [38]) Must-link set M: if  $(x_1, x_2)$  belongs to M, then point  $x_1$  and  $x_2$  have to be in the same cluster.

**Definition 2.2:** (k-th nearest neighbor distance) for a point  $p$  belongs to D, we call the distance between  $p$  and the k-th nearest neighbor of  $p$  the k-th nearest neighbor distance of  $p$ , denoted by P-Kdistance.

**Definition 2.3:** (k nearest neighbor list) for a point  $p$  belongs to D, a set of k nearest neighbors is called k nearest neighbor list of  $p$ , denoted by P-Kneighbor.

MDBSCAN is a new method incorporates pairwise constraints (must-link) in order to calculate parameters effectively and automatically which was used to deal with multi-density data sets. It makes use of some must-link constraints to calculate some parameters Eps in different density distributions; in latter step, it selects the best parameter Eps that reflects the current density distribution effectively for each density distribution by using a certain outlier detection algorithm; finally, MDBSCAN works on the multi-density data set with the calculated Eps.

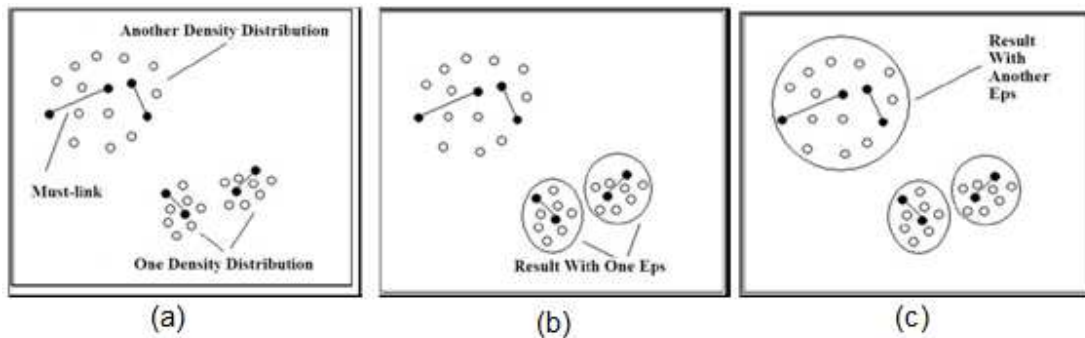


Figure 2.3 Framework of MDBSCAN algorithm.

In Figure 2.3(a), there are two density distributions in the dataset and there are four must-link constraints denoted by two black solid objects with a line. As we know, traditional DBSCAN algorithm does not perform well on the data set in Figure 2.3(a) with any value of parameter Eps. We can obtain four parameters Eps by using four must-link constraints, and then we must select two parameters Eps that reflect the density distribution. Figure 2.3(b) shows the result of DBSCAN with one parameter Eps; Figure 2.3(c) shows the result of DBSCAN with another parameter Eps. As we know, the k-th nearest neighbor distance of a point can approximately reflect the density distribution of area that the point is included. According to concept of must-

link, each must-link constraint  $(p, q)$ ,  $p$  and  $q$  must be in the same cluster; in other words, they are in the same density area, so we can consider that the  $k$ -th nearest neighbor distance of point  $p$  and point  $q$  are nearly the same. What's more, if  $p$  and  $q$  is in the same cluster,  $p$  and  $q$  should be density-connected with respect to certain parameter  $Eps$  and  $MinPts$ . In order to find the density distribution radius  $Eps$  which can let  $p$  density-connected to  $q$ , we check the  $k$ -th nearest neighbor distance of point  $p$  and point  $q$ . MDBSCAN make use of the must-link constraints to calculate the radius  $Eps$  of areas whose densities are different [17].

### **MDBSCAN Clustering Algorithm Problems:**

MDBSCAN is a very time-consuming clustering algorithm. It does not do well on the large datasets, and sometimes it gives the results after a very long time.

## **2.5 GMDBSCAN Clustering Algorithm**

Due to DBSCAN algorithm cannot choose parameter according to distributing of dataset. It simply uses a global  $MinPts$  parameter, so that the clustering result of multi-density database is inaccurate. In addition, when it is used to cluster large databases, it will cost too much time. For these problems, GMDBSCAN algorithm [45], based on spatial index and grid technique, is proposed.

### **The Process of GMDBSCAN Clustering Algorithm:**

The process of GMDBSCAN clustering algorithm is consist of six steps as follows:

1. Datasets input and Data standardization.
2. Dividing the data space into grids.
3. Statistics the grid density and Construct SP-Tree.
4. Bitmap forming.
5. Local-clustering and merging the similar sub-clusters.
6. Noises and Border processing.

Here is an illustration of the steps in more depth:

## Partitioned into grid

Partitioning is dividing the data space into grids. The neighborhood of a point is simulated by a grid, so the number of points in a grid and in the neighborhood is similar.

## Construct SP-Tree

For the non-empty grids, the processes to build SP-Tree index are as follows:

Each grid takes  $gNum$ (the number of points in the grid) as a keyword. Then start from the first dimension to look for the node in the corresponding layer of SP-Tree. If the corresponding number exists on SP-Tree, then it go to the next dimension. Repeat this until  $d+1$  dimension. After that, it creates a new leaf node to store grid. If corresponding number of the grid does not exist on SP-Tree, then it creates nodes from this layer [45].

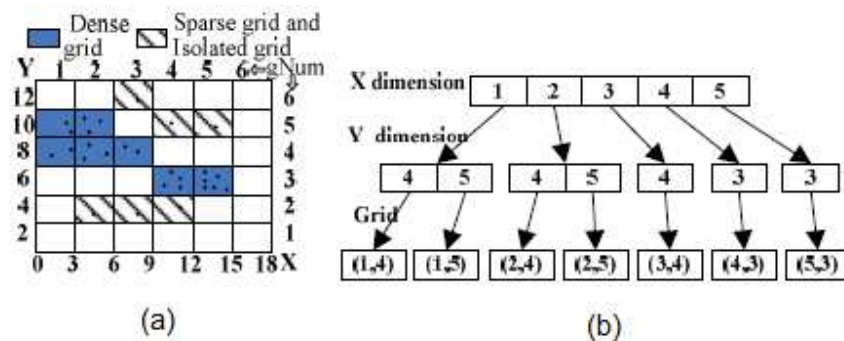


Figure 2.4 Framework of MDBSCAN algorithm.

In Figure 2.4(a), the entire space is divided into 36 grids. It only has seven dense grids. It creates the SP-Tree only on these grids in Figure 2.4(b).

## Bitmap Forming

In DBSCAN algorithm, we should calculate the distance between the data and the other and judge whether it is less than  $Eps$  repeatedly. So we do a preprocessing to calculate if the distance of each two data is less than or equal to  $Eps$ , and store the information in the bitmap. If the distance is less than or equal to  $Eps$ , it means the data are in each other's neighborhood. In fact, we only calculate the distance of two data which exists in the same or adjacent grids [45].

### Selecting Parameters of MinPts and Eps

Taking GD as *local*-MinPts approximation of the region where grid is in. The grid volume  $V_{Grid}$  is not equal to the Data points Neighborhood volume  $V_{Eps}$ , so it is a factor to correct it,  $factor = V_{Eps} / V_{Grid}$ . The relationship of GD and MinPts is:  $factor = MinPts / GD = V_{Eps} / V_{Grid}$ .  $Eps$  equals to the half length of the grid. The dataset has volume  $n$ , if each grid contains  $k$  ( $5 < k < 8$ ) data in average, then each dimension is divided to  $\lfloor \sqrt[k]{n/k} \rfloor$  parts. The length of each dimension is 1 and the length of side of grid is  $1 / \lfloor \sqrt[k]{n/k} \rfloor$  and the value of Eps is  $Eps = 2 / \lfloor \sqrt[k]{n/k} \rfloor$  [45].

### Clustering

GMDBSCAN mainly gets the idea of locally clustering, identifying a *local*-MinPts for each grid. For each grid, processing clustering with their *local*-MinPts to form a number of distributed local clusters. For the data density within the same cluster should be similar, if two sub-cluster which have same points and the similar density, can be merged to a single cluster. The algorithm sets a variable *similar* as the threshold of density similarity among clusters [45].

### The pseudo-code of GMDBSCAN is given below:

1. If each grid have been clustered  
    Then deal with boundary;
2. Output cluster, noises, outlier;
3. Else  
    Select grid whose Grid-Density is max and have not been clustered;  
    Compute  $Minpts = \lfloor factor * GD \rfloor$   
    For each data in grid  
    Cluster with DBSCAN algorithm;  
    If data is belong to other sub-cluster Then  
        If  $gds \geq similar$   
            Then merge sub-cluster;  
        Else assign it to the sub-cluster whose central point is most nearest from  
        this point;  
        End;  
    Else tag the data as a new cluster; End;
4. End;

### **Noises and Border Processing**

Noises distribution is not very sparse, but its amount is too small to form a cluster. So, GMDBSCAN algorithm sets a parameter according to the size of dataset. When the amount of data in a cluster is less than it, the entire cluster will be treat as noise [45].

### **GMDBSCAN Clustering Algorithm Problems:**

GMDBSCAN is a time consuming to perform well on large datasets, and sometimes it gives the output after a long time.

## **2.6 Cure Clustering Algorithm**

CURE is a hierarchical clustering algorithm that adopts a middle ground between the centroid-based and the all-point extremes [8]. CURE algorithm is more robust to outliers, and identifies clusters having non-spherical shapes and wide variances in size. It achieves this by representing each cluster by a certain fixed number of points that are generated by selecting well scattered points from the cluster, the scattered points capture the shape and extent of the cluster. And then shrinking them toward the center of the cluster by a specified fraction. Having more than one representative point per cluster allows CURE to adjust well to the geometry of non-spherical shapes and the shrinking helps to dampen the effects of outliers. The clusters with the closest pair of representative points are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. The scattered points approach employed by CURE alleviates the shortcomings of both the all-points as well as the centroid-based approaches. It enables CURE to correctly identify the clusters in Figure 2.5(a) - the resulting clusters due to the centroid-based and all-points approaches is as shown in Figures 2.5(b) and 2.5(c), respectively. CURE is less sensitive to outliers since shrinking the scattered points toward the mean dampens the adverse effects due to outliers are typically further away from the mean and are thus shifted a larger distance due to the shrinking. Multiple scattered points also enable CURE to discover non-spherical clusters like the elongated clusters shown in Figure 2.5(a). For the centroid-based algorithm, the space that constitutes the vicinity of the single centroid for a cluster is spherical. Thus, it favors spherical clusters and as shown in Figure 2.5(b), splits the elongated clusters. On the other hand, with multiple scattered points as representatives of a cluster, the space that forms



the vicinity of the cluster can be non-spherical, and this enables CURE to correctly identify the clusters in Figure 2.5(a).

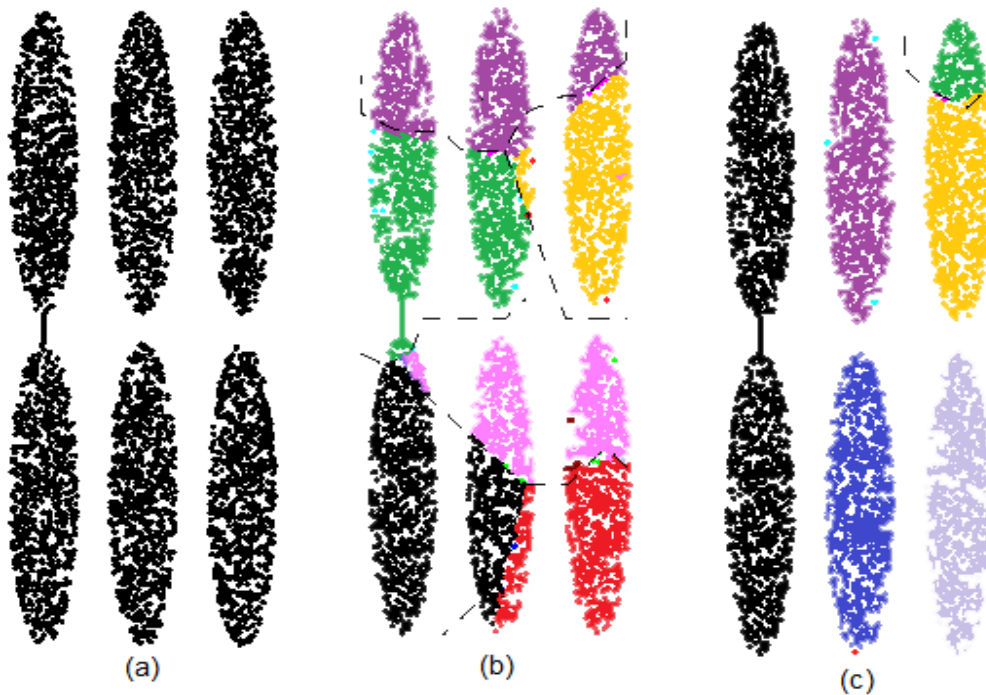


Figure 2.5 Clusters generated by hierarchical algorithms.

To handle large databases, CURE employs a combination of random sampling and partitioning. A random sample drawn from the data set is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters. The quality of clusters produced by CURE is much better than those found by existing algorithms. Furthermore, they demonstrate that random sampling and partitioning enable CURE to not only outperform existing algorithms but also to scale well for large databases without sacrificing clustering quality [8].

### **Random Sampling and Partitioning:**

CURE's approach to the clustering problem for large data sets is as follows. First, instead of pre-clustering with all the data points, CURE begins by drawing a random sample from the database. Random samples of moderate sizes preserve information about the geometry of clusters fairly

accurately, thus enabling CURE to correctly cluster the input. In particular, assuming that each cluster has a certain minimum size, CURE uses chernoff bounds to calculate the minimum sample size for which the sample contains, with high probability, at least a fraction  $f$  of every cluster. Second, in order to further speed up clustering, CURE first partitions the random sample and partially clusters the data points in each partition. After eliminating outliers, the pre-clustered data in each partition is then clustered in a final pass to generate the final clusters [8].

### **Labeling Data on Disk:**

Since the input to CURE's clustering algorithm is a set of randomly sampled points from the original data set, the final  $k$  clusters involve only a subset of the entire set of points. In CURE, the algorithm for assigning the appropriate cluster labels to the remaining data points employs a fraction of randomly selected representative points for each of the final  $k$  clusters. Each data point is assigned to the cluster containing the representative point closest to it. Note that approximating every cluster with multiple points instead a single centroid as is done in [46], enables CURE to, in the final phase, correctly distribute the data points when clusters are non-spherical or non-uniform. The final labeling phase, since it employs only the centroids of the clusters for partitioning the remaining points, has a tendency to split clusters when they have non-spherical shapes or non-uniform sizes (since the space defined by a single centroid is a sphere) [8].

## **2.7 Grid- Based Clustering Algorithms**

The grid-based clustering approach uses a multiresolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed.

In general, a grid-based clustering algorithm consists of the following five basic steps:

- Partitioning the data space into a finite number of cells (or creating grid structure).
- Estimating the cell density for each cell,
- Sorting the cells according to their densities,
- Identifying cluster centers,
- Traversal of neighbor cells.

The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space. It significantly reduces the computational complexity. Some typical examples of the grid-based approach include STING, which explores statistical information stored in the grid cells; WaveCluster, which clusters objects using a wavelet transform method; and CLIQUE, which represents a grid-and density-based approach for clustering in high-dimensional data space. OptiGrid, GRIDCLUS, GDILC, WaveCluster are also examples of grid-based clustering [47].

# Chapter 3

## Techniques and Background

This research aims to focus on a number of popular clustering algorithms that are density-based and to group them according to some specific baseline methodologies. This chapter is mainly concerned with presenting our adopted idea in explaining the integration of ideas in a number of previous algorithms which suffer from some problems, and we try to solve these problem by integrated the grid-based in addition to using representative points in our new proposed GMDBSCAN-UR algorithm as we will see in the next chapter. Here is more illustration of some techniques related to our work and ideas in this research.

### 3.1 Density-Reachability and Density Connectivity in DBSCAN Algorithm

Consider Figure 3.1 for a given  $\epsilon$  represented by the radius of the circles, and, say, let  $MinPts = 3$ . Based on the definitions was listed in the DBSCAN analysis in section 2.1:

1. Of the labeled points,  $p$ ,  $o$ , and  $r$  are core objects because each is in an  $\epsilon$ -neighborhood containing at least three points.
2.  $q$  is directly density-reachable from  $m$ .  $m$  is directly density-reachable from  $p$  and vice versa.
3.  $q$  is (indirectly) density-reachable from  $p$  because  $q$  is directly density-reachable from  $m$  and  $m$  is directly density-reachable from  $p$ . However,  $p$  is not density-reachable from  $q$  because  $q$  is not a core object. Similarly,  $r$  and  $s$  are density-reachable from  $o$ , and  $o$  is density-reachable from  $r$ .
4.  $o$ ,  $r$ , and  $s$  are all density-connected.

A density-based cluster is a set of density-connected objects that is maximal with respect to density-reachability. Every object not contained in any cluster is considered to be *noise*.

*"How does DBSCAN find clusters?"* DBSCAN searches for clusters by checking the  $\epsilon$ -neighborhood of each point in the database. If the  $\epsilon$ -neighborhood of a point  $p$  contains more than  $MinPts$ , a new cluster with  $p$  as a core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few

density-reachable clusters. The process terminates when no new point can be added to any cluster.

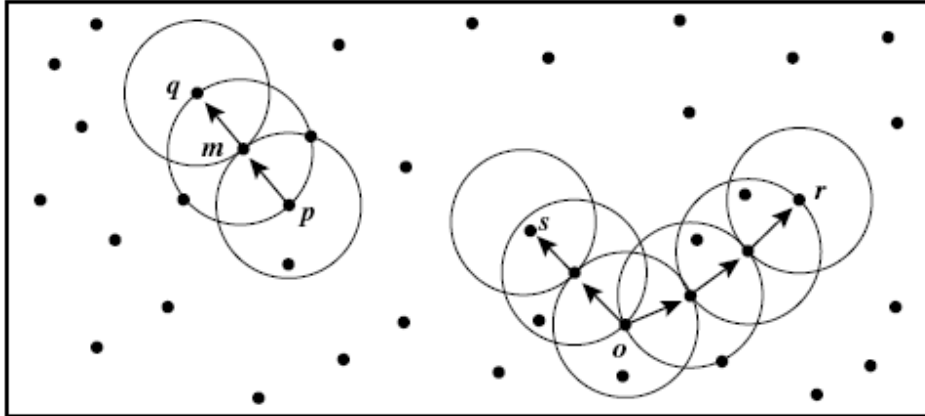


Figure 3.1: Density reachability and density connectivity in density-based clustering.

### 3.2 Data Types in Clustering Analysis

The type of data is directly associated with data clustering, and it is a major factor to consider in choosing an appropriate clustering algorithm. The attribute can be Binary, Categorical, Ordinal, Interval-scaled or Ratio-scaled [1].

**Binary:** Have only two states: 0 or 1, where 0 means that the variable is absent and 1 means that it is present.

**Categorical:** also referred to as nominal, are simply used as names, such as the brands of cars and names of bank branches. That is, a categorical attribute is a generalization of the binary variable; it can take on more than two states.

**Ordinal:** resembles a categorical variable, except that the M states of the ordinal value are ordered in a meaningful sequence. For example, professional ranks are often enumerated in a sequential order.

**Interval-scaled:** are continuous measurements of a linear scale such as weight, height and weather temperature.

**Ratio-scaled:** make a positive measurement on a nonlinear scale. For example an exponential scale and the volume of scales over time are ratio-scaled attributes.

There are many other data types, such as image data, though we believe that once readers get familiar with these basic types of data, they should be able to adjust the algorithms accordingly.

### 3.3 Similarity and Dissimilarity

Similarity and Dissimilarity (Distances) play an important role in cluster analysis. Similarity measures, similarity coefficients, dissimilarity measures, or distances are used to describe quantitatively the similarity or dissimilarity of two data points or two clusters, that how similar two data points are or how similar two clusters are: the greater the similarity coefficient, the more similar are the two data points. Dissimilarity measure and distance are the other way around: the greater the dissimilarity measure or distance, the more dissimilar are the two data points or the two clusters. Consider the two data points  $x$  and  $y$  example. The Euclidean distance between  $x$  and  $y$  is calculated as

$$d(x, y) = \left( \sum_{j=1}^d (x_j - y_j)^2 \right)^{1/2} \quad 2.1$$

The lower the distance between  $x$  and  $y$ , the more probability that  $x$  and  $y$  fall in the same cluster.

Every clustering algorithm is based on the index of similarity or dissimilarity between data points [46].

### 3.4 Scale Conversion

In many applications, the variables describing the objects to be clustered will not be measured in the same scales. They may often be variables of completely different types, some interval, others categorical. Scale conversion is concerned with the transformation between different types of variables. There are three approaches to cluster objects described by variables of different types. One is to use a similarity coefficient, which can incorporate information from different types of variable. The second is to carry out separate analyses of the same set of objects, each analysis involving variables of a single type only, and then to synthesize the results from different analyses. The third is to convert the variables of different types to variables of the same type, such as converting all variables describing the objects to be clustered into categorical variables [48]. Any scale can be converted to any other scale. Several cases of scale conversion are

described by Anderberg (1973) [48], including interval to ordinal, interval to nominal, ordinal to nominal, nominal to ordinal, ordinal to interval, nominal to interval, dichotomization (binarization) and so on.

### 3.5 GMDBSCAN-UR Related Definitions

**Definition 3.1:** *Cell:* is the smallest unit in the SP-Tree.

**Definition 3.2:** *SP-Tree* [49]: the structure of SP-Tree is generated by partition P of dataset D as follows:

1. SP-Tree has a root cell, and is consists of d+1 layer, in which d is the dimension of dataset;
2. Each dimension of dataset has its corresponding layer in SP-Tree and d+1 dimension is corresponding to all non-empty cells;
3. Except layer d+1, there are some internal cells whose forms are (gNum,nextLay) in layer i. gNum is the interval ID of a cell at the dimension i. nextLay is a pointer which points to a leaf cell in layer d . In other layers, next Lay points to the next layer cell which contains the IDs of all dissimilar non-empty cells of next dimension corresponding to the current cell;
4. The path from root cell to leaf cell is corresponding to a cell.

**Definition 3.3:** *Cell-Density:* The Cell-Density is denoted by CD, and defined as follow:

$$CD = \frac{\text{Amount of samples in the cell}}{\text{Volum of the cell}} \quad 4.8$$

**Definition 3.4:** *Dense Cell:* the cell that contains more than or equal to the threshold numbers of data points.

**Definition 3.5:** *Cell-Neighboring:* it is defined that c1 is the node neighborhood of c2, only if there is a point between these two cells.

**Definition 3.6:** *VCell:* The cell volume.

**Definition 3.7:** *VEps:* The Data point's Neighborhood volume.

# Chapter 4

## Proposed Algorithm "GMDBSCAN-UR"

The purpose of this research is to discover clusters with arbitrary shape, to regard clusters as dense regions of objects in the data space that are separated by regions of low density representing noise. In addition, the study is interested in algorithms that take into account the density to cluster the various real and artificial datasets. Our work in this research performs the density-based clustering in many stages as you can see in the following sections.

### 4.1 Our Adopted Idea

What exactly happens in practice is as follows: to begin with, the first stage is to input a multi-density dataset. We want to reduce the time complexity. In order to achieve this purpose, we propose a grid-based cluster technique, [22, 23] which divides the data space into cells. A number of well scattered points in each cell in the grid are chosen. These scattered points capture the shape and extent of the dataset as all. These scattered points after shrinking are used as representatives of its cell. The chosen scattered points are next shrunk towards the centroid of the cell by a fraction  $\alpha$ . The cells with the closest pair of representative points are the cells that are merged at each step of our work. The scattered points approach employed by our work alleviates the shortcomings of both the all-points as well as the centroid-based approaches [49]. Thus, our work in this research adopts a middle ground between the centroid-based and the all-point extremes. Next we treat all data in the same cell as an object, and all the operations of clustering are done on the this cell. This research deals with two approaches of making clustering data set with multi-densities; the two choices give the same result with flexible options. The first option is dealing with a specific cell with its local density so that; it is possible to vary the parameter  $Eps$  from cell to cell in the dataset and make the parameter  $MinPts$  to be constant. The second option is to make the parameter  $Eps$  constant over all cells and vary the parameter  $MinPts$  from cell to cell. Parameters choice depends on the local cell density. Next, make local clustering in each cell and merge between the resulted clusters. The next step is label the points, not chosen in the representative points , to the resulted clusters. After that, the post



processing is become needed in such a way to accurate the results. Finally and the necessary step in all clustering algorithms is to eliminate noise and outliers.

## **4.2 G MDBSCAN-UR Steps**

Our proposed clustering algorithm, G MDBSCAN-UR, consists of eight steps, these are as follows:

1. Datasets input and Data standardization.
2. Dividing dataset into smaller cells.
3. Chosen representative points in each cell.
4. Selecting Parameters of MinPts and Eps.
5. Bitmap Forming.
6. Local-Clustering and Merging the Similar Sub-clusters using DBSCAN algorithm.
7. Labeling and Post Processing.
8. Noise Elimination

Here we explain our new multi-density clustering algorithm based on grid and using representative points, G MDBSCAN-UR. Here we will illustrate the algorithm implementation steps in more details:

### **4.2.1 Datasets Input and Data Standardization**

Data standardization [51] makes data dimensionless. It is useful for defining standard indices. After standardization, all knowledge of the location and scale of the original data may be lost. It is necessary to standardize variables in cases where the dissimilarity measure, such as the Euclidean distance, is sensitive to the differences in the magnitudes or scales of the input variables (Milligan and Cooper, 1988)[52]. The approaches of standardization of variables are essentially of two types: global standardization and within-cluster standardization. Global standardization standardizes the variables across all elements in the data set. Within-cluster standardization refers to the standardization that occurs within clusters on each variable. Some forms of standardization can be used in global standardization and within-cluster standardization as well, but some forms of standardization can be used in global standardization only [47]. The

z-score is a well known form of standardization used for transforming normal variants to standard score form. Given a set of raw data D, the z-score standardization formula is defined as

$$x_{ij} = z_1(x_{ij}^*) = \frac{x_{ij}^* - \bar{x}_{ij}^*}{\sigma_j^*} \quad 4.1$$

Where  $\bar{x}_{ij}^*$  and  $\sigma_j^*$  are the sample mean and standard deviation of the jth attribute, respectively. The transformed variable will have a mean of 0 and a variance of 1. The location and scale information of the original variable has been lost. This transformation is also presented in (Jain and Dubes, 1988, p. 24) [53].

So, the first step in our proposed algorithm, G MDBSCAN-UR, is to input the dataset which is in a multi-density format like for example real dataset "adult" and artificial data set "chameleon", we name it like that because it has been used to evaluate chameleon algorithm [51]. And then make the data standardization step. Figure 4.1 below is a multi-density dataset, it has clusters with different densities.

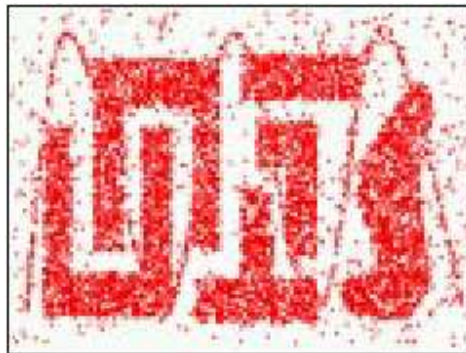


Figure 4.1: Multi-density dataset.

#### 4.2.2 Dividing Dataset into Smaller Cells

Partitioning divides the data space into smaller cells. So the cells numbers of points are not equal. Figure 4.2 show a multi-density dataset which is divided to cells as in Figure 4.3. Figure 4.3 shows that the top most left cell's number of points is not equal to top most right cell's number of points. So the second step is dividing the data space into cells in order to make local clustering in each cell. The number of cells per dimension is calculated in *SPTree.construct()* method. The SP-Tree is created only on these dense cells where  $CD \geq I$ .

where:  $I$  denotes the density threshold; it is an integer value and,  $CD$  is the cell density. Then each point is then assigned to a cell by the  $SPTree.insert()$  method. Cells i.e. the  $SPTree.leaves$ , have a collection of points.

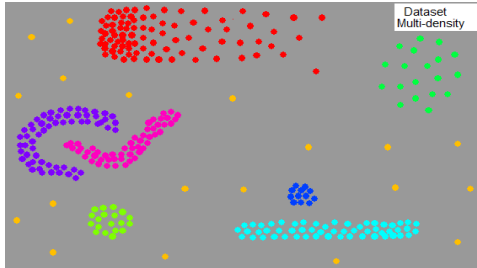


Figure 4.2: Multi-density Dataset.

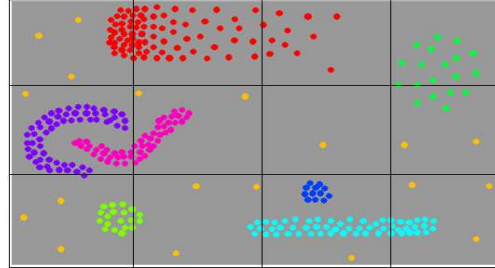


Figure 4.3: Dividing the dataset to smaller cells.

### 4.2.3 Chosen Representative Points

A number of well scattered points in each cell are chosen. The scattered points in the cell must capture the shape and extent of that cell as shown in Figure 4.4. The black points in Figure 4.4 below are the representative points, it is clear that the number of representative points is smaller than the number of points in the cell. And these representative points are well scattered over the original dataset points. This step leads to significant improvements in execution times in our new proposed G MDBSCAN-UR algorithm.

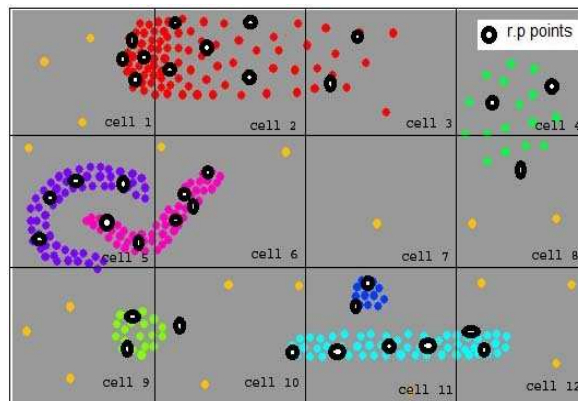


Figure 4.4: Taking a well scattered representative points in each cell.

So we visit all cells (leaves) in the tree and choose a percentage number of points, say half, to be the representative points in the cell. All the representative points in all cells are the representative points of the dataset. We put all representative points in a dataset\_Rep. At the same time we put the not chosen points from every cell and put all these points in another data set named,

dataset\_Remainder to be use it the labeling step later. Figure 4.5 below shows some dataset along with its representative points. It is clear that the chosen representative points Figure 4.5 (b) are actually represents the original dataset Figure 4.5 (a). Good representing the original dataset is very important issue in getting good final clustering results.

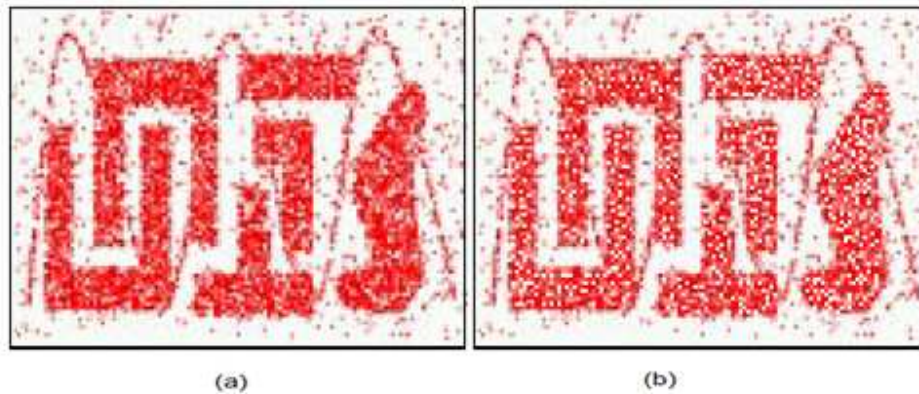


Figure 4.5: Dataset along with its representative points.

#### 4.2.4 Selecting MinPts and Eps Parameters

In each cell, one approach is used in selecting the MinPts and Eps. Either selects the MinPts for each cell individually and let the Eps to be constant for all cells or select the Eps for each cell individually and let the MinPts to be constant for all cells.

Firstly: when we use the same Eps with varying MinPts, then we have the following:

We apply the idea on the cells as shown below in Figure 4.6 using same MinPts in all cells to merge but in different Eps from cell to cell; i.e. the MinPts is 4 at all cells but, at the most left grid the Eps is the smallest because this cell is the most dense; at the middle cell the Eps is wider because this cell is less dense, at the right cell the Eps is the widest because it is the lowest density.

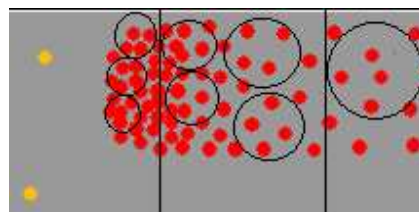


Figure 4.6: Using same MinPts with varying Eps.

Secondly: we apply the idea using the same *Eps* with varying *MinPts*, then we have the following: we apply the idea on two cells as shown below in Figure 4.7; using same *Eps* in all cells to merge but in different number of *MinPts* from cell to cell, i.e. at the left cell the *MinPts* is 4; the right choosing the cell *MinPts* to be 2 is enough.

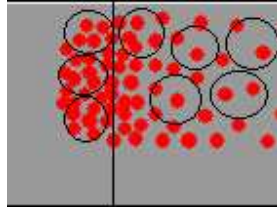


Figure 4.7: Using same *Eps* with varying *MinPts*.

#### 4.2.5 Bitmap Forming

Calculate the distance of two data which exists in the same or adjacent cells. Calculate the distances of each two data and compare with *Eps* then store the information in the bitmap. If the distance is less than or equal to *Eps*, it means the data are in each other's neighborhood [47].

Cell Density, denoted by *CD*, is defined as amount of data in a cell. Taking *CD* as local-*MinPts* approximation of the cell region. If the Cell Volume, denoted by, *VCell* is not equal to the data point's neighborhood volume, *VEps*, we set a factor to correct it,  $factor = VEps / VCell$ . The relationship of *CD* and *MinPts* is:

$$Factor = MinPts / CD = VEps / VCell \quad 4.2$$

$$MinPts = factor * CD \quad 4.3$$

This step make all necessary needed statistics which will be used in the next later steps.

#### 4.2.6 Local-Clustering and Merging the Similar Sub-clusters using DBSCAN Algorithm

In this step we apply the original DBSCAN method locally in each cell using the computed *MinPts* and *Eps* parameters. Our work in this study mainly gets the idea of locally clustering, identifying a local-*MinPts* for each cell in the dataset. For each cell, processing clustering with their local-*MinPts* to form a number of distributed local clusters.

The step is divided into consecutive steps to make local clustering and merging the similar sub clusters as you can see.

- 1- First is to select the cell whose density is maximum and has not been clustered. During that time we deal with boundary. Dense cells refer to those cells whose cell density,  $CD$ , is greater than or equal to some predefined threshold.
- 2- Then sparse cells which close to dense cells, but their cell densities are less than the pre-specified threshold. Data in sparse cell may be noise or border, it needs further study. Isolated cells refer to those whose cell density is less than threshold and not close to some dense cell. All data in isolated cells could be regarded as noise and isolated data. In DBSCAN, if the border object is in the scope of  $Eps$  neighborhood of different core objects, it is classified into the cluster to sort firstly. Here in our GMDBSCAN-UR algorithm, we set such object to the cluster whose core object is the nearest to this object. Second, we compute  $MinPts$  for each data in cell which its equation given by:

$$MinPts = factor * CD \quad 4.4$$

- 3- Then Cluster with original DBSCAN algorithm and for each unvisited point , $P$ , in dataset,  $D$ , mark  $P$  as visited and compute the neighbors of the point  $P$ , then compare this number with the  $MinPts$ . If neighbors are less than  $MinPts$  then label  $P$  as a noise, otherwise label it as a core point and so on. Then expand the current cluster.
- 4- If data belongs to another sub-cluster, then merge the two clusters, and if not, assign it to the cluster whose has the nearest representative point from this point and tag the data as a new cluster.

#### **4.2.7 Labeling and Post Processing**

Since the input to GMDBSCAN-UR's clustering algorithm is a set of well scattered chosen representative points from the original large dataset, the final  $k$  clusters involve only a subset of the entire set of points. In GMDBSCAN-UR, the algorithm for assigning the appropriate cluster labels to the remaining data points employs the selected representative points for each of the final  $k$  clusters. Each data point is assigned to the cluster containing the representative point closest to it. Note that approximating every cluster with multiple points instead a single centroid

enables GMDBSCAN-UR correctly distribute the data points when clusters are non-spherical or non-uniform. After that, we check the density for the resulted k cluster. If there are two clusters with nearly same density and very close to each others, they can be merged to a single cluster as a post processing step. The remerging step goes out with a very accurate final results. Thus, the remerging step applied on the resulted clusters is very necessary step [49].

This step, Labeling and Post Processing, labeling all points in the dataset\_Remainder data set, which are not entered in the clustering process. For every point in the dataset\_Remainder, we search for the nearest point from the dataset\_Rep dataset, which is clustered and becomes point belongs to some cluster. Once find the nearest point from the dataset\_Rep to the point from dataset\_ Remainder, we can label this non-clustered point to the cluster contains the nearest representative point. Now, to this end we have all points in the dataset become clustered.

#### **4.2.8 Noise Elimination**

Any data set almost always contains outliers. These do not belong to any of the clusters. That is, the neighborhoods of outliers are generally sparse compared to points in clusters, and the distance of an outlier to the nearest cluster is comparatively higher than the distances among points in points in the clusters themselves. Every clustering method needs mechanisms to eliminate outliers. In GMDBSCAN-UR, outliers due to their larger distances from other points, tend to merge with other points less and typically grow at a much slower rate than actual clusters. Thus the clusters which are growing very slowly are identified and eliminated as outliers. Also, since the number of points in a collection of outliers is typically much less than the number in a cluster and that outliers form very small clusters, we can easily identify such small groups and eliminate them. Consequently, the final step, the outlier elimination, is necessary step for good clustering.

### **4.3 GMDBSCAN-UR Algorithm Properties**

- Taking into account both the inter-connectivity as well as the closeness of the clusters
- Considering the internal characteristics of the clusters themselves.
- Operates successfully on data sets with various shapes.
- Dividing to Grid:

- Allows scaling to large datasets.
- Significantly, it reduces the computational complexity.
- Does not depend on user-supplied model.
- GMDBSCAN-UR clustering algorithm is very sensitive to two parameters, MinPts and Eps.
- GMDBSCAN-UR clustering algorithm divides its work into three separate steps as follows:

**First:** Make the main clustering after dividing the data space to cells and choose the representative point to enter the clustering process with the DBSCAN algorithm.

**Second:** After getting the clusters and for the result to be more accurate, we need to perform such as a post processing step to get more accurate results of the clusters, that the result we get may contain more small size clusters due to over clustering which is not a good choice in clustering. So, here in GMDBSCAN-UR we run a remerging method to get more accurate result. This remerging method is not a time consuming and we can run such a method comfortably without worrying about the time to increase in the clustering algorithm.

**Third:** Up to this point we have a set of points that do not enter the clustering process and do not belong to any of the resulting clusters. Here the role of the labeling step is to get every point that is not in the chosen representative set of points, i.e. not clustered points, and search for the nearest point to it from the representative points, then assign the non-clustered point to the cluster that the representative clustered point belongs to. To this end, we have all data points are clusters and each belongs to the true cluster.



## 4.4 GMDBSCAN-UR Algorithm Pseudo-Code

**Algorithm 4.1:** GMDBSCAN-UR algorithm.

**Purpose:** Cluster a multi-density complex dataset in an efficient way.

**Inputs:** Multivariate dataset.

**Outputs:** Clusters.

**Procedure:**

1. if each cell has been clustered  
    Then deal with boundary;  
    Output cluster, noises, outlier;
2. Else
3. While there is a cell in the data space not clustered  
    Select half of the cell points and put them in the dataset\_Rep  
  
    Put the other half of the cell points and put them in the dataset\_Remainder  
    and remove them from the corresponding cell  
  
    Select cell whose Node-Density is max and has not  
    been clustered;  
  
    Compute  $\text{MinPts} = \lfloor \text{factor} * \text{ND} \rfloor$   
    For each data in cell  
    Cluster with DBSCAN algorithm;  
    If data belong to other sub-cluster Then  
        If  $\text{gds} \geq \text{similar}$  Then merge sub-cluster;  
        Else assign it to the sub-cluster whose central point is  
        most nearest from this point;  
    End;  
    Else tag the data as a new cluster;  
    End;
4. End while
5. For each point in dataset\_Remainder  
    Search for the nearest point in all the resulted clusters and assign this point  
    to the cluster contains the nearest point.
6. End;
7. For each resulted cluster  
    Apply the remerge method
8. End
9. End;

# Chapter 5

## Experimental Results

In this chapter we will show a sufficient number of results with various types of datasets with various numbers of points also, and with these inputs we will compare between four algorithms namely: DBSCAN, MDBSCAN, G MDBSCAN and, our proposed algorithm, G MDBSCANUR for clustering 2 features (*age* versus *fnlwgt* (final weight) - subsets of UCI "adult" dataset). The following sections contain figures and tables which show the corresponding times and the number of clusters for each comparable algorithm.

### 5.1 Adult Dataset

This dataset comes from the UCI repository of machine learning databases [50]. Table 5.1 show the necessary information about adult dataset. It is a multivariate dataset as we want to show the effectiveness of our proposed G MDBSCAN-UR algorithm. Adult dataset has 48842 categorical and integer instances. It has 14 attributes. The following subsections are more specifications about the adult dataset. Figure 5.1 below shows adult dataset. In our experiments we choose two numerical, integers, attributes. So, the dataset contains three clusters each has a different density.

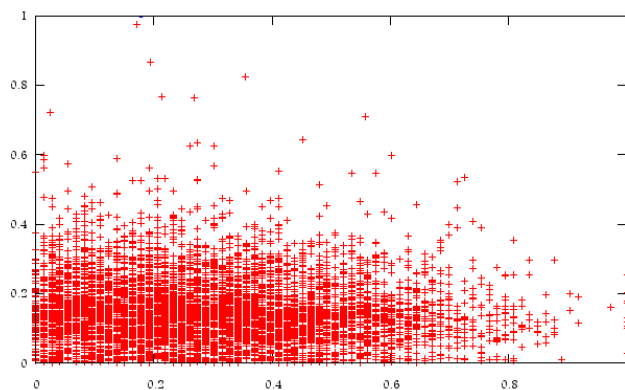


Figure 5.1: Adult dataset.

Table 5.1: Adult dataset specification.

<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	48842
<b>Attribute Characteristics:</b>	Categorical, Integer	<b>Number of Attributes:</b>	14

### Number of Instances

- 48842 instances, mix of continuous and discrete (train=32561, test=16281)
- 45222 if instances with unknown values are removed (train=30162, test=15060)

### Number of Attributes

6 continuous, 8 nominal attributes.

### Some of Attribute Information:

1. age: continuous.
2. fnlwgt: continuous.
3. sex: Female, Male.
4. hours-per-week: continuous.

## 5.2 Chameleon Dataset

Chameleon dataset is an artificial dataset, we use it to evaluate our proposed algorithm [54]. It has 8000 data points. We choose two attributes of double values. Chameleon dataset has a nested not simple shape dataset and it consists of six multi-densities clusters. Each cluster has arbitrary shape and many noises. Figure 5.2 below shows chameleon dataset.

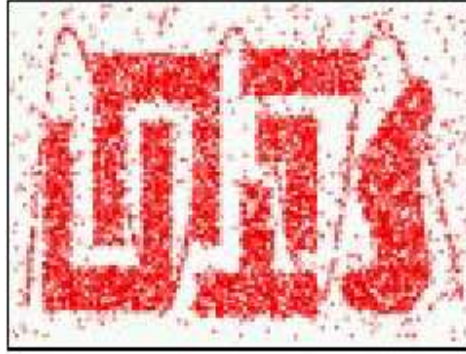


Figure 5.2: Chameleon dataset.

### 5.3: DBSCAN Results

In this section we are going to explain DBSCAN clustering algorithm results. Figure 5.3 and Figure 5.4 below show DBSCAN algorithm clustering results. Figure 5.3 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from adult dataset with DBSCAN clustering algorithm. And Figure 5.4 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from chameleon dataset with DBSCAN clustering algorithm. DBSCAN fails in clustering the multi-densities datasets like adult. It fails in clustering various numbers of points; that it cannot discover all clusters in the dataset as you can see in all Figure 5.3 parts below. DBSCAN can only discover one or two clusters and fail to discover the others, see Figure 5.3 (a)-(f). In clustering 250 points, it can discover only one cluster, while the others disappeared because DBSCAN was not able to appear them. May be the wrong result of disability to discover all clusters when clustering with part from the dataset is a normal result, due to the absent of the remainder of dataset points leads to another dataset with another characteristics and another density distribution. So, here we must talk only about clustering with all points, 8000 points, in the dataset. The clustering output when dealing with all data points of dataset has a meaning. Here, Figure 5.3 (f) is the result of clustering all the 8000 point from adult dataset with DBSCAN clustering algorithm. Although we use all the dataset's data points, DBSCAN only discover one cluster.

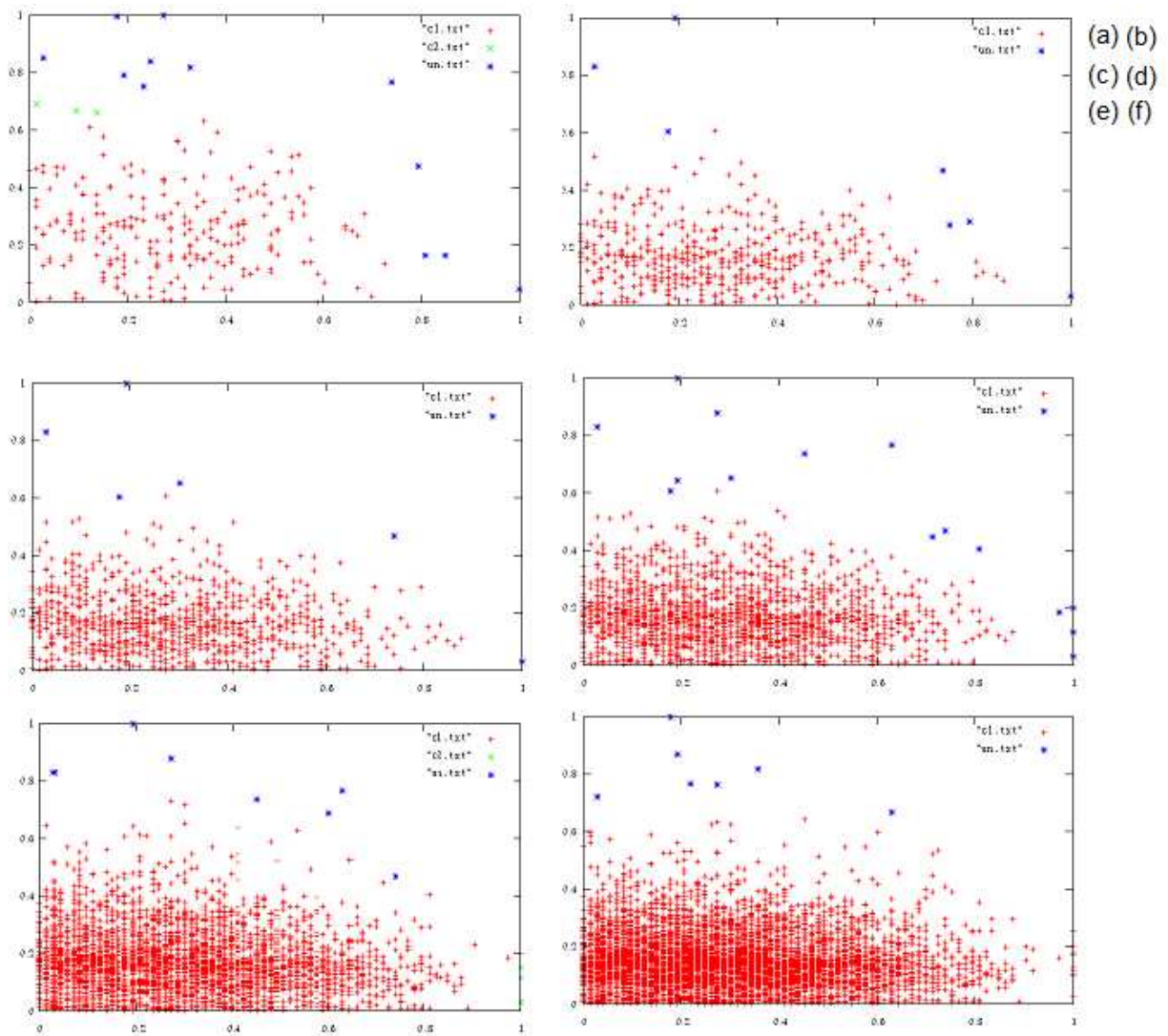


Figure 5.3: Adult dataset clustering results using DBSCAN algorithm.

Also DBSCAN algorithm gives the same results equalities when cluster with chameleon dataset. It fails to discover clusters in such a multi-density datasets like the artificial chameleon dataset Figure 5.2. Chameleon dataset consists of six different densities from each other. In our experiments DBSCAN can discover only one cluster as in Figure 5.4 (a)-(f). Figures 5.4 (a)-(f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points from chameleon dataset respectively. DBSCAN merges between different clusters whereas it is impossible to merge between them. The previous wrong merges between clusters result with a wrong final results. So, DBSCAN is a very bad clustering algorithm with a multi-density datasets.

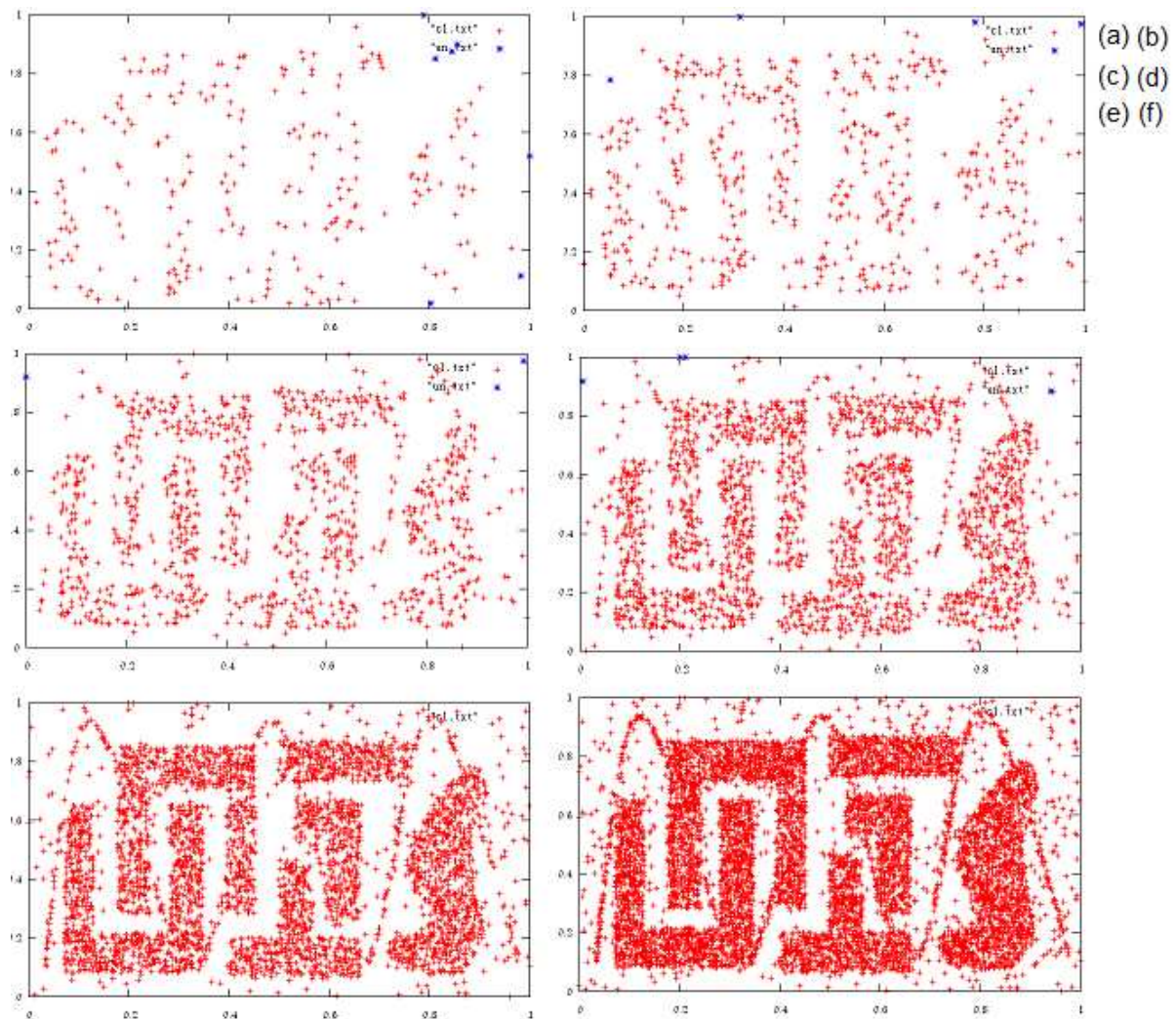


Figure 5.4: Chameleon dataset clustering results using DBSCAN algorithm.

Table 5.2 is the resulting times and equalities of clustering adult and chameleon datasets with DBSCAN clustering algorithm. The table's first column represents the number of points used to make the clustering process. These numbers are 250,500,1000,2000,4000 and 8000 points. The table's second and fourth columns are the multi-density datasets, adult and chameleon, clustering times in milliseconds. While the third and the fifth table's columns are the corresponding resulted numbers of clusters. We notice that the clustering time is directly promotional to the number of input points. DBSCAN algorithm takes 390855 milliseconds and 159115 milliseconds to cluster 8000 points from adult and chameleon datasets respectively. And also the quality is bad, it must be three clusters in adult dataset while it outputs only one cluster. And it must be six clusters in chameleon dataset while it outputs only one cluster. Thus, DBSCAN algorithm is not a good choice in both the quality of the resulting output clusters and in the execution time.

Table 5.2: Adult and chameleon datasets clustering results summary using DBSCAN algorithm.

DBSCAN				
No. Points	Time (ms) "adult"	No. of clusters	Time (ms) "chameleon"	No. of clusters
250	47	2	47	1
500	140	1	141	1
1000	796	1	453	1
2000	5522	1	2527	1
4000	41341	2	19282	1
8000	390855	1	159115	1

Thus clustering multi-densities datasets, like adult and chameleon datasets, by DBSCAN algorithm, we get bad quality results with long times as we see from the above table. Thus, DBSCAN clustering algorithm is a time consuming algorithm when dealing with large multi-densities datasets. This is due to Eps parameter value which is very important for DBSCAN algorithm, but it's calculation is a time-consuming. In other words, clustering algorithms is in need to discover a better version of DBSCAN algorithm to deal with these special multi-densities datasets.



## 5.4: MDBSCAN Results

Here in this section, we will offer the results of the second comparison algorithm. Figure 5.5 and Figure 5.6 below show MDBSCAN algorithm clustering results. Figure 5.3 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from adult dataset with MDBSCAN clustering algorithm. And Figure 5.6 (a), (b) and (c) are the results of clustering 250, 500 and 1000 points respectively from chameleon dataset with MDBSCAN clustering algorithm. MDBSCAN gives results better than DBSCAN results in clustering the multi-densities datasets like adult as you see in Figure 5.5 (a)-(f). MDBSCAN was not able to discover all clusters in the dataset correctly. In clustering 250 points, it can discover only two cluster, while in clustering 500 points, it can discover only one cluster. The most important point to talk about is the result of clustering the dataset at all, 8000 points. Figure 5.3 (f) is the result of clustering all the 8000 point from adult dataset with DBSCAN clustering algorithm. MDBSCAN algorithm makes more splits to output seven clusters instead of three clusters.



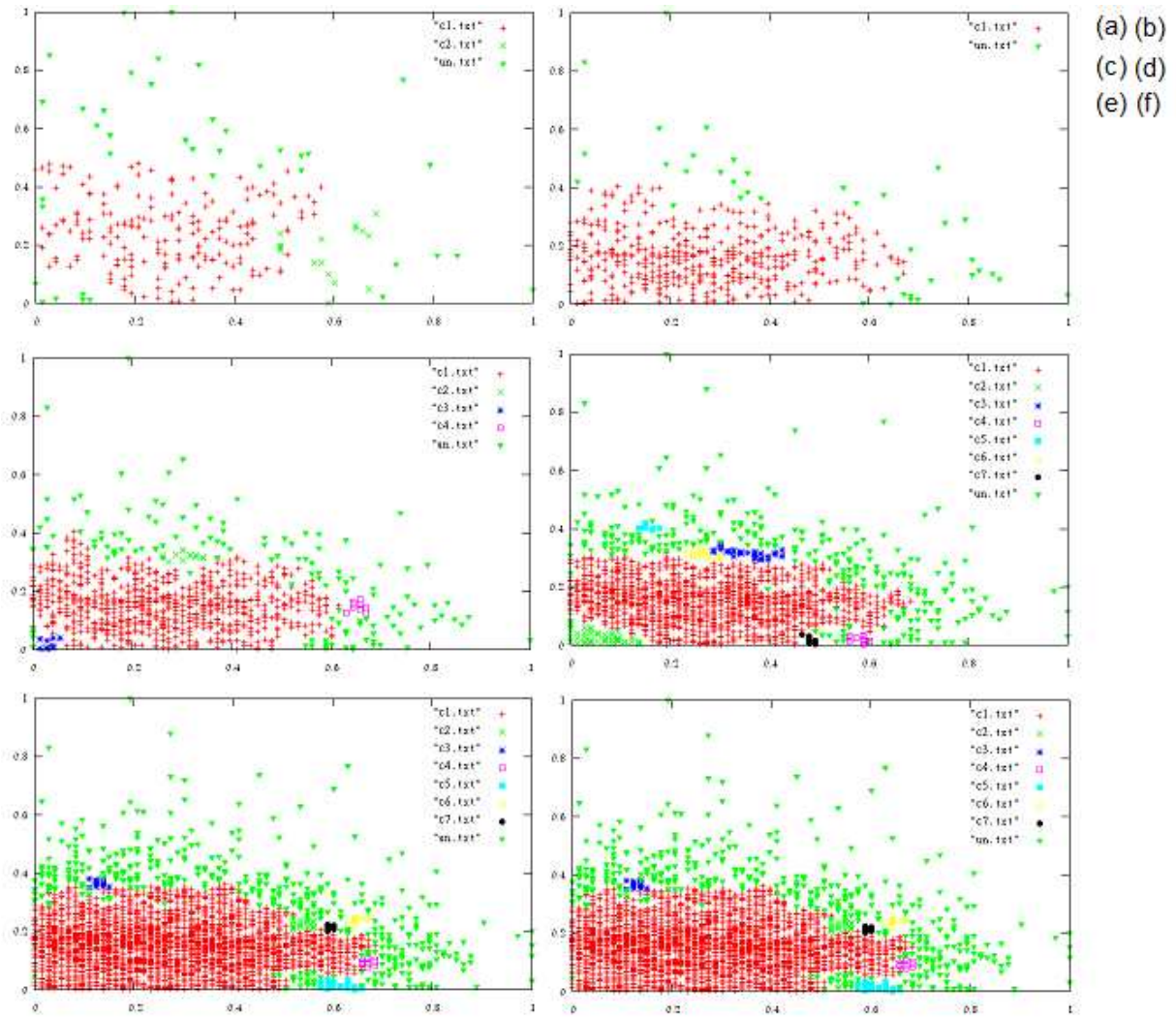


Figure 5.5: Adult dataset clustering results using MDBSCAN algorithm.

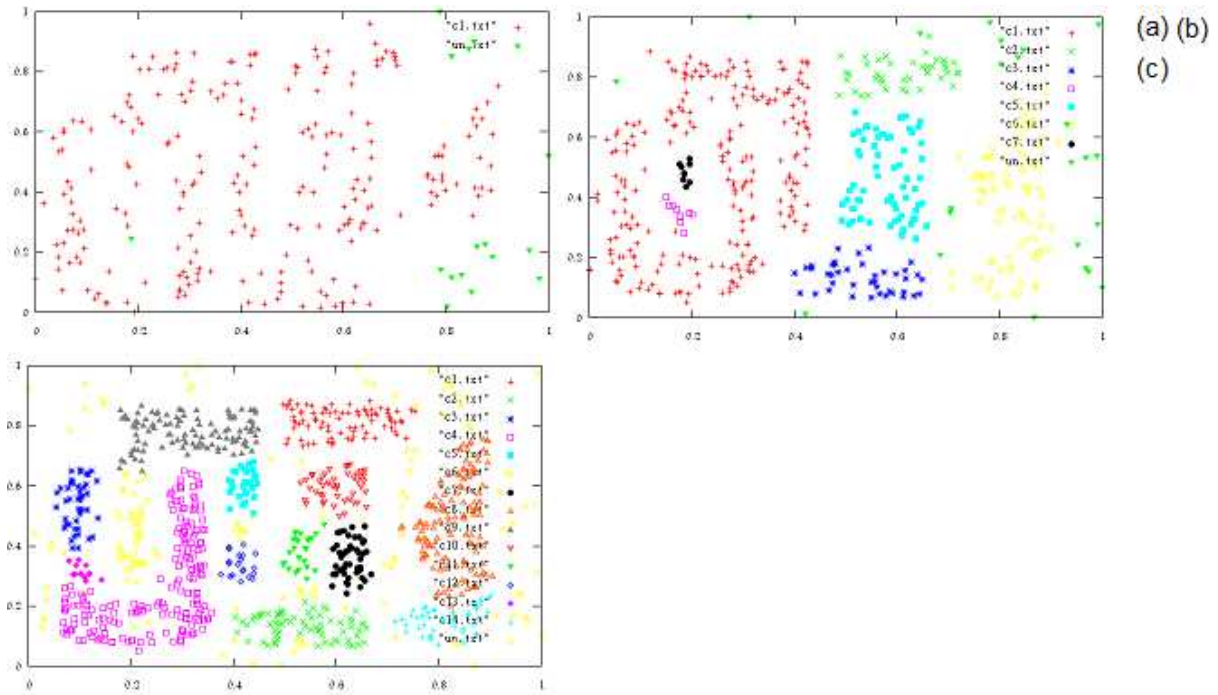


Figure 5.6: Chameleon dataset clustering results using MDBSCAN algorithm.

Clustering chameleon dataset with MDBSCAN algorithm gives also bad results. MDBSCAN algorithm fails in discovering clusters in such a multi-density datasets as in Figure 5.6. In our experiments, MDBSCAN makes wrong splits and wrong merges as in Figure 5.6 (a)-(c). Figures 5.4 (a)-(c) are the results of clustering 250, 500 and 1000 points from chameleon dataset respectively. MDBSCAN takes a very long time to cluster 1000 points from the dataset. It was enough to display only 250, 500 and 1000 points clustering results. Displaying more MDBSCAN results will not needed because it is clear that MDBSCAN clustering algorithm is a very bad one with large multi-density datasets. Clustering with MDBSCAN is a very time consuming process. For this reason, we stopped clustering more points beyond the 1000 points. So, MDBSCAN is a very time consuming clustering algorithm and not a good choice when clustering a large datasets.

Table 5.3 is the resulted times of clustering adult and chameleon datasets with MDBSCAN clustering algorithm. The table's first column represents the number of points used to make the clustering process. These numbers are 250,500,1000,2000,4000 and 8000 points. The table second and fourth columns are the multi-density datasets, adult and chameleon, clustering times in milliseconds. While the third and the fifth columns are the corresponding resulted numbers of

clusters. MDBSCAN algorithm is a very time consuming algorithm and does not work well with large datasets like adult and chameleon. MDBSCAN gives better quality results than DBSCAN clustering algorithm, but it is the worst in time compared with the all the other density-based clustering algorithms. So, MDBSCAN algorithm is a not a good choice in clustering large multi-density datasets due to its spending a very long time in Eps and MinPts parameters calculations. Thus, the researchers do their best efforts to discover an alternative multi-density clustering algorithms to face MDBSCAN algorithm problems.

Table 5.3: Adult and chameleon datasets clustering results summary using MDBSCAN algorithm.

MDBSCAN				
No. Points	Time (ms) "adult"	No. of clusters	Time (ms) "chameleon"	No. of clusters
250	499	2	1763	1
500	1623	1	64608	7
1000	5631	4	426040	14
2000	20891	7	very long time	-
4000	502486	7	very long time	-
8000	very long time	7	very long time	-

## 5.5: GMDBSCAN Results

This is an improved version of DBSCAN algorithm. It is a multi-density clustering algorithm. We can see that GMDBSCAN algorithm gives good results but it takes a bit more time which make us to search for a better one in clustering time. Figure 5.7 and Figure 5.8 below show GMDBSCAN algorithm clustering results. Figure 5.7 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from adult dataset with GMDBSCAN clustering algorithm. And Figure 5.8 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from chameleon dataset with GMDBSCAN clustering algorithm. GMDBSCAN algorithm results is better than MDBSCAN algorithm's results in both quality and time. We are going to talk in more specific about GMDBSCAN algorithm results. GMDBSCAN algorithm makes more not needed splits in adult dataset clustering which resulted in 18 clusters as in Figure 5.7 (f). In chameleon clustering,

GMDBSCAN gives a correct result as in Figure 5.8 (f), six clusters, but in a quite long time. So we still in need to more improvements in clustering process time.

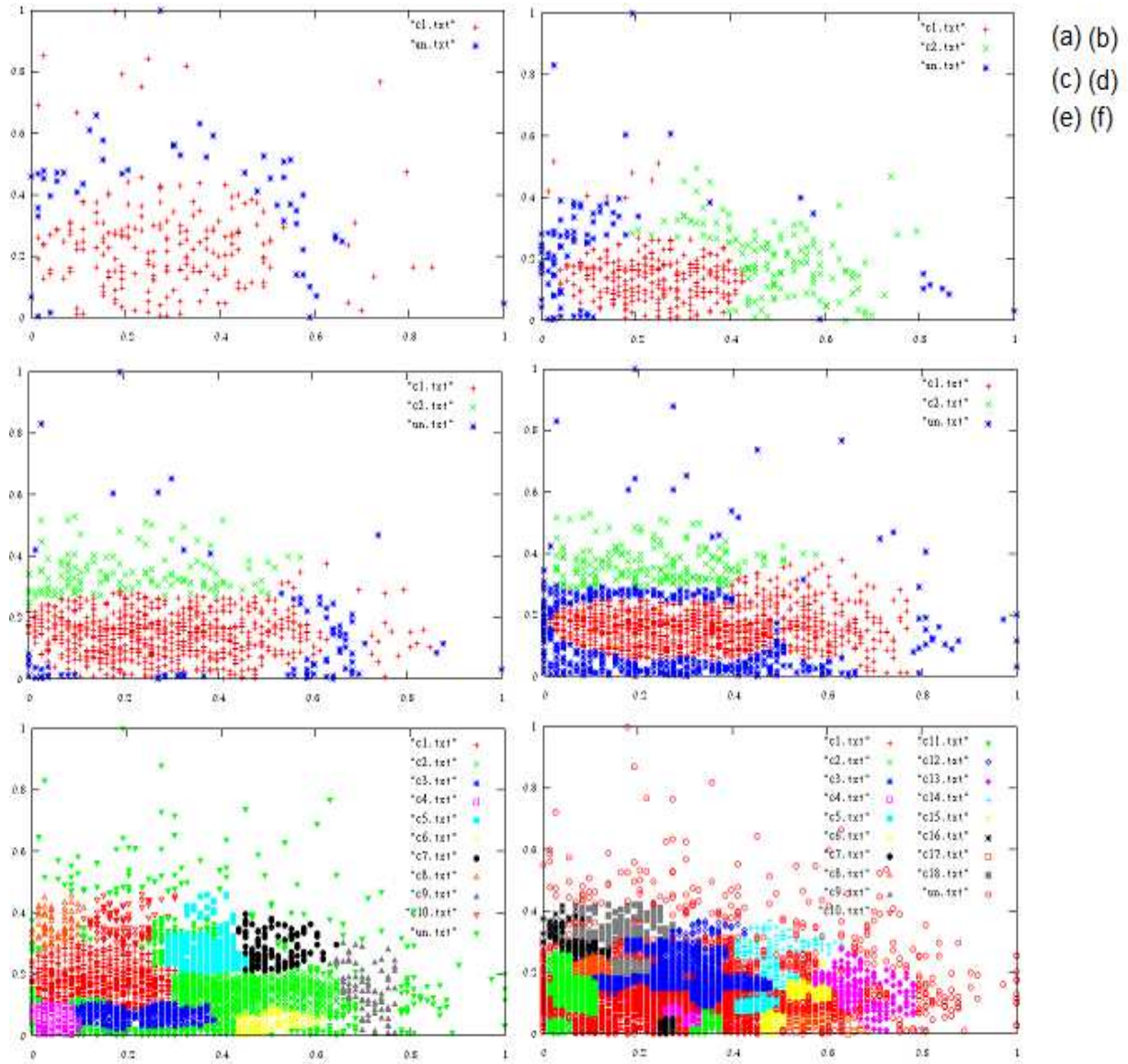


Figure 5.7: Adult dataset clustering results using G MDBSCAN algorithm.



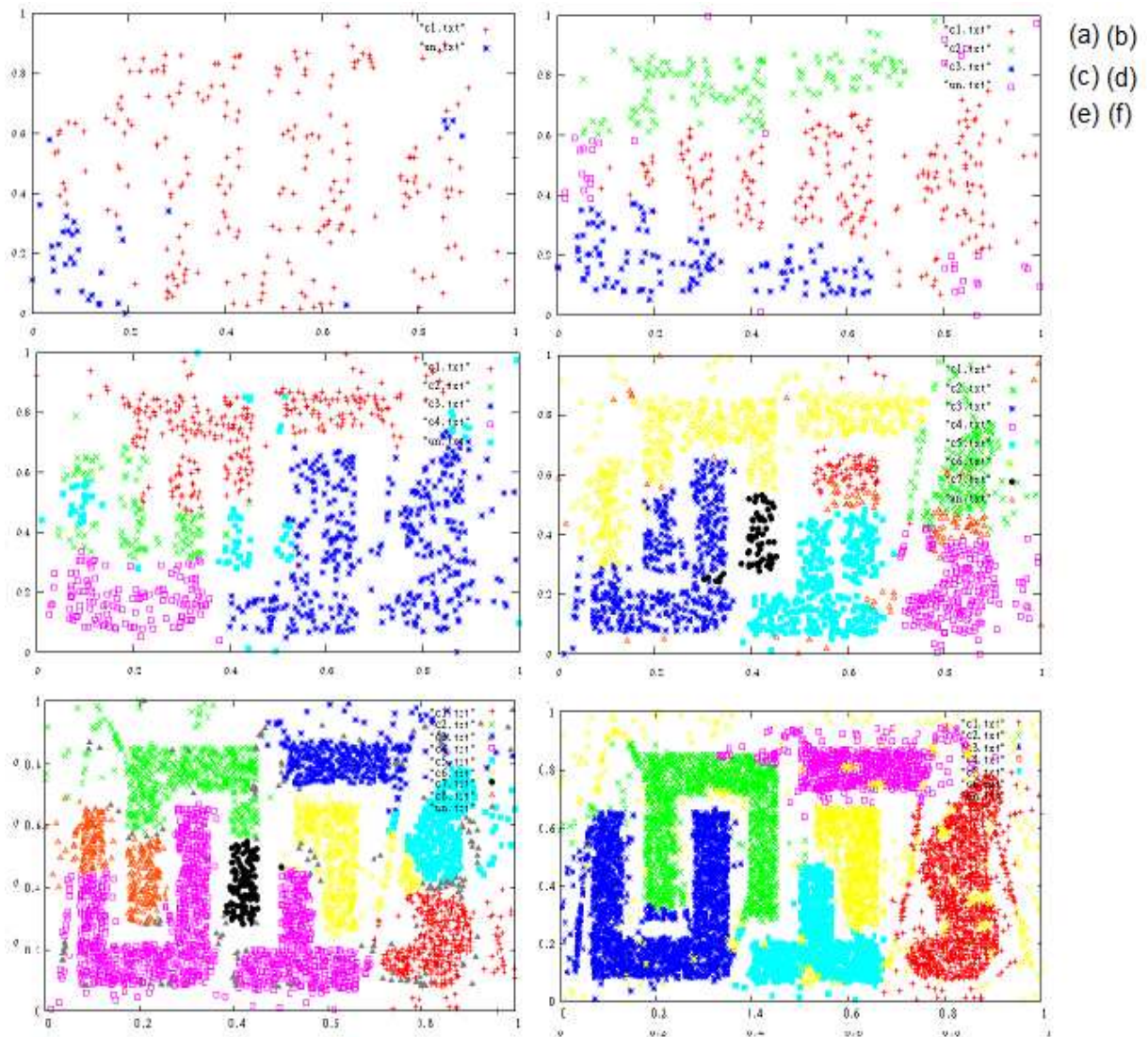


Figure 5.8: Chameleon dataset clustering results using GMDSCAN algorithm.

Table 5.4 is the resulted times of clustering adult and chameleon datasets with GMDSCAN clustering algorithm. Table's first column represents the number of points used to make the clustering process. These numbers are 250,500,1000,2000,4000 and 8000 points. The table's second and fourth columns are the multi-density datasets, adult and chameleon, clustering times in milliseconds. While the third and the fifth columns are the corresponding resulted numbers of clusters. We see that GMDSCAN algorithm is not bad and it is a good clustering algorithm. It gives good results but it is not fast enough. So we still in need to a faster clustering algorithm like the proposed one in our current study.

Table 5.4: Adult and chameleon datasets clustering results summary using GMDBSCAN algorithm.

GMDBSCAN				
No. Points	Time (ms) "adult"	No. of clusters	Time (ms) "chameleon"	No. of clusters
250	270	1	712	1
500	820	2	250	3
1000	3181	2	859	4
2000	15592	2	5448	7
4000	31672	10	17410	8
8000	227995	18	59470	6

## 5.6 : GMDBSCAN-UR Results

GMDBSCAN-UR clustering algorithm solves all problems we faced in all previous algorithms as we can see from the results in this section. Figure 5.9 and Figure 5.10 below show GMDBSCAN-UR algorithm clustering results. Figure 5.9 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from adult dataset with GMDBSCAN-UR clustering algorithm. And Figure 5.10 (a), (b), (c), (d), (e) and (f) are the results of clustering 250, 500, 1000, 2000, 4000 and 8000 points respectively from chameleon dataset with GMDBSCAN-UR clustering algorithm. Here, we will talk in more depth about the clustering process with our new proposed GMDBSCAN-UR clustering algorithm. In this section, we evaluate the performance of GMDBSCAN-UR, and compare it with previous density-based clustering algorithms. All problems we faced in all previous density-based clustering algorithms are solved with GMDBSCAN-UR algorithm. Figure 5.9 (f) is the result of clustering 8000 data points from adult dataset. GMDBSCAN-UR clustering algorithm is an effective and efficient clustering algorithm. GMDBSCAN-UR gets better results than GMDBSCAN results. GMDBSCAN-UR can recognize noise and outliers from the datasets. Figure 5.10 (f) is the results of clustering all chameleon data points, 8000 points. Chameleon dataset has 8000 data points of six clusters. Each cluster has arbitrary shape and many noises. Our proposed clustering algorithm, GMDBSCAN-UR, succeeded in clustering adult and chameleon multi-density datasets and gives good quality results with a short times. The performance of GMDBSCAN-UR algorithm is superior to GMDBSCAN, MDBSCAN and DBSCAN algorithms as the volume of data increases. In GMDBSCAN-UR algorithm, if the data points of dataset increased, the runtime complexity increases linearly as the volume of data increases. At the same time, the

improvements in time compared with the previously mentioned density-based algorithms are increased very fast.

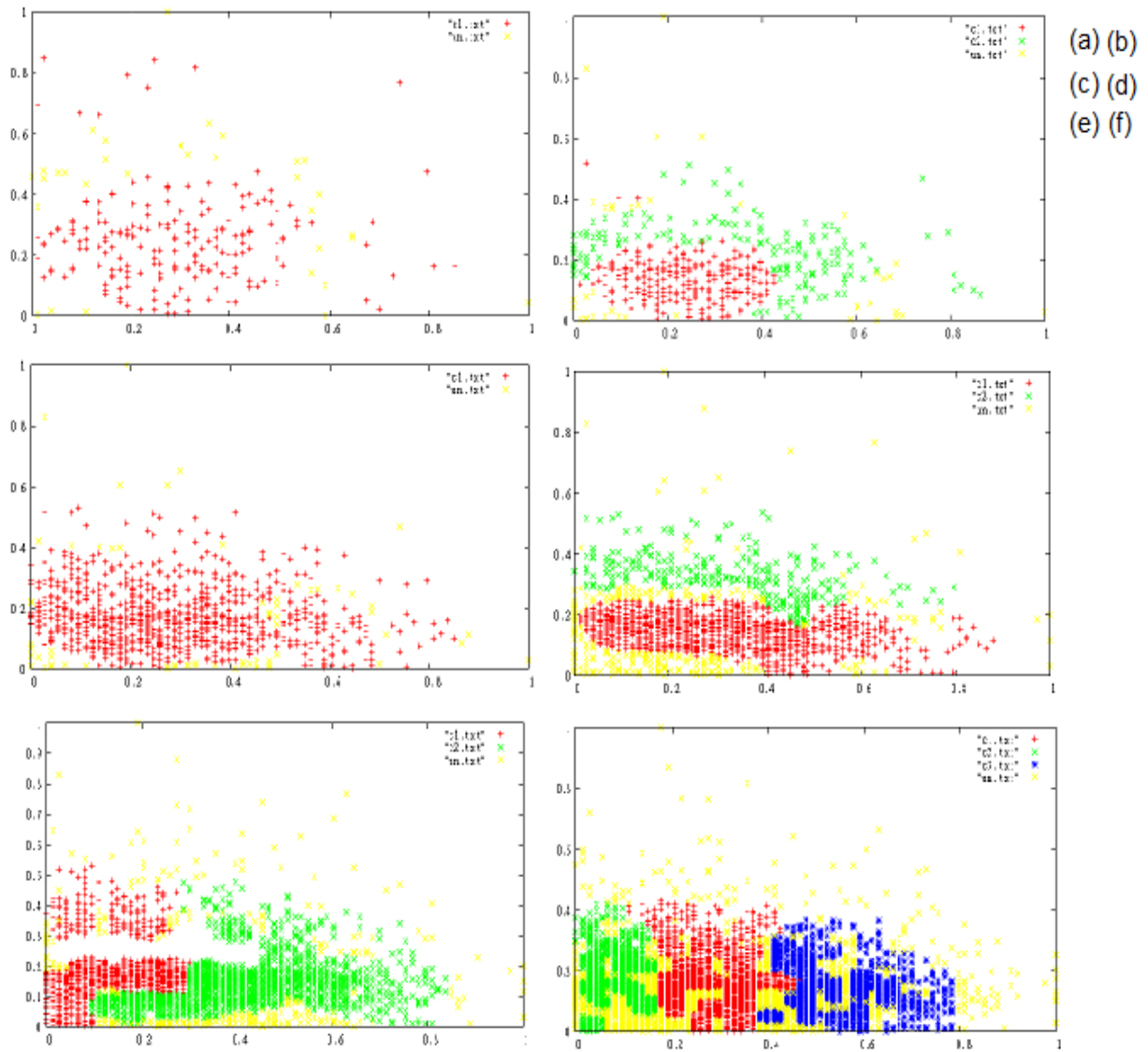


Figure 5.9: Adult dataset clustering results using GMDBSCAN-UR algorithm.

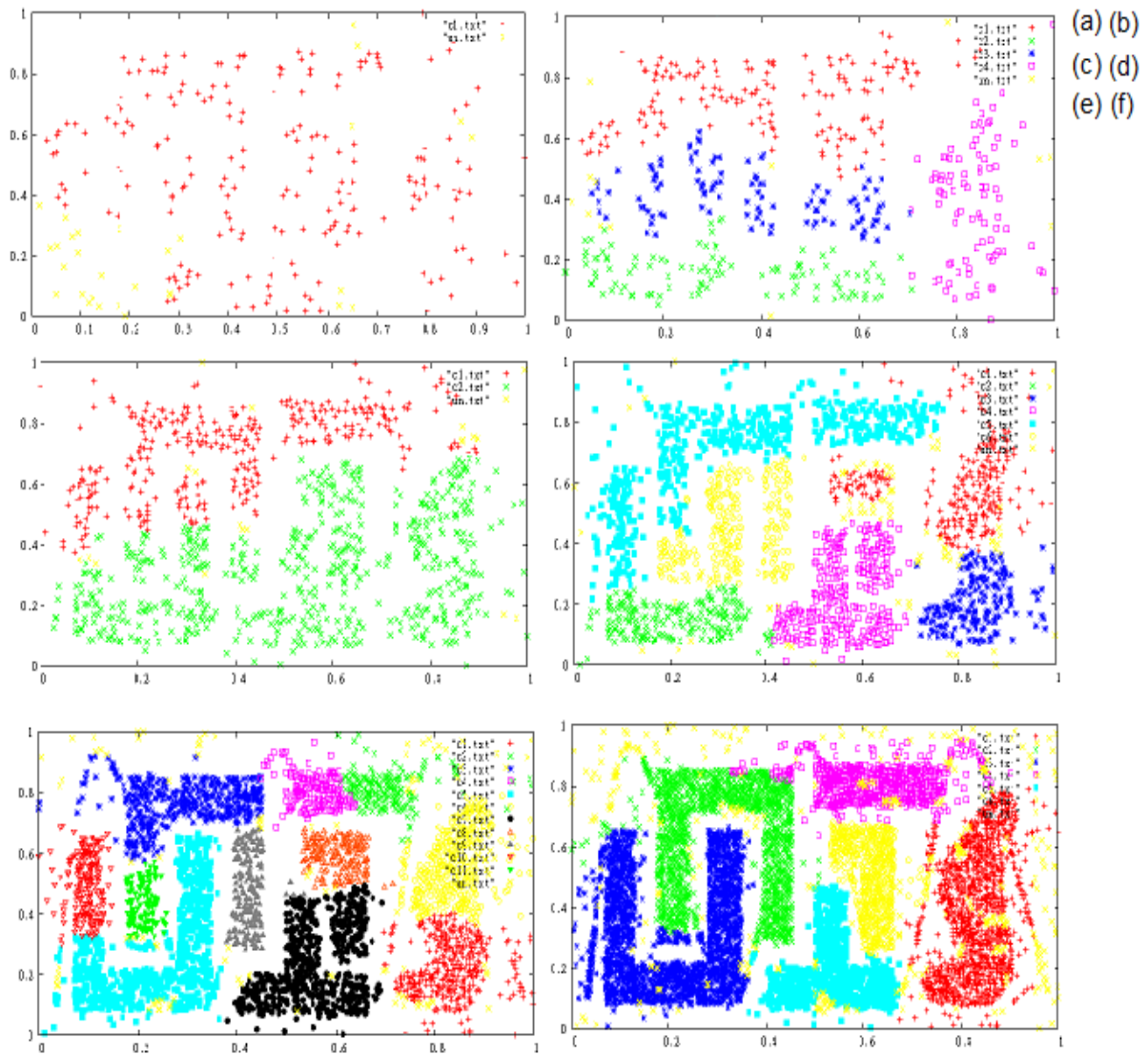


Figure 5.10: Chameleon dataset clustering results using GMDBSCAN-UR algorithm.

Table 5.5 shows the results of clustering various numbers of data points in both adult and chameleon datasets, and illustrates the differences in times between our algorithm and the density-based previous ones. Take for example the case of clustering all dataset data points, 8000 point, in adult dataset. You can see how much the times differences in milliseconds between the four clustering algorithms. Times in milliseconds are: 390855 in DBSCAN, 227995 in GMDBSCAN, a very long time in MDBSCAN and 83975 in our proposed algorithm, GMDBSCAN-UR. The difference is clear between our proposed algorithm, GDBSCAN-UR, and the comparable other algorithms. The same for chameleon dataset, clustering all data points,



8000 data points, in chameleon dataset, we saw that the difference is awesome. The clustering times in milliseconds are: 159115 in DBSCAN, very long time in MDBSCAN, 59470 in G MDBSCAN and 30767 in G MDBSCAN-UR as shown in Table 5.2, Table 5.3, Table 5.4 and Table 5.5 below. G MDBSCAN-UR clustering time is more better than all other comparable density-based clustering algorithms.

Table 5.5: Adult and chameleon datasets clustering results summary using G MDBSCAN-UR algorithm.

<b>G MDBSCAN-UR</b>				
No. Points	Time (ms) "adult"	No. of clusters	Time (ms) "chameleon"	No. of clusters
250	140	1	94	1
500	405	2	203	4
1000	1436	1	796	2
2000	5696	2	2809	6
4000	14978	2	10751	11
8000	83975	3	30767	6

As we mentioned in the previous chapter that G DBSCAN-UR algorithm's work is summarized in three main steps. The three steps are: choosing the representative data points, then dividing the data space to smaller cells and make the original DBSCAN clustering, and the third step is labeling the remainder data points, post processing with noise elimination. Here, we show the result of each step separately in details accompanied by figures to proof and demonstrate the improvements, in quality and time, achieved by G MDBSAN-UR, our new proposed algorithm.

Figure 5.11 below shows the result of choosing the representative points step only. You notice from the figure how our proposed algorithm succeeded in choosing representative points which actually represents the dataset as all. Representative points must give the actual and exact shape of the entire dataset. We achieved this in our algorithm because we choose points from everywhere, i.e. every cell in the dataset, and keeping some points not chosen from everywhere also.



Figure 5.11: The result of choosing representative points from "chameleon" with GDBSCAN-UR algorithm.

Figure 5.12 below shows the result after executing both steps; choosing the representative points and labeling the remainder points. The dataset in the figure below contains all points, i.e. both the selected as representative points and the not selected points also. This is clear because the dataset density in Figure 5.12 is higher than the dataset density in Figure 5.11. Figure 5.12 has a not tolerant error rate. Cluster 1 and cluster 4 must be one cluster, also cluster 2 and cluster 8 must be one cluster. A remerging process must be done as a post processing. Thus, the clustering result from the first two steps is not sufficient and still need the third step, post processing. Figure 5.13 is the result after the third step, the post processing step.

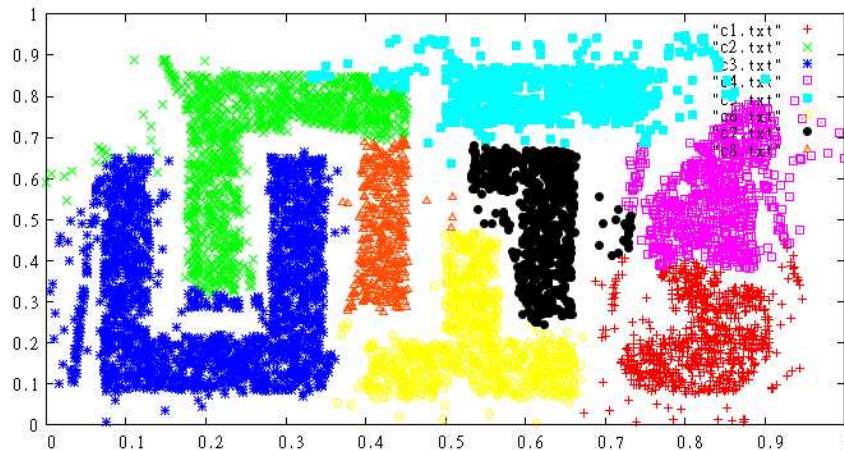


Figure 5.12: The result of choosing representative points from chameleon dataset with GDBSCAN-UR algorithm and labeling the remainder points of the dataset.

Figure 5.13 shows the final result, after the three steps of the proposed algorithm, of clustering all the 8000 data points from chameleon dataset. The artificial data set contains six clusters; each

has a different density. Clustering this artificial dataset by GMDBSCAN-UR algorithm takes nearly half the time the GMDBSCAN algorithm takes. From the result in Figure 5.13 we see that GMDBSCAN-UR algorithm gets accurate clusters, and also recognizes noises with sparse distribution. GMDBSCAN-UR algorithm does better in case of a lot of noises with more intensive distribution existence.

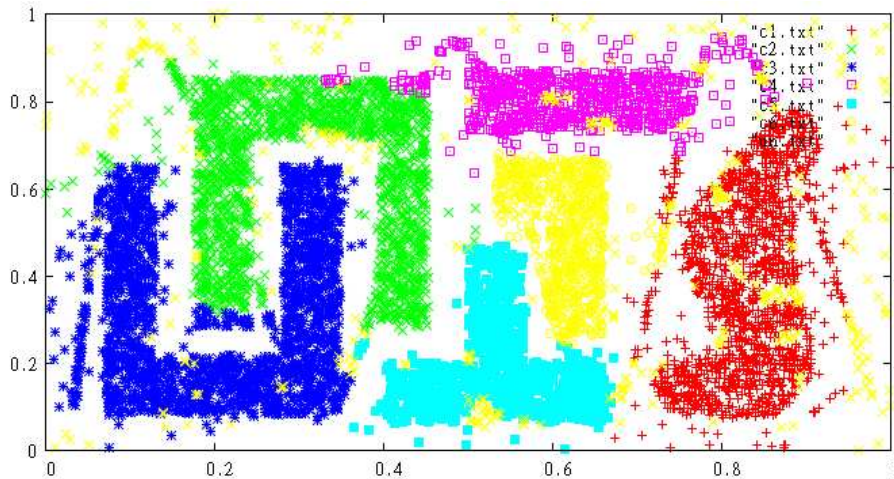


Figure 5.13: The final result of choosing representative points from "chameleon", labeling the remainder points of the dataset and remerging steps with GDBSCAN-UR algorithm

Figure 5.14 shows the final result of clustering adult dataset with GMDBSCAN-UR clustering algorithm. Adult dataset consists of three arbitrary shapes clusters and many noises. Figure 5.14 below shows the clustering quality of our proposed algorithm, GMDBSCAN-UR. It is more better than Figure 5.3 (f), Figure 5.5 (f) and Figure 5.7 (f), the clustering results of DBSCAN, MDBSCAN and GMDBSCAN respectively. GMDBSCAN-UR gets this result in nearly 83 seconds i.e. this time is more better than the other density-based algorithms' clustering times as we see in the following section.

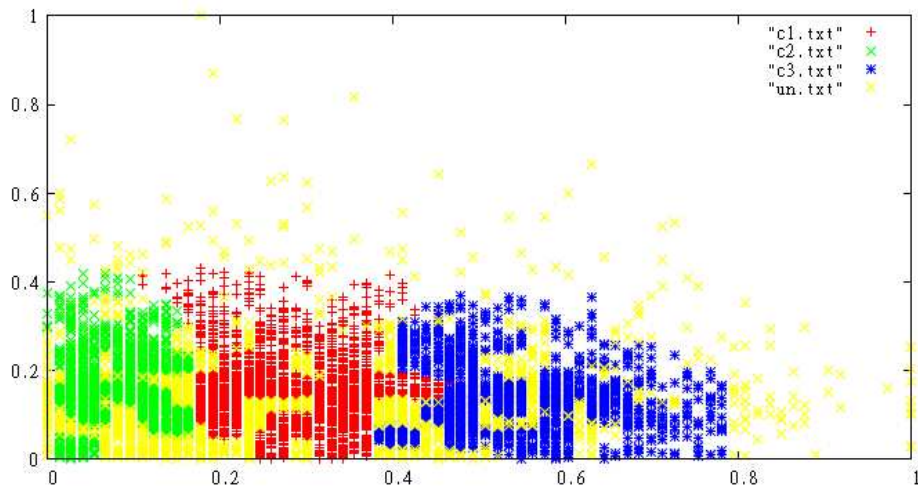


Figure 5.14: The result of clustering adult dataset with GDBSCAN-UR algorithm.

## 5.7 The Results Summary

### 5.7.1 Adult Final Results:

Here we will illustrate, in more deeply, the comparison between the four clustering algorithms in terms of both the quality of the resulting clusters and the time each takes to produce there results. Table 5.6 first row represents the density-based clustering algorithms G MDBSCAN-UR, G MDBSCAN, M DBSCAN and D BSCAN. Table 5.6 is the resulted times of clustering "adult" and "chameleon" datasets with the above mentioned four clustering algorithm. Table's first column represents the number of points used to make the clustering process. These numbers are 250,500,1000,2000,4000 and 8000 points, the table's first column. The table second and fourth columns are the multi-density dataset's, adult and chameleon, clustering times in milliseconds. While third and fifth columns are the corresponding resulted numbers of clusters.

By looking at the results in more depth at the following table, we will note how the differences in time between the four algorithms are clear for a various numbers of points used. The G MDBSCAN takes more than twice the time G MDBSCAN-UR algorithm takes. And D BSCAN nearly takes the time equal to five times the time G MDBSCAN-UR takes to cluster 8000 points in adult dataset. Our experiments' results show how wonderful the output of our new proposed algorithm, G MDBSCAN-UR, with respect to other related algorithms in both the quality and in time. It is really strong, active and fast in finding clusters effectively, and Figure 5.16, which

draws the four clustering algorithms clustering times for various data points numbers, proves the GMDBSCAN-UR efficiency.

Note that clustering small datasets made all the four clustering algorithms take nearly the same times, and this is not our concern. We are interested in clustering large datasets which contains a number of thousands data points to show the efficiency and the quality of the clustering algorithm.

Table 5.6: Adult dataset comparative clustering results summary using the four algorithms.

No. Points	GMDBSCAN-UR	GMDBSCAN	MDBSCAN	DBSCAN
	Time (ms)	Time	Time	Time
<b>250</b>	140	218	499	47
<b>500</b>	405	826	1623	140
<b>1000</b>	1436	3027	5631	796
<b>2000</b>	5696	13579	20891	5522
<b>4000</b>	14978	29329	502486	41341
<b>8000</b>	83975	186506	very long time	390855

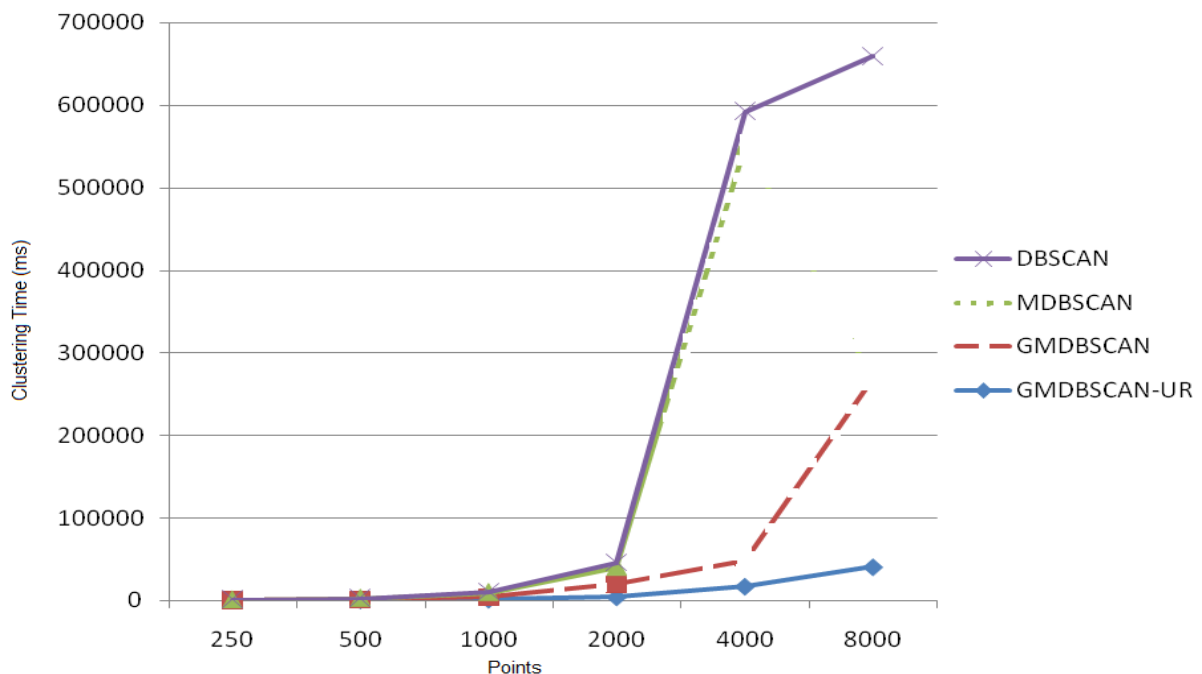


Figure 5.16: Adult dataset clustering results curves using the four algorithms.

## 5.7.2 Chameleon Final Results

By comparing between the four algorithms tables results, numbers of clusters and the times they take to produce the results, we notice the following:

By clustering few number of points, say 250 point, we see that DBSCAN algorithm is the smallest numbers of times and the MDBSCAN is the worst one and in some cases when the dataset is large that contains a large number of points, MDBSCAN takes a very long time to cluster it. In a large and complex datasets, using a large number of data points, say 4000 or 8000 points, the GMDBSCAN-UR is the best one in results for both quality and the times it takes to produce the results, see Tables 5.6,5.7 and Figures 5.16,5.17.

To support our idea in proving that the GMDBSCAN-UR is the best one, we use two datasets, chameleon and adult as we see from the above figures in the previous four sections.

Table 5.7: Chameleon dataset comparative clustering results summary using the four algorithms.

No.	<b>GMDBSCAN-UR</b>	<b>GMDBSCAN</b>	<b>MDBSCAN</b>	<b>DBSCAN</b>
Points	Time (ms)	Time	Time	Time
<b>250</b>	94	125	1763	47
<b>500</b>	203	203	64608	141
<b>1000</b>	796	967	426040	453
<b>2000</b>	2809	6008	Failed	2527
<b>4000</b>	10751	19579	Failed	19282
<b>8000</b>	30767	59470	Failed	159115

GMDBSCAN-UR clustering algorithm runs in three separate steps. Each step takes its separate time. For example, the three steps times for clustering chameleon dataset with GMDBSCAN-UR algorithm is as follows:

- The time after the first step, main clustering step is : 17044 ms.
- The time after the second step, remerge step: 19449 ms.
- The time after the third step, labeling step : 30767 ms.

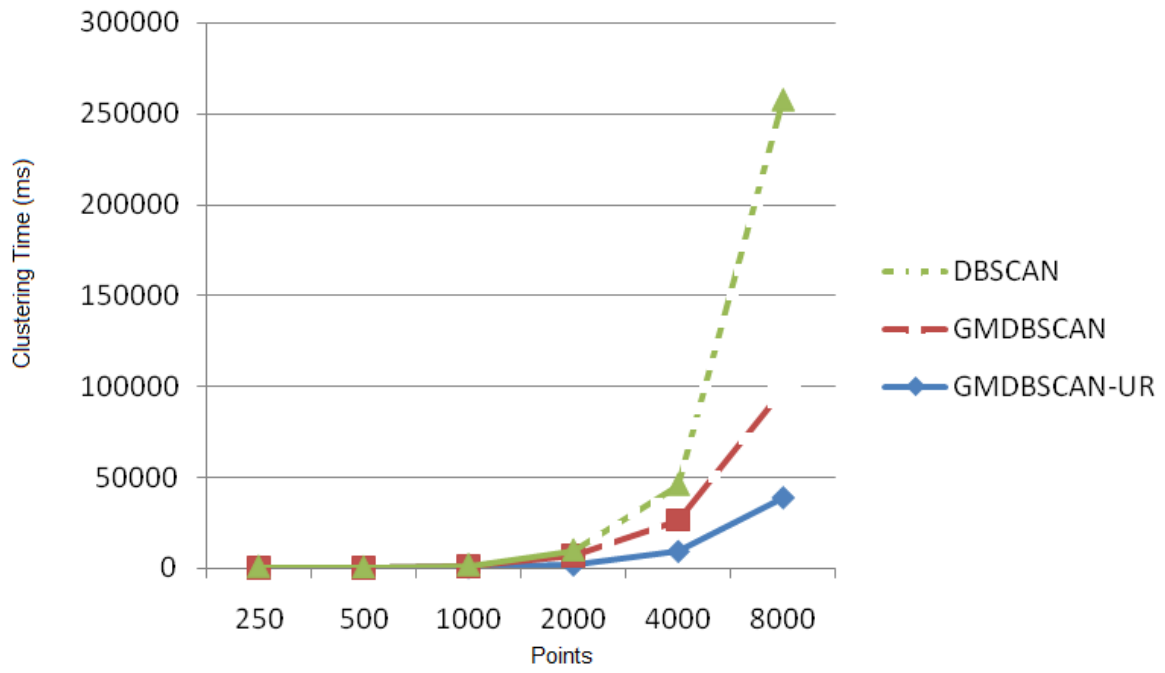


Figure 5.17: Chameleon dataset clustering results curves using the four algorithms.

# Chapter 6

## Conclusion and Future Work

### 6.1 Conclusion

In this study, we introduce a new multi-density clustering algorithm based on grid and use representative points that take the general shape of the data in the dataset. We perform an experimental evaluation to the performance of GMDBSCAN-UR using real data. The results of our experiments show that GMDBSCAN-UR is effective and efficient. In this study, in addition to handling data sets which are high dimensional, we also use representative points technique to work with reduced number of points in the dataset which result in a high saving in time. We investigated using representative points not all data set points for improving the performance of our algorithm. We developed a novel effective clustering algorithm which improved the performance of the DBSCAN algorithm. The proposed clustering algorithm uses SP-tree, and divides its work into three main steps which are main clustering after getting the representative points, remerging clusters to get a more accurate result and last labeling the remainder data points which are not entered in the clustering process to the true clusters that they belong to.

The GMDBSCAN-UR algorithm that we presented is specific to clustering more complex and with large number of points data sets. The proposed clustering algorithms have a great saving in running time and giving amazing results. Experimental results are shown in this thesis to demonstrate the effectiveness of the proposed algorithm. WE illustrated the time complexity and the performance of classifying complex data sets. We proved that the proposed algorithms can classify complex data sets more accurately than other previous algorithms.

### 6.2 Future Work

The work reported in this thesis may be extended in a number of ways, some of which are discussed below:

We can use a special algorithm for selecting the representation points of the data set and this algorithm can result with a better representation of the data set and this will obviate using of the so-called remerging the resulting clusters and post processing methods, and that in turn leads to



more saving in time and result in best performance of the algorithm. We say this because some nodes in the data set contain odd numbers of points and that leads it hard to pick points and leave the other for the labeling method later in the algorithm after the clustering stage.

## References

- [1] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, 2006.
- [2] "Clustering Algorithms" available at <http://theory.stanford.edu/~nmishra/cs369C-2005.html> , last visit 22 Aug 2011
- [3] J. Han and M. Kamber. "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, 2001.
- [4] L. Kaufman and P. Rousseeuw, "Finding groups in Data: an Introduction to cluster," John Wiley & Sons, 1990.
- [5] J. Han, M. Kamber, and A. K. H. Tung, "Spatial Clustering Methods in data mining," A Survey, Geographic Data Mining and Knowledge Discovery, 2001.
- [6] P. Bradley, U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," In proc. 1998 Int. Conf. Knowledge Discovery and Data mining, 1998.
- [7] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," In Proc. 1996 ACM SIGMOD Int. Conf. Management of data (SIGMOD'96), 1996.
- [8] S. Guha, R. Rastogi, and K. Shim, "Cure : An efficient clustering algorithm for large databases," In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), 1998.
- [9] M. Ester, H. P. Kriegel, J. sander, and X. Xu, "A density based algorithm for discovering clusters in large spatial databases," In Proc. 1996 Inc. Conf. Knowledge discovery and Data mining (KDD'96).

- [10] M. Ankerst, M. Breunig, H.P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," In Proc. 1999 ACM-SIGMOD Int. Conf. Management of data (SIGMOD'96), 1999.
- [11] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," In Proc. 1998 Int. Conf. . Knowledge discovery and Data mining (KDD'98), 1998.
- [12] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," In Proc. 1997 Int. Conf. Very Large Data Bases ( VLDB'97), 1997.
- [13] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wave Cluster : A multi- resolution clustering approach for very large spatial databases," In Proc. 1997 Int. Conf. Very Large Data Bases ( VLDB'97), 1998.
- [14] R. Agrawal, J. Gehrke and D. Gunopulos, P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining application," In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), 1998.
- [15] J. W. Shavlik and T.G. Dietterich, "Reading in machine learning," 1990.
- [16] T. Kohonen, "Self organized formation of topologically correct feature maps," Biological Cybernetics, 1982.
- [17] T. Huang ,Y. Yu, K Li and W. Zeng, "Reckon the Parameter of DBSCAN for Multi-density Data Sets with Constraints," Dept. of Computer Science, School of Mathematics & Computer Science Fujian Normal University Fuzhou, China; 2009.
- [18] Martin Ester and Hans-Peter Kriegel, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc of 2nd Int. Conf. on KDD'96, 1996: 226-231.

- [19] Zhou S, Zhou A and et al, "A Fast Density-Based Clustering Algorithm," [J]. Journal of Computer Research and Development, 2003, 37(11):1287-1292.
- [20] Zhou S, Fan Y and Zhou A. SDBSCAN, "A Sampling- Based DBSCAN Algorithm for Large-Scale Spatial Databases," Journal of Chinese Computer Systems[J], 2000, 21(12): 1270-1274.
- [21] Zhou A, Zhou S, Cao J and et al. "Approaches for scaling DBSCAN algorithm to large spatial database," [J].Journal of computer science and technology,2000,15(06): 509-526.
- [22] Zeng Donghai, "The Study of Clustering Algorithm Based on Grid-Density and Spatial Partition Tree," XiaMen University, PRC, 2006.
- [23] A. H. Pilevar. M. Sukumar, "GCHL: a grid-clustering algorithm for high-dimensional Data Mining," SBIA 2004, LNAI 3171.
- [24] C. H. Chou, M. C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," Pattern Analysis and Applications, vol. 7, no. 2, pp. 205–220, July 2004.
- [25] G. Hammerly, and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," Proc. ACM on Information and Knowledge Management, pp. 600– 607, November 2002.
- [26] P.S. Bradley, and U.M. Fayyad, "Refining Initial Points for K-Means Clustering," ICML 1998, pp. 91–99, January 1998.
- [27] W. Wang, J. Yang, and R. Muntz, "STING: A statistical Information grid approach to spatial data mining," VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25-29, 1997.
- [28] P. Berkhin, "Survey of Clustering Data Mining Techniques," Accrue Software, Technical Report, 2002.

- [29] J. Kogan, "Introduction to Clustering Large and High-Dimensional Data," Cambridge University Press, New York, 2007.
- [30] J. Valente de Oliveira, and W. Pedrycz, "Advances in Fuzzy Clustering and its Applications," John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, vvvEngland, 2007.
- [31] M. Sato-Ilic, and L. C. Jain, "Innovations in Fuzzy Clustering," Springer-Verlag Berlin Heidelberg, New York, 2006.
- [32] W. Pedrycz, "KNOWLEDGE-BASED CLUSTERING From Data to Information Granules," John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.
- [33] A. K. Jain and R. C. Dubes, "Algorithms for Clustering Data," Prentice Hall, 1988.
- [34] K. Fukunaga (1990), "Introduction to Statistical Pattern Recognition," 2nd edition. Computer Science and Scientific Computing. San Diego, CA: Academic Press.
- [35] J. Macqueen (1967), "Some methods for classification and analysis of multivariate observations," In Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, volume 1, pages 281–297. Berkeley, CA: University of California Press.
- [36] R. Sharan and R. Shamir (2000), "CLICK: A clustering algorithm with applications to gene expression analysis," In Proceedings of the eighth international conference on intelligent systems for molecular biology (ISMB), pages 307–316. Menlo Park, CA: AAAI Press.
- [37] S. Basu, A. Banerjee and R. Mooney, "Active semisupervision for pairwise constrained clustering," Proc. of the SIAM Int. Conf. on Data Mining (SDM-2004), MIT Press, Apr. 2004, pp.333-344, doi: 10.1.1.5.877.

- [38] K. Wagstaff, C. Cardie, S. Rogers and S. Schroedl, "Constrained K-Means clustering with background knowledge," Proc. of 18th Int. Conf. on Machine Learning (ICML-2001), Morgan Kaufmann Publishers, Jun. 2001, pp.577-584, doi: 10.1.1.20.7363.
- [39] N. Grira, Crucianu and N. Boujema, "Semi-supervised fuzzy clustering with pairwise-constrained competitive agglomeration," IEEE Int. Conf. on Fuzzy Systems (Fuzz- IEEE 2005), IEEE Press, May 2005, pp.22-25, doi: 10.1.1.109.938.
- [40] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: theoretical and empirical results," Proc. of Principles of Knowledge Discovery from Databases (PKDD 05), Springer, Oct. 2005, pp.59-70, doi: 10.1007/11564126\_11.
- [41] I. Davidson and S. S. Ravi, "Intractability and clustering with constraints," Proc. of 24th International Conference on Machine Learning (ICML 2007), ACM Press, Jun. 2007, pp.201-208, doi: 10.1145/1273496.1273522.
- [42] T. Wei, X. Hui, Z. Shi and W. Jie, "Enhancing semisupervised clustering: a feature projection perspective," Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 07), ACM Press, Aug. 2007, pp. 707-716, doi: 10.1145/1281192.1281268.
- [43] M. Ester and H. Kriegel. "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc of 2nd Int. Conf. on KDD'96, 1996: 226-231.
- [44] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A densitybased algorithm for discovering clusters in large spatial database with noise," Proc. of 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining (KDD 96), AAAI Press, Aug. 1996, pp.226-231, doi: 10.1.1.72.8521.
- [45] C. Xiaoyun, M. Yufang, Z. Yan and W.Ping, "GMDBSCAN: Multi-Density DBSCAN Cluster Based on Grid," *School of Information Science and Engineering, Lanzhou University Lanzhou 730000, PRC China.*

- [46] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96), pages 103–114, Montreal, Canada, June 1996.
- [47] Gan, G., Ma, C., and Wu, J., "Data Clustering: Theory, Algorithms, and Applications,"
- [48] M. Anderberg (1973), "Cluster analysis for applications," New York: Academic Press.
- [49] S. Guha, R. Rastogi and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," Stanford University.
- [50] Asuncion, A. & Newman, D.J. UCI Machine Learning Repository, available at [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. 2007, last visit 22 Aug 2011
- [51] A. H. Pilevar and M. Sukumar, "GCHL: a grid-clustering algorithm for high-dimensional Data Mining," SBIA 2004, LNAI 3171.
- [52] G. Milligan and M. Cooper (1988), "A study of standardization of variables in cluster analysis," Journal of Classification, 5:181–204.
- [53] A. Jain and R. Dubes (1988). Algorithms for Clustering Data. Englewood Cliffs, NJ: Prentice–Hall.
- [54] "CLUTO - Software for Clustering High-Dimensional Datasets" available at <http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download/> , last visit 7 Oct 2011