AN ADAPTIVE DENSITY-BASED METHOD FOR CLUSTERING EVOLVING DATA STREAMS

AMINEH AMINI

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2014

AN ADAPTIVE DENSITY-BASED METHOD FOR CLUSTERING EVOLVING DATA STREAMS

AMINEH AMINI

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2014

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **AMINEH AMINI** (I.C./Passport No.: **L95236745**)

Registration/Matrix No.: **WHA090004**

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

AN ADAPTIVE DENSITY-BASED METHOD FOR CLUSTERING EVOLVING DATA STREAMS

Field of Study: **DATA STREAM MINING**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

Density-based method has emerged as a worthwhile class for clustering data streams. It has the abilities to discover clusters of arbitrary shapes, handle noise, and cluster without prior knowledge of number of clusters. The characteristics of data stream includes infinite volume, dynamically changing, allowing only one or a small number of scans, and demanding fast response time. Due to these characteristics the traditional density-based clustering is not applicable.

Recently, a number of density-based algorithms have been developed for clustering data streams. However, existing density-based data stream clustering algorithms are not without problems. The first problem refers to the high computation time required for the clustering process. The second problem is the dramatic decrease in the quality of clustering when there is a range in density of data. In this research, these problems are taken into account and a new method is proposed.

This study proposes a density-based algorithm for clustering evolving data streams. The proposed method, which is called MuDi-Stream (Multi Density clustering algorithm for evolving data Stream), is an online-offline algorithm with four main components. Three of components are applied in the online phase while the other one is used in the offline phase. The prominent tasks of these components are keeping synopsis information, pruning these information, and forming final clusters.

In the first component, a hybrid method comprised of density grid and micro clustering techniques is applied to maintain summary information in the form of core mini clusters while mapping outlier to the grids. The data points inside the grid form a new core mini cluster in case it reaches a density threshold in the second component. Furthermore, grid and core mini clusters are pruned using a pruning technique in the last component of online phase in order to keep the memory limited. A new multi density-based clustering

method forms final clusters using both summarized synopsis information and statistical information.

The quality of the algorithm is comprehensively evaluated on various synthetic and real datasets with different characteristics using variety of quality metrics. The complexity analysis shows that it uses limited time and memory which makes MuDi-Stream applicable for data stream. Furthermore, the scalability results prove that the proposed algorithm is scalable in terms of both dimension and number of clusters. Finally, the experimental results show that the proposed method in this study improves clustering quality in multi-density environments while minimizing the computation time.

ABSTRACT

Kaedah berasaskan berkepadatan telah muncul sebagai kelas yang amat bernilai untuk kelompok aliran data. Ia mempunyai kebolehan untuk menemui kelompok berbentuk rawak, mampu mengatasi masalah gangguan (noise) dalam aliran data dan kelompok yang asalnya tidak diketahui jumlah kelompoknya. Ciri-ciri aliran data adalah termasuk isipadunya infiniti (infinite volume), ia berubah secara dinamik, ia membolehkan hanya satu atau sebilangan kecil imbasan, dan ia memerlukan masa tindak balas yang pantas. Oleh kerana ciri-ciri ini, kelompok berasaskan berkepadatan tradisional tidak dapat diaplikasikan.

Baru-baru ini, beberapa algoritma berasaskan berkepadatan telah dibangunkan untuk kelompok aliran data. Walau bagaimanapun, terdapat masalah pada algoritma untuk pengelompokan aliran data berdasarkan kepadatan (density-based data stream clustering algorithms) sedia ada. Masalah pertama merujuk kepada masa pengiraan yang tinggi yang diperlukan untuk memproses kelompok itu. Masalah kedua adalah penurunan dramatik dalam kualiti kelompok apabila terdapat pelbagai kepadatan data. Dalam tesis ini, masalah-masalah ini diambil kira dan kaedah baru adalah dicadangkan.

Kajian ini mencadangkan satu algoritma berdasarkan kepadatan bagi pengelompokan aliran data yang berkembang. Kaedah yang dicadangkan, yang dipanggil MudiStream (Kelompok algoritma pelbagai ketumpatan untuk merubah aliran data) adalah algoritma dalam talian - luar talian (online - offline) dengan empat komponen utama. Tiga komponen digunakan dalam fasa dalam talian manakala yang lain digunakan dalam fasa luar talian. Tugas-tugas utama komponen ini menyimpan maklumat sinopsis, memangkas maklumat yang disimpan dan membentuk kelompok akhir.

Dalam komponen pertama, kaedah hibrid terdiri daripada grid kepadatan dan teknik kelompok mikro digunakan untuk mengekalkan maklumat ringkasan dalam bentuk kelom-

pok teras mini sambil melakukan pemetaan titik terpencil kepada grid. Titik data dalam grid akan membentuk teras kelompok mini baru apabila ia mencapai ambang ketumpatan dalam komponen kedua. Tambahan pula, grid dan teras mini kelompok dipangkas menggunakan teknik pemangkasan dalam komponen terakhir fasa dalam talian untuk menyimpan memori yang terhad. Kaedah kelompok berasaskan pelbagai kepadatan (multi density-based clustering) membentuk kelompok yang terakhir dengan menggunakan kedua-dua ringkasan maklumat sinopsis dan maklumat statistik.

Kualiti algoritma dinilai secara komprehensif menggunakan pelbagai dataset sintetik dan sebenar dengan ciri-ciri yang berbeza dengan menggunakan pelbagai kualiti metrik. Analisis kerumitan(complexity analysis) menunjukkan bahawa ia menggunakan masa dan memori yang terhad yang menjadikan Mudi-Stream dapat diaplikasikan untuk aliran data. Tambahan pula, keputusan berskala membuktikan bahawa algoritma yang dicadangkan itu adalah berskala(scalable) dari segi dimensi dan jumlah kelompok. Akhir sekali, keputusan eksperimen menunjukkan bahawa kaedah yang dicadangkan dalam kajian ini akan memperbaiki kualiti kelompok dalam persekitaran pelbagai kepadatan (multi density environment) dan pada waktu yang sama akan mengurangkan masa pengiraan

ACKNOWLEDGEMENTS

First and above all, I thank the Almighty for superior protection and guidance throughout the lows and highs of my Ph.D journey. All my thanks goes to all people involved in this process of growth as a being that thinks.

My foremost and utmost gratitude goes to my supervisor Associate Professor Dr. Teh Ying Wah for his dedicated supervision during my Ph.D study. His endless help, care, kindness, patience, generosity, and thoughtful consideration is greatly valued. Thank you very much for your remarkable advices.

I want to express my deep thanks to Associate Professor Dr. Hassan Abolhassani for his inspiration and offering valuable advice.

To my parents, Maryam Karimi and Jafar Amini, words are not enough to express my love. I am forever indebted to you. Your unconditional, and selfless support have been the foundation of my life.

A great part of this project would not exist without the help of my husband Hadi Saboohi, a source of wisdom, encouragement, and strength. Thank you Dr. Hadi to act as my co-supervisor. I have very much enjoyed our discussions. Your most trenchant comments always shaped my work better.

Last but not least thanks to my lovely princess, Saba, for making my life more meaningful. I really miss every minute which I am not spending with her for the sake of study. To my daughter Saba: your smile is the most beautiful thing in the world and I love you so much.

Amineh, September 2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF APPENDICES

LIST OF ACRONYMS

ARI	Adjusted Rand Index
CMC	Core-mini-cluster
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EDS	Evolving Data Stream
FCM-Component	Forming Core-mini-clusters Component
FFC-Component	Forming Final Clusters
JI	Jaccard Index
M-DBSCAN	Multi-density DBSCAN
mcd	mini core distance
MDS	Multi-density Dataset
MM-Component	Merging/Mapping Component
MuDi-Stream	Multi Density-based Clustering Algorithm for Data Stream
NMI	Normalized Mutual Information
PGCM-Component	Pruning Grid and Core-mini-clusters Component
RI	Rand Index

CHAPTER 1

INTRODUCTION

1.1 Background

Ever-growing volume of data production is the reality we are living in. Social networks, smart cities, telephone networks are some of the examples which are the generator of huge amount of data in the modern world. Now we face-off a new kind of data produced continuously over time called data stream (Severien, 2013).

Different from traditional datasets, data stream flow in and out ceaselessly. They are enormous, rapid changing, and potentially limitless. Due to the huge volume of data stream, it may be impractical to record the whole data stream or to look over it multiple times (Han, Kamber, & Pei, 2011).

Mining data stream is related to extracting knowledge structure represented in streams information (Han & Kamber, 2006; Hahsler & Dunham, 2011). Clustering is one of the prominent methods for mining data stream which has drawn lots of attention in the last few years due to ever-growing presence of data stream. The goal of clustering is to group the streaming data into meaningful classes. Data stream creates additional challenges on clustering such as clustering in limited memory and limited time, examining the data only once as it arrives, and handling data in evolving manner to capture the underlying changes in clusters.

Clustering algorithms in general have been categorized into five types: partitioning, hierarchical, density-based, grid-based, and model-based methods (Han et al., 2011). In most of the clustering methods, the clusters are formed based on the distance between objects. These methods discover only spherical-shaped clusters, and have difficulty to

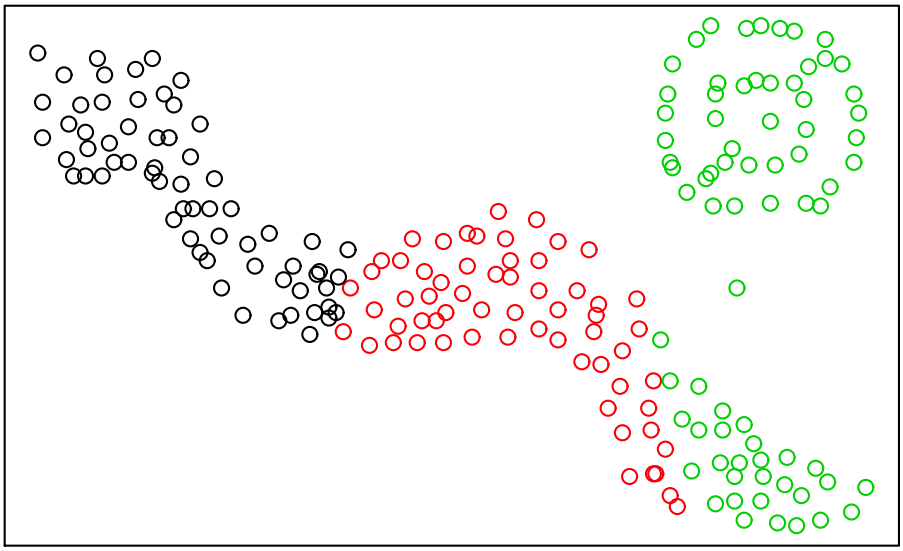
both find arbitrary shape clusters and handle noise.

In fact, many of notable clustering algorithms cannot cope well with the data in which the actual clusters have non-spherical shapes. Furthermore, clustering data stream needs the ability to detect and remove noise and outliers (Ester, 2013). Density-based clustering method has the aforementioned characteristics and yet no assumption about the number of clusters. Density-based method has been developed based on the concept of density. The clusters are formed as dense areas which are separated from sparse regions. The main idea is to continuously grow a given cluster as long as the density (number of objects or data points) in the neighborhood exceeds a threshold.

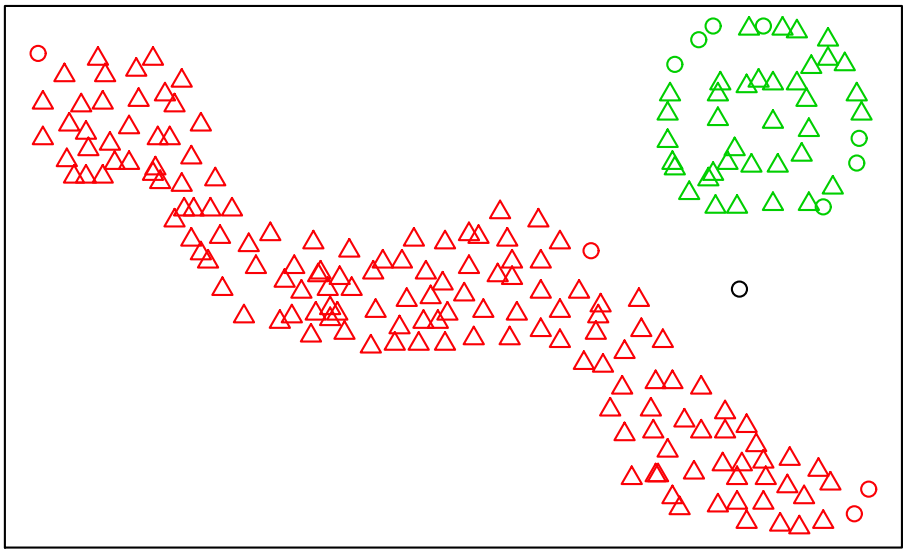
Figure 1.1 displays a comparison on a partitioning-based clustering such as k-means (MacQueen, 1967) versus DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester, Kriegel, Sander, Wimmer, & Xu, 1998), a density-based clustering algorithm. In clustering methods such as k-means, the objective, which is to minimize the average squared distance of points from the corresponding cluster center, leads to form clusters regardless of the shape of the actual cluster (Ester, 2013). Density-based clustering algorithms instinctively handle noise by preventing to put them into clusters. It is observed that the arbitrary shape clusters and noise are detected accurately using density-based clustering algorithm while k-means falsely detects the noise as a part of clusters (Ester, 2013).

Therefore, density-based method (Ester et al., 1998) has come out as a valuable class for clustering data stream.

Definition 1 ***(Data Stream).*** *A data stream consists of a set of d-dimensional records* $x_1, \ldots, x_i, \ldots$ *arriving at time stamps* $t_1, \ldots, t_i, \ldots$ $, x_i = (x_i{}^1, \ldots x_i{}^d)$. *It is massive (e.g., terabytes in volume), temporally ordered, fast changing, and potentially infinite (Zhou, Cao, Qian, & Jin, 2008; Han & Kamber, 2006; Aggarwal & Reddy, 2013).*



(a) Partitioning-based Clustering (k-means, k=3)



(b) Density-based Clustering (DBSCAN, $MinPts = 5,\ \varepsilon = 20$)

Figure 1.1: Data Stream Clustering Comparison on a Synthetic Dataset

With substantial growth in computer network during the past few decades, data streams are being collected continuously, for example, an earthquake monitoring system has up to 7,000 sensor systems to collect seismic data. These data are continuously gathered into processing centers for further analysis. The data of such nature are called data stream, which flow through computer systems at high speeds. The data stream often are too large to fit in the main memory and the time is limited to process these fast changing data.

Definition 2 ***(Evolving Data Stream).*** *In evolving data stream, the behavior of stream is considered as an evolving process over time, and further it is possible that in the new data arrival, a new cluster appears that has never appeared in the stream before. Moreover, it may happen that a class appears, which has not been in the stream for a long time.*

Definition 3 ***(Clustering Data Stream).*** *The process of partitioning d-dimensional records* $x_1,\ldots,x_i,\ldots$ *arriving at time stamps* $t_1,\ldots,t_i,\ldots,x_i = (x_i{}^1,\ldots x_i{}^d)$ *into number of groups* $C = \{C_1,C_2,\ldots,C_k\}$ *such that the intra-cluster similarity is maximized, and the inter cluster similarity is minimized. Similar records are categorized together as one cluster* C_i *using similarity function while dissimilar objects separated in different groups or the group of noise (noisy points).*

Clustering data stream requires a process able to continuously cluster objects within memory and time restrictions (Silva et al., 2013). Clustering data stream has to achieve the following requirements: (i) provide timely results by performing fast and incremental processing of data points; (ii) rapidly adapt to evolving data stream, which means algorithms should detect when new clusters may appear, or others disappear; (iv) provide a compact model representation which is not growing with the number of data points processed; (v) detect the presence of outliers (Babcock, Babu, Datar, Motwani, & Widom, 2002; Babcock, Datar, & Motwani, 2002; Barbará, 2002; Tasoulis, Ross, & Adams, 2007; Bifet, Holmes, Kirkby, & Pfahringer, 2010).

Definition 4 ***(Density-based Clustering).*** *Density-based method is a prominent class in clustering. The main advantage of density-based clustering is that it can find the arbitrary shape of clusters and also provide natural protection against outliers. Furthermore, they do need the number of clusters in advance. Density-based clusters are connected, dense areas in the data space separated from each other by sparser areas. Additionally, the*

density within the areas of noise is assumed to be lower than the density in any of the clusters.

Since density-based clusters are not certainly groups of points with a low pairwise within-cluster dissimilarity as measured by a dissimilarity function, dense connected areas in the data space can have arbitrary shapes. Sparse areas in the data space are treated as noise and are not assigned to any cluster (Kriegel, Kröger, Sander, & Zimek, 2011; Aggarwal & Reddy, 2013).

Definition 5 ***(Adaptive Density-based Clustering for Evolving Data Stream).*** *The density-based method is an attractive clustering algorithm for data stream since it can find arbitrarily shaped clusters yet handle noises. Furthermore, they do not need the number of clusters in advance. Therefore, density-based method is adopted for data stream. However, due to data stream characteristics the traditional density-based clustering algorithm is not applicable for them. There are some problems with the existing density-based algorithms: they have high computation time in clustering data streams and low quality for multi-density data. Thus, in this thesis, we choose density-based method for clustering data stream according to their salient features. However, we solve the problems of the existing algorithms to be applicable for data streams.*

Definition 6 ***(Multi-density data).*** *In a multi-density dataset, there is a range of densities for the clusters and multi-density clusters refer to the clusters that are formed in different densities.*

1.2 Motivation

In real world applications (Cao, Ester, Qian, & Zhou, 2006; Hershberger, Shrivastava, & Suri, 2009):

- naturally occurring clusters are typically not spherical in shape. For instance, in habitat monitoring using wireless sensor networks, the sensors produce a steady stream of geographic data including objects' locations being tracked. In this kind of applications, clusters may have arbitrary shape because of the restraints put by geographic entities such as mountain and rivers.

- there are large amounts of noise or outliers in some of them, for instance, due to the influence of different factors such as temporary failure of sensors in data stream scenario, some random noises appear occasionally. Detecting noise is one of the important issues specifically in evolving data stream in which the role of real data changes to noise over time.

- there is not a priori knowledge in many real-life data to be considered as the number of clusters.

The prototype of density-based clustering has been developed to address all of the aforementioned requirements. However, density-based method has some drawbacks as follows:

Firstly, it has high computation time because the process, which finds the nearest neighbors to form clusters, is time consuming. Recently, a number of density-based clusterings are developed for clustering data stream. In data stream environments finding neighbors to form clusters are performed on summarized data. Some of the algorithms (Isaksson, Dunham, & Hahsler, 2012; Forestiero, Pizzuti, & Spezzano, 2013) try to reduce the number of comparisons in order to reduce the computation time but it is still high to be applicable for data stream.

Secondly, density-based clustering fails to handle the local density variation that exists within a cluster. Multi-density data are prevalent in some applications such as clustering GPS data stream to determine the traffic accurately. There are a few number of

density-based clustering algorithms for multi-density datasets (Carmelo, Alfredo, Rosalba, Giuseppe, & Alfredo, 2013; X. Chen, Liu, Chen, Zhang, & Zhang, 2012) which try to solve this problem for static datasets. Nonetheless, these algorithms have some difficulties to be used for data stream. First, they need the whole data for their processing. In data stream, it is impossible to have the whole data in advance since the data arrives continuously over time. Second, they require two processes over the data to get density distribution and then they can cluster based on the related information. However, clustering data stream has to be performed in single pass. Accordingly, there is not an effective algorithm to get the accurate density of the data stream with multi-density.

1.3 Problem Statement

The problem of this study is defined as follows:

Existing density-based clustering algorithms for evolving data stream have high computation time and low quality in multi-density data.

Density-based clustering has two important problems. First, they have high computation time. Accordingly, some research efforts have been made to present methods for density-based data stream clustering; however, they still suffer from their computation time.

Since density-based clustering method uses global parameters, it cannot capture the intrinsic cluster structure of data stream. The existing density-based clustering algorithms cannot choose parameters according to the distribution of data. Using the global parameters leads to inaccurate clustering result of multi-density data. As it is shown in Figure 1.2, it is impossible to detect all the clusters properly. Using global parameters, the clustering results consist of either C1, C2, and C3 as clusters and A and B as noise, or A, B, and C as clusters in which C1, C2, and C3 are not separated. Therefore, the clustering quality of existing methods is lessened remarkably if the data has various densities.

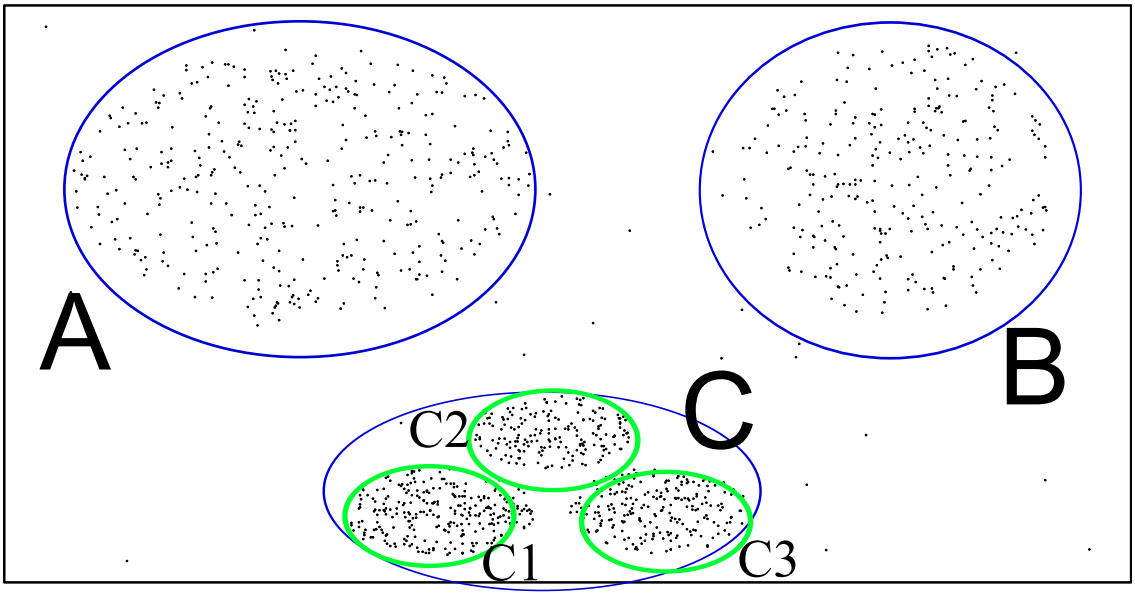


Figure 1.2: Multi-density Data (Cassisi et al., 2013)

1.4 Research Questions

The research questions which this thesis will answer are as follows:

- Q1. Which method is more appropriate for summarizing data stream?
- Q2. How to handle evolving data stream?
- Q3. What are the reasons of high computation time?
- Q4. How to lower the computation time?
- Q5. What are the issues that impede the clustering quality in multi-density environments?
- Q6. How to increase the clustering quality in multi-density data?

1.5 Research Objectives

The objectives of this research are as follows:

- **To propose and develop a new density-based algorithm for clustering evolving data stream.**

- **To improve the quality of clustering for multi-density data.** The clustering quality has to be high for normal distribution data as well as for the data with dissimilar density distributions.

- **To reduce the computation time.** The computation time has to be low enough to cope with the speed of arriving data stream.

- **To evaluate** the capability of the proposed method in improving the Quality.

The first research objective answers research questions RQ1 and RQ2. Research questions RQ3, and RQ4 are answered by the third research objective. Finally, research questions RQ5, and RQ6 are answered by the second research objective.

1.6 Scope of Research

- This thesis studies in the scope of clustering *evolving* data stream. In other words, we present a method for which the data stream evolves over time, therefore some clusters disappear while new clusters are formed.

- We provide a data stream clustering approach that is applicable in two perspectives. It can cluster evolving data stream with various density distributions and also uniform density distribution.

- The aim of this research is improving quality for multi-density data in which the clusters have different densities and there is no nested cluster. Although a slight modification of our approach (in its offline phase) can achieve nested clusters, the focus of this thesis is not on the nested clusters.

- There are different kinds of data attributes in data streams including numerical, categorical, and uncertain data. The focus of this thesis is only on numerical (continuous) data. The majority of test sets have only numeric attributes.

- The focus is not on the high dimensional data streams. In high dimensional data, the computation time is not low since we have a lot of empty grids to process. For clustering high dimensional data, the proposed method needs some improvements in its online phase.

1.7 Thesis Outline

The rest of the thesis is structured as follows:

- Chapter 2 presents a brief and rather general overview on different clustering methods on data stream. Furthermore, it has a comprehensive review on the existing density-based methods for clustering data stream. This chapter reviews the existing multi density-based clusterings for static datasets as well.

- Chapter 3 describes the research methodology applied to achieve the research objectives. It has a general overview of Multi Density-based clustering algorithm for evolving data Streams (MuDi-Stream) and a brief review on its components.

- Chapter 4 presents the proposed density-based clustering framework. It describes all the new notations applies in the proposed method. The components of the new method are explained in details. Most of the research questions are answered in this chapter.

- Chapter 5 presents the experimental setup including: the datasets used for evaluation of the proposed method, and implementation and environment are discussed in details. Moreover, the chapter describes the experimental evaluation of the proposed method on different datasets, evaluation measures, and the performance. Finally, the parameter settings are discussed, and the final results are reported.

- Chapter 6 concludes the study, reviews the main contributions of the thesis and outlines the possibilities for future work.

CHAPTER 2

RELATED WORK

2.1 Overview

In this chapter, we review the existing density-based clustering for data as well as multi-density algorithms for datasets. Section 2.2 overviews main clustering methods for data streams. Furthermore, basics in data stream clustering as well as challenges in clustering data streams are presented in Section 2.2. Section 2.3 introduces two main categories of density-based data stream clustering algorithms. Section 2.4 elaborates in details one of the well-known density-based clustering methods called DBSCAN. The first category is density micro-clustering algorithm and is discussed in Section 2.5. Density grid-based clustering algorithms are introduced in Section 2.6. Density-based clustering algorithms for multi-density dataset are explained in Section 2.7. Section 2.8 addresses an important issue of clustering process regarding the quality assessment of the clustering results. Section 2.9 is a comprehensive discussion on density-based data stream clustering algorithms as well as multi-density algorithms.

2.2 Clustering Data Streams

Clustering is a key data mining task (Aggarwal, 2007; OCallaghan, Meyerson, Motwani, Mishra, & Guha, 2002; Barbará, 2002; Guha, Meyerson, Mishra, Motwani, & O'Callaghan, 2003; Aggarwal, Han, Wang, & Yu, 2003; Ackermann et al., 2010) which classifies a given dataset into groups (clusters) such that the data points in a cluster are more similar to each other rather than the points in different clusters.

Unlike clustering static datasets, clustering data streams poses many new challenges. Data stream comes continuously and the amount of data is unbounded. Therefore, it is

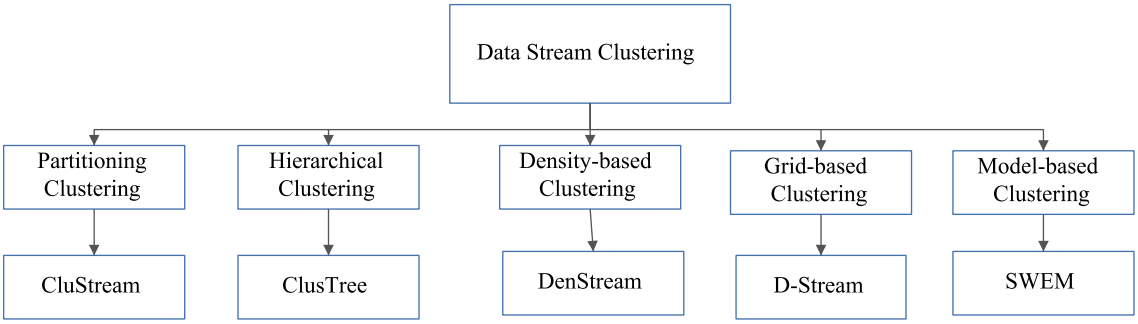


Figure 2.1: Data Stream Clustering Algorithms

impossible to keep the entire data stream in main memory. Data stream passes only once, so multiple scans are infeasible. Moreover, data stream requires fast and real time processing to keep up with the high rate of data arrival and mining results are expected to be available within short response time.

There is an extensive number of clustering algorithms for static datasets (Jain & Dubes, 1988) and (Jain, 2010) where some of them have been extended for data streams. Generally, clustering methods are classified into five major categories (Han & Kamber, 2006): partitioning, hierarchical, density-based, grid-based, and model-based methods (Figure 2.1).

A **partitioning-based clustering** algorithm organizes the objects into some number of partitions, where each partition represents a cluster. The clusters are formed based on a distance function like k-means algorithm (MacQueen, 1967; Lloyd, 1982) which leads to finding only spherical clusters and the clustering results are usually influenced by noise. Two of the well-known extensions of k-means on data streams are presented in (Guha, Mishra, Motwani, & O'Callaghan, 2000) where k-means algorithm clusters the entire data stream and in STREAM (OĆallaghan et al., 2002; Guha et al., 2003) which has LOCALSERACH algorithm based on K-median for data streams. Aggarwal et al. proposed an algorithm called CluStream (Aggarwal et al., 2003) based on k-means for clustering evolving data streams. CluStream introduces online-offline framework for clustering data streams which has been adopted for the majority of data stream clustering

algorithms.

A **hierarchical clustering** method groups the given data into a tree of clusters which is useful for data summarization and visualization. In hierarchical clustering once a step (merge or split) is done, it can never be undone. However, methods for improving the quality of hierarchical clustering have been proposed such as integrating hierarchical clustering with other clustering techniques, resulting in multiple-phase clustering such as BIRCH (T. Zhang, Ramakrishnan, & Livny, 1996) and Chameleon (Karypis, Han, & Kumar, 1999). BIRCH is extended for data stream as micro-cluster in (Aggarwal et al., 2003). Furthermore, ClusTree (Kranen, Assent, Baldauf, & Seidl, 2011) is a hierarchical index for maintaining cluster feature. In fact, ClusTree builds a hierarchy of micro-clusters at different levels.

Grid-based clustering is independent of distribution of data objects. In fact, it partitions the data space into a number of cells which forms the grids. Grid-based clustering has fast processing time since it is not dependent on the number of data objects. Some examples of the grid-based approach include STING (Wang, Yang, & Muntz, 1997), which explores statistical information stored in the grid cells; WaveCluster (Sheikholeslami, Chatterjee, & Zhang, 2000), which clusters objects using a wavelet transform method; and CLIQUE (Agrawal, Gehrke, Gunopulos, & Raghavan, 1998), which represents a grid-based and density-based approach. Grid-based methods are integrated with density-based methods for clustering data streams which are referred to as density grid-based. In density grid-based clustering methods data points are mapped into the grids. Then, the grids are clustered based on their density. Some of the density grid-based clustering algorithms are D-Stream (Y. Chen & Tu, 2007; Tu & Chen, 2009) and MR-Stream (Wan, Ng, Dang, Yu, & Zhang, 2009).

Model-based clustering methods attempt to optimize the fit between the given data and some mathematical models like EM (Expectation Maximization) algorithm (Demp-

ster, Laird, & Rubin, 1977). EM algorithm can be viewed as an extension of the k-means. However, EM assigns the objects to a cluster based on a weight representing the membership probability. In (Dang, Lee, Ng, Ciptadi, & Ong, 2009), SWEM (clustering data streams in a time-based Sliding Window with Expectation Maximization technique) is proposed which is a clustering data stream using EM algorithm.

Density-based methods have been developed based on the notion of density. The clusters are formed as dense areas which are separated from sparse regions. The main idea is to continuously grow a given cluster as long as the density (number of objects or data points) in the neighborhood exceeds some threshold. Such a method can be used to filter out noise or outliers and to discover clusters of arbitrary shape. The main density-based algorithms include: 1) DBSCAN (Ester, Kriegel, Sander, & Xu, 1996) which grows clusters according to a density-based connectivity analysis, 2) OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999) which extends DBSCAN to produce a cluster ordering obtained from a wide range of parameter settings, 3) DENCLUE (Hinneburg & Keim, 1998) which clusters objects based on a set of density distribution functions. Extensions of density-based algorithms for data stream are proposed as well.

2.2.1 Basics in Clustering Data Streams

In clustering data streams an important issue is how to process the infinite data which is evolving over time or how to keep the huge amount of data for later processing. There are some methods such as processing in one-pass, evolving and in online-offline manner as well as different methods for summarization of data streams. A short description of these methods is described as follows.

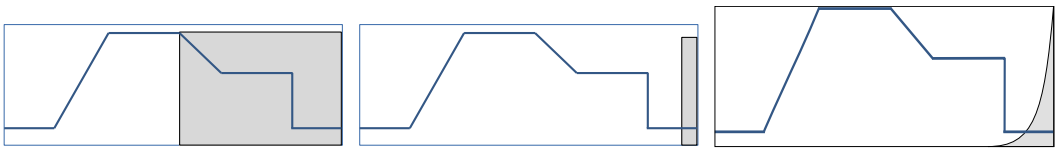
1. **Processing**

   One pass: In the one-pass, data streams are clustered by scanning data streams only once with the assumption that data objects arriving in chunks like k-means was

extended to be used for data streams (Charikar, O'Callaghan, & Panigrahy, 2003; Guha et al., 2000, 2003). Another well-known algorithm is STREAM (OĆallaghan et al., 2002; Guha et al., 2003), which partitions the input stream into chunks and computes (for each chunk) a cluster using a local search algorithm from (Guha et al., 2000). DUC-Stream (Gao, Li, Zhang, & Tan, 2005) is a one-pass grid-based clustering algorithm which assumes the arrival of data in chunks.

Evolving: In the one-pass approaches the clusters are computed over the entire data streams; however, data streams are infinite and they continuously evolve with time. Hence, the clustering results may change considerably over time. In the evolving approaches, the behaviors of streams are considered as an evolving process over time and processed in different forms of window model. Different clustering algorithms such as (Aggarwal et al., 2003; Wan et al., 2009; Zhou et al., 2008; Aggarwal, Han, Wang, & Yu, 2004, 2005; Babcock, Datar, Motwani, & O'Callaghan, 2003; Tu & Chen, 2009) are developed based on this approach. In window model, the data is separated into several basic windows and these basic windows are used as updating units. Three kinds of window models are as follows (Ng & Dash, 2010):

- Landmark window model: the window is determined by a specific time point called landmark and the present. It is used for mining over the entire history of the data streams (Figure 2.2a).
- Sliding window model: data is considered from a certain range in the past to a present time. The idea behind "sliding window" is to perform detailed analysis over both the most recent data points, and the summarized version of the old ones (Figure 2.2b).
- Fading (Damped) window model: a weight is given for each data point based on a fading function (Aggarwal et al., 2004), and more weights are given to



(a) Landmark Window Model (b) Sliding Window Model (c) Fading Window Model

Figure 2.2: Window Models (Matysiak, 2012)

Table 2.1: Window Models in Clustering Data Streams

Window Model	Definition	Pros	Cons	Example(s)
Landmark window model	Analyze *the entire history* of data stream	Suitable for one-pass clustering algorithms	All the data are equally important and the amount of data inside the window would quickly grow to unprocessable sizes	(Guha et al., 2003)
Sliding window model	Analyze *the most recent* data points	Suitable for applications where interest exists only in the most recent information like stock marketing	Ignores part of streams	(Zhou et al., 2008; Ren, Ma, & Ren, 2009)
Fading (damped) window model	Assign *different weights* to data points	Suitable for applications where old data has an effect on the mining results, the effect decreases as time goes on diminishes the effect of the old data	Unbounded time window (the window captures all historical data, its size keeps growing as time elapses)	(Cao et al., 2006; Y. Chen & Tu, 2007; Wan et al., 2009)

recent data compared to outdated data. The use of a damped window model is to diminish the effect of the old data on the mining result (Figure 2.2c).

The summarization of the window models with some example of clustering algorithm as well as their pros and cons are presented in Table 2.1. All the models have been considered in clustering data streams. Choice of the window model depends on the applications' needs (Ng & Dash, 2010).

Online-offline: sometimes a data stream clustering algorithm needs to investigate the clusters over different parts of stream. A different window model is used for tracing evolving behavior of data streams. However, we cannot perform dynamic

clustering over all possible time horizons of data streams. Therefore, online-offline approach is introduced by Aggarwal et al. in (Aggarwal et al., 2003). The online component keeps summary information (overcoming real-time and memory constraints) about fast data streams and offline component gives an understanding of the clusters. The majority of data stream clustering developed for evolving data streams use CluStream's two-phase framework (Aggarwal et al., 2003; Wan et al., 2009; Zhou et al., 2008; Aggarwal et al., 2004, 2005; Y. Chen & Tu, 2007; Tu & Chen, 2009).

2. **Summarization:** The large volume of data streams put space and time constraints on the computation process. Data streams are massive and infinite, so it is impossible to record the entire data. Therefore, synopsis information can be constructed from data items in the streams. The design and choice of a particular synopsis method depends on the problem being solved. A brief description about different methods of summarization is as follows (Han & Kamber, 2006; Aggarwal, 2007):

   Sampling methods: instead of recording the entire data streams which seems impossible, we can make a sampling from data stream. Reservoir sampling (Vitter, 1985) is a technique which is used to select an unbiased random sample of data streams and it is useful for data streams.

   Histograms: histogram based methods are used for static datasets; however, their extension for data streams is a challenging task. Some of the methods are discussed in (M. Garofalakis, Gehrke, & Rastogi, 2002) for data streams. One of the recent algorithms, called SWClustering (Zhou et al., 2008), keeps summary information of data streams in the form of histogram.

   Wavelets: wavelets are popular multi-resolution techniques for data streams' summarization. Wavelets are traditionally used for image and signal processing. They

are used for multi-resolution hierarchy structures over an input signal, in this case, the stream data. Furthermore, wavelet-based histograms can be dynamically kept over time (Aggarwal & Yu, 2007; M. N. Garofalakis, 2009; Gilbert, Kotidis, Muthukrishnan, & Strauss, 2003).

Sketches: sketch is a probabilistic summary technique for analyzing data streams. Sketch-based methods can be considered as a randomized version of wavelets technique. While other methods emphasis on small part of data, sketches summarize the entire dataset at multiple levels of details (Aggarwal, 2007).

Micro-cluster: micro-cluster (Aggarwal et al., 2003) is a method to keep statistical information about the data locality. It can adjust well with evolution of the underlying data streams. We will elaborate on the micro-cluster further in Section 2.5.

Grid: in this method, the data space is partitioned into some small segments called grids and the data points in streams are mapped to them. Each grid has a characteristic vector which keeps a summary about all the data points mapped to it (Y. Chen & Tu, 2007). More details of the grid method in Section 2.6.

According to the reviewed papers, the most applicable summarization methods for density-based clustering algorithms are micro-clustering and grid-based. Therefore, we categorize the reviewed algorithms based on these two summarization methods (Amini, Ying Wah, & Saboohi, 2014; Aggarwal & Reddy, 2013).

2.2.2 Challenges in Clustering Data Streams

Considering their dynamic behavior, clustering over data streams should address the following challenges (Guha et al., 2000; Kranen et al., 2011; Wan et al., 2009; Han & Kamber, 2006; J. Gama & (Eds), 2007; Han et al., 2011; Aggarwal & Reddy, 2013):

- **Handling noisy data**: any clustering algorithm must be able to deal with random noises present in the data since outliers have great influence on the formation of clusters.

- **Handling evolving data**: the algorithm has to consider that the data streams considerably evolve over time.

- **Limited time**: data streams arrive continuously, which requires fast and real-time response. Therefore, the clustering algorithm needs to handle the speed of data streams in the limited time.

- **Limited memory**: the huge amount of data streams are generated rapidly, which needs an unlimited memory. However, the clustering algorithm must operate within memory constraints.

- **Handling high dimensional data**: some of data streams are high dimensional in their nature such as gene expression or clustering text documents. Therefore, the clustering algorithm has to overcome this challenge in case of its data being high dimensional.

We will discuss how different density-based clustering algorithms over data streams address aforementioned challenges in Section 2.9.1.

2.3 Density-based Data Stream Clustering

Based on a comprehensive review on existing density-based clustering algorithms on data stream, these algorithms are categorized in two broad groups called density micro-clustering algorithms and density grid-based clustering algorithms (Amini et al., 2014; Aggarwal & Reddy, 2013) (Figure 2.3).

In density micro-clustering algorithms, micro-cluster keeps summary information about data and clustering is performed on these synopsis information. The reviewed al-

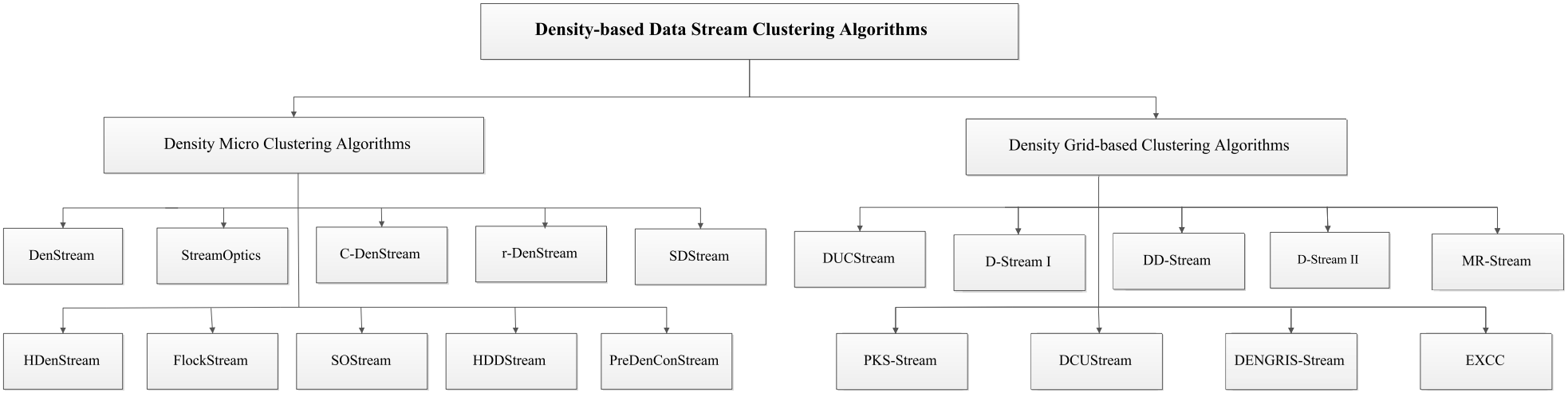


Figure 2.3: Density-based Data Stream Clustering Algorithms' Categorization

gorithms in this category include: DenStream (Cao et al., 2006), StreamOptics (Tasoulis et al., 2007), C-DenStream (Ruiz, Menasalvas, & Spiliopoulou, 2009), r-DenStream (Lixiong, Jing, Yun-fei, & Hai, 2009), SDStream (Ren et al., 2009), HDenStream (Lin & Lin, 2009), FlockStream (Forestiero et al., 2013), SOStream (Isaksson et al., 2012), HDDStream (Ntoutsi, Zimek, Palpanas, Kröger, & Kriegel, 2012), and PreDeConStream (Hassani, Spaus, Gaber, & Seidl, 2012).

In density-grid based clustering algorithms group, the data space is divided into grids, data points are mapped to these grids, and the clustering are formed based of the density of grids. The reviewed algorithms in this category include: DUC-Stream (Gao et al., 2005), D-Stream I (Y. Chen & Tu, 2007), DD-Stream (Jia, Tan, & Yong, 2008), D-Stream II (Tu & Chen, 2009), MR-Stream (Wan et al., 2009), PKS-Stream (Ren, Cai, & Hu, 2011), DCUStream (Y. Yang, Liu, Zhang, & Yang, 2012), DENGRIS-Stream (Amini & Ying Wah, 2012), and ExCC (Bhatnagar, Kaur, & Chakravarthy, 2013).

In the following sections, firstly, DBSCAN (Ester et al., 1998), a remarkable density-based clustering algorithm, is elaborated. Furthermore, we will discuss in details about the algorithms in each aforementioned category, and their pros and cons. Additionally, we examine how they address the challenging issues in clustering data streams.

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

Density-based clustering has the abilities to discover arbitrary-shape clusters and to handle noises. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) is one of the density-based algorithms, which is adopted for data stream algorithms, described in details as follows.

DBSCAN is developed for clustering large spatial databases with noise, based on connected regions with high density. The density of each point is defined based on the number of points close to that particular point called point's neighborhood. The

dense neighborhood is defined based on two user-specified parameters: the radius (ε) of the neighborhood (ε-neighborhood), and the number of the objects in the neighborhood (*MinPts*). The basic definitions in DBSCAN are introduced in the following. D is a current set of data points.

- ε-neighborhood of a point: the neighborhood within a radius of ε. Neighborhood of a point p is denoted by $N_\varepsilon(p)$:

$$N_\varepsilon(p) = \{q \in D | dist(p,q) \leq \varepsilon\}, |N_\varepsilon(p)| >= MinPts \tag{2.1}$$

  where $dis(p,q)$ denotes the Euclidean distance between points p and q.

- *MinPts*: minimum number of points around a data point in the ε-neighborhood

- Core Point: a point which its cardinality of ε-neighborhood is at least *MinPts*

- Border Point: a point which the cardinality of its ε-neighborhood is less than *MinPts* and at least one of its ε-neighborhood is a core point

- Noise Point: a point which the cardinality of its ε-neighborhood is less than *MinPts* and no core point is in the ε-neighborhood of that point

- Directly density reachable: a point p is directly density reachable from point q, if p is in the ε-neighborhood of q and q is a core point

- Density reachable: a point p is density reachable from point q, if p is in the ε-neighborhood of q and q is not a core point but they are reachable through chains of directly density reachable points

- Density-connected: if two points p and q are density-reachable from a core point o, p and q are density-connected

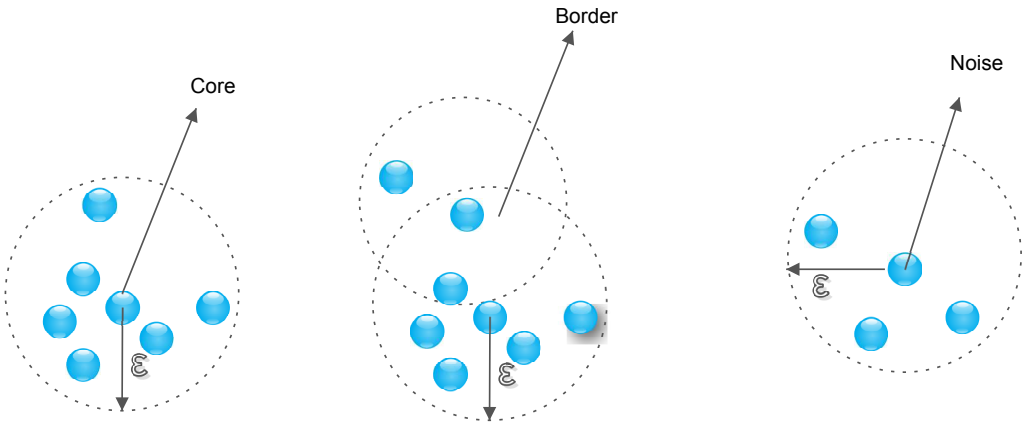


Figure 2.4: DBSCAN: Core, Border, and Noise Points

Algorithm 1 DBSCAN(D, $MinPts$, ε)

Input: a data set D
Output: arbitrary shape clusters

```
1:  for each data point p in D do
2:    if p is not mark as 'seen' then
3:      mark p as 'seen'
4:      Find N_ε(p,D) /* find ε neighborhood of data point p */
5:      if |N_ε(p,D)| <= MinPts then
6:        mark data point: ClusterId=noise
7:      else
8:        ClusterId=ClusterId+1
9:      end if
10:     for all q ∈ N_ε(p,D) do
11:       mark data point q as 'seen'
12:       find N_ε(q,D)
13:       if |N_ε(q,D)| > MinPts then
14:         give data point q a ClusterId
15:       end if
16:     end for
17:   end if
18: end for
```

- A cluster: a maximal set of density-connected points

Core, border and noise points are shown in Figure 2.4.

DBSCAN starts by randomly selecting a point and checking whether the ε-neighborhood of the point contains at least $MinPts$ points. If not, it is considered as a noise point, otherwise the point is considered as a core point and a new cluster is created. DBSCAN iteratively adds the data points, which do not belong to any cluster and directly density reachable (Ester et al., 1996) from the core points of a new cluster. If

the new cluster can no longer be expanded, the new cluster is completed. In order to find the next cluster, DBSCAN randomly selects an unvisited data point and the clustering process continues until all the points are visited and no new point is added to any cluster. The overall architecture of DBSCAN algorithm is outlined in Algorithm 1.

Therefore, a density-based cluster is a set of density-connected data objects with respect to density reachability. The points that are not placed in any cluster are considered as noise. Figure 2.5 shows DBSCAN algorithm performing on a small synthetic dataset. Figures 2.5a, 2.5b, and 2.5c are the steps of the clustering and Figure 2.5d is the final clustering results.

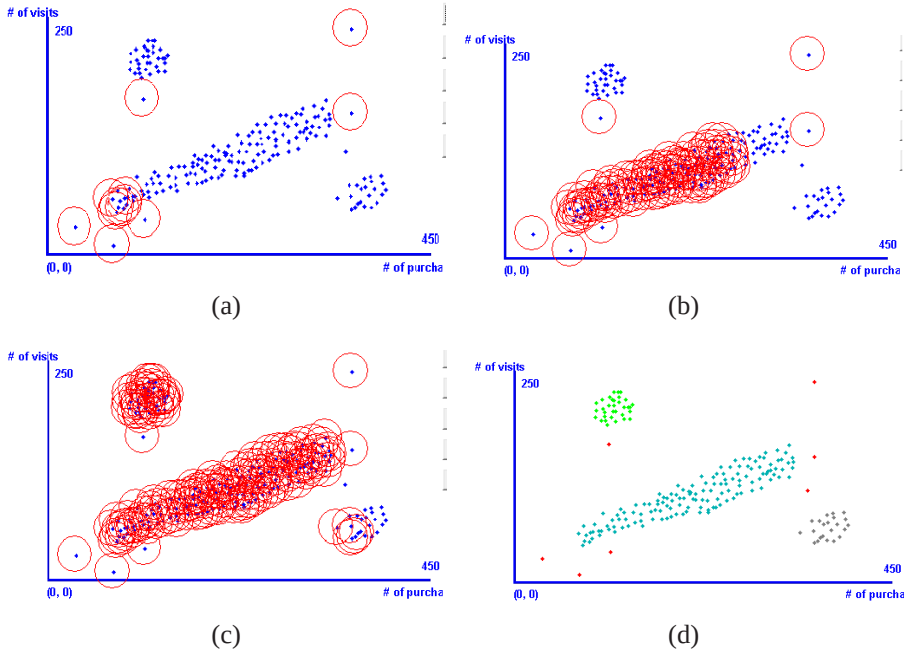
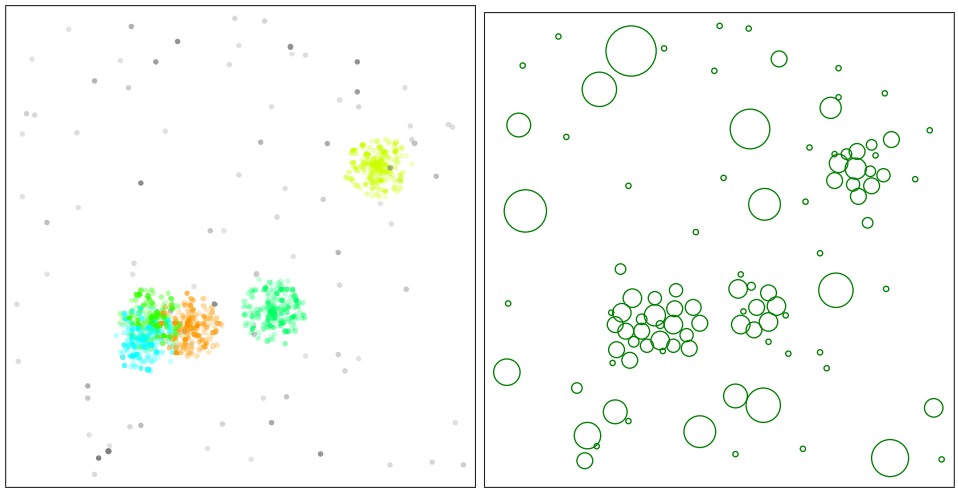


Figure 2.5: DBSCAN algorithm on synthetic data set: $\varepsilon = 20$, $MinPts = 5$

2.5 Density Micro-Clustering Algorithms on Data Streams

Micro-clustering is a remarkable method in stream clustering to compress data streams effectively and to record the temporal locality of data (Aggarwal, 2007). The micro-cluster concept was first proposed in (T. Zhang et al., 1996) for large datasets, and subsequently adapted in (Aggarwal et al., 2003) for data streams. The micro-cluster concept is described as follows (Figure 2.6):



(a) (b)

Figure 2.6: Micro-Clusters in density-based clustering generated by MOA

Micro-Cluster is a temporal extension of cluster feature (CF) (T. Zhang et al., 1996), that is a summarization triple maintained about a cluster. The triple vector comprises the number of data points, the linear sum of data points, and their squared sum. Therefore, a micro-cluster for a set of *d-dimensional* points $p_{i1} \dots p_{in}$ is defined as the $(2.d+3)$ tuple $(\overrightarrow{CF2^x}, \overrightarrow{CF1^x}, CF2^t, CF1^t, n)$.

- $\overrightarrow{CF2^x}$: for each dimension, the sum of squares of data values is maintained in $CF2^x$. Therefore, the *p-th* entry of $CF2^x$ is equal to $\sum_{j=1}^{n} = \left(x_{ij}{}^p\right)^2$.
- $\overrightarrow{CF1^x}$: for each dimension, the sum of the data values is maintained in $\overrightarrow{CF1^x}$. Therefore, the *p-th* entry of $CF1^x$ is equal to $\sum_{j=1}^{n} = x_{ij}{}^p$.
- $CF2^t$: sum of squares of timestamps $T_{i1} \dots T_{in}$.
- $CF1^t$: sum of timestamps $T_{i1} \dots T_{in}$.
- n: number of data points.

The micro-cluster for a set of points C is denoted by $CFT(C)$.

Micro-clustering method uses micro-cluster to save summary information about the data streams, and performs the clustering on these micro-clusters.

2.5.1 DenStream

Feng et al. in (Cao et al., 2006) proposed a clustering algorithm, termed as DenStream, for evolving data stream, which has the ability to handle noises as well. The algorithm extends the micro-cluster concept as core micro-cluster, potential micro-cluster, and outlier micro-cluster in order to distinguish between real data and outliers. The core-micro-cluster synopsis is designed to summarize the clusters with arbitrary shape in data streams. Potential and outlier micro-clusters are kept in separate memories since they need different processing. DenStream is based on the online-offline framework. In the online phase it keeps micro-clusters with real data and removes micro-clusters with noises. In the offline phase density-based clustering is performed on the potential micro-clusters which have the real data.

DenStream extends the micro-cluster concepts to core-micro-cluster, potential-micro-cluster, and outlier-micro-clusters (Figure 2.7) which are described for a group of close points $p_{i1} \dots p_{in}$ with timestamps $T_{i1} \dots T_{in}$, as follows:

Core-micro-cluster: is defined as $CMC(w, c, r)$.

- $w = \sum_{j=1}^{n} = f(t - T_{ij})$, is the weight and $w \geq \mu$
- $c = \frac{\sum_{j=1}^{n} f(t-T_{ij}) p_{ij}}{w}$ is the center
- $r = \frac{\sum_{j=1}^{n} f(t-T_{ij}) dist(p_{ij}, c)}{w}$, $r \leq \varepsilon$ is the radius. $dist(p_{ij}, c)$ is *Euclidean distance* between point p_{ij} and the center c.

  Note that the weight of a micro-cluster must be above a predefined threshold μ in order to be considered as a core.

Potential micro-cluster: at time t is defined as $(\overrightarrow{CF^1}, \overrightarrow{CF^2}, w)$.

- $w = \sum_{j=1}^{n} f(t - T_{ij})$, is the weight and $w \geq \beta\mu$. β is the parameter to determine the threshold of the outlier relative to *c-micro-clusters* ($0 < \beta < 1$).

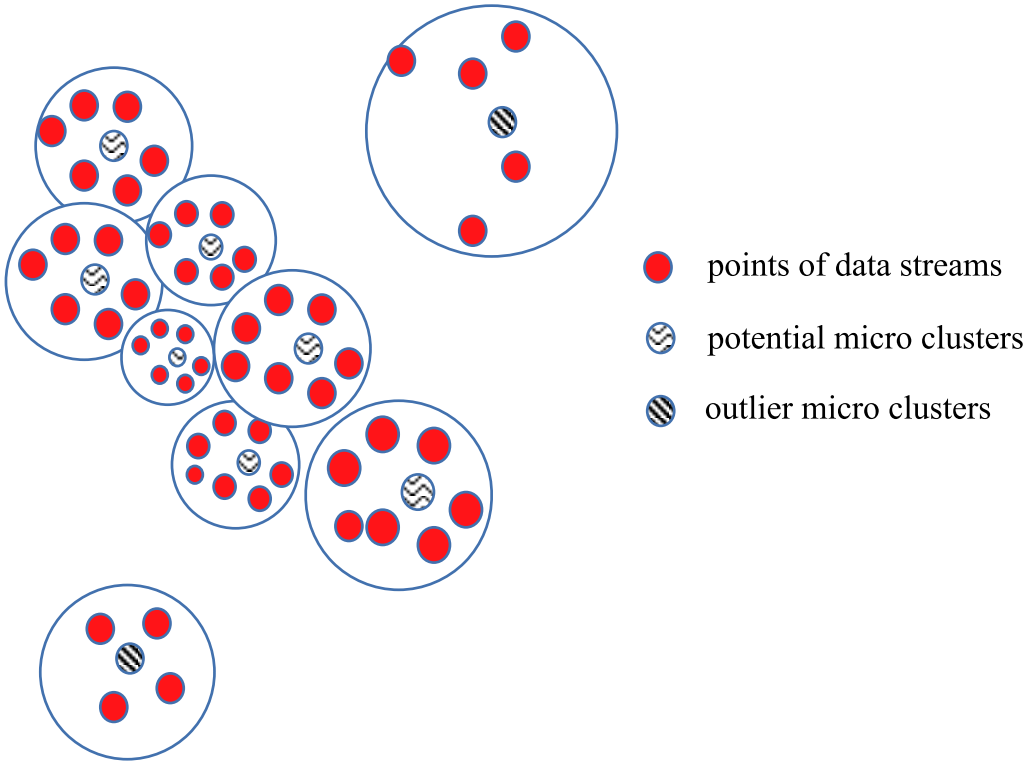


Figure 2.7: Potential and Outlier Microclusters

- $\overrightarrow{CF^1} = \sum_{j=1}^{n} f(t - T_{ij}) p_{ij}$, is the weighted linear sum of the points.
- $\overrightarrow{CF^2} = \sum_{j=1}^{n} f(t - T_{ij}) p_{ij}^2$, is the weighted squared sum of the points.

The center of potential micro-cluster is $c = \frac{\overrightarrow{CF^1}}{w}$. And the radius of potential micro-cluster is $r = \sqrt{\frac{|\overrightarrow{CF^2}|}{w} - (\frac{|\overrightarrow{CF^1}|}{w})^2}$ $(r \leq \varepsilon)$.

Outlier micro-cluster: is defined as $(\overrightarrow{CF^1}, \overrightarrow{CF^2}, w, t_0)$. The definition of w, CF^1, CF^2, center, and radius are the same as in the potential-micro-cluster. $t_0 = T_{i1}$ denotes the creation time of the outlier micro-cluster. In an outlier micro-cluster the weight w must be below the fixed threshold, thus $w < \beta\mu$. However, it could grow into a potential micro-cluster when, by adding new points, its weight exceeds the threshold.

Weights of micro-clusters are periodically calculated and decision about removing or keeping them is made based on the weight threshold.

Online Phase: For initialization of the online phase, DenStream uses the DBSCAN algorithm on the first initial points, and forms the initial potential micro-clusters. In fact,

for each data point, if the aggregate of the weights of the data points in the neighborhood radius is above the weight threshold, then a potential micro-cluster is created. When a new data point arrives, it is added to either the nearest existing potential micro-cluster or outlier micro-cluster. The Euclidean distance between the new data point and the center of the nearest potential or outlier micro-cluster is measured. A micro-cluster is chosen with the distance less than or equal to the radius threshold. If it does not belong to any of them, a new outlier micro-cluster is created and it is placed in the outlier buffer.

Offline phase: adopts DBSCAN to determine the final clusters on the recorded potential micro-clusters.

DenStream has a pruning method in which it frequently checks the weights of the outlier-micro-clusters in the outlier buffer to guarantee the recognition of the real outliers. The algorithm defines a density threshold function which calculates the lower limit of density threshold. If the outlier micro-cluster weighs below the lower limit, it is a real outlier and it can be omitted from the outlier buffer.

Merits and limitations: DenStream handles the evolving data stream effectively by recognizing the potential clusters from the real outliers. DenStream creates a new micro-cluster if the arriving records are incorporated into existing micro-clusters. However, the algorithm does not occupy any memory space for the new micro-cluster by either deleting a micro-cluster or merging two old micro-clusters. Furthermore, the storage for the new micro-cluster is repeatedly allocated until it is eliminated in the pruning phase. Nevertheless, the pruning phase for removing outliers is a time consuming process in the algorithm. The merging time is also time consuming since it uses two lists for keeping potential and outlier micro clusters.

2.5.2 StreamOptics

In (Tasoulis et al., 2007), Tasoulis et al. developed a streaming cluster framework which graphically represents the cluster structure of data stream. It addresses visualization challenges in clustering data streams. The algorithm is called StreamOptics that extends the OPTICS (Ordering Points To Identify the Clustering Structure) algorithm (Ankerst et al., 1999) for data streams using micro-cluster concept. Core-distance and reachability distance from OPTICS algorithm are changed in the form of micro-cluster as follows.

Definition 7 ***(Micro-Cluster core-distance).*** *Micro-Cluster core-distance is defined to be equal to micro-cluster radius. In OPTICS, core distance for a data point is defined as the smallest of ε (neighboring radius) that makes a data point as a core object. If data point is not a core object, its core-distance is undefined.*

Definition 8 ***(Reachability-distance).*** *The reachability-distance is the same as OPTICS. Reachability-distance of an object p_1 with respect to another object p_2 is chosen based on the maximum value between Euclidean distance of p_1, p_2 and the core distance of p_2. If p_2 is not a core object, the reachability-distance between $p1$ and p_2 is undefined. However, in StreamOptics the distance is calculated between the potential micro-clusters. Reachability-distance between micro-cluster mc_1 and mc_2 is chosen based on the maximum value between Euclidean distance of mc_1 and mc_2 and the core distance of mc_2. If mc_2 is not a core object, the reachability-distance between mc_1 and mc_2 is undefined.*

StreamOptics also uses potential micro-cluster and outlier micro-cluster from DenStream. StreamOptics keeps an ordered list from potential micro-clusters and discards outlier micro-clusters. Therefore, micro-cluster neighborhood and cluster ordering is defined based on the potential micro-clusters as follows.

Definition 9 ***(Micro-cluster neighborhood).*** *Micro-cluster neighborhood is defined based on the Euclidean distance between two potential micro-clusters.*

Definition 10 ***(Cluster Ordering).*** *Cluster ordering orders the potential micro-clusters based on their reachability distance.*

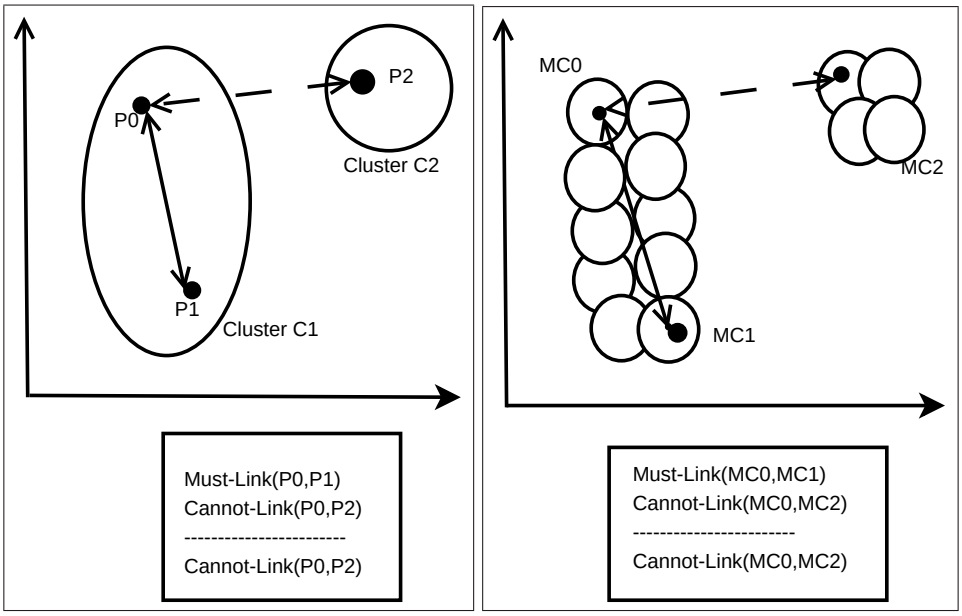
In StreamOptics, firstly the neighborhoods of each potential micro-cluster is determined, and an ordered list of potential micro-clusters are made based on their reachability distance. StreamOptics produces a reachability plot that represents the micro-cluster structure using OPTICS algorithm.

Since data streams are changed by time, in StreamOptics, time is considered as the third dimension which is added to the two dimensional plots of OPTICS. The StreamOptics plot allows the user to recognize the changes in cluster structure in terms of emerging and fading clusters.

Merits and limitations: StreamOptics is based on micro-clustering framework, which uses OPTICS algorithm to provide the three dimensional plot that shows the evolution of the cluster structure over the time. However, it is not a supervised method for cluster extraction; it needs manual checking of the generated three dimension plot.

2.5.3 C-DenStream

Ruiz et al. in (Ruiz et al., 2009) developed a density-based clustering algorithm with constraints for data streams. The algorithm is referred to as C-DenStream, which extends the concept of instance-level constraints from static data to stream data. Instance-level constraints are a particular form of background knowledge, which refer to the instances that must belong to the same cluster (Must-Link constraints) and those that must be assigned to different clusters (Cannot-Link constraints) (Ruiz et al., 2009). In C-DenStream, instance level constraints are converted to potential micro-clusters level con-



(a) (b)

Figure 2.8: Micro-cluster Constraint

straint (Figure 2.8) and final clusters are generated on the potential micro-clusters using C-DBSCAN (Ruiz, Spiliopoulou, & Menasalvas, 2007).

Merits and limitations: C-DenStream includes domain information in the form of constraints by adding the constraints to the micro-clusters. The algorithm is very useful in the applications which have a priori knowledge on the group membership of some records. It prevents the formation of the clusters which is included in the applications' semantics. However, the algorithm needs an expert to define its constraints. Moreover, the algorithm has DenStream limitations as well.

2.5.4 rDenStream (DenStream with retrospect)

In (Li-xiong et al., 2009), the authors developed a density-based clustering algorithm for applications with a large amount of outliers. The algorithm is a three-step clustering algorithm based on DenStream, which is referred to as rDenStream (DenStream with retrospect). rDenStream improves the accuracy of the clustering algorithm by forming a classifier from the clustering result. In the retrospect step of the algorithm, the mis-interpreted discarded data points get a new chance to be re-learned and to improve the

robustness of the clustering.

In rDenStream the potential and the outlier micro-clusters are determined like DenStream. However, instead of discarding the outlier micro-cluster, which can not be converted to a potential-micro-cluster or to satisfy the density requirements, they are placed in a historical outlier buffer. In retrospect phase, final clusters from performing DBSCAN on potential micro-clusters, are used to form a classifier. This classifier is applied to re-learn the outlier micro-cluster in the historical outlier buffer. In this phase, the micro-clusters which were chosen wrongly as outliers are modified to improve the clustering accuracy.

Merits and limitations: rDenStream is useful for extracting knowledge pattern from the initial arriving data streams. However, the memory usage and the time complexity is high since it retains and processes the historical buffer. rDenStream is only applicable in the applications with a large amount of outliers, which are worthwhile to spend time and memory to gain better accuracy. The space complexity of rDenStream is similar to DenStream; however, it needs extra memory for keeping the historical outlier buffer.

2.5.5 SDStream

The *SDStream* algorithm (Ren et al., 2009) has the ability to discover the clusters with arbitrary shapes over sliding window (Ng & Dash, 2010). In the algorithm, the distribution of the most recent data stream is considered and the data points that are not accommodated in sliding window length are discarded. It uses potential and outlier micro-clusters; however, they are stored in the form of exponential histogram. It is also an offline-online phase algorithm.

In the online phase, the new data points are added to the nearest micro-cluster. The nearest micro cluster is either potential-micro-cluster or outlier-micro-cluster, if the new radius of micro-cluster is less than or equal to the threshold radius. Otherwise, a new micro-cluster is created. Since the number of micro-clusters are limited, either a micro-

cluster has to be deleted or two clusters be merged. For deleting a micro-cluster, the outdated micro-cluster is chosen according to its time value: if the time value does not belong to the length of sliding window. In merging case, the two nearest micro-clusters, which are density-reachable (Ester et al., 1996), are merged (Zhou et al., 2008). In the offline phase, the final clusters of arbitrary shape are generated on potential micro-clusters using a modified DBSCAN.

Merits and limitations: SDStream uses the sliding window model, processing the most recent data and summarizing the old data. In the real applications, users are interested in the distribution characteristics of the most recent data points. The authors did not clarify the main usage of exponential histogram for their algorithm.

2.5.6 HDenStream

HDenStream (Lin & Lin, 2009) is a density-based clustering over evolving heterogeneous data stream. It adopts potential and outlier micro-cluster concepts from DenStream algorithm and uses the method for measuring distance in case of categorical data from HCluStream (C. Yang & Zhou, 2006). HDenStream adds another entry to potential and outlier micro-cluster concept which is a two dimensional array keeping the frequency of categorical data. In fact, for measuring distance between two micro-clusters with categorical data, the distance between two categorical attributes and continuous attributes are calculated separately. The algorithm has online and offline phases and the pruning phase is similar to DenStream as well.

Merits and limitations: The algorithm can cover categorical and continuous data which makes it more useful since in the real-world applications, we may have numerical, categorical, and continuous data. However, the algorithm does not discuss how to save categorical features in an efficient way for data stream environment.

2.5.7 SOStream

SOStream (Self Organizing density-based clustering over data Stream) (Isaksson et al., 2012) detects structure within fast evolving data streams by automatically adapting the threshold for density-based clustering. The algorithm has only online phase in which all mergings and updatings are performed. SOStream uses competitive learning as introduced for SOMs (Self Organizing Maps) (Kohonen, 1982) where a winner influences its immediate neighborhood. When a new data point arrives a winner cluster is defined based on Euclidean distance of existing micro-clusters. If the calculated distance is less than a dynamically defined threshold, the micro-cluster is considered as a winner micro-cluster and the new data point will be added to it. It also affects the micro-cluster neighbors of the winner cluster. The neighbors are defined based on *MinPts* parameters of DBSCAN algorithm. The algorithm finds all the clusters overlapping with the winner. For each overlapping cluster its distance to the winning cluster is calculated. Any cluster with a distance less than that of the merge-threshold will be merged with the winner. If the new point is not added to any existing micro-cluster, a new micro-cluster is created for it. SOStream dynamically creates, merges, and removes clusters in an online manner.

Merits and limitations: SOStream is a density-based clustering algorithm that can adapt its threshold to the data stream. SOM is a time consuming method which is not suitable for clustering data streams. SOStream is a micro-cluster based algorithm; however, it compares its result with two grid based methods.

2.5.8 HDDStream

HDDStream (Ntoutsi et al., 2012) is a density-based algorithm for clustering high dimensional data streams. It has online and offline phases. The online phase keeps summarization of both points and dimensions and the offline phase generates the final clusters based on a projected clustering algorithm called PreDeCon (Bohm, Kailing, Kriegel, &

Kroger, 2004). The algorithm uses DenStream concepts; however, it introduces prefer vector for each micro-cluster which is related to prefer dimension in high dimensional data. A prefer dimension is defined based on variance along this dimension in micro-cluster. A micro-cluster prefers a dimension if data points of micro-clusters are more dense along this dimension. The micro-cluster with preferred vector is called a projected micro-cluster. Projected term shows that the micro-cluster is based on a subspace of feature space and not the whole feature space. Based on this concept, the algorithm changes the potential and outlier micro-clusters to projected potential micro-clusters and projected outlier micro-clusters respectively. HDDStream has pruning time similar to DenStream in which the weights of the micro-clusters are periodically checked.

Merits and limitations: HDDStream can cluster high dimensional data stream; however, in the pruning time it only checks micro-cluster weights. Since the micro-cluster fades over time the prefer vector should be checked as well because it may change over time.

2.5.9 PreDeConStream

PreDeConStream (Hassani et al., 2012) is similar to HDDStream; however, PreDeConStream improves the efficiency of the HDDStream by working on the offline phase. This algorithm also introduces a subspace prefer vector which is defined based on the variance of micro-clusters and their neighbors. The algorithm keeps two lists including potential and outlier micro-clusters.

In the pruning time, the neighbors of newly inserted potential micro-clusters are checked as well as deleted potential micro-clusters. The subspace prefer vector of these neighboring micro-clusters are updated and put in a list as affected micro-clusters. The affected micro-cluster list is used in the offline phase as expanding clusters to improve the efficiency of the offline phase.

Merits and limitations: The algorithm can cluster high-dimensional data stream based on the density method. However, searching the affected neighboring clusters is a time consuming process.

2.5.10 FlockStream

FlockStream (Forestiero et al., 2013) is a density-based clustering algorithm based on a bio-inspired model. It is based on flocking model (Kennedy, Kennedy, & Eberhart, 2001) in which agents are micro-clusters and they work independently but form clusters together. It considers an agent for each data point which is mapped in the virtual space. Agents move in their predefined visibility range for a fixed time, if they visit another agent, they join to form a cluster in case they are similar to each other. It merges online and offline phases since the agents form the clusters at any time. In fact, it does not need to perform offline clustering to get the clustering results.

Since, FlockStream only compares each new point with the other agents in its agent visibility distance, it reduces the number of comparisons in the neighborhood of each point. The visibility distance has a threshold which is defined by the users. The agents have some rules in order to move in the virtual space such as cohesion, separation and alignment (Forestiero et al., 2013). These rules are executed for each agent over the time. FlockStream has three kinds of agents: basic representative agents for new data point and p-representative, and o-representative agents which are based on potential- and outlier-micro-clusters respectively. Actually, when the similar basic agents merge to each other, they form a p-representative or an o-representative agent based on their weights.

Merits and limitations: FlockStream is a single pass clustering algorithm which merges online and offline phases of clustering. It also reduces the number of comparisons in clustering. FlockStream uses a flocking model which generates clustering results any time without performing frequently offline phase. However, the flocking model is based

on moving agent in data space which makes the execution time dependent on the number of agents. A heuristic search on these agents is time consuming which leads to high time complexity of FlockStream. Although, the algorithm forms an outlier agent to handle noise, there is not any clear strategy to show when and how to remove the outliers from the agents list.

Table 2.2 summarizes some of the main characteristics of the reviewed density micro-clustering algorithms.

Table 2.2: Main Characteristics of Density Micro-clustering Algorithms

Name	Year	Type of data	Input parameters	Results	Objective
DenStream (Cao et al., 2006)	2006	Continuous	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	clustering evolving data streams
StreamOptics (Tasoulis et al., 2007)	2007	Continuous	potential micro-cluster list, core distance, reachability distance	cluster structure plot over time	cluster visualization
C-DenStream (Ruiz et al., 2009)	2009	Continuous	cluster radius, minimum number of points in the neighborhood, outlier radius, decay factor, a stream of instance level constraint	arbitrary shape clusters with constraint	applying constraint in clustering
rDenStream (Li-xiong et al., 2009)	2009	Continuous	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	improving accuracy
SDStream (Ren et al., 2009)	2009	Continuous	sliding window size, cluster radius, cluster weight	arbitrary shape clusters over sliding window	clustering over sliding window
HDenstream (Lin & Lin, 2009)	2009	Continuous, Categorical	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	improve quality
SOSStream (Isaksson et al., 2012)	2012	Continuous	cluster radius	clustering threshold	Automate clustering threshold selection
HDDStream (Ntoutsi et al., 2012)	2012	Continuous	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	clustering high dimensional data
PreDeConStream (Hassani et al., 2012)	2012	Continuous	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	clustering high dimensional data
FlockStream (Forestiero et al., 2013)	2013	Continuous	cluster radius, cluster weight, outlier threshold, decay factor	arbitrary shape clusters	density-based clustering using flocking model

2.6 Density Grid-based Clustering Algorithms on Data Streams

Using density-based and grid-based methods, researchers developed several hybrid clustering algorithms for data streams referred to as density grid-based clustering algorithms (Y. Chen & Tu, 2007; Wan et al., 2009; Tu & Chen, 2009). In these algorithms, data space is partitioned into small segments called grids. Each data point in data streams is mapped into the grid and then grids are clustered based on their density. Density grid-based algorithms not only can discover arbitrary shape clusters and detect the outliers, but also have fast processing time which only depends on the number of cells (Figure 2.9).

According to the reviewed algorithms, there are some definitions, which form the basis of the density grid-based algorithms. In these algorithms, the data space is partitioned into density grids and each data point $x = \{x_1, x_2, \cdots, x_d\}$ is mapped to a density grid $g(x)$. Based on these assumptions the following concepts are described:

- Density coefficient: for each data point, a density coefficient is considered to capture the dynamic changes of the clusters. The density of each grid is associated with a decay factor, which is decreased over time. In fact, the grids are processed in the form of fading window model.

- Grid density: the density of each grid is defined based on the aggregation of density coefficients of all the data points in that grid (Y. Chen & Tu, 2007). However, in an algorithm called DUC-Stream (Gao et al., 2005), the density of the grid is defined based on its number of data points.

- Dense, sparse and transitional grid: density grid-based algorithms consider a threshold for the density of each grid. This density threshold categorizes the grid as dense, sparse, and transitional. A grid is considered as dense if its density is higher than a special threshold. If the grid density is lower than another special threshold,

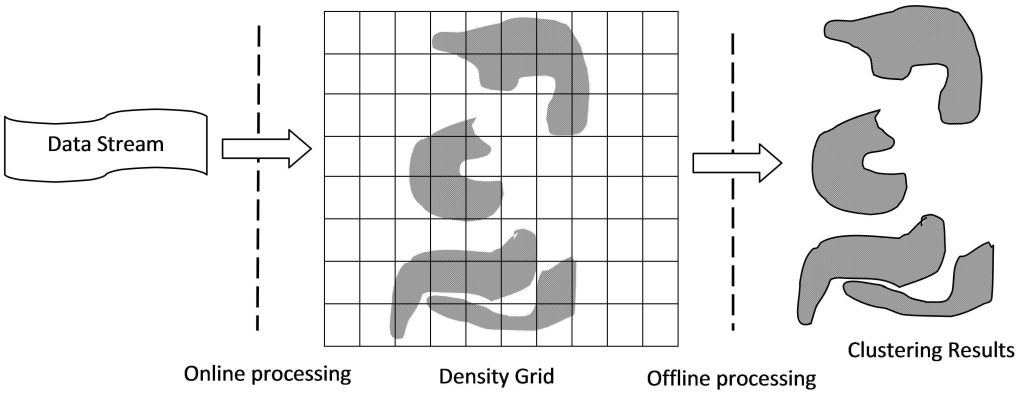


Figure 2.9: Density-Grid based clustering Framework

the grid is a sparse grid. The grid with density between the dense and sparse density thresholds is considered as a transitional grid.

- Characteristic vector: keeps some information about the data points, which are mapped to the grid, such as grid density, update time, creation time, and grid type.
- Grid cluster: a group of dense neighboring grids, which has higher density than the surrounding grids, form a grid cluster (Y. Chen & Tu, 2007).

In the following sections, we explain the density grid-based algorithms in details and we discuss their pros and cons.

2.6.1 DUC-Stream

Gao et al. (Gao et al., 2005) have proposed an incremental single pass clustering algorithm for data streams using dense unit, which is referred to as DUC-Stream. DUC-Stream assumes the arrival of data in chunks which contain some points. The density of each unit is its number of points and if it is higher than a density threshold, it is considered as a dense unit. The algorithm introduces the local dense unit in order to keep only the units, which are most probably converted to dense unit. In DUC-Stream, the clusters are identified as a connected component of a graph in which the vertices show the dense units and edges show their relation. Therefore, when a dense unit is added, if there is no related

cluster, a new cluster is created; otherwise, the new dense unit is absorbed to the existing clusters.

Furthermore, DUC-Stream keeps the clustering results in bits, which is called clustering bits, to retain little amount of memory. The clustering bit is a bit string, which keeps the number of dense units. In fact, the clustering result is created in an incremental manner. The time complexity and the memory space of the DUC-Stream is claimed to be low due to utilizing the bitwise clustering.

Merits and limitations: DUC-Stream checks the density of each unit. If the unit does not receive enough data points over time, its density is decreased so it is not considered for clustering. Since DUC-Stream processes the data in chunks, it relies on the user to determine the size of the chunks of data.

2.6.2 D-Stream I

Chen et al. (Y. Chen & Tu, 2007) proposed a density-based clustering framework for clustering data streams in the real time which is termed as D-Stream I. D-Stream I has online and offline phases. The online phase reads a new data point, maps it into the grid, and updates the characteristic vector of the grid.

The offline phase adjusts the clusters in each time interval gap. The time interval gap is defined based on the minimum conversion time of different kinds of grids. In the first time interval, each dense grid is assigned to a distinct cluster. After that, in each time interval, clusters are adjusted by determining dense and sparse grids. A threshold is considered for the grid density. If the grid density is higher than the special threshold, it is a dense grid otherwise is considered as a sparse grid. If the grid is dense, it is merged with neighboring grids with higher density and forms a cluster. Otherwise, if it is sparse, the grid is removed from the cluster. In fact, D-Stream I firstly updates the density of the grids and then performs the clustering based on a standard method of density-based

clustering.

An important motivation behind this framework is handling the outliers by considering them as sporadic grids. Sporadic grid is a kind of sparse grid, which has very few data and does not have any chance to be converted to a dense grid. D-Stream I defines a lower limit for density threshold based on density threshold function. If a sparse grid density is less than the lower limit of density threshold, it is considered as a sporadic grid. It has also a pruning phase which happens in each time interval gap. In this phase, the clusters are adjusted and the sporadic grids are removed from the grid list. D-Stream I uses a hash table for keeping the grid list.

Merits and limitations: D-Stream I clusters data streams in real time based on the grid and the density. It also proposes a density decaying to adjust the clusters in real time and captures the evolving behavior of data streams and has techniques for handling the outliers. However, for determining the time interval gap, the algorithm considers the minimum time for a dense grid to be converted to sparse and vice versa. Therefore, the gap depends on many parameters. In fact, it could be better that the algorithm would define the time gap based on only the conversion of dense to sparse grids, since the conversion of sparse to dense grid has already been considered in the weight of the grid. Furthermore, it cannot handle the high dimensional data because it assumes that the majority of the grids are empty in the high-dimensional situation.

2.6.3 DD-Stream

DD-Stream algorithm (Jia et al., 2008) is an extension of D-Stream I, which improves the cluster quality by detecting the border points in the grids. The boundary points are extracted before performing any adjustment on the grids. The online phase performs merely like D-Stream I. The offline component runs in each time interval gap (defined like D-Stream I) and extracts boundary points, detects dense and sparse grids and clus-

ters the dense grids using density-based methods. DD-Stream assigns the points on the borders based on their distance from the center of the neighboring grids. If the distances are equal, the neighboring grid with higher density is chosen. The information about the center of the grids is kept in the characteristic vector of the grid.

Merits and limitations: DD-Stream extracts the boundary points from the grids to improve the quality of the clustering. However, the border points are extracted whenever the data is mapped to the grids which is a time consuming process. It is better to detect the border point in each time interval gap before merging the grids rather than arrival time of the data points. Furthermore, the algorithm recognizes the sparse and dense grids based on their density, but it does not have any clear strategy for removing the sporadic grids.

2.6.4 D-Stream II

Tu et al. (Tu & Chen, 2009) proposed an algorithm for clustering data streams based on grid density and attraction. The algorithm is based on the observation that many density-based clustering algorithms do not consider the positional information of data in the grid. The idea is based on using grid attraction for the grids. Grid attraction (Tu & Chen, 2009) shows that to what extent the data in one neighbor is closer to another neighbor. In fact, the algorithm is an extension of D-Stream I, and we refer to it as D-Stream II. The clustering procedure of D-Stream II is similar to D-Stream I; however, in D-Stream II, two dense grids are merged in case that they are strongly correlated. Two girds are called strongly correlated if their grid attractions are higher than a pre-defined threshold. D-Stream II has pruning techniques, like D-Stream I, to adjust the clusters in each time interval gap and to remove the sporadic grids mapped by the outliers.

Merits and limitations: D-Stream II improves the quality of clustering to some extent by considering the position of the data in the grids for clustering. However, the algorithm still has the problems that are already mentioned in D-Stream I. Nevertheless,

it keeps the grid list in a tree rather than a table which makes the processing of the grid list faster and it reduces the memory space.

2.6.5 MR-Stream

Li Wan et al. (Wan et al., 2009) developed an algorithm for density-based clustering of data streams at multiple resolutions, termed as MR-Stream. The algorithm improves the performance of density-based data stream clustering algorithm by running the offline component at constant times. The algorithm determines the right time for the users to generate the clusters.

MR-Stream partitions the data space in cells and a tree-like data structure which keeps the space partitioning. Each time a dimension is divided in two, a cell can be further divided in 2^d where d is the data set dimensionality. The tree data structure keeps the data clustering in different resolutions. Each node has the summary information about its parent and children.

MR-Stream has online and offline phases. In the online phase, when a new data point is arrived, it is mapped to its related grid cell. In the tree structure, if there is not any sub-node, a new sub-node is created for the new data point, and updates parent's weights up to the root of the tree. In each time interval gap, the tree is pruned in two ways: from the root to maximum height and vise versa. In pruning from leaf to root, the sparse grids are detected and density of dense grids are added to their parents. In the pruning from root to the maximum height, the dense grids are detected and sparse grids are merged to form noise clusters. The sporadic grid cell is also removed by comparing its density with lower limit of density threshold function.

The offline phase, generates clusters at a user defined height. It determines all the reachable dense cells at a special distance and marks them as one cluster. The noise clusters are removed by checking their size and density with size and density thresholds

respectively.

The authors of MR-Stream proposed a memory sampling method to recognize the right time to trigger the offline component. In this method, the algorithm makes a relation between nodes in the tree and evolution of clusters.

Merits and limitations: MR-Stream introduces a memory sampling method in order to define the right time for running the offline component, which improves the performance of the clustering. However, MR-Stream keeps the sparse grids and merges them for consideration as a noise cluster. It is better not to let the noise cluster to be formed by checking the density of the sparse grids. Furthermore, the algorithm cannot work properly in high dimensional data.

2.6.6 PKS-Stream

Ren et al. in (Ren et al., 2011) proposed an algorithm for clustering data streams based on the grid density for high dimensional data streams referred to as PKS-Stream. The algorithm is based on the observation that in grid based clustering, there are a lot of empty cells specially for the high dimensional data. The idea is based on using pks-tree for recording non-empty grids and their relations as well. For keeping the non-empty cells, PKS-Stream introduces the k-cover grid cell concept. A grid is a k-cover, if it has the minimum density threshold and it is not covered by any other grid. In fact, k-cover shows the non-empty grids in the neighboring of the leaf node grids.

PKS-Stream has online and offline phases. The online phase maps the data records to the related grid cells in the pks-tree, if there is a grid cell for the data record. Otherwise, a new grid cell is created. The offline phase forms the clusters based on the dense neighboring grids. In each time interval gap, the pks-tree is adjusted and the sparse grids are removed from the tree.

Merits and limitations: PKS-Stream is a density grid-based clustering, which han-

dles the high dimensional data stream. However, it does not have any pruning on the tree after adding a new data point to any of the cells of the tree. PKS-Stream depends on K, which affects the clustering result. It also affects the k-cover, which defines the resolution of cluster.

2.6.7 DCUStream

DCUStream (Y. Yang et al., 2012) is a density-based clustering algorithm over uncertain data stream. For each data point in the stream a tuple which includes data point, existence probability of the data point and its arrival time are considered. Each data point is mapped into the grid. The algorithm considers an uncertain tense weight for each data point which is calculated based on temporal feature of data stream and its existence probability. By aggregation of uncertain tense weight, the algorithm defines the uncertain data density. DCUStream introduces the core dense grid which is a dense grid with sparse neighbors. By considering threshold for uncertain data density, dense and sparse grid are defined. For clustering, DCUStream examines all the grids to find core dense grid. It uses depth first search algorithm to find neighbor grids. The process continues for all unlabeled dense grids. All sparse grids are considered as noise.

Merits and limitations: DCUStream algorithm improves density-based clustering algorithm for uncertain data stream environment. However, searching the core dense grids and finding their neighbors are time consuming processes.

2.6.8 DENGRIS-Stream

DENGRIS-Stream (Amini & Ying Wah, 2012) is a density grid-based clustering for stream data over sliding window. The algorithm maps each input data into a grid, computes the density of each grid, and clusters the grids using density concepts within time window units. DENGRIS-Stream can capture the distribution of recent records precisely using sliding window model which is more preferable in data stream applications.

It introduced the expired grid concept for detecting and removing the grids which their time stamps are not in the sliding window. Furthermore, DENGRIS-Stream removes the expired grids before any processing on the grid list which leads to save time and memory.

Merits and limitations: DENGRIS-Stream is the first density grid-based clustering algorithm for evolving data streams over sliding window model. However, there is no evaluation to show its effectiveness in compare to other state-of-the-art algorithms.

2.6.9 ExCC

ExCC (Exclusive and Complete Clustering) (Bhatnagar et al., 2013) is an exclusive and complete clustering algorithm for heterogeneous data stream. It is an online-offline algorithm. Online phase keeps synopsis in the grids and offline phase forms the final clusters on demand. The algorithm maps the numerical attributes to the grid and the categorical attributes are assigned granularities according to distinct values in respective domain sets. ExCC is a complete algorithm since it uses pruning based on the speed of data stream not a window model such as fading one. ExCC introduces fast or slow stream based on the average arrival time of the data points in the data stream. Furthermore, it is an exclusive clustering algorithm since it uses grid for the distribution of data. The algorithm detects noise in the offline phase using wait and watch policy. For detecting real outliers, it keeps the data points in the hold queue which is kept separately for each dimension. ExCC uses a user specified threshold for detecting dense and sparse grids. ExCC can filter out noise using cell density and cluster density threshold which is specified by user. However, the algorithm estimates the threshold based on the granularity of the grid, the data dimension, and the average number of points in each grid. In order to generate the clusters, it considers a pool for dense and recent grids. The dense neighboring grids are chosen from this pool by considering eight neighboring of each grid. For categorical data the equality of the attributes are considered.

Merits and limitations: ExCC can cover data stream with mix attributes (numeric and categorical). Furthermore, the algorithm compares the results with micro-clustering methods. However, since it is a grid-based algorithm the results have to be compared with grid-based algorithms to be fair. The hold queue strategy needs more memory and processing time since it is defined for each dimension. Moreover, using pool for keeping dense grids requires more memory to keep and more time to process.

We summarize the main characteristics of the density grid-based clustering algorithms in Table 2.3.

Table 2.3: Main Characteristics of Density Grid-based Clustering Algorithms

Name	Year	Type of data	Input parameters	Results	Objective
DUC-Stream (Gao et al., 2005)	2005	undefined	chunks of data streams	clusters as the connected components of the graph	one-scan clustering algorithm
D-Stream I (Y. Chen & Tu, 2007)	2007	Continuous	data stream, decay factor, dense grid threshold, sparse grid threshold	arbitrary shape clusters	real-time clustering
DD-Stream (Jia et al., 2008)	2008	Continuous	data stream, decay factor, dense grid threshold, sparse grid threshold	arbitrary shape clusters	improving quality
D-Stream II (Tu & Chen, 2009)	2009	Continuous	data stream, decay factor, dense grid threshold, sparse grid threshold	arbitrary shape clusters	improving quality
MR-Stream (Wan et al., 2009)	2009	Continuous	data stream, decay factor, dense cell threshold, sparse cell threshold	clusters in multiple resolutions	improving performance
PKS-Stream (Ren et al., 2011)	2011	Continuous	pks-tree, density threshold	arbitrary shape clusters	clustering high dimensional data
DCUStream (Y. Yang et al., 2012)	2012	Continuous	data stream dimension, density threshold	arbitrary shape clusters	clustering uncertain data
DENGRIS-Stream (Amini & Ying Wah, 2012)	2012	Continuous	data stream, sliding window size	arbitrary shape clusters	clustering over sliding window
ExCC (Bhatnagar et al., 2013)	2013	Continuous, Categorical	grid granularity	arbitrary shape clusters	clustering heterogeneous data streams

2.7 Density-based Clustering Algorithms for Multi-Density Dataset

In this section, existing density-based clustering algorithms for multi density dataset are introduced and discussed in details.

2.7.1 GMDBSCAN

GMDBSCAN (Xiaoyun, Yufang, Yan, & Ping, 2008) is a multi-density clustering algorithm which uses a grid technique to define multi-density clusters. The algorithm determines local MinPts parameters using grid-density and the grids are clustered applying DBSCAN by related local MinPts. The clusters are integrated based on density similarity. Furthermore, GMDBSCAN uses distance-based method to adjust boundaries. The algorithm also uses a special kind of tree structure, which keeps the positional information of grids, and results in the faster finding of the neighborhoods. It is a two-pass clustering; hence, not applicable for data streams.

2.7.2 MSDBSCAN

MSDBSCAN (Esfandani & Abolhassani, 2010) is a density-based clustering algorithm which localizes the concept of core points in DBSCAN based on the position of their neighbors. It introduces a new definition for core points called local core distance (lcd) which represents the distance in which there is at least MinPts objects. MSDBSCAN calculates the lcd for all data points and then the data point is a core point if the values of its lcd vector is similar. The algorithm constructs the large clusters by merging the core points. The time complexity of MSDBSCAN is high which makes it inapplicable for data streams.

2.7.3 Multi-DBSCAN

Multi-DBSCAN (Huang, Yu, Li, & Zeng, 2009) covers the problems of multi-density datasets by determining values for ε. It uses “must link constraint” and “k-th”

nearest distance to calculate the ε values for different densities. Multi-DBSCAN chooses the best ε for each density distribution using an outlier detection algorithm. After that, DBSCAN is performed on the dataset using the calculated ε. Multi-DBSCAN improves DBSCAN in multi-density datasets by determining different ε values for various density distributions.

2.7.4 Multi Level

In (X. Li, Ye, Li, & Ng, 2010), a hierarchical clustering algorithm is developed to cluster nested and multi-density clusters. It uses a multi-level approach to detect hierarchical clustered structures in datasets. Agglomerative k-means is used to create a cluster tree for both nested clusters and clusters with different densities. The algorithm's procedure is as follows: firstly, the agglomerative k-means algorithm is used to discover the number of clusters which are generated in this level. Then, a cluster validation technique is used to identify the atomic and composite clusters. The atomic clusters are the clusters that do not need to be divided any more. For the other clusters, the agglomerative k-means is applied to further partitioning. These kind of clusters are named as composite clusters. The process is repeated to generate a tree of clusters. Using agglomerative k-means helps to determine atomic and composite clusters in each level. The validation metrics are based on compactness of the clusters which are measured based on the scattering of a cluster. Scattering value is determined based on the goodness of fit for Gaussian multi-model clusters. If the cluster scattering is large or small, the node in the cluster tree is considered as a composite or an atomic cluster respectively. It is also a two-pass clustering algorithm.

2.7.5 IS-DBSCAN

IS-DBSCAN (Carmelo et al., 2013) is a method which is proposed to improve DBSCAN algorithm in terms of reduced number of parameters and the ability to cluster

multi-density data. The method proposed a new concept named as space ranking which ranks the data points in the space based on density metrics. IS-DBSCAN uses Influence Space (IS) concept which has been introduced in INFLO (INFLuenced Outlierness) (Jin, Tung, Han, & Wang, 2006). IS gives a better estimation of the neighborhoods' density distribution and improves separation of clusters with different densities. IS uses both the nearest neighbors (NNs) and reverse nearest neighbors (RNNs). Though, the ranking method is based on a linear combination of INFLO and kNN (k-Nearest Neighbor) distances. After ranking the data density, IS-DBSCAN adds one more dimension to the original data, whose value for each point represents the sum of distances of IS. The number of input parameters are reduced by providing a proper way to set them. The method establishes a cluster around a point until a border point or an outlier is reached. A border point is recognized by checking the size of IS. When the algorithm reaches a point p whose size of $IS(p)$ is below a predefined threshold, the subset is not processed and the point is detected as noise. In fact, in this method, the traditional way of finding ε-neighborhood is replaced by an approach using the advantages of influence space (IS) for density-based clustering.

2.7.6 SCDM2

SCDM2 (X. Chen et al., 2012) is an extension of SCDM (Y.-Q. Yu, Huang, Guo, & Li, 2008) which is a semi-supervised clustering algorithm. SCDM2 is developed for multi-density data, and uses constraints to guide the clustering process. It improves the SCDM by adding 2 more steps to it. In SCDM, DBSCAN's ε parameter is calculated based on "must link constraint". Therefore, if one cluster has no must link constraint, this cluster may not appear in the final clustering results. SCDM2 algorithm has five phases: 1) getting referenced ε with must link sets, 2) adding referenced ε, 3) selecting representative ε by applying cannot-link sets, 4) multi-stage DBSCAN clustering by using

illustrative ε, and 5) assigning boundary clusters. In SCDM the list of referenced ε is only made from must link constraint; however, in SCDM2 the list of cannot-link is also added to reference and then the representative ε.

2.7.7 DBSCAN-DLP

DBSCAN-DLP (Multi-density DBSCAN based on Density Levels Partitioning) (Xiong, Chen, Zhang, & Zhang, 2012) determines the parameters for each cluster in order to automatically discover the clusters with various densities using density level partitioning. In this method, firstly, a dataset is divided into different density levels based on statistical information of its density variation. Then, the ε is defined for each density level. In the last step of the algorithm, DBSCAN is adopted to perform clustering on each density level with its related ε to get clustering result. Statistical information of density variation is calculated based on the k-nearest neighbor distance, which is a distance between the data point *p* and its k-th nearest neighbors. K-neighborhood density is determined based on the k-nearest neighbor distance; hence, the smaller k distance the denser the cluster. Density Level Set (DLS) consists of points whose densities are almost similar. DBSCAN-DLP algorithm generalizes the traditional DBSCAN to find the clusters with different densities through density level partitioning. DBSCAN-DLP is a two-pass clustering algorithm which has high computation time to be applicable for data streams.

2.7.8 GDCLU

GDCLU (Esfandani, Sayyadi, & Namadchian, 2012) is a density-based clustering algorithm based on the grid method. It proposed a definition for grid density based on the density of their neighbors. The algorithm is also scale independent. The algorithm is based on local density and neighboring density. This method also uses a similar radius in forming micro clusters.

2.7.9 DSCLU

DSCLU (Namadchian & Esfandani, 2012) is an algorithm for density-based clustering of data streams. It has online-offline phases. The algorithm improves the offline phase to be applicable for clustering data stream in multi-density environments. The algorithm uses the micro-cluster method for density-based clustering. DSCLU determines the dominant micro-clusters based on neighbors' weights. These micro-clusters are dense and their densities are similar to density of their dense neighbors. In the offline phase, the clustering is performed on these dominant micro-clusters.

2.8 Clustering Evaluation Metrics

One of the important issues in clustering algorithms is evaluating (validating) the goodness of the clustering results that the evaluation measures. A multitude of evaluation metrics were introduced in the literature for measuring cluster quality. Evaluation quality metrics can be categorized into two main classes, internal and external measures. The main difference is whether or not the external information is used for the cluster evaluation (Kremer et al., 2011). Some of the internal and the external evaluation measures are: C-index (L. J. Hubert & Levin, 1976), sum of squared distance (SSQ) (Han & Kamber, 2006), silhouette coefficient (Kaufman & Rousseeuw, 2005), Rand Index (Wu, Xiong, & Chen, 2009; Rand, 1971), purity (Zhao & Karypis, 2004), van Dongen (Dongen & Dongen, 2000), B Cubed precision (Han et al., 2011), V-measure (Rosenberg & Hirschberg, 2007), variation of information (Meilă, 2005), F-measure (Rijsbergen, 1979), precision (Rijsbergen, 1979), and recall (Rijsbergen, 1979). A list is available in (Milligan, 1981) for the internal and the external measures.

When the ground truth is available, it can be compared with a clustering to evaluate the clustering results. In this thesis, a number of external quality metrics is used such as Purity, Normalized Mutual Information (NMI), Entropy, Rand Index, Adjusted Rand

Index, Fowlkes–Mallows index (FM), Jaccard Score, F-measure to evaluate the quality of the proposed method.

However, when the ground truth is not available, the internal index is used. In this study, silhouette coefficient is applied as an internal quality evaluation. There are different internal quality metrics such as SSQ; however, they need cluster center which are more applicable for spherical shapes clusters. In fact, SSQ measures how closely related are the objects in the cluster. It defines the compactness of the spherical clusters in convex approaches. Therefore, the silhouette coefficient is selected which is not dependent on the clusters shapes.

The clustering evaluation metrics except Entropy have values ranging from 0 to 1, where 1 is related to the case when ground truth and finding clusters are identical. Therefore, the bigger criteria values are preferred. For Entropy the lower value shows better clustering result.

Each of the mentioned evaluation criteria has its own benefit and there is no consensus of which criterion is better than other criteria in the data mining community.

The most often evolution quality metrics used in the clustering data streams in the reviewed algorithms are purity, and NMI (Amini et al., 2014). However, in this thesis, a number of other metrics is used to get more fair results. The evaluation methods used in this thesis are elaborated in the following sections as shown in Figure 2.10.

2.8.1 External Metrics

There are different available metrics for clustering quality. The metrics have their own benefits and limitations. For example, purity and entropy are more towards small clusters since they reach a maximal value in case that all clusters are of size one. Combining precision and recall using balanced F-measure, on the other hand, favors coarser clusterings, and random clusterings do not receive zero values. Finally, according to

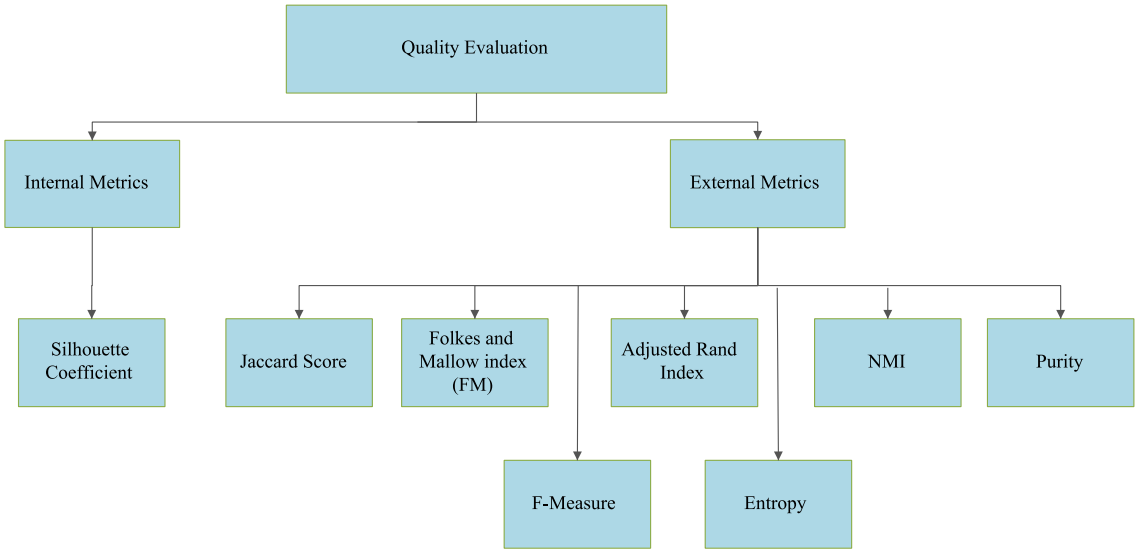


Figure 2.10: Quality Evaluation Metrics

Strehl's study (Strehl, 2002), Mutual Information (Strehl & Ghosh, 2003) has the best properties because it is unbiased and symmetric in terms of the cluster distribution. This kind of information is very helpful to determine which metric to be used in a specific clustering scenario (Amigó, Gonzalo, Artiles, & Verdejo, 2009).

There are several performance indices for cluster evaluation. Indices are measures of correspondence between two partitions of the same data and are based on how pairs of objects are classified in a contingency table. Consider $G = \{G_1, G_2, \ldots, G_i\}$ as ground truth clusters, and $C = \{C_1, C_2, \ldots, C_j\}$ as the clusters made by a clustering algorithm under evaluations. Table 2.4 can be formed to indicate group overlap between G and C.

In Table 2.4, an entry, n_{ij}, represents the number of data points in the class G_i and in the cluster C_j. b_j is the number of points in cluster j, a_i is the number of points in class i and n is the number of data points. From the total number of possible combinations of pairs $\binom{n}{2}$, four different pairs $(|TP|, |TN|, |FN|, and |FP|)$ can be represented as follows:

Let $|TP|$ (True Positive) be the number of data points in the same class G and same cluster C, therefore it is defined as follows:

$$|TP| = \sum_{ij} \binom{n_{ij}}{2} \tag{2.2}$$

Table 2.4: Contingency Table

Class \ Cluster	C_1	C_2	...	C_k	Sums
G_1	n_{11}	n_{12}	...	n_{1k}	a_1
G_2	n_{21}	n_{22}	...	n_{2k}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
G_j	n_{j1}	n_{j2}	...	n_{jk}	a_j
Sums	b_1	b_2	...	b_k	n

$|TN|$ (True Negative) is defined as the number of pairs of data points in the same class G but not the same cluster C. It is denoted as:

$$|TN| = \sum_i \binom{a_i}{2} - \sum_{ij} \binom{n_{ij}}{2} \tag{2.3}$$

Similarly, $|FP|$ (False Positive) is defined as the number of pairs of data points in the same cluster in C but not in the same class in G. It is written as:

$$|FP| = \sum_j \binom{b_j}{2} - \sum_{ij} \binom{n_{ij}}{2} \tag{2.4}$$

$|FN|$ (False Negative) is defined as the number of pairs of data points that are not in the same class in G and not the same cluster in C. Since $|TP|+|TN|+|FP|+|FN| = \binom{n}{2}$, therefore d is calculated as follows:

$$|FN| = \binom{n}{2} - |TP| - |TN| - |FP| \tag{2.5}$$

The external evaluation metrics are described as follows:

2.8.1 (a) Purity

The purity of each cluster is defined based on the class which is most frequent in it by dividing by number of data points in that cluster. Purity is defined as follows:

$$purity(C,G) = \sum_i \frac{a_i}{n}(max_j \frac{n_{ij}}{a_i}) \tag{2.6}$$

The purity in data stream is calculated only for the points arriving in a predefined window, since the weight of points diminishes continuously. Bad clusterings have purity values close to 0, and a perfect clustering has a purity of 1. However, high purity is easy to achieve when the number of clusters is large. Purity is used for evaluation in data stream clustering in various studies (Hassani et al., 2012; Y. Chen & Tu, 2007; Forestiero et al., 2013; Bhatnagar et al., 2013).

2.8.1 (b) Normalized Mutual Information (NMI)

The normalized mutual information (NMI) (Strehl & Ghosh, 2003) is a well known information theoretic measure that assesses how similar two clusterings are. In fact, in order to make trade-off between the quality of the clustering against the number of clusters. NMI is also applicable when the number of classes is different from clustering. The normalized mutual information $NMI(C,G)$ is defined as:

$$NMI(C,G) = \frac{I(C;G)}{[H(C)+H(G)]/2} \tag{2.7}$$

I is mutual information:

$$I(C,G) = \sum_i \sum_j \frac{n_{ij}}{n} \log \frac{n * n_{ij}}{a_i * b_j} \tag{2.8}$$

$$H(C) = -\sum_j \frac{b_j}{n} \log \frac{b_j}{n} \tag{2.9}$$

$$H(G) = -\sum_i \frac{a_i}{n} \log \frac{a_i}{n} \tag{2.10}$$

2.8.1 (c) Entropy

Entropy (Song & Zhang, 2008; Rohlf, 1974) measures the purity of the clusters with respect to the given class labels. Thus, if all clusters consist of objects with only a single class label, the entropy is 0. However, as the class labels of objects in a cluster become more varied, the entropy increases.

$$Entropy(C,G) = -\sum_i \frac{a_i}{n} \sum_j \frac{n_{ij}}{a_i} \log \frac{n_{ij}}{a_i} \tag{2.11}$$

2.8.1 (d) Rand Index

Rand index (Rand, 1971) is one of the popular indices and the most used one for clustering evaluation (Santos & Embrechts, 2009). It is also used for evaluating density-based data stream clustering (Ruiz, Spiliopoulou, & Menasalvas, 2010). It measures the agreement between two partitions, that is, how the clustering results are close to the ground truth.

$$RandIndex = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{2.12}$$

2.8.1 (e) Adjusted Rand Index (ARI)

ARI (L. Hubert & Arabie, 1985) is an improvement of RI. It is recommended as the index of choice for measuring agreement between two partitions in clustering analysis with different numbers of clusters.

$$AdjustedRandIndex = \frac{index - expectedindex}{maximumindex - expectedindex} \tag{2.13}$$

$$ARI = \frac{\binom{n}{2}(|TP| + |TN|) - [(|TP| + |FN|)(|TP| + |FP|) + (|Fp| + |TN|)(|FN| + |TN|)]}{\binom{n}{2}^2 - [(|TP| + |FN|)(|TP| + |FP|) + (|FP| + |TN|)(|FN| + |TN|)]} \tag{2.14}$$

2.8.1 (f) Fowlkes and Mallow index (FM):

Fowlkes–Mallows index (Fowlkes & Mallows, 1983) is used to determine the similarity between two clusterings (clusters obtained after a clustering algorithm). It is easily generalized to a measure for clusterings with different numbers of clusters. A higher value for the Fowlkes–Mallows index indicates a greater similarity between the clusters and the ground truth.

$$FM = \sqrt{\frac{|TP|}{|TP|+|FP|} \cdot \frac{|TP|}{|TP|+|FN|}} \tag{2.15}$$

2.8.1 (g) Jaccard Index

Jaccard index (Jaccard, 1901) is one of the external metrics that has been used in various studies as external index (Chaovalit, 2009; Papapetrou & Chen, 2011; Kremer et al., 2011). The Jaccard score is defined as:

$$Jaccard(C,G) = \sqrt{\frac{|TP|}{|TP|+|FN|+|FP|}} \tag{2.16}$$

2.8.1 (h) F-Measure

F-Measure (also F-score or F1 score) (Rijsbergen, 1979) considers both the precision and the recall to compute the score. Precision is the number of correct results divided by the number of all returned results and recall is the number of correct results divided by the number of results that should have been returned. The F-Measure can be considered as a weighted average of the precision and recall, where F-Measure reaches its best value at 1 and worst score at 0 (Meesuksabai, Kangkachit, & Waiyamai, 2011).

$$precision(C,G) = \sqrt{\frac{|TP|}{|TP|+|FP|}} \tag{2.17}$$

$$Recall(C,G) = \sqrt{\frac{|TP|}{|TP|+|TN|}} \tag{2.18}$$

$$F-measure(C,G) = \frac{2 * precision(C,G) * Recall(C,G)}{precision(C,G) + Recall(C,G)} \tag{2.19}$$

2.8.2 Internal Metrics

2.8.2 (a) Silhouette Coefficient

Silhouette coefficient (Rousseeuw, 1987; Brun et al., 2007) is used in measuring cluster quality when the ground truth of a dataset is not available. The silhouette coefficient is calculated as follows (Han et al., 2011):

For a data set, D, of n objects, suppose D is partitioned into k clusters, $C_1, \ldots, C_k$. For each object $o \in D$, we calculate $a(o)$ as the average distance between o and all other objects in the cluster to which o belongs. Similarly, $b(o)$ is the minimum average distance from o to all clusters to which o does not belong to. Formally, suppose $o \in C_i (1 \leq i \leq k)$; then

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} distance(o,o')}{|C_i - 1|} \tag{2.20}$$

$$b(o) = min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} distance(o,o')}{|C_j|} \right\} \tag{2.21}$$

According to 2.20 and 2.21, the silhouette coefficient of o is defined as:

$$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}} \tag{2.22}$$

The value of the silhouette coefficient is between -1 and 1. The value of $a(o)$ reflects the compactness of the cluster to which o belongs. The smaller the value, the more com-

pact the cluster. The value of $b(o)$ captures the degree to which o is separated from other clusters. The larger $b(o)$ is, the more separated o is from other clusters. Therefore, when the silhouette coefficient value of o approaches 1, the cluster containing o is compact and o is far away from other clusters, which is the preferable case. However, when the silhouette coefficient value is negative (i.e., $b(o) < a(o)$), this means that, in expectation, o is closer to the objects in another cluster than to the objects in the same cluster as o (Han et al., 2011).

Silhouette function is the ratio of the difference between the average inter-cluster and the average intra-cluster distances, to the maximum of the inter-cluster and intra-cluster distances (Kaufman & Rousseeuw, 2009). The positive Silhouette values approaching 1 indicate good cluster quality, and negative Silhouette values approaching -1 indicate incorrect cluster assignment. The Silhouette function measures the compactness and separation of the produced clusters.

The internal validation with indices such as Sum of SQuared Error (SSQ) and Silhouette was employed in evaluation, especially when there was a lack of ground truth. Among internal validity criteria, SSQ is based on the distance of every data point in a cluster to its centroid. Despite its popular usage in the evaluation of clustering results in the literature, SSQ favored spherical clusters, as opposed to arbitrarily-shaped clusters. In our experiments, we found that the Silhouette function was a more appropriate measure that did not have this limitation. Therefore, it was included in the evaluation. A Silhouette width measures the compactness of the cluster, which was more appropriate to discover non-spherical clusters. In addition, the Silhouette measure is supported by a comparative study on various cluster validity indices as the most appropriate measure for internal indices (Brun et al., 2007).

Tools and Software: The MOA (Massive On-line Analysis) framework (Bifet, Holmes, Pfahringer, et al., 2010) is an open source benchmarking software for data streams that is built on the work of WEKA (Weka Group Project, 2008; Holmes, Donkin, & Witten, 1994). MOA has a set of stream clustering algorithms and a collection of evaluation measures. MOA has considered stream classification algorithms; however, recently they added stream clustering evaluation tool (Kranen et al., 2010). Furthermore, another evaluation measure called Cluster Mapping Measure (CMM) (Kremer et al., 2011) is integrated to MOA for evolving data streams. CMM has a mapping component which can handle emerging and disappearing clusters correctly. Kremer et al. (Kremer et al., 2011) show that the proposed measure can reflect the errors in data stream context effectively. SAMOA (Scalable Advanced Massive Online Analysis) (De Francisci Morales, 2013) is another upcoming tool for mining big data streams. The goal of SAMOA is to provide a framework for mining data streams using a cluster/cloud environment.

2.9 Discussion

Figure 2.11 depicts the distribution of the reviewed papers for density-based data stream clustering algorithms over years. There are two peaks in 2009 and 2012 for both categories. However, it can be observed that micro-clustering methods are more popular than grid methods.

In Figure 2.12, we show the chronological order of the reviewed algorithms as well as how the algorithms relate to each other. It can be observed from the figure that the most remarkable algorithms are DenStream and D-Stream I in micro-clustering and the grid group respectively. Other algorithms in each of the categories try to improve two mentioned algorithms in different aspects such as improving efficiency or quality or handling different kinds of data by adding some features which are listed in Table 2.5.

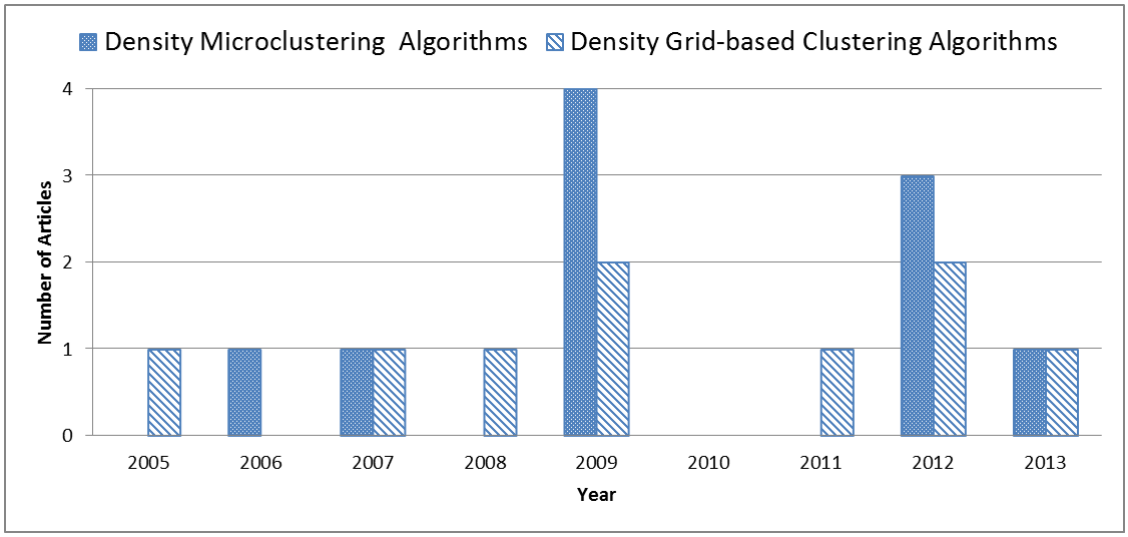


Figure 2.11: Distribution of the Reviewed Papers for Density-based Data Stream Clustering Algorithms

2.9.1 Density-based Data Stream Clustering Algorithm and Challenging Issues

In this subsection, we briefly describe how the algorithms overcome the challenges in clustering data streams.

- Handling noisy data: in micro-clustering algorithms outlier micro-cluster is introduced. The outlier and the real data are retained in different forms of micro-clusters, which helps to distinguish between the seeds of the new clusters from the outliers. In the grid methods, sporadic grid is introduced which has a limited number of data points mapped by outliers.

- Handling evolving data: density-based clustering algorithms over data streams either micro-clustering or grid based have the ability to handle evolving data streams using different kinds of window models such as fading and sliding window models. DUCStream does not handle evolving data because it considers the behavior of data streams as the data points arriving in chunks.

- Limited time: D-Stream II has the lowest time complexity which enables the processing of data stream in a limited time. Other algorithms' time complexity grows linearly as data streams are generated. However, the algorithms such as rDenStream

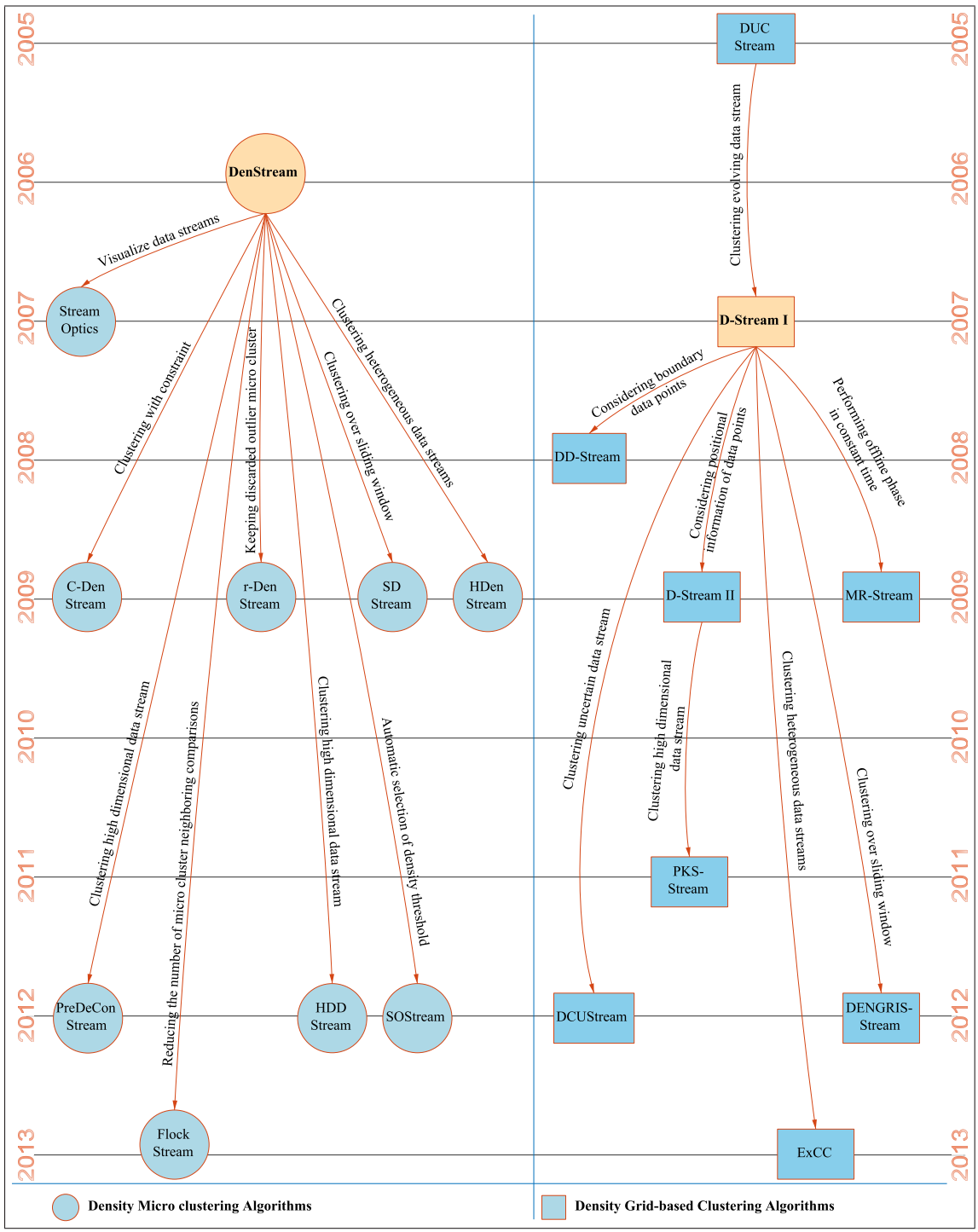


Figure 2.12: Chronological Order of the Reviewed Density-based Data Stream Clustering Algorithms

and C-DenStream need more time for processing historical buffer and constraints respectively. SOStream has the highest time complexity compared to other algorithms.

- Limited memory: the aforementioned algorithms use micro-clusters or grid to keep summary about the data stream to process data points. However, the algorithms

Table 2.5: Algorithms' Relations

Name	Added feature	Objective
DenStream (Cao et al., 2006)	Main algorithm	density micro data stream clustering
StreamOptics (Tasoulis et al., 2007)	DenStream+Visualization	graphically represents the cluster structure of the data stream
C-DenStream (Ruiz et al., 2009)	DenStream+Constraint	guiding clustering process using domain information
rDenStream (Li-xiong et al., 2009)	DenStream+Retrospect phase	using discarded micro-cluster to improve accuracy
SDStream (Ren et al., 2009)	DenStream+Sliding window	clustering more recent data
HDenstream (Lin & Lin, 2009)	DenStream+Categorical data	achieve higher cluster purity
SOSStream (Isaksson et al., 2012)	automate DenStream parameters	removing difficulties in choosing unsuitable parameters
HDDStream (Ntoutsi et al., 2012)	DenStream+High dimensional data	density-based projected clustering over high dimensional data streams
PreDeConStream (Hassani et al., 2012)	DenStream+High dimensional data	improve efficiency of offline phase in density-based projected clustering over high dimensional data streams
FlockStream (Forestiero et al., 2013)	DenStream+Bio model	avoid the computing demanding offline cluster computation
DUC-Stream (Gao et al., 2005)	Clustering data stream in chunks	density-grid single pass clustering
D-Stream I (Y. Chen & Tu, 2007)	Main algorithm	density grid-based data stream clustering
DD-Stream (Jia et al., 2008)	D-Stream I+Considering boundary points	improve quality
D-Stream II (Tu & Chen, 2009)	D-Stream I+Grid attraction	considering positional information of the data in that grid to improve quality
MR-Stream (Wan et al., 2009)	D-Stream I+Removing offline phase	improve quality
PKS-Stream (Ren et al., 2011)	D-Stream II+High dimensional data	clustering high dimensional data streams
DCUStream (Y. Yang et al., 2012)	D-Stream I+Uncertain data	improves density-based clustering algorithm for uncertain data stream environment
DENGRIS-Stream (Amini & Ying Wah, 2012)	D-Stream I+Sliding window	clustering more recent data streams
ExCC (Bhatnagar et al., 2013)	D-Stream I+Categorical data	exclusive and complete clustering for mix attributes data streams

such as rDenStream, C-DenStream, FlockStream and ExCC need more memory.

- Handling high dimensional data: if the algorithms are used for the high dimensional data, the time complexity would be high which is not acceptable in clustering of data streams. PKS-Stream, HDDStream, and PreDeConStream are the algorithms with the ability to handle high dimensional data streams.

Table 2.6 summarizes how the algorithms address the mentioned challenging issues.

2.9.2 Existing Algorithms Evaluation

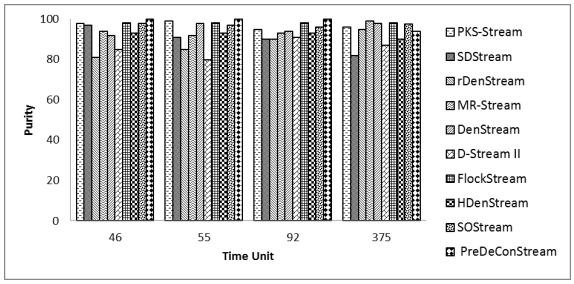
We compared the algorithms based on the evaluation metrics. The algorithms with same metrics are compared together, for example, algorithms using purity (DenStream, rDenstream, SDStream, PKS-Stream, MR-Stream, FlockStream, HDenStream,

Table 2.6: Density-based Clustering Algorithms and Challenging Issues

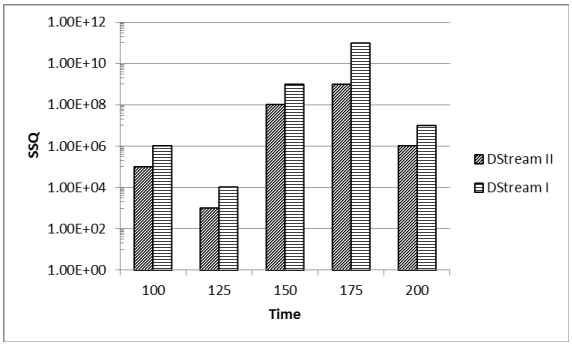
Density-based Clustering Algorithms	Handling Noisy Data	Handling Evolving Data	Limited Time	Limited Memory	Handling High Dimensional Data
DenStream	✓	✓	✓	✓	-
StreamOptics	✓	✓	-	-	-
C-DenStream	✓	✓	-	-	-
rDenStream	✓	✓	-	-	-
SDStream	✓	✓	-	✓	-
HDenStream	✓	✓	✓	✓	-
SOStream	✓	✓	-	✓	-
HDDStream	✓	✓	-	✓	✓
PreDeConStream	✓	✓	-	✓	✓
FlockStream	✓	✓	✓	-	-
DUCStream	✓	-	✓	✓	-
D-Stream I	✓	✓	-	-	-
DD-Stream	✓	✓	-	-	-
D-Stream II	✓	✓	✓	✓	-
MR-Stream	✓	✓	-	-	-
PKS-Stream	✓	✓	-	-	✓
DCUStream	✓	✓	-	✓	-
DenGRIS-Stream	✓	✓	-	✓	-
ExCC	✓	✓	-	-	-

SOStream, PreDeConStream) (Figure 2.13a) and algorithms using SSQ (D-Stream I, D-Stream II) (Figure 2.13b). However, C-DenStream is the only algorithm which uses Rand Index and it is compared with DenStream (Figure 2.13c). FlockStream also uses NMI (Normalized Mutual Information) (Manning, Raghavan, & Schtze, 2008) and is compared to DenStream to measure quality (Figure 2.13d). NMI is measured based on different time units which is chosen by FlockStream. All the comparisons are based on the real dataset KDD CUP99 (Rosset & Inger, 2000). Purity is measured based on the various time units in which at least an attack exists.

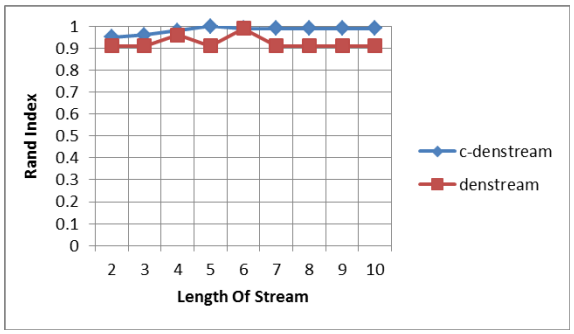
The high quality of DenStream and MR-Stream benefit from their effective pruning strategies, which promptly get rid of the outliers while keep the potential clusters to form final clusters. In terms of high dimensional data, PreDeConStream has better qual-



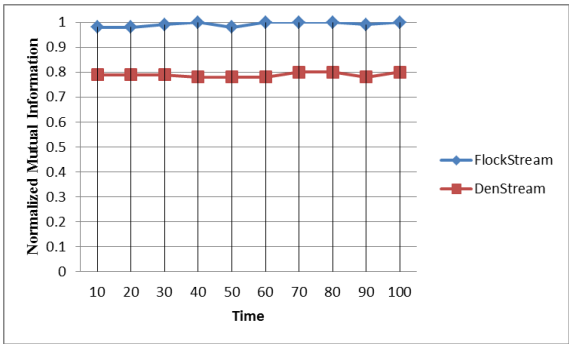
(a) Quality Comparison-Purity



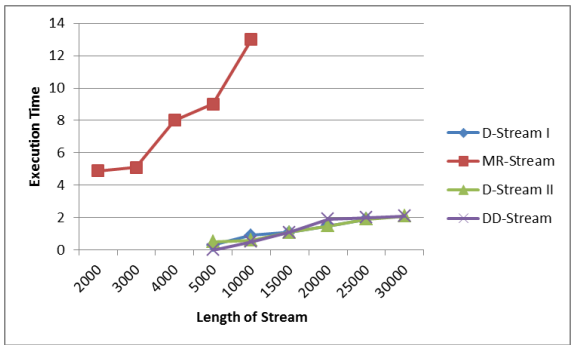
(b) Quality Comparison-SSQ



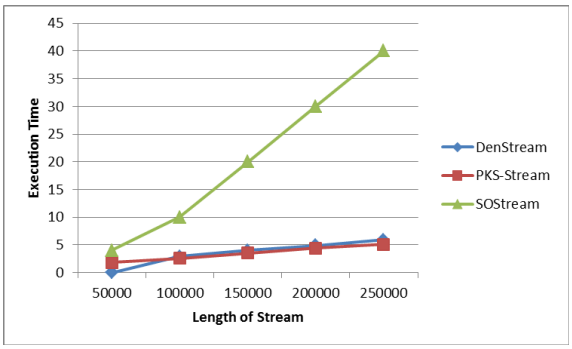
(c) Quality Comparison-RI



(d) Quality Comparison-NMI



(e) Execution Time Comparison-Length of Stream<50*k*



(f) Execution Time Comparison-Length of Stream>50*k*

Figure 2.13: Algorithm Evaluation

ity than PKS-Stream since it has a method to improve the offline phase of the algorithm. SDStream has acceptable quality in the initial time unit; however, the quality reduces specifically in time 375, when more attacks should be detected. The quality of rDen-Stream gradually improves since it makes classifier from clustering result. C-DenStream has a quality better than DenStream which shows that using the background knowledge for guiding the clustering improves the clustering quality. Even though FlockStream uses approximate nearest neighbor, it has higher quality compared to DenStream in terms of purity and normalized mutual information. D-Stream II has better quality in compare to

D-Stream I, since it considers the positional information about the data points inside the grid. HDenStream has quality less than DenStream which shows that it cannot improve DenStream to be used for data stream with categorical attributes.

We compared algorithms' performance as well. Execution time is measured based the number of data points (length of stream) with respect to the time in seconds. We divided algorithms comparisons based on the length of stream less than 50000 ($< 50k$) data points and more than 50000 ($\geq 50k$) (shown in Figures 2.13e and 2.13f) respectively to make fairly comparison. The comparison is based on the real dataset KDD Cup99 Network Intrusion Detection. The algorithms which are not in Figure 2.13 used another dataset or they are measured only on synthetic datasets or do not have any evaluation on their execution time.

It can be observed that SOStream has the highest execution time since finding the winner micro-cluster is time consuming. MR-Stream also has a high execution time even in smaller length of streams since the pruning method is time consuming. D-Stream I, DD-Stream, and D-Stream II have almost the same execution time; however, D-Stream II has a better time performance than the others. D-Stream II uses tree structure for keeping grid list which makes the algorithm faster. DenStream's execution time is similar to PKS-stream. It shows that PKS-Stream clusters high dimensional data with acceptable execution time.

Table 2.7 compares the quality metrics, memory usage, time complexity and application domain of the reviewed algorithms which will be discussed in the following subsection.

2.9.3 Discussion on Multi-Density Algorithms

A couple of clustering algorithms are developed for multi-density datasets. They use different methods to discover clusters with various densities such as localizing MinPts,

Table 2.7: Evaluation on Density-based Data Stream Clustering Algorithms

Name	Quality Metric	Memory Usage	Time Complexity	Application Domain
DenStream (Cao et al., 2006)	Purity	m	$O(m)$	Network Intrusion Detection System
StreamOptics (Tasoulis et al., 2007)	-	m	$O(m*log(m))$	Environment monitoring
C-DenStream (Ruiz et al., 2009)	Rand Index	$m+m_c$	$O(m+m_c)$	Environment monitoring
rDenStream (Li-xiong et al., 2009)	Purity	$m+S_{hb}$	$O(m)+T_h$	Network Intrusion Detection System
SDStream (Ren et al., 2009)	Purity	n_{sw}	N/A	Network Intrusion Detection System
HDenstream (Lin & Lin, 2009)	Purity	m	$O(m)$	Network Intrusion Detection System
SOStream (Isaksson et al., 2012)	Purity	m	$O(n^2 logn)$	Network Intrusion Detection System
HDDStream (Ntoutsi et al., 2012)	Purity	m	$o(m)+o(m_p)$	Environment monitoring, Network Intrusion Detection System
PreDeConStream (Hassani et al., 2012)	Purity	m	$o(m)+o(m_{ip})+o(m_{dp})$	Network Intrusion Detection System
FlockStream (Forestiero et al., 2013)	Purity, NMI	$m+n_{agent}$	$o(m)+o(n^2_{agent})$	Network Intrusion Detection System
DUC-Stream (Gao et al., 2005)	SSQ	n_d	$O(c_b)$	Network Intrusion Detection System
D-Stream I (Y. Chen & Tu, 2007)	SSQ	g	$O(1)+o(g)$	Network Intrusion Detection System
DD-Stream (Jia et al., 2008)	N/A	g	$O(g^2)$	Network Intrusion Detection System
D-Stream II (Tu & Chen, 2009)	SSQ	$log_{\frac{1}{\lambda}} g$	$O(\log\log_{\frac{1}{\lambda}} g)$	Network Intrusion Detection System
MR-Stream (Wan et al., 2009)	Purity	$g*H$	$O(g*H)+O(2^g*H)+O(g*log(N))$	Network Intrusion Detection System
PKS-Stream (Ren et al., 2011)	Purity	log^g_k	$O(logk)$, $O(k)$	Network Intrusion Detection System
DCUStream (Y. Yang et al., 2012)	Average quality of clusters	g	$o(g)$	Environment monitoring
DENGRIS-Stream (Amini & Ying Wah, 2012)	N/A	g	$o(g)$	N/A
ExCC (Bhatnagar et al., 2013)	Purity	$g+S_{Pool}+S_{HQ}$	$O(g^{xk})$	Network Intrusion Detection System

n: number of data points, m: number of micro-clusters in main memory, m_c: number of micro-cluster constraints, S_{hb}: size of historical buffer, T_h: time for processing historical buffer, n_{SW}: sliding window length, n_{agent}: number of agents, $o(m_p)$: number of potential micro-clusters, m_{ip}: number of inserted potential micro-clusters, m_{dp}: number of deleted potential micro-cluster ,n_d: number of dense units, c_b: clustering bits, g: number of grids in grid list, λ: decay factor, H: level of clustering, k: pks-tree degree, S_{Pool}: size of pool for dense grids, S_{HQ}: size of hold queue for noise, xk: number of discovered clusters

localizing ε, using grid method to map the data and to determine the density of data points, ranking the clustering density, and forming a cluster tree for multi-density data. The algorithms are more suitable when the whole data is available or in case that the processing should not be done in a limited time. However, in data stream environments, these algorithms are not applicable due to the following drawbacks:

1. They need two-pass of data such as GMDBSCAN (Xiaoyun et al., 2008), and IS-DBSCAN (Carmelo et al., 2013). In these algorithms, they first extract useful information about the distribution of data and then cluster the data based on the extracted information. This situation is impossible in the data stream, since data streams arrive continuously, and the clustering algorithms have to be performed in a single scan.

2. Some of the existing multi-density clustering algorithms need the whole data (Xiaoyun et al., 2008; X. Li et al., 2010; Esfandani et al., 2012).

3. Algorithms like (X. Chen et al., 2012; Esfandani & Abolhassani, 2010; Huang et al., 2009) have high execution time which makes them not applicable for data streams because they need fast processing time.

DSCLU (Namadchian & Esfandani, 2012), a density-based clustering for data stream in multi-density environments, considers similar radiuses for all micro clusters even with different data point distributions.

Table 2.8 is a comparison of the existing algorithms for multi-density data.

2.9.4 Density-based Data Stream Clustering Algorithms' Applications

The literature on density-based clustering for data streams is usually centered around concrete methods rather than application contexts. Nevertheless, in this section, we would like to bring examples of several possible scenarios where density-based clustering can be used.

The density-based method has been used for earth environments since a long time ago (Sander, Ester, Kriegel, & Xu, 1998). Recently it has been utilized for medical purposes such as a pre-processing phase for prediction of Alzheimer disease (Plant et al., 2010) and for skin cancer (Mete, Kockara, & Aydin, 2011).

Table 2.8: A Comparison of Density-based Clustering Algorithms on Multi-Density data

Algorithm Name	Method	Data Stream Supported	Disadvantages for data stream
GMDBSCAN (Xiaoyun et al., 2008)	Grid density, local MinPts	✗	Two scan clustering
Multi-DBSCAN (Huang et al., 2009)	k nearest distance, must link constraints	✗	High execution time
MSDBSCAN (Esfandani & Abolhassani, 2010)	localizing core point concept	✗	High execution time
(X. Li et al., 2010)	Hierarchical cluster tree, k-means	✗	Need whole data
SCDM2 (X. Chen et al., 2012)	Using constraints in clustering	✗	Needs number of clusters (semi supervised)
GDCLU (Esfandani et al., 2012)	Grid-based clustering, local density	✗	Need whole data
DSCLU (Namadchian & Esfandani, 2012)	Dominant micro cluster	✔	Using similar radius
DBSCAN-DLP (Xiong et al., 2012)	Density level partitioning	✗	Two scan clustering
ISDBSCAN (Carmelo et al., 2013)	Space ranking	✗	Two scan clustering

Real world applications may have any shape clusters and generate noisy data in some situations. Furthermore, they do not require the number of clusters in advance. Since density-based clusterings have some abilities in their nature, they are applicable in different applications such as:

- Network intrusion detection system: in this system, sensors capture all network traffic and the system analyzes the content of individual packets for malicious traffic (Cao et al., 2006).
- In environment observations: for example, in applications which are used to monitor flood, hurricane, tsunami, earthquake and forest fire detection (Ruiz et al., 2009).
- Medical systems: clustering medical data streams such as anatomical and physiological sensors, incidence records, health information systems, and patient monitoring system (Mete et al., 2011; Plant et al., 2010).
- Stock trade analysis: for example clustering one million transaction records throughout the trading hours of a day (D. Yang, Rundensteiner, & Ward, 2011).

- Social network analysis: clustering micro-blogging text streams (e.g. Twitter), in order to obtain temporal and geo-spatial features of real-world events (Lee, 2012).

- Moving objects applications: such as animal migration analysis, vehicle traffic management (Y. Yu, Wang, Wang, Wang, & He, 2013).

Applications like patient monitoring and sensor networks in seismic studies, for example, work in bounded data space. Therefore, it is more preferable to use grid-density based methods. In these applications, a data point is either a member of a cluster or an outlier. Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All the clustering operations are performed on the grid structure, i.e. on the quantized space. The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space. In this method if the quality is the most important factor and time and memory are second and third factors respectively, MR-Stream is the best choice. In the case of the importance of execution time such as environmental observations, for example, for Tsunami detection the best choice is D-stream II since it has the lowest execution time. However, the quality of grid-based method is highly dependent on the granularity of the grid and further, defining the grid granularity to get the proper result is challenging.

Another important class of density-based algorithms over data streams is the density micro-clustering group. The quality of these algorithms is better than the grid-based methods. In the grid-based method, if we want to get more accurate results we have to fine the grids that leads to high time complexity. Density based micro-clustering has better quality with reasonable time complexity. Micro-clustering method has limited memory usage which depends on the number of micro-clusters. In the micro-clustering method, when the data points arrive they are assigned to the related micro-clusters and at the same

time the outlier micro-clusters are removed based on the density threshold. Therefore, clustering results can be generated any time. However, it has some limitations; finding the proper micro-cluster is time consuming. In some cases because of the limitations in the memory usage, some real data is removed due to the appearance of an outlier.

However, choosing a proper density micro-clustering algorithm depends on the type of the application. For example, in clustering GIS applications the best choice is C-DenStream because it considers the real world constraints such as the city, rivers and highway networks. If the application needs limited processing time with good quality, FlockStream is a better choice rather than DenStream since it decreases the number of micro-clustering comparisons. If quality is the first priority, r-DenStream is the choice; however, it needs more memory usage and execution time compared to the other algorithms. If there is any application which its thresholds' settings (like similarity threshold or grid size) are difficult to be manually done, SOStream is the best choice because it automatically adapts the thresholds. For detecting clusters in the recent data such as identifying malicious attacks (clusters) in current network traffic or recent stock trades in stock exchange, SDStream and DENGRIS-Stream are more applicable since they cluster within the most recent portion of the stream.

Another aspect of choosing an algorithm is the type of data generated by the application such as uncertain, high-dimensional or heterogeneous. Most of the algorithms in micro-cluster and grid groups only cover the continuous data. Therefore, if we have for example biomedical data with the categorical attributes, we have only ExCC in the grid group and HDenStream in the micro-cluster group. Furthermore, in some sensor-based applications the output of sensor networks is uncertain because of the noise in the sensor inputs or errors in wireless transmission. In this case, the algorithm has to cover the uncertain data as well. In this situation the best choice is DCUStream. Moreover, if the data is high-dimensional in its nature, we can choose between HDDStream and PreDe-

ConStream in the micro-clustering group and PKS-Stream in the grid-based algorithms.

In summary, the task of choosing a proper density-based clustering algorithm depends on the kind of data produced as well as the application requirements such as limited time, high quality, high accuracy, handling high noisy data and many other requirements which are defined based on the application's objectives.

2.10 Summary

Data stream are infinitive, massive and evolve over time which make the clustering more challenging. Density-based method is one of the significant classes in clustering which has prominent features such as detecting arbitrary shape clusters, handling noise, and it does not require the number of clusters in advance.

Recently, a number of density-based clustering are developed for data stream. These algorithms use different methods to address the challenges in clustering data streams. The methods are referred to as density micro clustering and density grid-based clustering. Micro-clustering algorithms have high computation time while grid-based methods have low quality and neither of them have the ability to cluster multi-density data. Some algorithms have been proposed for multi-density clustering. However, these algorithms are only for static datasets and they are not applicable for data stream due to some problems such as high execution time, two-pass clustering and the need for the whole data.

Based on the comprehensive review on the existing density-based clustering for evolving data stream and multi-density algorithms, it is concluded that the existing density-based data stream clustering algorithms have high computation time and low quality when there is a range of densities in data. Our intention in this study is to develop a density-based clustering algorithm with low time complexity and high quality even when the data has various density distributions.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview

This chapter explains the research methodology used in the study. We explain in details the research methodology steps which includes an overview of the existing methods, how to approach the problem statement, and how to achieve the objectives. Furthermore, the evaluation method and the analysis of evaluation results are presented.

3.2 Approaches to Research

The research methodology framework is shown in Figure 3.1. Every step of the research methodology is described as follows.

3.2.1 Reviewing Related Works

We reviewed existing density-based clustering algorithms as well as challenges in clustering data stream. We defined to what extend the existing clustering algorithms can overcome the challenges. Furthermore, we extracted the problems of existing density-based methods for clustering data streams. The analysis of the existing approaches gives a wider perspective of the problems in density-based clustering of data streams.

3.2.2 Problem Formulation

Reviewing the existing methods turned out that density-based clustering algorithms have high computation time for clustering data streams. Furthermore, density-based clustering cannot work well in multi-density data stream. In fact, they have low quality in the environments with various densities.

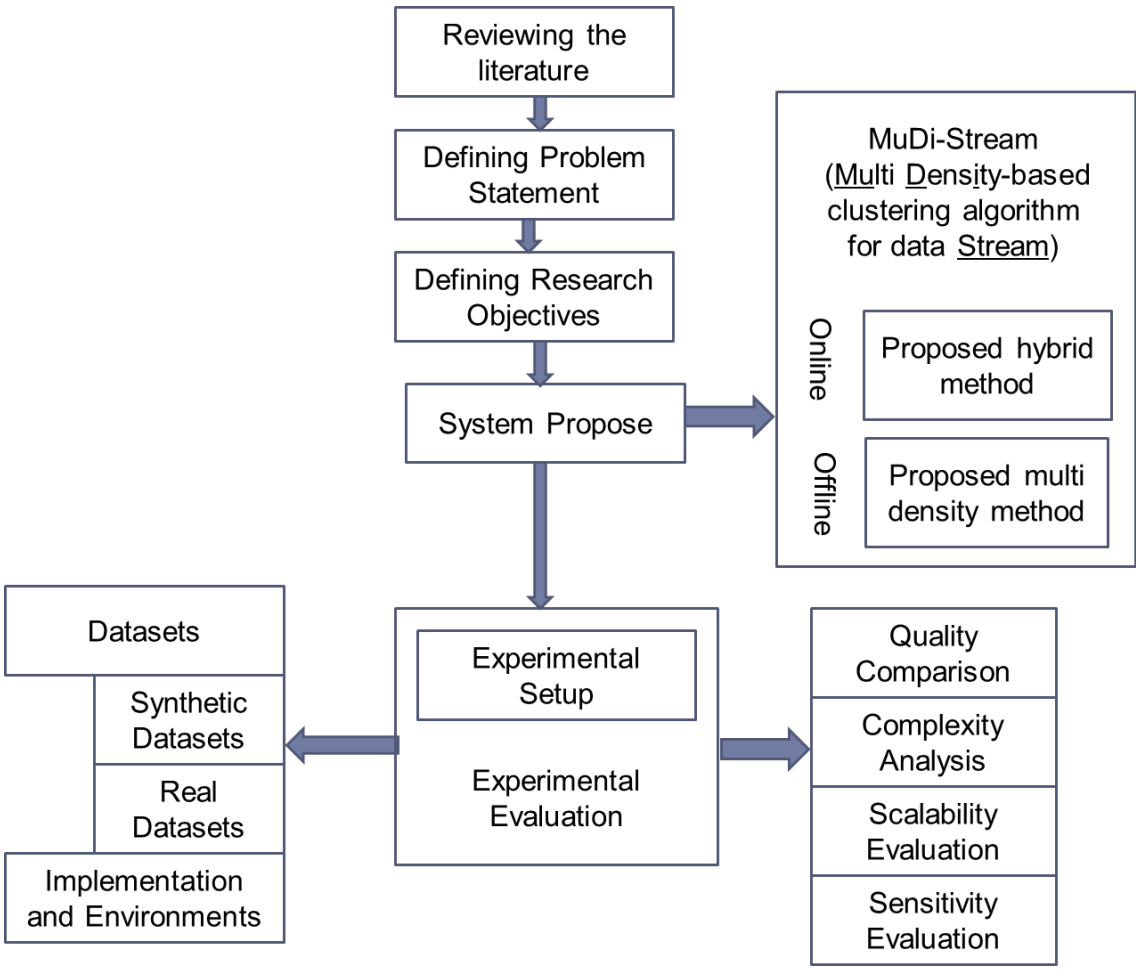


Figure 3.1: Research Methodology Framework

3.2.3 Defining the Research Objectives

The objectives of our research based on the problem statement are as follows:

- To propose and develop a density-based clustering algorithm for evolving data streams. This objective needs the following methods:
  - To develop a method with low computation time
  - To develop a new method to save summary information about data stream in multi-density environments
  - To develop a method to prune the summary information
  - To develop a method to perform macro clustering on synopsis data

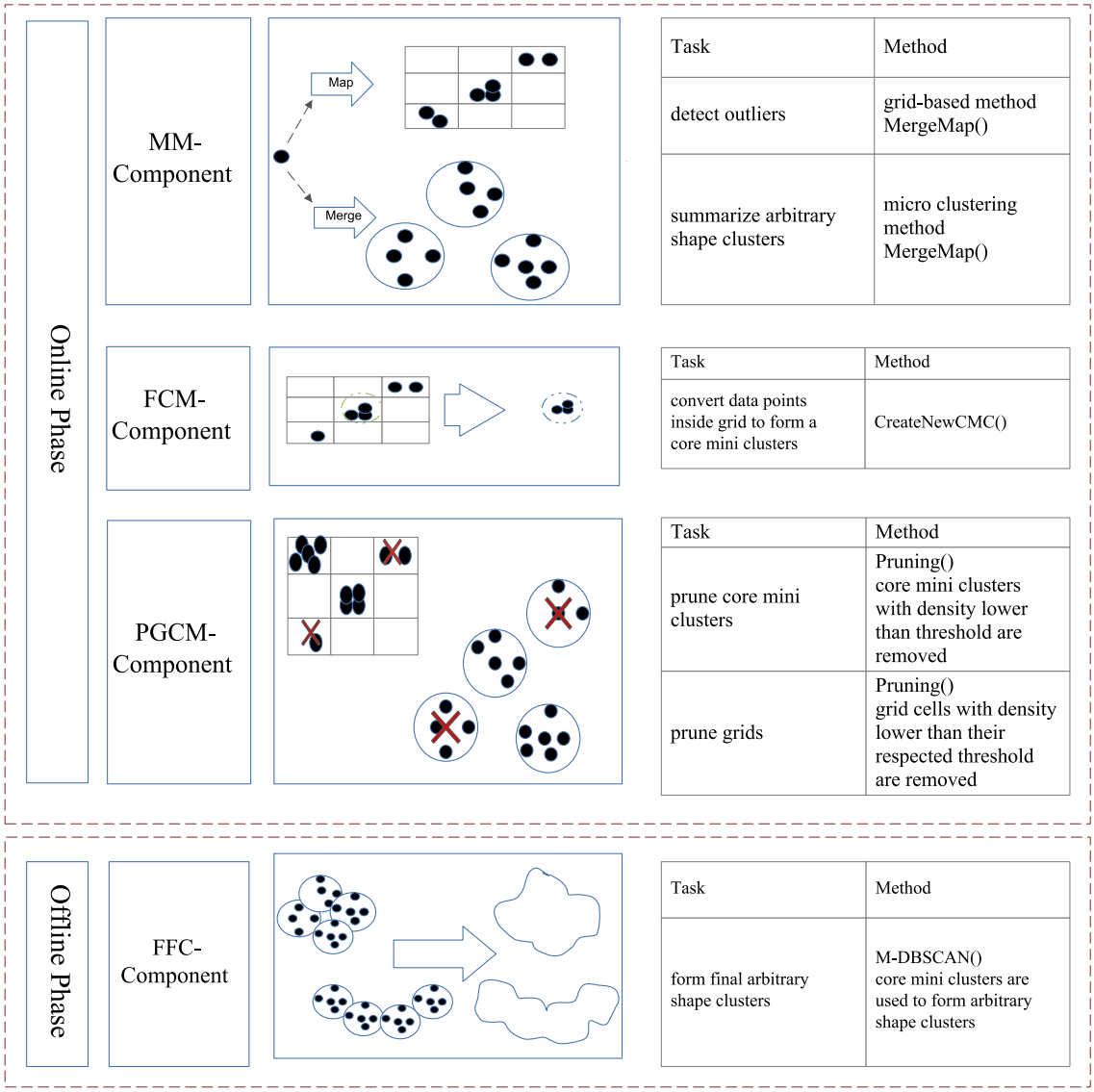


Figure 3.2: MuDi-Stream Algorithm

- To evaluate the capability of the proposed method in improving the quality of clustering in multi-density environments
- To evaluate the capability of the proposed method to perform clustering in low computation time

3.2.4 System Propose

In order to achieve the objectives, a method which we called it MuDi-Stream (Multi Density clustering algorithm for evolving data Stream), is proposed. The algorithm, which is illustrated in Figure 3.2, is an online-offline one and it has the following components:

- Merging or Mapping (MM-Component): in this component, core mini cluster is introduced to keep summary information about arbitrary shape clusters while a grid list is used to keep outliers. When a data point arrives, the algorithm checks to add it to the nearest core mini cluster in case its distance is less than the core mini cluster's radius. However, if it cannot be added to any core mini cluster, the data point is considered as noise and it is mapped to the density grid.

- Forming Core Mini clusters (FCM-Component): if the grid density is more than a predefined threshold, a new core mini cluster is formed from the data points inside the grid cell. Each core micro cluster is formed based on a different radius. The radius is calculated according to the distribution of data inside the micro-cluster.

- Pruning Grids and Core Mini clusters (PGCM-Component): for each core mini cluster, if no new point is added, its weight will decay gradually. Furthermore, there are some grids which do not receive data points for a long time and become sporadic. These kinds of core mini clusters and grid cells should be removed from the mini clusters and the grid list respectively. The decision for removing grids and mini clusters is made based on a comparison of their weights and a specific threshold.

- Forming Final Clusters (FFC-Component): final clusters are formed from pruned core mini clusters. In this phase, a modified multi-density based clustering algorithm is used to perform the clustering on the core mini clusters.

In the following, the motivation of each component of the proposed multi-density based clustering algorithm for evolving data streams is presented:

- **Motivation for MM-Component**:

Data stream consists of indefinitely and possibly time-evolving sequences (Michalak, DuBois, DuBois, Wiel, & Hogden, 2012; Chu et al., 2006; X. Zhang, Furtlehner, Germain-Renaud, & Sebag, 2013; J. a. Gama, Rodrigues, & Lopes, 2011; X. Zhang et al., 2013). The omnipresence of data stream poses new challenges for clustering. Data stream are infinite and evolve over time. The challenge is how to keep summary information from this huge amount of data generated over time. Two prominent categories in clustering data stream is using grid synopsis or micro-clustering. These methods are discussed extensively in Chapter 2. In this thesis, a hybrid method using grid and micro clusters is used for keeping summary information. In real applications the data is noisy which makes detection of clusters more challenging since data streams evolve over time. The motivation of using this hybrid method is that the micro clusters keep the real data while grid is used for the noise or outliers. When data points in the grid structure gain enough weight, it is converted to micro clusters.

- **Motivation for FCM-Component**:

  Data stream clustering is a challenging problem because of two important properties: its infinite length and evolving nature (Masud, Gao, Khan, Han, & Thuraisingham, 2008; Read, Bifet, Holmes, & Pfahringer, 2012). Therefore, data stream needs a synopsis structure which is updated as its nature changes over time. For example, as the time passes the outliers may convert to real data and vice versa. Therefore, in evolving data stream, a method to check the density threshold is vital. Since a hybrid method is used, outliers are mapped to the grids and real data form core mini clusters. The main duty of this component is forming core mini clusters by controlling the aggregation of data points' weight inside the grid in case the weight value is more than a threshold.

- **Motivation for PGCM-Component**:

  In evolving data stream, the role of the clusters and outliers frequently change. A number of core mini clusters are formed as data stream proceeds. Nevertheless, the problem is that the number of outliers are also growing as data stream proceeds. The worst case happens when a lot of outliers exist. Therefore, the algorithm requires an effective pruning strategy which quickly eliminates the outliers while keeps the potential core mini clusters. Instead of pruning too frequently, a pruning time is calculated which is the minimum timestamp for a core mini cluster to be converted to an outlier (Y. Li, Li, Wang, & Zhai, 2014).

- **Motivation for FFC-Component**:

  In the online phase of MuDi-Stream, all core mini clusters capture the density areas of data stream. However, a clustering algorithm is required to get meaningful clusters. In this component, an algorithm named M-DBSCAN is developed which has the ability to cluster core mini clusters with various densities. M-DBSCAN is an extension of DBSCAN algorithm (Kriegel et al., 2011; Aggarwal & Reddy, 2013) with the ability to cluster multi-density data.

3.2.5 Experimental Setup

Various datasets including real and synthetic ones are selected for evaluation purposes. For all datasets, we used the following normalization technique in order to have all the data points in the range of $[0,1]$:

$$Normalized(e) = \frac{e - E_{min}}{E_{max} - E_{min}}$$

where E_{min} is the minimum value of variable e, and E_{max} is its maximum value. If E_{max} and E_{min} are equal then $Normalized(e)$ is set to 0.5.

MuDi-Stream has some parameters including *GridGranularity*, λ, and α. For each dataset, the range of parameters are determined.

MuDi-Stream with all its components are implemented in Java. One of our future work is to implement it as an add-on in MOA.

3.2.6 Evaluation Method

As it is explained in Section 2.8 of Chapter 2, the quality metrics are categorized in internal and external indices. The most common evaluation method to measure quality is purity and NMI. However, we added some other metrics which are chosen from the data stream clustering in the literature. The list of metrics used in this thesis include: Purity, Normalized Mutual Information (NMI), Rand Index (RI), Adjusted Rand Index (ARI), Jaccard index, Fowlkes and Mallow index (FM), and F-measure.

These metrics are used in order to show to what extend the quality is improved compared to the existing methods.

The focus of this study is on improving the quality; however, the scalability, complexity, and sensitivity of the proposed model are also calculated to prove its feasibility theoretically. For scalability, time execution and memory usage are measured. For complexity, time and space complexity are discussed, and finally for sensitivity the range of all parameters are determined and compared to the clustering quality metrics. We evaluated the algorithm on various real and synthetic datasets. The pseudo-code for all methods are provided separately. The results are compared with the existing well-known and state-of-the-art algorithms.

Figure 3.3 depicts the evaluation method of MuDi-Stream algorithm.

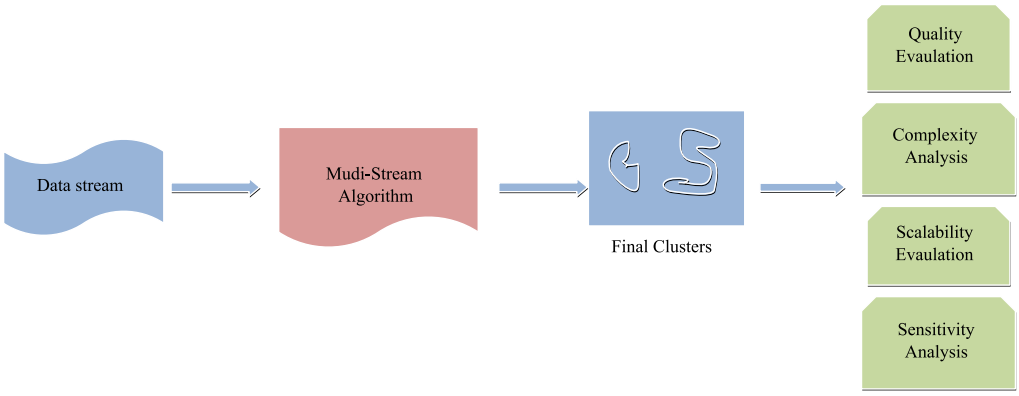


Figure 3.3: Evaluation Steps

3.3 Summary

The chapter explains the research methodology of the thesis. A method is developed based on the problem statement and the research objectives. It is called MuDi-Stream which is a density-based clustering algorithm with the ability to cluster multi-density data stream while it has low computation time. In MuDi-Stream, a hybrid method using grid- and micro-cluster techniques is proposed to solve high computation time problem. In our proposed method, micro clusters keep the data for arbitrary shape clusters and grid method is used for outliers. Compared to the existing methods, mapping to the grid is much more faster than adding to related outlier micro clusters. For handling multi-density data, in the online phase data stream's summary is kept in a way to handle multi-density data. Furthermore, a new algorithm is developed for the offline phase for multi-density arbitrary shape clusters with noise using statistical information about data points' densities. All the tasks are done through some components consisting MM-Component (either merging or mapping of a new data point), FCM-Component (forming core mini cluster from a grid cell), PGCM-Component (pruning grids and core mini clusters), and FFC-Component (forming final clusters). More details about each component are explained in the next chapter. Furthermore, we briefly explained the evaluation method. Different kinds of datasets including real and synthetic datasets are used to evaluate the quality of the proposed method using various metrics.

CHAPTER 4

PROPOSED SYSTEM

4.1 Overview

In this chapter, the proposed model, MuDi-Stream (Multi Density-based clustering method for evolving data streams), is explained in details. Sections 4.2 and 4.3 generally discuss about the proposed method and Section 4.4 analyzes how the proposed algorithm satisfies the challenges in clustering data stream. In Section 4.5, a general view of the model is presented. The basic concepts of the proposed method are introduced in Section 4.6. The Algorithm has online and offline phases. Each phase has some components which are elaborated in the subsequent sections. The online phase has three components which are described in details in Sections 4.7.1, 4.7.2, and 4.7.3 respectively. The offline phase's component is described in Section 4.8.1.

4.2 The Proposed Hybrid Clustering Method

One of the challenging issues in density-based clustering algorithms is how to reduce the computation time. For this purpose, a hybrid method of density grid-based and micro clustering is proposed. Using the hybrid method leads to decrease the computation time. When a data point arrives, firstly, we try to find a suitable micro cluster for it. If the data point cannot be placed in any existing micro cluster then the grid method is used and the data point is considered as an outlier. Despite the existing methods which form another micro cluster for an outlier data. In our method, it is mapped to the grid and if the grid's weight reaches to a specific threshold which is the micro cluster threshold's weight, it is converted to a micro cluster. Using the grid method significantly affects the searching time since it replaces the search in an outlier micro cluster list with a mapping method.

4.3 The Proposed Multi-Density Clustering Method

Another challenging issue is how to cluster multi-density data. In clustering data streams, while data points of the stream arrive some summary information are kept. These summaries are incrementally updated over time. Our proposed method has online and offline phases. A new concept called core mini clusters (cmc) is introduced to keep summary information about multi-density arbitrary shape clusters. A new multi density-based clustering method called M-DBSCAN is also proposed in the offline phase to form final clusters from the summary information. The proposed multi density-based method also uses statistical information to form final arbitrary shape clusters.

4.4 The Proposed Method and Challenging Issues

There are some challenging issues regarding the solutions for the aforementioned problems as follows.

- Evolving data stream: The data stream is evolving and so some clusters may disappear and the others appear. A weight value is considered for each data point in order to keep track of the time it appears.
- Handling noise: the noise are mapped to the grids and they are pruned frequently if they do not have chance to be converted to core mini clusters.
- Limited time: The algorithm has bounded time to cluster data which is discussed in Sections 5.6.2 and 5.7.1.
- Limited memory: We try to keep the required memory limited by pruning grids and core mini clusters. We analyze memory usage later in Sections 5.6.1 and 5.7.2.

4.5 An Overall View of MuDi-Stream Algorithm

In this research, a new multi density-based clustering algorithm for evolving data stream called MuDi-Stream (Multi Density-based clustering algorithm for evolving data

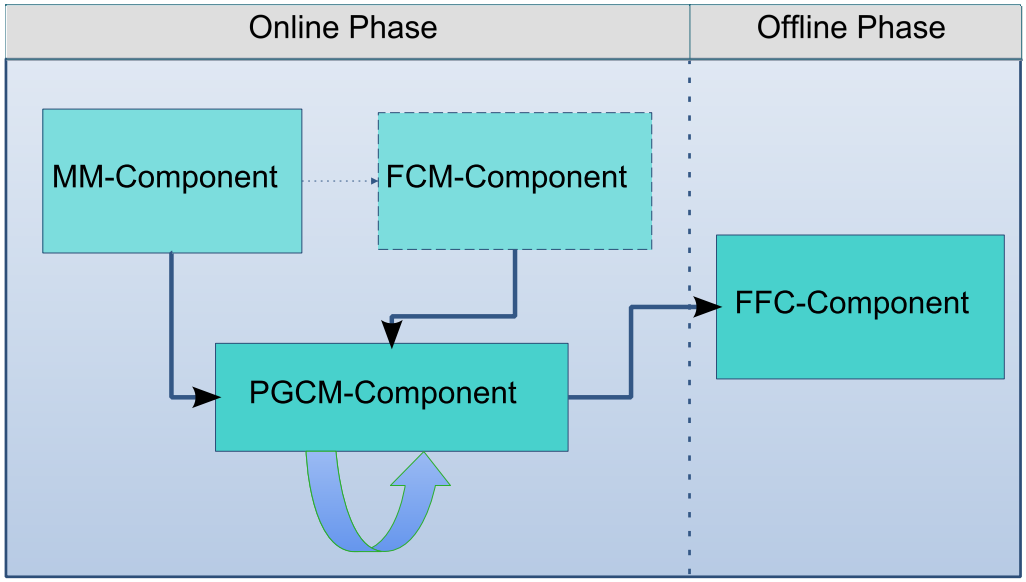


Figure 4.1: Overall View of MuDi-Stream Algorithm

Stream) is proposed. MuDi-Stream has online and offline phases which use a proposed hybrid method for the online phase. A proposed multi-density method is also used for the offline phase. MuDi-Stream keeps summary information about evolving data stream in the form of core mini clusters in its online phase. The grid-based method is used as an outlier buffer to handle noises and multi-density data and yet is used to reduce the merging time of clustering. Furthermore, a novel pruning strategy is designed to handle the weights of the core mini clusters and the grids. The offline phase generates the final clusters using a new multi-density method.

The tasks of online phase MuDi-Stream are divided into three components: MM-component, FCM-component, and PGCM-component. Moreover, the offline phase has FFC-component. We elaborate further on the components as follows. Figure 4.1 depicts an overall view of MuDi-Stream algorithm.

The online phase has the following components:

- Merging or Mapping (MM-component): merging data points to existing core mini clusters or mapping them to the grid.

- Forming Core Mini clusters (FCM-component): if the density of data points inside

Table 4.1: MuDi-Stream Components and Algorithms

Component	Algorithm(s)	Input	Output
MM-component	MergeMap (Algorithm 3)	data points of the stream, current timestamp, and density threshold$(x, t_c, \alpha, \lambda, N)$	core mini clusters, grid$(\{cmc\}, g)$
FCM-component	CreateNewCMC (Algorithm 4)	a grid cell, current timestamp (g, t_c)	a core mini cluster (cmc)
PGCM-component	Pruning (Algorithm 5)	core mini clusters, grid list, current timestamp, density threshold $(\{cmc\}, g, t_c, \alpha, \lambda, N)$	pruned core mini clusters and grid list $(\{cmc\}, g)$
FFC-component	M-DBSCAN (Algorithm 7)	core mini clusters $(\{cmc\}, g, MinPts)$	arbitrary shape clusters (C)

grid cells is higher than a predefined threshold, a new core mini cluster is formed out of the cell with a related radius.

- Pruning Grids and Core Mini clusters (PGCM-component): the grid cells' as well as core mini clusters' weights are periodically checked in a defined pruning time.

Moreover, the offline phase has one component:

- Forming Final Clusters (FFC-component): The final clusters are formed based on the pruned core mini clusters. Each core mini cluster is considered as a virtual point for clustring using a modified DBSCAN which we call it M-DBSCAN.

Figure 4.2 provides a detailed view of the proposed method, MuDi-Stream. Table 4.1 lists the algorithms which are related to every component. Further, the overall view of the algorithm is outlined in Algorithm 2. MuDi-Stream algorithm parameters for online and offline phase are listed in Table 4.2

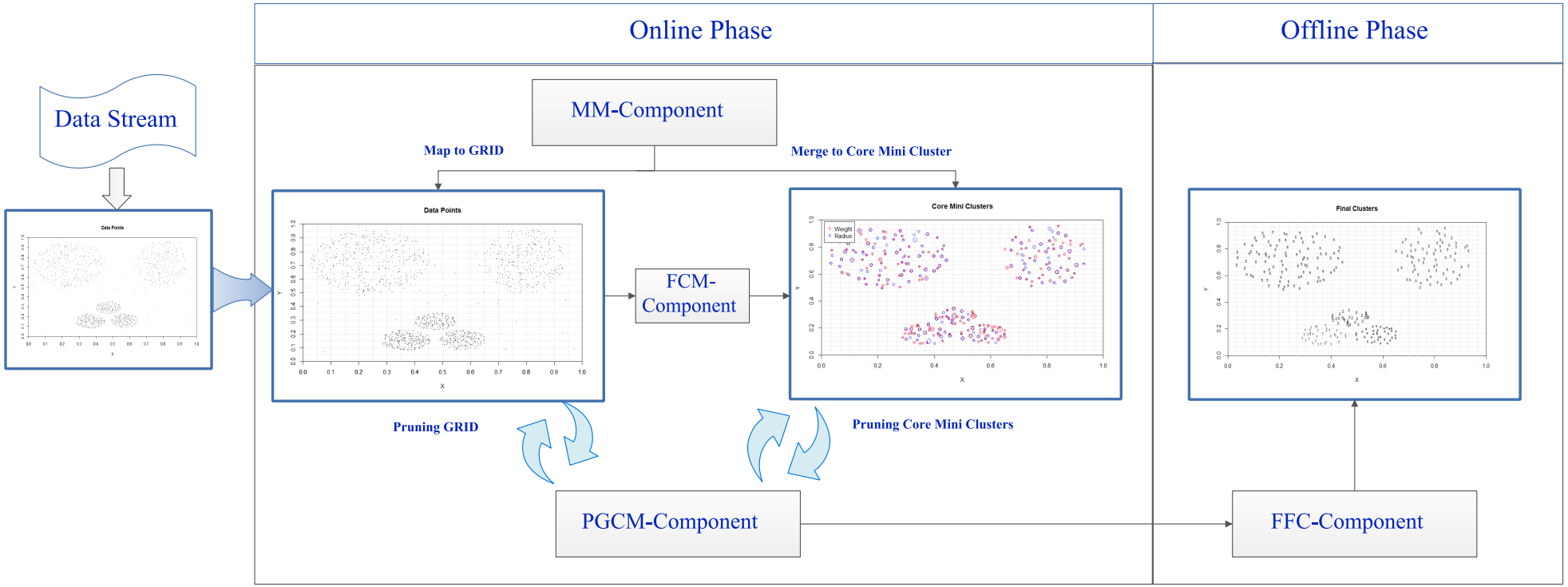


Figure 4.2: A Detailed View of MuDi-Stream Algorithm

Algorithm 2 MuDi-Stream(DS,λ,α,N)-Online Phase

Input: a data stream
Output: arbitrary shape clusters

1: $t_{pt} = \frac{1}{\lambda}\log_2^{\frac{\alpha}{\alpha-1+2^{-\lambda}}}$
2: $t_c \leftarrow 0$;
3: **while** not end of stream **do**
4: Read a data point, x, from data stream;
5: MergeMap$(x, t_c, \alpha, \lambda, N)$;
6: **if** t_c mod $t_{pt} == 0$ **then**
7: Pruning$(cmc, g, t_c, \alpha, \lambda, N)$;
8: **end if**
9: $t_c \leftarrow t_c + 1$;
10: **end while**

Table 4.2: MuDi-Stream Algorithm Parameters

Parameters	Explanation
λ	Density threshold
α	Outlier threshold
cmc	core mini cluster
g	grid cell
$MinPts$	minimum number of point
t_{pt}	pruning time
N	number of grid cells
mcd	mini core distance
n_g	Number of point inside grid
t_p	Last time grid update
w_g	grid weight
w_{cmc}	core mini cluster weight
r_cmc	core mini cluster radius
mcd_{cmc}	mini core distance of core mini cluster
t_c	current time
$owt(t_c, t_p)$	density threshold function
$N_g(cmc)$	cmc-grid-neighborhoods
$N_{sh}(cmc_q)$	MinPts-nearest-neighbors
N_{core}	core-neighboring

4.6 Basic Concepts of MuDi-Stream Algorithm

In this section, we introduce the new concepts of MuDi-Stream algorithm.

Definition 11 ***(Density Grids).*** *In this research, it is presumed that the input data has d dimensions, and each data point is defined within the space:*

$$S = S_1 \times S_2 \times ... \times S_d, \tag{4.1}$$

S_i is the definition space for the i^{th} dimension. The d-dimensional space S is divided into density grids. For each dimension, its space S_i, $i = 1,\ldots,d$ is partitioned to gridGranularity partitions as:

$$S_i = S_{i,1} \bigcup S_{i,2} \bigcup \ldots S_{i,gridGranularity} \tag{4.2}$$

Then, the data space S is partitioned into $N = \prod_{i=1}^{d} gridGranularity_i$ density grids. Each density grid g is comprised of $S_{1,j_1} \times S_{2,j_2} \times \ldots \times S_{d,j_d}$, $j_{i=1,\ldots,gridGranularity}$, which is defined as follows:

$$g = (j_1, j_2, \ldots, j_d) \tag{4.3}$$

A data point $x = (x_1, x_2, \ldots, x_d)$ is mapped to a density grid $g(x)$ as: $g(x) = (j_1, j_2, \ldots, j_d)$, $x_i \in S_{i,j_i}$.

According to the discussion for clustering evolving data streams in Section 2.2.1, there are different window models. One of the remarkable window models which is used in a number of existing methods (Cao et al., 2006; Y. Chen & Tu, 2007; Wan et al., 2009; Forestiero et al., 2013; Amini et al., 2014) is fading window model. This thesis also uses the fading window model since it can capture evolving nature of data stream very well. Therefore, for each data point, a weight is considered which decreases exponentially with time using a fading function. The fading function that we use in MuDi-Stream is defined as follows:

$$f(t) = 2^{-\lambda t} \tag{4.4}$$

where $\lambda > 0$ (Ng & Dash, 2010). We use the fading function as a weight coefficient for each data point.

Definition 12 ***(Data point's weight coefficient).*** *For each data point x in the data*

stream, a weight coefficient, w_x, is assigned which decreases over time. If x arrives at time t, its weight coefficient at t_c is ($t_c > t$):

$$w_x(t_c,t) = 2^{-\lambda(t_c-t)} \tag{4.5}$$

The initial value, w_c, of each data point is 1.

Definition 13 ***(Grid weight).*** *For a grid g at current time t_c, the grid weight is defined based on sum of the weight coefficients of data points which are mapped to it:*

$$w_g(t_c) = \sum_{x \in g} 2^{-\lambda(t_c-t_x)} \tag{4.6}$$

Definition 14 ***(Grid weight update).*** *The grid weight is updated in t_c with the last updated value t_p as follows ($t_c > t_p$):*

$$w_g(t_p,t_c) = 2^{-\lambda(t_c-t_p)} * w_g(t_p) + 1 \tag{4.7}$$

This saves the computation time. When a new data point arrives, it is only needed to update the grid weight of the grid cell which the data is mapped to it. So, all other grids' weights are not required to be updated.

Lemma 1 *The maximum weight, w_{max}, of all data points is $\frac{1}{1-2^{-\lambda}}$.*

Proof. Assume that all the data points of the data stream are mapped to the same grid cell. Therefore, according to Definition 13, we have $w_g(t) = \sum_{t'=0}^{t} 2^{-\lambda(t-t')}$ which can be transformed with the sum formula for geometric series as following:

$$w_g(t) = \sum_{t'=0}^{t} 2^{-\lambda(t-t')} = \frac{1-2^{-\lambda(t+1)}}{1-2^{-\lambda}} \tag{4.8}$$

Thus, the maximum weight of a grid g is:

$$w_{max} = lim_{t\to\infty}\frac{1-2^{-\lambda(t+1)}}{1-2^{-\lambda}} = \frac{1}{1-2^{-\lambda}} \tag{4.9}$$

Average grid density: The sum of all data points' weights has an upper bound of $\frac{1}{1-2^{-\lambda}}$. Since we have the total number of N density grids (according to Definition 11), the average density of each grid is $\frac{1}{N(1-2^{-\lambda})}$.

Definition 15 ***(Grid synopsis).*** *The grid synopsis of a grid g is a tuple* $GS(n_g, t_p, w_g)$ *where,*

- n_g *is the number of data points inside the grid,*
- t_p *is the last updated timestamp of the grid, and*
- w_g *is the grid weight.*

Grid synopsis keeps summary information about the data points inside the grid. When a new data point is added to a grid cell, its grid synopsis is updated. In fact, instead of saving timestamps and all its data points' weights, it is adequate to save the grid synopsis.

Definition 16 ***(Dense grid).*** *Grid g is dense at time t if*

$$w_g(t) >= \frac{\alpha}{N(1-2^{-\lambda})} \tag{4.10}$$

Because the overall weight cannot be more than $\frac{1}{N(1-2^{-\lambda})}$, α is a controlling threshold.

Definition 17 ***(Mini core distance (mcd)).*** *In grid g with* $p_1, p_2, \ldots, p_n$ *data points, mini core distance is the maximum distance from the mean of all the data points in the grid as*

o to all other neighborhoods.

$$\forall p \in \{neighbors_g(o)\} = Maximum(distance(o,p)), \tag{4.11}$$

$$|neighbors_g(o)| >= MinPts$$

$\{neighbors_g(o)\}$ is a set of all the data points inside cell boundaries of a grid g which includes point o.

For the data points are on a border of a grid, we assign them to the neighboring grid with higher density.

Definition 18 ***(Core mini cluster (cmc)).*** *A cmc at time t is defined as $CMC(w,c,r,mcd)$ for a group of very close data points $p_{i1} \ldots p_{in}$ with timestamp $T_{i1}, \ldots, T_{in}$ as follows:*

- $w_{cmc} = w_g$,
- $c_{cmc} = \frac{\sum_{j=1}^{n} 2^{-\lambda(t-T_{ij})}(p_{ij})}{w_{cmc}}$,
- $r_{cmc} = \frac{\sum_{j=1}^{n} 2^{-\lambda(t-T_{ij})} distance(c_{cmc}, p_{ij})}{w_{cmc}}$, $r_{cmc} \leq mcd_{cmc}$,
- $mcd_{cmc} = \forall p \in \{neighbors_g(c_{cmc})\} = Maximum(distance(c_{cmc}, p))$.

$distance(c_{cmc}, p_{ij})$ is an Euclidean distance between cmc's center and the data points in the grid.

Assume a core mini cluster $cmc_p(w_{cmc_p}, c_{cmc_p}, r_{cmc_p}, mcd_{cmc_p})$. If a data point p is merged to it the core mini cluster is updated as follows:

$$cmc_p = (2^{-\lambda(t_c - t_p)} w_{cmc_p}(t_p) + 1, c_{cmc_p} + p^2, r_{cmc_p} + p, mcd_{cmc_p})$$

4.7 Online Phase of MuDi-Stream Algorithm

Prior to the arrival of the very first data point, an empty grid list is initialized.

The components that are designed for this phase are described as follows.

4.7.1 MM-Component (Merging and Mapping)

In order to discover the clusters in an evolving data stream, a set of core mini clusters *cmc*s are maintained. Furthermore, all the outliers are maintained in density grids which is in a separate memory space. In this component, two important concepts are core mini clusters and density grids. The structures of the core mini clusters and the grids keep sufficient information for final clustering. Merging and mapping are the major tasks in MM-component. Based on the data attributes, the decision is made either to merge the data points to existing core mini clusters or to map them to the grids.

In MM-component, when a new data point arrives, the component tries to add it to a core mini cluster if there is any in which the data point can fit in. However, if the data point cannot be added to any core mini cluster, it may be either a seed of a new core mini cluster or an outlier. This kind of data point is mapped to the grid in the outlier buffer and the decision is postponed to a later time.

In the initial arrival of data points from data streams, the majority of the points are mapped to the grids. However, when time passes, a number of core mini clusters are formed. MM-component works with FCM-component in the sense that whenever a grid's density is more than a predefined threshold, FCM-component is called and it forms a new core mini cluster.

The flow of the MM-component is described in details as follows. When a data point arrives, various tasks are invoked:

Task 1: Merging

1. If there is any core mini cluster (*cmc*), MuDi-Stream finds the nearest *cmc* to the

new data point. The nearest core mini cluster (cmc_s) is determined by the minimum distance between data point p and $center_{cmc}$. Core mini cluster (cmc) with the lowest distance is selected and denoted as cmc_s.

$$\forall cmc_i \in \{cmc\}, distance(p, center_{cmc_s}) = Min\{distance(p, center_{cmc_i})\} \quad (4.12)$$

2. If the new data point's distance to $center_{cmc_s}$ is less than mini core distance (mcd) of the nearest core mini cluster, it will be added to that particular core mini cluster (cmc). When the data point is added to a cmc_s, its center, radius, and weight are updated.

$$distance(p, center_{cmc_s}) \leq mcd_{cmc_s} \rightarrow cmc_s + p \quad (4.13)$$

Task 2: Mapping

3. Otherwise, the data point has to be mapped into the grid in the outlier buffer. A data point $p = (p_1, p_2, \ldots, p_d)$ is mapped to a density grid $g(p)$ as:

$$g(p) = (j_1, j_2, \ldots, j_d), p_i \in S_{i,j_i} \quad (4.14)$$

4. Update the grid synopsis $GS(n_g, t_p, w_g)$ of a grid g with its new values as follows:

$$n_g \leftarrow n_g + 1; \quad (4.15)$$

$$t_p \leftarrow t_c; \quad (4.16)$$

$$w_g(t_c) \leftarrow 2^{-\lambda(t_c - t_p)} w_g(t_p) + 1 \quad (4.17)$$

Algorithm 3 MergeMap(x,t_c,α,λ,N)

Input: a data point x from data stream
Input: current timestamp t_c
Input: density threshold α
Input: outlier threshold λ
Input: number of grid cells N
Output: *cmc* list
Output: grid list

1: $cmc_s \leftarrow$ find the nearest *cmc* to x in *cmc* list;
2: **if** distance $(x, center_{cmc_s}) \leq mcd_{cmc_s}$ **then**
3: $cmc_s \leftarrow cmc_s$+x; {Merge x to the *cmc*}
4: **else**
5: map x to the grid;
6: $n_g \leftarrow n_g + 1$;
7: $w_g \leftarrow 2^{-\lambda(t_c - t_p)} w_g(t_p) + 1$;
8: $t_p \leftarrow t_c$;
9: Update $GS(n_g, t_p, w_g)$;
10: **if** $n_g > 1$ and $w_g >= \frac{\alpha}{N(1-2^{-\lambda})}$ **then**
11: $cmc_{new} \leftarrow$ CreateNewCMC(g,t_c);
12: remove grid g from grid list;
13: **end if**
14: **end if**

5. Check number of data points in the grid cell and the cell's weight. If it has more than one data point and the grid cell's weight is more than the density threshold, the CreateNewCMC algorithm in FCM-component is invoked to form a new core mini cluster out of the grid cell.

MM-component of MuDi-Stream is shown in Algorithm 3. In the algorithm, which is called MergeMap Algorithm, the new data point is added to the nearest core mini cluster if it fits (Lines 1-3). Otherwise, it is mapped to the grid and the grid summary information is updated (Lines 5-9). At the same time if the grid's weight is more than the weight threshold and it has more than one data point, a new core micro cluster is formed from data points inside that grid (Lines 10-13).

The flowchart of MM-component is shown in Figure 4.3. Moreover, The procedure of MM-component is illustrated in Figure 4.4.

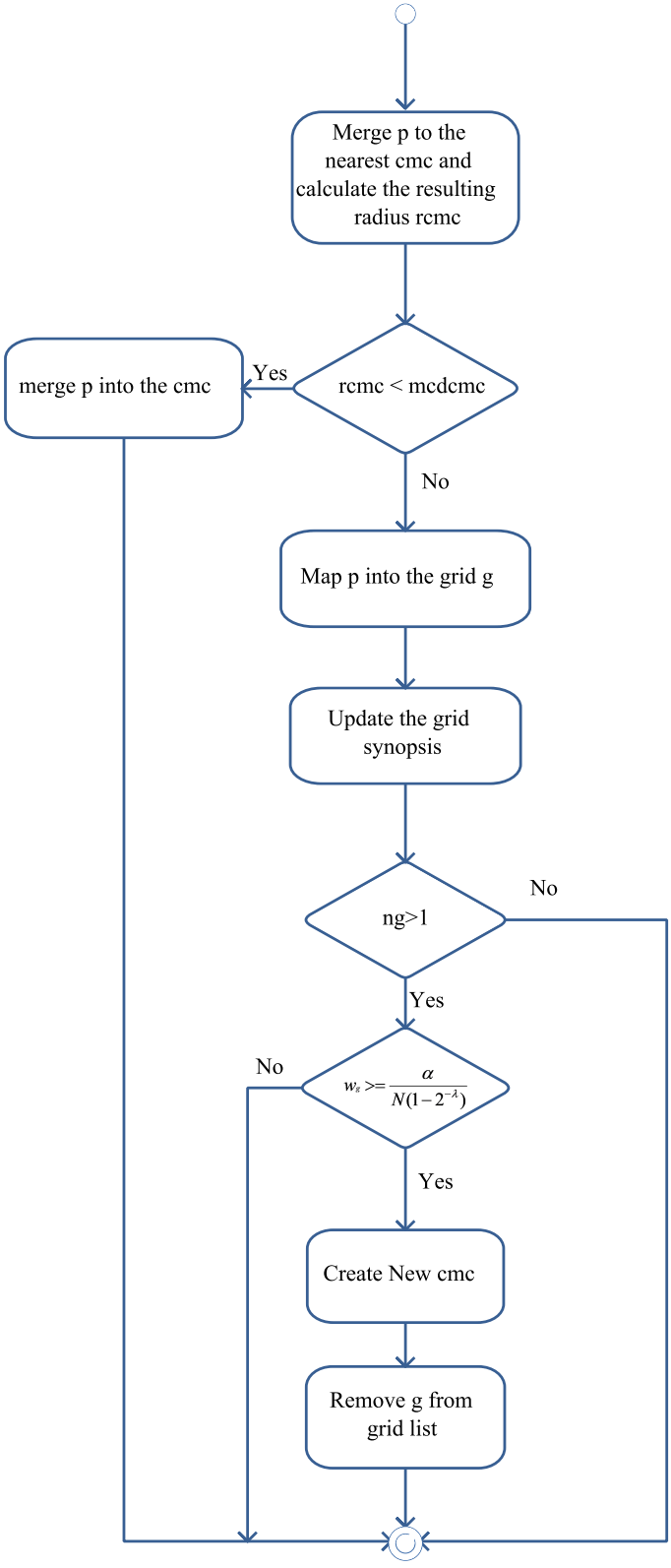


Figure 4.3: Flowchart for MM-Component and FCM-Component of MuDi-Stream

4.7.2 FCM-Component (Forming Core Mini Clusters)

Since data stream evolves over time, the number of data points inside the grids change over time. If the grid's weight is above a threshold, it means that it can form

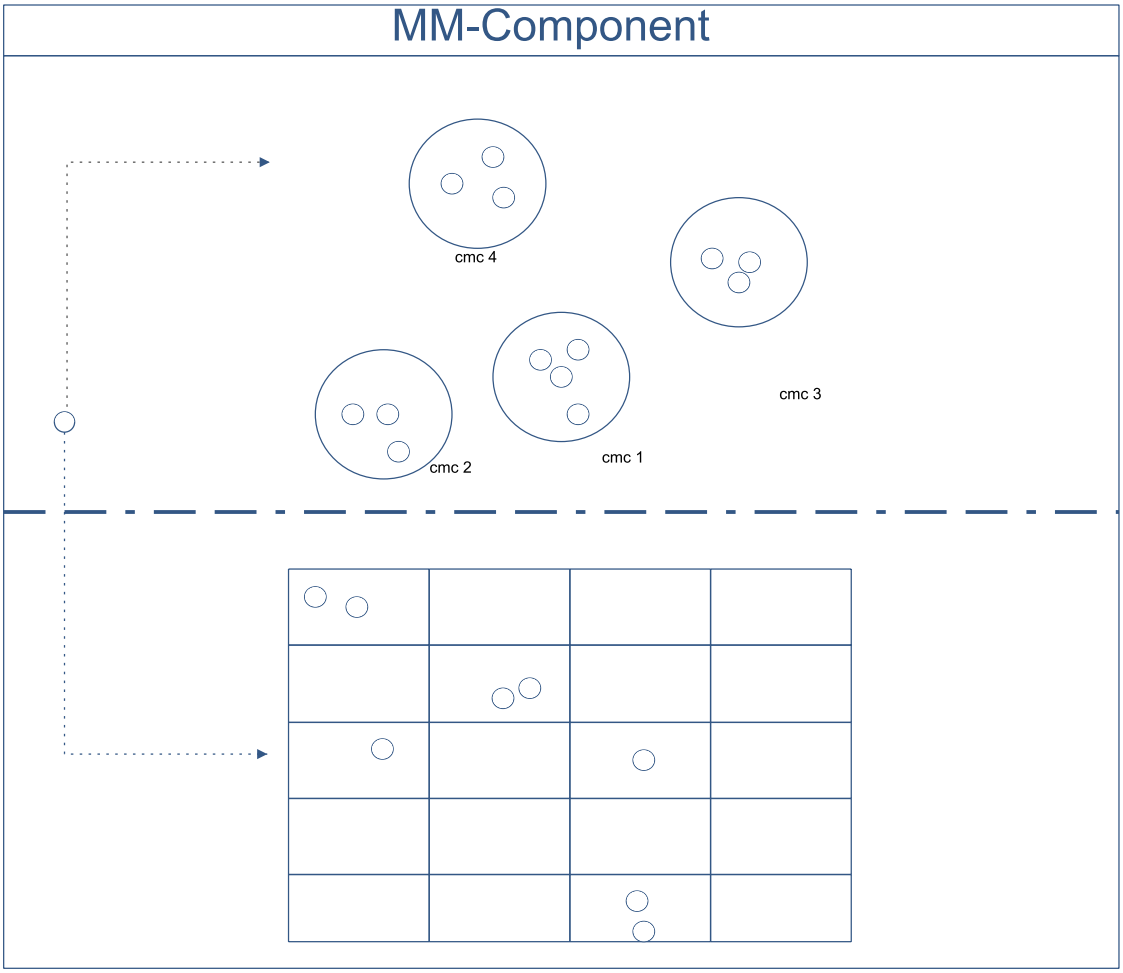


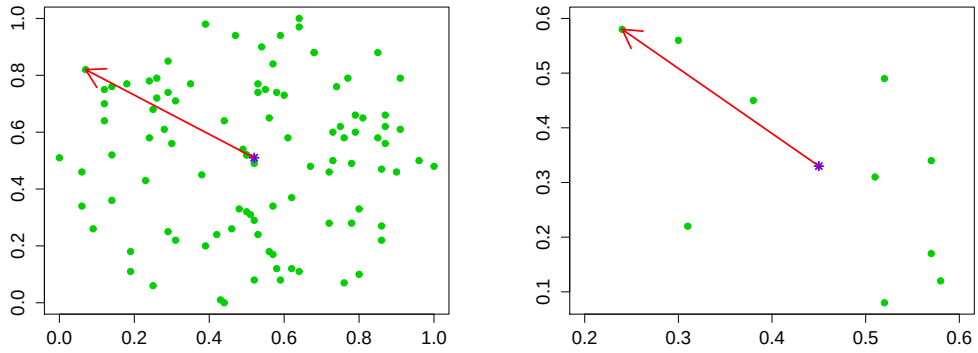
Figure 4.4: Merging to Existing Core Mini Clusters or Mapping to the Grid

a core mini cluster. Therefore, the data points are removed from the grid and a new core mini cluster is created.

Two important tasks in this component are 1) forming core mini clusters and 2) determining mini core distances.

In the former task, the grid weight is a main factor to make a decision regarding the generation of a new core mini cluster. If the grid weight is more than a threshold, it is not an outlier any more and it can form a new core mini cluster. This core mini cluster may attend in final clustering later.

The latter task determines mini core distance. This distance is used as radius threshold. For each core mini cluster, different radius thresholds are considered to have final clusters with different densities. Mini core distance *mcd* is maximum distance from cen-



(a) mcd=0.54, 100 data points (b) mcd=0.33, 10 data points

Figure 4.5: *mcd* in different data distributions inside a grid

ter to farthest point in the grid. Figure 4.5 shows an example in which how various values of *mcd* are for different distributions inside the grids.

This component is invoked by MM-component. Whenever a data point is mapped to the grid, if the conversion criteria is satisfied, the data points inside the grid form a core mini cluster.

The procedure of FCM-Component is explained as follows:

1. If the number of data points inside grid n_g is more than one, then we check the grid weight w_g with the density threshold. If the grid weight w_g is higher than the dense grid threshold then we form a new *cmc* out of the data points in this grid.

$$n_g > 1, w_g >= \frac{\alpha}{N(1-2^{-\lambda})} \tag{4.18}$$

2. The *cmc* is formed based on the data points inside the grid. *cmc* attribute values such as radius, center, weight and its *mcd* are determined as explained in Definition 18.

3. The related grid *g* of the new *cmc* is discarded from the grid list.

Algorithm 4 CreateNewCMC(g, t_c)

1: $w_{cmc} \leftarrow w_g$;
2: $c_{cmc} \leftarrow \frac{\sum_{i=1}^{n} f(t_c - T_i)(p_i)}{w_{cmc}}$;
3: $r_{cmc} = \frac{\sum_{i=1}^{n} f(t_c - T_i) distance(p_{ij}, c_{cmc})}{w_{cmc}}$
4: **for** data points p_i in the grid g **do**
5: $mcd_{cmc} \leftarrow$Maximum$\{distance(c_{cmc}, p_i)\}$;
6: **end for**
7: return $cmc(w_{cmc}, c_{cmc}, r_{cmc}, mcd_{cmc})$

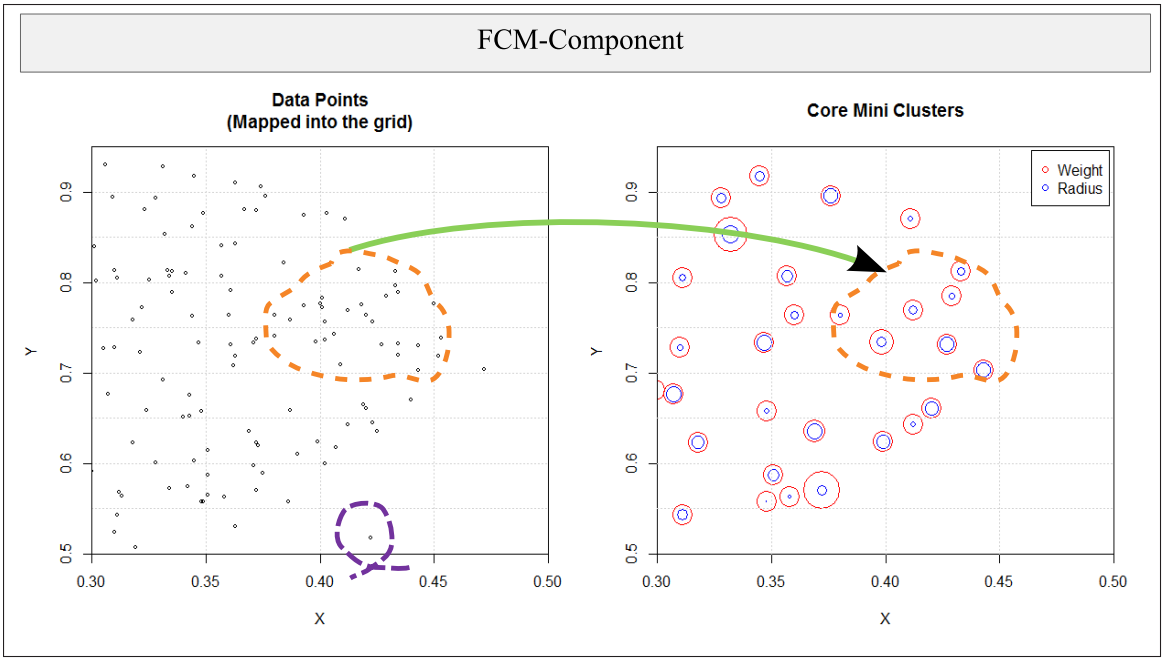


Figure 4.6: FCM-component: Forming Core Mini Clusters from Data Points inside Grid

FCM-component is outlined in Algorithm 4. In CreateNewCMC Algorithm, the grid weight is assigned as the weight for the new core mini cluster (Line 1). The center and radius is calculated based on the data points in the grid (Lines 2-3). Mini core distance is also calculated from the data points inside the grid. Each core mini cluster has its own mini core distance.

FCM-component is shown in Figure 4.6.

4.7.3 PGCM-Component (Pruning Grid and Core Mini Clusters)

Since data stream evolves over time, the role of real data is changed to outliers and vice versa. A technique is required to filter out outliers and detect the real data. This filtering should be performed on both the grid and the core mini clusters. The problem is

that the number of outliers may increase over time as data stream proceeds. It becomes worse when a lot of outliers are available in the data. Therefore, we need to periodically prune the real outliers from the the outlier buffer, i.e. the grid.

One of the challenges for the grid-based clustering method is the large number of grids, especially for high-dimensional data. For example, if the *gridGranularity* equals to 20, there will be 20^d possible grids. However, most of the grids in the space are empty or receive data very seldom. In our implementation, we allocate memory to store the *grid synopsis* for those grids that are not empty, which form a very small subset in the grid space. However, in practice, this is still not efficient enough since the appearance of outlier data which lead to continual increase of non-empty grids that will be processed during clustering. We call such grid as *scattered* grids since they contain very few data. Since a data stream flows in by huge volume in high speed and it could run for a very long time, due to noises in the data stream, more and more grids will be occupied during the process which many of them contain only very few data. If these *scattered* grids are not checked, the total number of grids in the grid list will keep increasing and become extremely large. Therefore, it is critical to detect and remove such scattered grids periodically.

Once a scattered grid is deleted, its density is reset to zero since its *grid synopsis* is deleted. A deleted grid may be added back to grid list if there are new data records mapped to it later, but its previous records are discarded and its density restarts from zero. Such a dynamic mechanism maintains a moderate size of the grids in memory, saves computing time, and prevents infinite accumulation of scattered grids in memory. We define a Outlier Weight Threshold (OWT) function for detecting scattered grids:

Definition 19 ***(Outlier Weight Threshold function (OWT)).*** *If the last updated time of a grid g is t_p then at current time t_c, the Outlier Weight Threshold (OWT) is defined as*

follows ($t_c > t_p$):

$$OWT(t_p, t_c) = \frac{\alpha}{N} \sum_{i=0}^{t_c - t_p} 2^{-\lambda i} = \frac{\alpha(1 - 2^{-\lambda(t_c - t_p + 1)})}{N(1 - 2^{-\lambda})} \tag{4.19}$$

See Appendix A for the proof.

Theorem. The size of grid list at most $L = \frac{1}{\lambda} \log_{\frac{\alpha}{N+\alpha}} N$ is the total number of grid and λ is the decay factor.

See Appendix A for the proof.

If a grid g is detected as a scattered grid, is it possible that g can be non-scattered. If it has not been previously deleted from grid list? It is answered in the following result.

Proposition 2. Assume the last time a grid g is deleted as scattered grid is t_k and the last updated of grid g is t_p. If at current time t, we have $w_g(t) < owt(t_p, t)$, then we also have $w_g(t) < owt(0, t)$

See Appendix A for the proof.

Proposition 2 is important because it shows that deleting a scattered grid will not cause a dense grid be falsely deleted. It shows that, if g is deleted as a scattered grid at t since $w_g(t) < owt(t_p, t)$, then even if all the previous deletions have not happened, it is still scattered and cannot be a dense grid.

The important issue is the length of the time interval for grid density checking. This time which we referred as pruning time cannot be too large or too small. If pruning time is too large, dynamical changes of data streams will not be properly recognized. If pruning time is too small, it will result in frequent computation by the offline component. Therefore, the processing speed of the offline component may not compete the speed of the input data stream.

Furthermore, since our method gradually reduces the weight of the data points. If a core mini cluster does not receive any data for a long time, it has to be removed form

Algorithm 5 Pruning $(\{cmc\}, g, t_c, \alpha, \lambda, N)$

Input: $\{cmc\}$
Input: $\{g\}$
Output: $\{cmc\}$

update the weights of all grids in grid list ($w_g(t_c) = 2^{-\lambda(t_c - t_p)} * w_g(t_p)$);
for all grid g **do**
  $OWT(t_c, t_p) \leftarrow \frac{\alpha(1-2^{-\lambda(t_c-t_p+1)})}{N(1-2^{-\lambda t_p})}$;
  if $w_g < OWT$ **then**
    remove grid g from the grid list;
  end if
end for
for all $\{cmc\}$ **do**
  if $w_{cmc} < \frac{\alpha}{N(1-2^{-\lambda})}$ **then**
    remove cmc from $\{cmc\}$;
  end if
end for

the list. A core mini clusters is formed from a grid with density higher than $\frac{\alpha}{N(1-2^{-\lambda})}$, therefore, if the density is less than dense grid threshold it should be removed form the list.

Therefore, the density of grid and core mini clusters should be inspected after a period of time. The pruning time is considered as the time needed for a core mini cluster to convert to an outlier which is defined as follows:

Definition 20 ***(Pruning time).*** *We check the cmc' weight as well as grids' in a specific time called t_{pt}. t_{pt} is the minimum time for a cmc in timestamp t_1 to be converted to an outlier in t_2 ($t_2 > t_1$), which is formally defined as follows:*

Lemma 2

$$t_{pt} = \frac{1}{\lambda} \log_2^{\frac{\alpha}{\alpha - 1 + 2^{-\lambda}}}$$

See Appendix A for the proof.

The main tasks of PGCM-component, which are performed in pruning time, are as follows:

1. Check the grid weight with *Outlier Weight Threshold function (OWT)*. If it is less

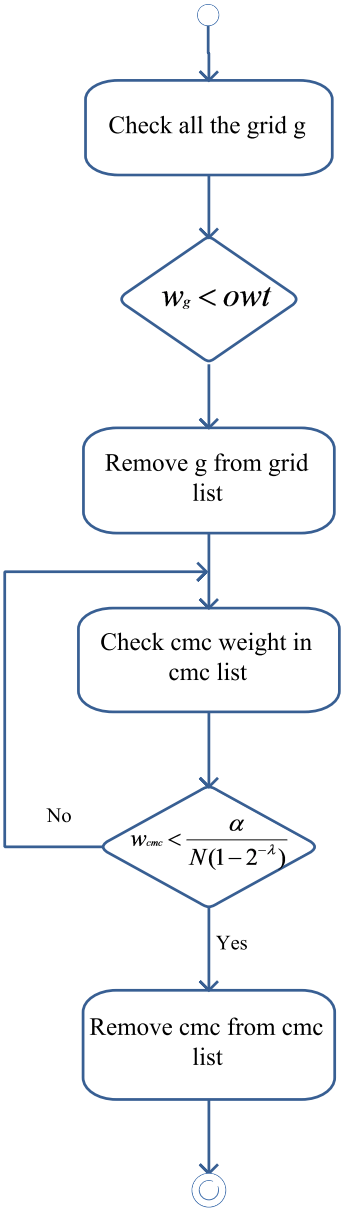


Figure 4.7: Flowchart of PGCM-Component of MuDi-Stream

than this threshold then the grid is detected as scattered grid and will be removed from the grid list.

$$RemovalList\{g\} \leftarrow w_g < OWT(t_p, t_c) \tag{4.20}$$

2. Check the weights of the *cmc* list, if there is any *cmc* with the weight less than the dense grid threshold, that particular *cmc* is removed from the *cmc* list.

$$RemovalList\{cmc\} \leftarrow w_{cmc} < \frac{\alpha}{N(1-2^{-\lambda})} \tag{4.21}$$

The PGCM-component's procedure is shown in Algorithm 5. In Algorithm 5 , the grid weights are updated by current time in Line 1. After that, in lines 2-7, the weight of all grid are checked to remove scattered grid. Finally, the removal list of core mini clusters are determined in lines 8-12 of algorithm. The flowchart is depicted in Figure 4.7 as well. Figure 4.8 depicts the pruning component.

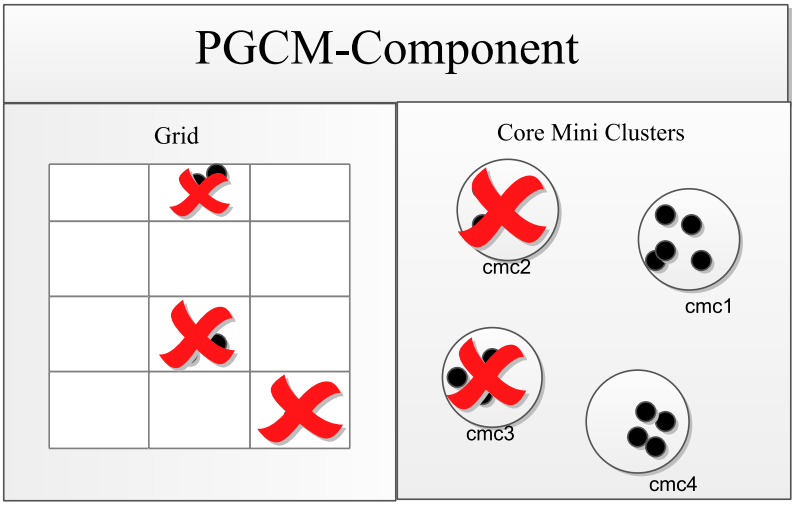


Figure 4.8: Pruning Grids and Core Mini Clusters

The online phase of MuDi-Stream with all its components are given in Algorithm 6.

Algorithm 6 MuDi-Stream Online Phase(DS, λ, α, N)

Input: a data stream
Output: core mini clusters

$t_{pt} \leftarrow \frac{1}{\lambda} \log_2^{\frac{\alpha}{\alpha-1+2^{-\lambda}}}$
$t_c \leftarrow 0$;
while not end of stream **do**
  Read data point x from Data Stream
  $cmc_s \leftarrow$ find the nearest cmc to x in cmc list;
  if distance $(x, c_{cmc_s}) \leq mcd_{cmc_s}$ **then**
    $cmc_s \leftarrow cmc_s$+x; {/*Merge x to the cmc*/}
  else
    map the new data point x to the grid;
    $n_g \leftarrow n_g + 1$;
    $w_g \leftarrow 2^{-\lambda(t_c - t_p)} w_g(t_p) + 1$;
    $t_p \leftarrow t_c$;
    Update $GS(n_g, t_p, w_g)$;
    if $n_g > 1$ and $w_g >= \frac{\alpha}{N(1-2^{-\lambda})}$ **then**
      $w_{cmc} \leftarrow w_g$;
      $c_{cmc} \leftarrow \frac{\sum_{i=1}^{n} f(t_c - T_i)(p_i)}{w_{cmc}}$;
      $r_{cmc} \leftarrow \frac{\sum_{i=1}^{n} f(t_c - T_i) distance(p_{ij}, c_{cmc})}{w_{cmc}}$
      for data points p_i in the grid g **do**
        $mcd_{cmc} \leftarrow$ Maximum$\{distance(c_{cmc}, p_i)\}$;
      end for
    end if
  end if
  if t_c mod $t_{pt} == 0$ **then**
    update the weight of all grids in grid list $(w_g(t_c) = 2^{-\lambda(t_c - t_p)} * w_g(t_p))$;
    for all grid g **do**
      $OWT(t_c, t_p) \leftarrow \frac{\alpha(1-2^{-\lambda(t_c - t_p + 1)})}{N(1-2^{-\lambda})}$;
      if $w_g < OWT(t_c, t_p)$ **then**
        remove grid g from the grid list;
      end if
    end for
    update the weight of all core micro clusters $(w_{cmc}(t_c) = 2^{-\lambda(t_c - t_{cmc})} * w_{cmc}(t_c mc))$;
    for all $\{cmc\}$ **do**
      if $w_{cmc} < \frac{\alpha}{N(1-2^{-\lambda})}$ **then**
        remove cmc from $\{cmc\}$;
      end if
    end for
  end if
  $t_c \leftarrow t_c + 1$;
end while

4.8 Offline Phase of MuDi-Stream Algorithm

The offline phase has one component called FFC-component. An adapted density based algorithm is proposed in this phase to form arbitrary multi-density shape clusters from synopsis data.

4.8.1 FFC-Component (Forming Final Cluster)

In FFC-component, a new density-based clustering, called M-DBSCAN is proposed which needs only one parameter (*MinPts*) to discover final clusters. The existing density-based data stream clustering algorithms apply DBSCAN in their offline phase; however, since it uses a global set of parameters, they cannot cover multi-density data. In our proposed algorithm, M-DBSCAN, instead of finding neighbors in constant radius ε, the neighboring radius is determined based on the distribution of data around the core using mean and standard deviation values.

When a clustering request arrives, M-DBSCAN algorithm is applied on the set of online maintained core mini clusters to get the clustering result. Each core mini cluster *cmc* is considered as a virtual point located at the center of *cmc*. M-DBSCAN replaces the ε values of DBSCAN by local cluster density. In M-DBSCAN, the *core-neighboring* concept is introduced in which core mini clusters are added to existing clusters if they have similar values of mean with some acceptable difference defined by standard deviation of the core.

Using online phase information and statistical analysis of the distribution of data inside core mini clusters, M-DBSCAN is able to generate clusters with different densities.

Some new concepts are applied in this component which are described as follows:

Definition 21 ***(Neighboring grids*** (N_g)***).*** *Two density grids* $g_1 = (j_1^1, j_2^1, \ldots, j_d^1)$ *and* $g_2 = (j_1^2, j_2^2, \ldots, j_d^2)$ *are neighbors if there exists m,* $1 \leq m \leq d$ *such that:*

- $j_i^1 = j_i^2, \quad i = 1, \ldots, m-1, m+1, \ldots, d$*; and*

- $|j_m^1 - j_m^2| = 1$

Then g_1 and g_2 are neighboring grids in the m^{th} dimension.

Definition 22 ***(cmc-grid-neighborhoods ($N_g(cmc)$)).*** *All core mini clusters which are placed in the neighboring grids of g.*

$$\left\{N_g(cmc)\right\} \leftarrow \forall cmc_p \in \left\{N_g\right\} \tag{4.22}$$

Definition 23 ***(MinPts-nearest-neighbors ($N_{sh}(cmc)$)).*** *In order to determine MinPts-nearest-neighbor for the cmc_p, firstly the distance from cmc_p to all cmc_p-grid-neighborhoods are calculated. After that, MinPts neighbors are selected with minimum distances.*

$$\left\{N_{sh}(cmc_q)\right\} \leftarrow Minimum(distance(cmc_p, N_g(cmc_p))), |N_{sh}(cmc_q)| \geq MinPts \tag{4.23}$$

Definition 24 ***(Core-neighboring (N_{core})).*** *A core mini cluster with its MinPts-nearest-neighbors become core-neighboring if the following condition is satisfied:*

$$\{N_{core}\} \leftarrow \forall cmc_q \in \left\{N_{sh}(cmc_p)\right\},$$

$$\mu(Dist_{cmc_q}) \in [\mu(Dist_{core}) - \sigma(Dist_{core}), \mu(Dist_{core}) + \sigma(Dist_{core})] \tag{4.24}$$

$$\mu(Dist_{core}) \leftarrow \mu(distance(N_{sh}(cmc_p), cmc_p))$$

$$\sigma(Dist_{core}) \leftarrow \sigma(distance(N_{sh}(cmc_p), cmc_p))$$

$$\mu(Dist_{cmc_q}) \leftarrow \mu(distance(N_{sh}(cmc_q), cmc_q))$$

$$\sigma(Dist_{cmc_q}) \leftarrow \sigma(distance(N_{sh}(cmc_q), cmc_q))$$

The new definition of core-neighboring expresses that, in order to be in the list of core neighbors, the values of mean of distance from core mini cluster to its neighbors should lie within a certain range from core.

FCC-component works as follows:

- Initially, all core mini clusters are marked as "unvisited." M-DBSCAN randomly chooses an unvisited object, a core mini cluster (cmc_p), and marks it as "visited." Then, it checks whether its *cmc-grid-neighborhood* contains at least *MinPts*. If not, mark the core mini cluster cmc_p as noise. Otherwise, a new cluster is created and the core mini cluster cmc_p is added to that cluster. After that, the algorithm finds the *MinPts-nearest-neighborhood* of core mini cluster cmc_p. This gives us a shorted list of the neighbors. This technique is a kind of filtering which prevents formation of a single cluster out of multiple close dense clusters.

- Next, the Euclidean distances from the core mini cluster cmc_p to all its shorted list neighbors are calculated. Mean and standard deviation of these distances are determined as well. A new core mini cluster with its neighbors are added to the existing cluster if the condition of a *core-neighboring* is satisfied.

- The algorithm continues with the unvisited neighbors of core mini cluster cmc_p. The core-neighboring list, mean and standard deviation are updated whenever a new core mini cluster and its neighbors are added to an existing cluster. M-DBSCAN adds core mini clusters until the cluster cannot be expanded any more. Therefore, the cluster is complete. To find the next cluster, M-DBSCAN randomly selects another unvisited core mini cluster from the remaining ones. The clustering process terminates when all core mini clusters are visited.

Algorithm 7 M-DBSCAN($MinPts$,g, $\{cmc\}$)- MuDi-Stream's Offline Phase

1: mark all *cmc*s as unvisited;
2: **repeat**
3: randomly choose an unvisited cmc_p;
4: mark cmc_p as visited;
5: $\{N_g(cmc_p)\} \leftarrow cmc_p$-grid-neighborhood
6: **if** $|\{N_g(cmc_p)\}| \geq MinPts$ **then**
7: create a new cluster C, and add cmc_p to C;
8: $\{N_{core}\} \leftarrow$ find $MinPts$-nearest neighbors in $\{N_g(cmc_p)\}$ from cmc_p
9: calculate $\mu(Dist_{core})$, and $\sigma(Dist_{core})$
10: **for** each cmc_q in $\{N_{core}\}$ **do**
11: **if** cmc_q is unvisited **then**
12: mark cmc_q as visited;
13: $\{N_g(cmc_q)\} \leftarrow cmc_q$-grid-neighborhood
14: **if** $|N_g(cmc_q)| \geq MinPts$ **then**
15: $\{N_{sh}(cmc_q)\} \leftarrow$ find $MinPts$-nearest-neighbors in $\{N_g(cmc_q)\}$ from cmc_q
16: calculate $\mu(Dist_{cmc_q})$, and $\sigma(Dist_{cmc_q})$
17: **if** $\mu(Dist_{cmc_q}) \in [\mu(Dist_{core}) - \sigma(Dist_{core}), \mu(Dist_{core}) + \sigma(Dist_{core})]$ **then**
18: $\{N_{core}\} \leftarrow \{N_{core}\} \bigcup \{N_{sh}(cmc_q)\}$;
19: update $\mu(Dist_{core})$, and $\sigma(Dist_{core})$;
20: **end if**
21: **end if**
22: **end if**
23: **if** cmc_q is not assigned to any cluster **then**
24: add cmc_q to cluster C;
25: **end if**
26: **end for**
27: **else**
28: mark cmc_p as noise;
29: **end if**
30: **until** no *cmc* is unvisited;

The procedure is outlined in Algorithm 7 and the FCC-component is depicted in Figure 4.9.

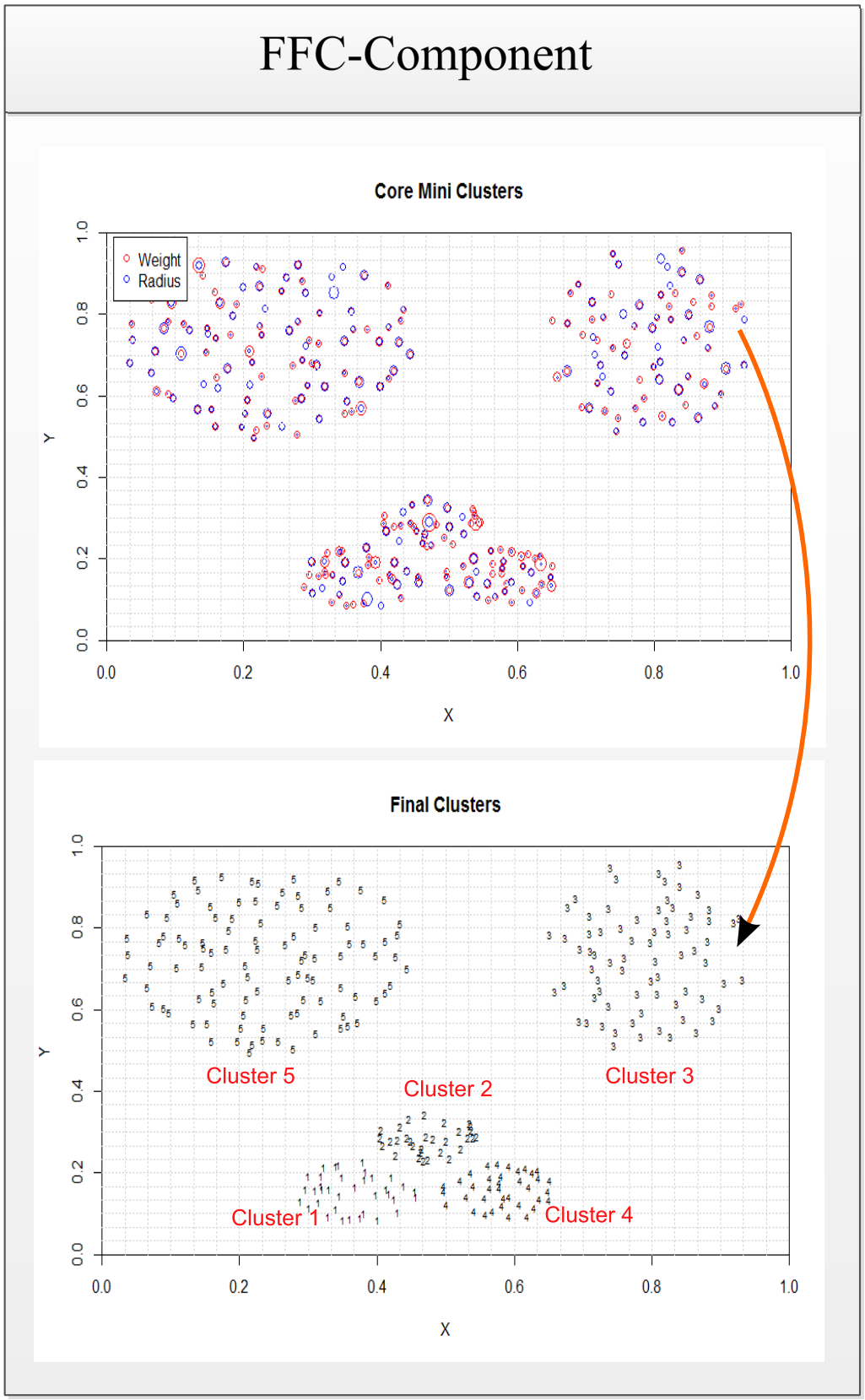


Figure 4.9: Forming Final Clusters from Pruned Core Mini Clusters

4.9 Summary

In this chapter, we explained the proposed method, MuDi-Stream, in details. The algorithm has online and offline phases and has the ability to cluster multi-density data with low computation time. MuDi-Stream is using a hybrid method in its online phase and a multi-density method in its offline phase.

The algorithm has four different components. The components of the online phase are MM-component, FCM-component, and PGCM-component. MM-component has a hybrid method using grid and micro clustering methods. It has a list of core mini clusters and also a grid as the outlier buffer. If the new data point cannot be merged to any existing core mini cluster, it is mapped to the grid. FCM-component works simultaneously with MM-component. If the density of grid in the outlier buffer is more than a density threshold, it is converted to a core mini cluster by the FCM-component. Data stream evolves over time, therefore some grids may not receive any data for a long time or a core micro clusters' weight gradually decreases over time. PGCM-component has a technique to check the weight of these synopsis data. M-DBSCAN algorithm in FFC-component in the offline phase is proposed to cluster synopsis data in order to get final clusters. M-DBSCAN has the ability to cluster multi-density data using information about density distribution of data points in the stream.

MuDi-Stream can effectively handle noise by mapping them in the grid. These outliers may either be changed to core mini clusters or be removed form the grid cells. Furthermore, it has the ability to handle evolving data stream by considering weight coefficient which decreases over time. Another prominent feature of MuDi-Stream is clustering multi-density data by keeping information in the online phase and using them in a new method in the offline phase. It has also low computation time since instead of searching in the outlier list of micro clusters, it maps the outliers into the grids.

CHAPTER 5

EXPERIMENTAL EVALUATION AND ANALYSIS

5.1 Overview

In this chapter, datasets which are used for evaluation purposes are introduced. Some real datasets and further some synthetically generated datasets are used to evaluate the proposed method. The real datasets are selected based on the literature. These datasets are the most applicable ones in evaluation of both data stream clustering and multi-density data clustering. The synthetic datasets are generated with different number of data points, various clusters and densities based on existing datasets in the literature. These real and synthetic datasets are fair benchmarks for evaluating the performance of the proposed approach, MuDi-Stream, w.r.t. state-of-the-art methods.

Evaluation of the clustering results is one of the difficulties in data stream clustering. As it is explained in Chapter 2, there are some evaluation metrics for measuring clustering quality. The evaluation process of MuDi-Stream includes:

- **Quality evaluation**: The quality of clustering results is measured using seven quality metrics on ten different datasets. The results are explained in Sections 5.3, 5.4, and 5.5.

- **Complexity analysis**: Time and space complexity of MuDi-Stream are measured and the results are discussed in Section 5.6.

- **Scalability evaluation**: Section 5.7 describes the scalability results which are measured in terms of execution time and memory usage of MuDi-Stream algorithm.

- **Sensitivity evaluation**: A comprehensive analysis of MuDi-Stream's parameters which affect the clustering results, is elaborated in Section 5.8.

5.2 Experimental Setup

5.2.1 Datasets

In this section, we introduce the datasets which are applied to show the effectiveness of the proposed approach for handling evolving multi-density data streams. The evaluations of the proposed method are performed on real and synthetic datasets.

First, the proposed method is evaluated on synthetic datasets. Different synthetic datasets are generated with various number of data points and clusters by considering noise. Synthetic dataset generation is based on the reviewed papers from the literature (Xiong et al., 2012; Huang et al., 2009; Xiaoyun et al., 2008; X. Li et al., 2010; Carmelo et al., 2013; Cao et al., 2006). Furthermore, some of them are generated based on different distributions of multi-density datasets. Synthetic datasets are generated based on normal and Gaussian distributions. Additionally, some of the synthetic datasets are combined to simulate evolving data stream over time. All the synthetic datasets except the Gaussian one are generated using a program which is written in Java. The program has the ability to generate any kind of dataset with different numbers of data points, clusters, and density distributions.

Furthermore, real world datasets of various characteristics are tested which are taken from different sources (Cao et al., 2006; Forestiero et al., 2013; Y. Chen & Tu, 2007; X. Li et al., 2010). The datasets have different number of clusters, densities, and data points. Table 5.1 lists the applied datasets for the experiments.

5.2.1 (a) Mashaal Dataset (DS1)

This dataset is generated similar to (Forestiero et al., 2013; Cao et al., 2006) which is applied to evaluate the quality of clustering. It contains 10000 data points with 5% noise.

Table 5.1: List of Datasets

No	Dataset	Size	Features	Classes	Type	Ref
1	Mashaal Dataset (DS1)	10000	2	4	Synthetic	(Cao et al., 2006)
2	Smile Dataset (DS2)	10000	2	4	Synthetic	citeForestiero13
3	FourCircles Dataset (DS3)	10000	2	4	Synthetic	(Forestiero et al., 2013)
4	Evolving Data Stream (EDS)	30000	12	2	Synthetic	-
5	Multi Density Dataset (MDS1)	12131	5	2	Synthetic	-
6	Multi Density Dataset-House (MDS2)	1097	5	2	Synthetic	(Mitra & Nandy, 2011)
7	Multi Density Dataset-5Cirlce (MDS3)	1360	5	2	Synthetic	(X. Chen et al., 2012)
8	Evolving Multi Density Data Stream (EMDS)	2457	10	2	Synthetic	-
9	Multi Density Cylinder-Cube (MDS4)	10000	3	3	Synthetic	-
10	Gaussian Multi Density Dataset (GMDS5)	10000	5	2	Synthetic	-
11	Network Intrusion Detection	424021	42	7	Real	(Frank & Asuncion, 2010)
12	Landsat Satellite Data	4435	36	6	Real	(X. Li et al., 2010)
13	Forest Cover Type	581012	54	7	Real	(Forestiero et al., 2013)

It has four classes with various shapes. The dataset is depicted in Figure 5.1.

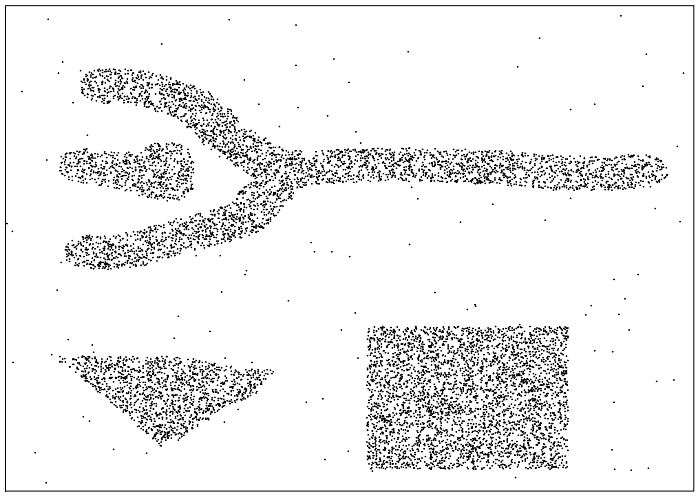


Figure 5.1: Mashaal Dataset (DS1) - 10000 data points, 3% noise

5.2.1 (b) Smile Dataset (DS2)

The smile dataset has 10000 data points with 4% noise which is chosen from (Forestiero et al., 2013; Cao et al., 2006). It has four arbitrary shape clusters. The dataset is shown in Figure 5.2.

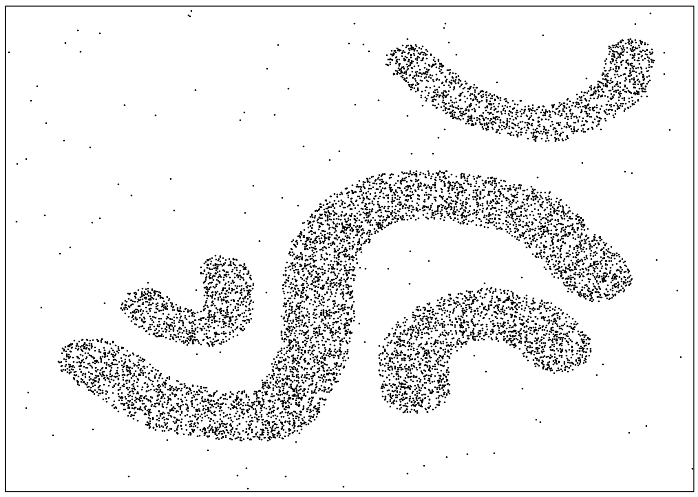


Figure 5.2: Smile Dataset (DS2) - 10000 data points, 4% noise

5.2.1 (c) FourCircles Dataset (DS3)

Adopted from the works presented in (Ester et al., 1996; Cao et al., 2006; Forestiero et al., 2013), the FourCircle dataset is generated. It has 10000 data points with 5% noise. The dataset is depicted in Figure 5.3.

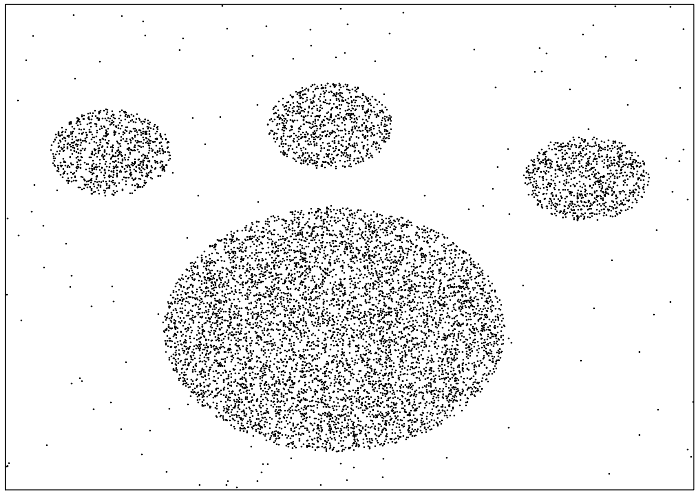


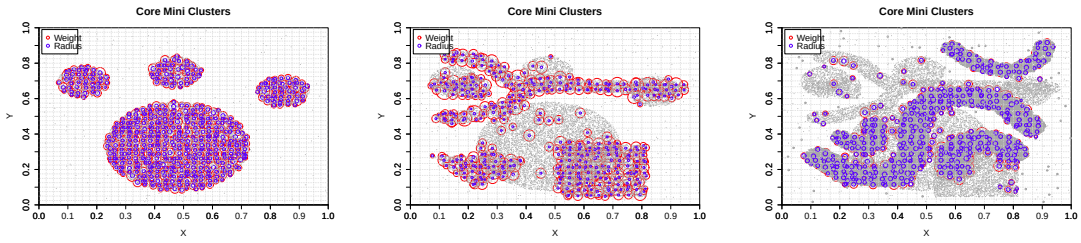
Figure 5.3: FourCircles Dataset (DS3) - 10000 data points, 5% noise

5.2.1 (d) Evolving Data Streams (EDS)

One of the important features of our proposed method is capturing clusters of evolving data streams. Therefore, we generated an evolving data stream (EDS) by randomly selecting one of the datasets (DS1, DS2 and DS3) (Cao et al., 2006; Forestiero et al., 2013). For each iteration, the chosen dataset forms a 10000 points part of the data stream, so the total length of the evolving data stream is 30000. Figure 5.4 shows the evolving data stream's dataset. It is depicted how the previous clusters disappear while the new arrival ones are the clusters which are identified using MuDi-Stream.

5.2.1 (e) Multi-density Dataset (MD1)

The aim of generating this dataset is to demonstrate how the proposed method can handle multi-density clusters. A two-dimensional well-separated dataset with 12131 data



(a) (b) (c)

Figure 5.4: Evolving Data Stream (EDS)

points is generated. It is composed of five clusters of different densities as shown in Figure 5.5.

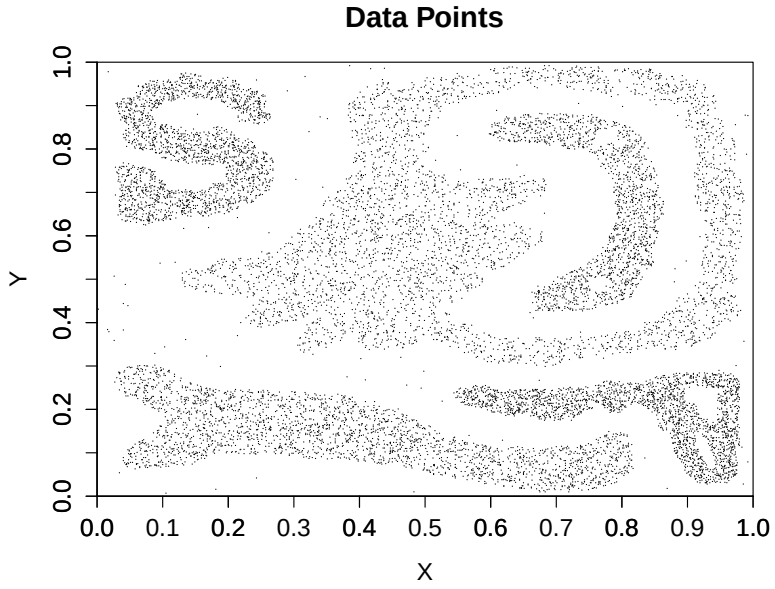


Figure 5.5: Multi-density Dataset (MDS1) - 12131 data points with 3% noise

5.2.1 (f) Multi-density Dataset - House (MDS2)

In this experiment, we generated a more complicated two-dimensional dataset to show the effectiveness of our algorithm. It has 1097 data points and five clusters with different densities with 4% noise which is adopted from (Mitra & Nandy, 2011). The MDS2 is depicted in Figure 5.6. The dataset consists of five clusters with different sizes, shapes, and densities.

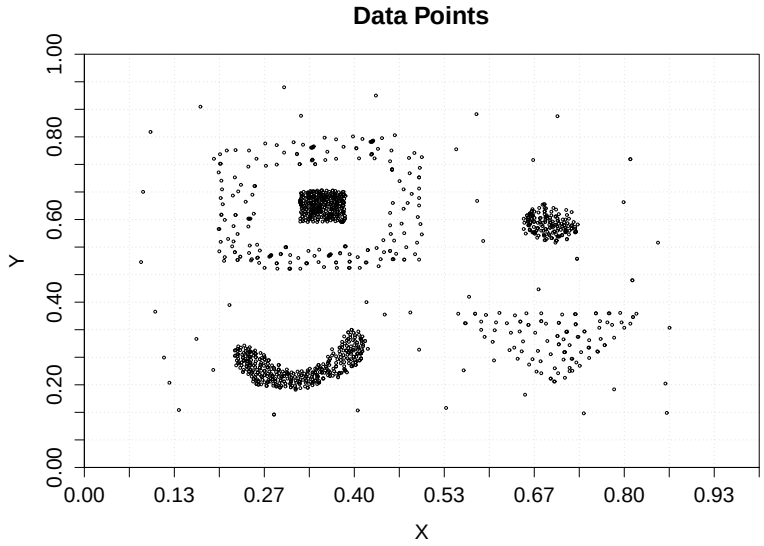


Figure 5.6: Multi-density Dataset (MDS2) - 1097 data points with 4% noise

5.2.1 (g) Multi-density Dataset - 5Circle (MDS3)

MDS3 dataset is generated with different density distributions. It is one of the datasets which is usually applied for the evaluation of method on multi-density datasets (X. Chen et al., 2012; Duan, Xu, Guo, Lee, & Yan, 2007). It has 1360 data points with 2% noise with five clusters. Figure 5.7 shows the MDS3 dataset.

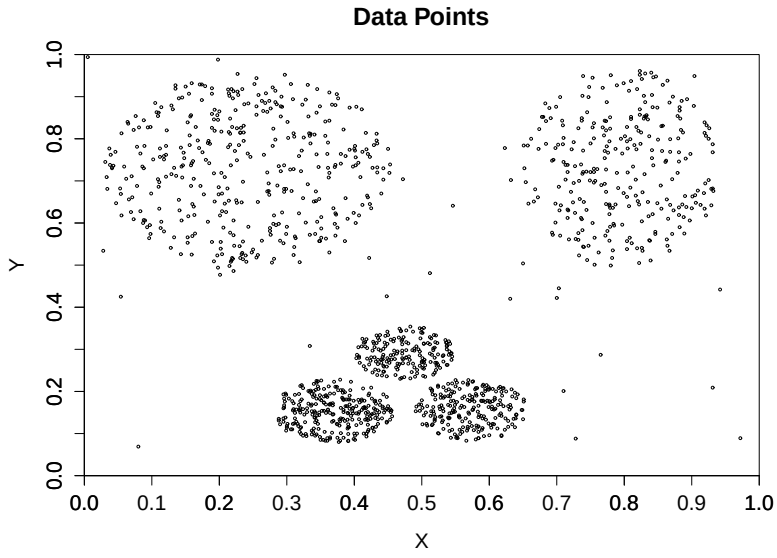
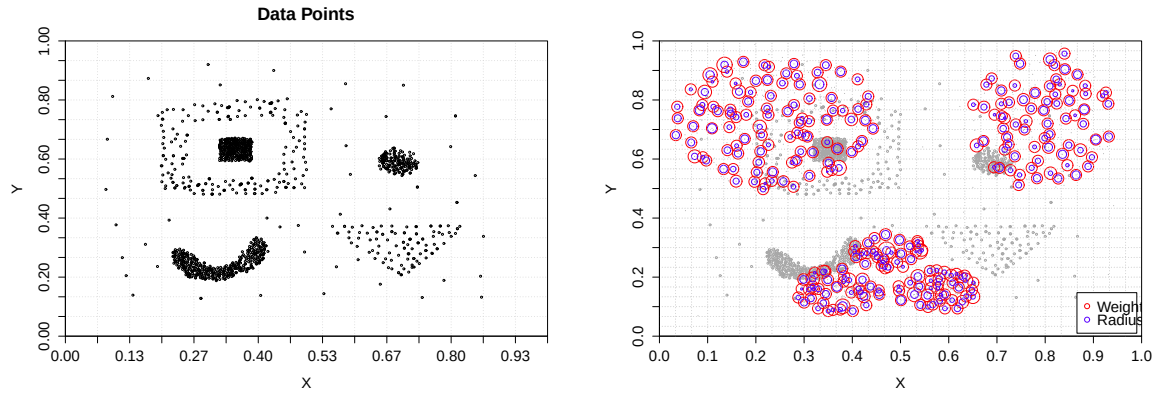


Figure 5.7: Multi-density Dataset - 5Circle (MDS3) - 1360 data points with 2% noise



(a) DS6, t=6 (b) Cmc, t=12

Figure 5.8: Evolving Multi-density Data Stream (EMDS)

5.2.1 (h) Evolving Multi-density Data Stream (EMDS)

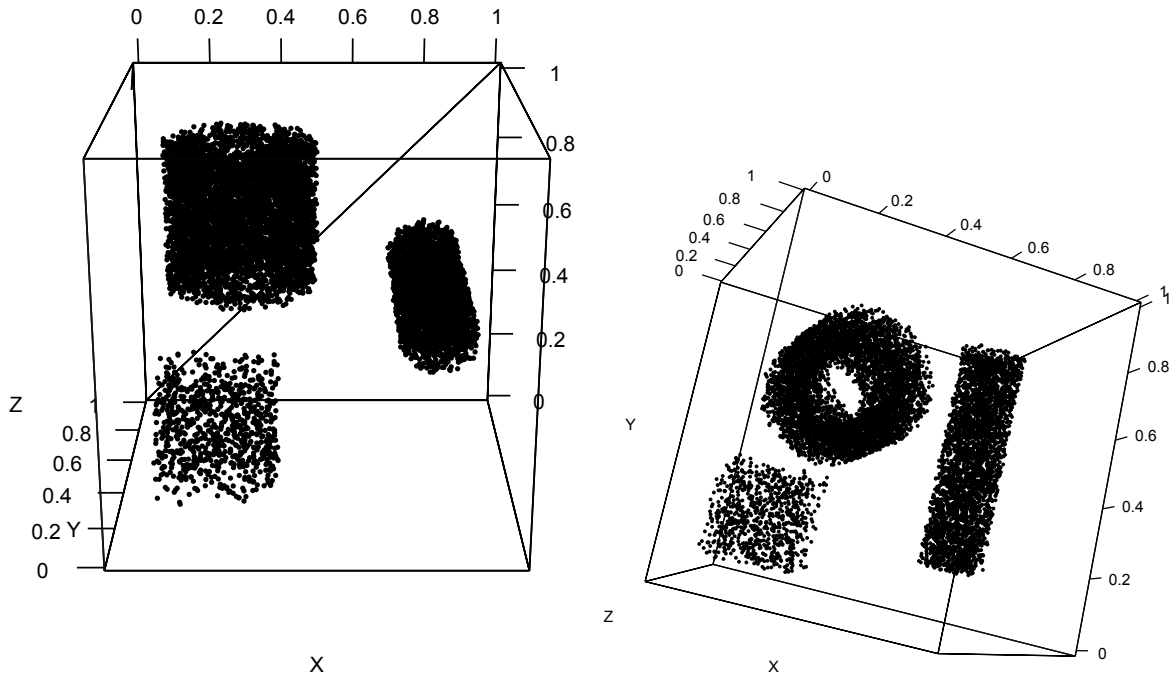
MD2 and MD3 datasets are also combined to simulate an evolving multi-density data stream (EMDS) over time (Figure 5.8). Therefore, we have a dataset with 2457 data points. This dataset is used to evaluate the clustering quality of MuDi-Stream in multi-density data which is evolving over time. Figure 5.8a shows MD2 at time 6 while Figure 5.8b depicts core-mini-clusters at time 12, when data stream evolves and MD3 data points arrive.

5.2.1 (i) Multi-density CylinderCube (MD4)

Multi-density CylinderCube (MD4) is another multi-density dataset with different density distributions. It has 10000 data points with three clusters and three dimensions without noise. In fact, we wanted to evaluate the ability of algorithm for a three-dimensional multi-density data without noise. Figures 5.9a, 5.9b depict MD4 from different perspectives.

5.2.1 (j) Gaussian Multi-density Dataset (GMDS)

A Gaussian dataset is also generated to evaluate the proposed method. It is generated to show the effectiveness of the proposed algorithm in Gaussian distributions. This dataset is generated using GaussianMoving() function in R language. GaussianMoving() is used



(a) (b)

Figure 5.9: Multi-density CylinderCube (MD4) - 10000 Points

for simulating evolving data stream in Gaussian distributions. Figure 5.10 shows the Gaussian dataset with 1000 data point with 3% noise and five clusters.

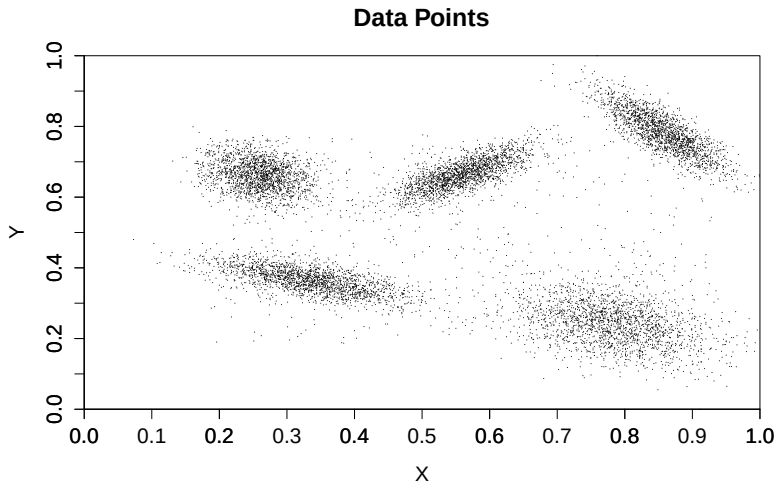


Figure 5.10: Gaussian Multi-density Dataset (GMDS) - 1000 data points with 3% noise

5.2.1 (k) Network Intrusion Detection

The Network Intrusion dataset (KDD Cup'99) (Frank & Asuncion, 2010) contains TCP connection logs from two weeks of LAN network traffic. The dataset comes from the 1998 DARPA Intrusion Detection. It contains training data consisting of 7 weeks

Table 5.2: List of Network Intrusion Detection features with their classes

Class Label	Relevant Features
Normal	1, 6, 12, 15, 16, 17, 18, 19, 31, 32, 37
Smurf	2, 3, 5, 23, 24, 27, 28, 36, 40, 41
neptune	4, 25, 26, 29, 30, 33, 34, 35, 38, 39
Land	7
teardrop	8
ftp_{write}	9
back	10,13
$guess_{pwd}$	11
$buffer_{overflow}$	14
warezclient	22

of network-based intrusions inserted in the normal data, and 2 weeks of network-based intrusions and normal data for a total of 4,999,000 connection records described by 42 characteristics. Each record corresponds to a normal connection or an attack. The attacks fall into 4 main categories and 22 more specific types: DOS (i.e., denial-of-service), R2L (i.e., unauthorized access from a remote machine), U2R (i.e., unauthorized access to local superuser privileges), and PROBING (i.e., surveillance and other probing). Three biggest classes Normal, Neptune, and Smurf appear in chunks, whereas smaller attack classes are scattered throughout the dataset.

All 34 continuous attributes of KDD CUP99 are used as in (Cao et al., 2006; Tu & Chen, 2009; Ntoutsi et al., 2012; Forestiero et al., 2013). It is converted into data stream by taking the data input order as the order of streaming. Figure 5.11 plots the distribution of data from KDD CUP99 for selected features. Figure 5.12 depicts class labels that appears in “10% KDD” dataset which is adopted from (Kayacik, Zincir-Heywood, & Heywood, 2005). Moreover, Table 5.2 shows the most discriminative class labels for each feature.

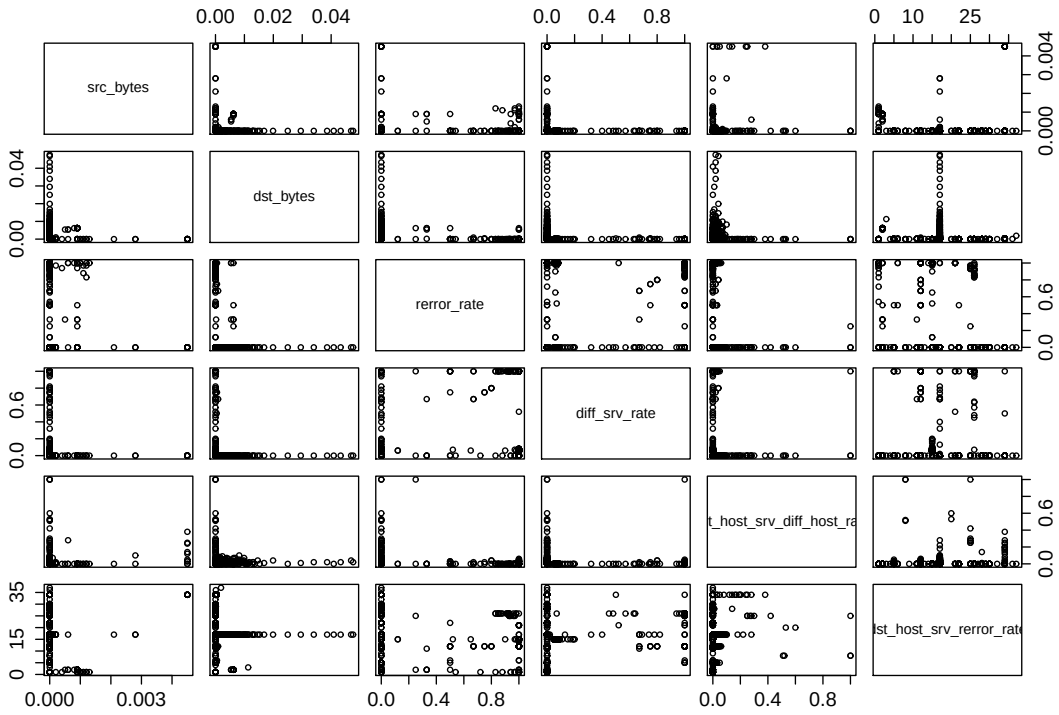


Figure 5.11: Data Distribution on KDD

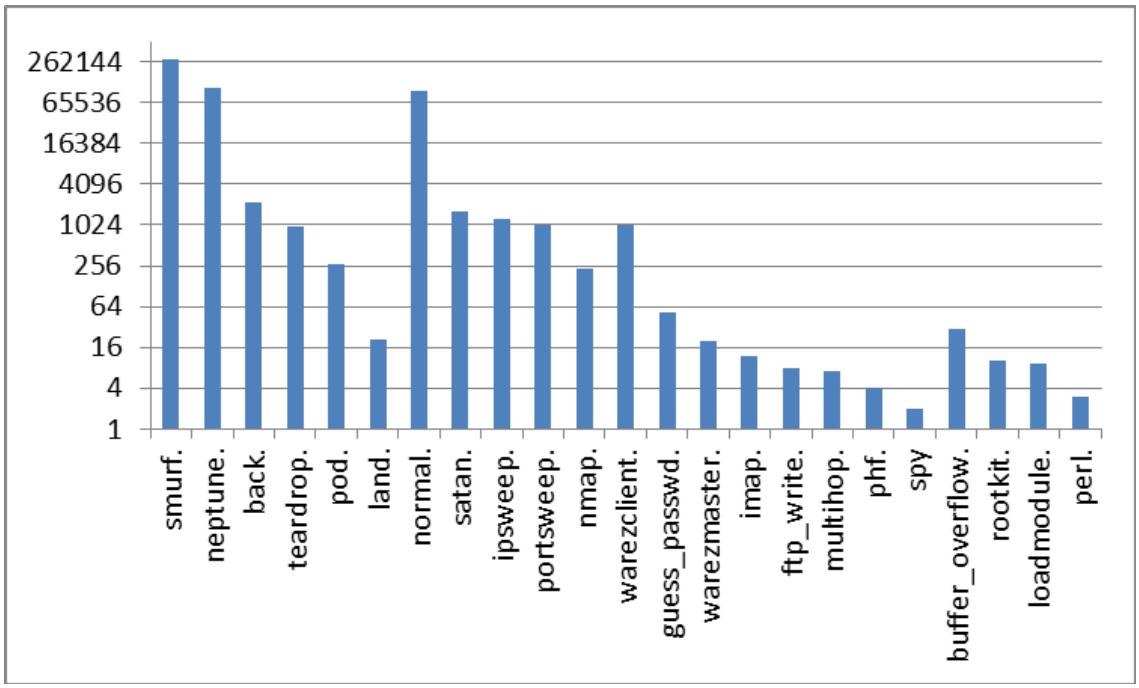


Figure 5.12: Class labels that appears in “10% KDD” dataset

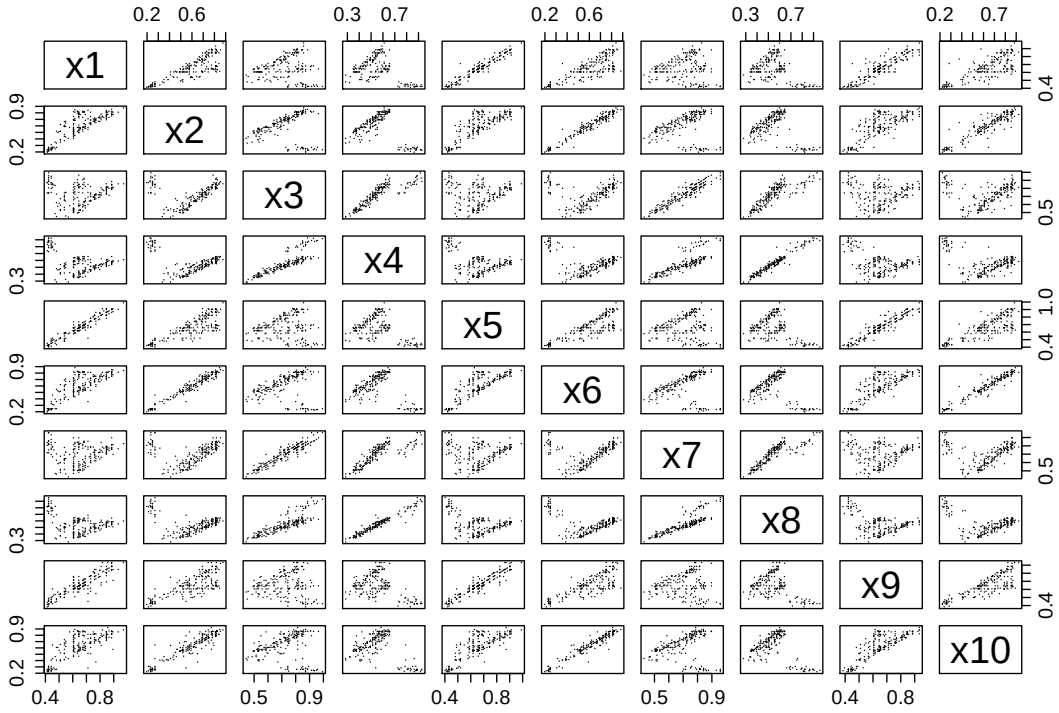


Figure 5.13: LandSat Dataset

5.2.1 (l) Landsat Satellite Data

Landsat Satellite Dataset from UCI Machine Learning Repository [1] is a dataset composed of 4435 objects. It is obtained from remote-sensing satellite images. Each object represents a region and each sub-region is recorded by the four measurements of intensity taken at different wavelengthes. Therefore, each object has 36 attributes. A class label indicating the type of the central sub-region is also given for each object. This dataset is used in (X. Li et al., 2010) to evaluate multi-density data. Figure 5.13 shows a sample of ten attributes of Landsat data points distribution. Furthermore, the class labels and the number of associated objects are shown in Figure 5.14.

5.2.1 (m) Forest Cover Type

The Forest Cover Type dataset from UCI KDD Archive (Bache & Lichman, 2013) contains data of different forest cover types. It contains forest cover type for 30x30 me-

[1] http://archive.ics.uci.edu/ml/

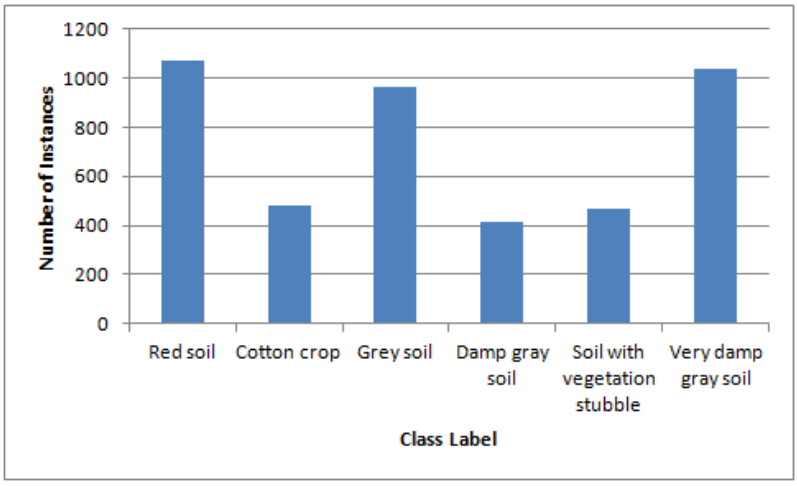


Figure 5.14: LandSat Class Distribution

ter cells obtained from US Forest Service (USFS) Region Resource Information System (RIS) data. It has 581,012 records and 54 attributes, out of which the 7 continuous ones, representing seven different types of forest types are picked. It has been used in several papers on data stream clustering (Forestiero et al., 2013; Bhatnagar et al., 2013; Zliobaite, Bifet, Read, Pfahringer, & Holmes, 2014). However, there are in fact only 12 different attributes. The last two are categorical and have been encoded as binary columns (4 and 40 respectively). The output is also categorical, but is encoded as a number between 1 and 7. This dataset contains 581012 observations and each observation consists of 54 attributes, including 10 numeric variables, 4 binary wilderness areas and 40 binary soil type variables. In our evaluation, similar to (Forestiero et al., 2013), 10 numeric variable is used. There are seven forest cover type classes. Figure 5.15 depicts the data distribution of forest cover type dataset.

5.2.2 Implementation and Environment

We have implemented MuDi-Stream as well as the comparative method, DenStream (Cao et al., 2006), in Java with graphical interface of R language. DenStream is one of the remarkable algorithms which is used as a benchmark for density-based clustering of data streams. DenStream has a longer chain of comparisons with existing algorithms

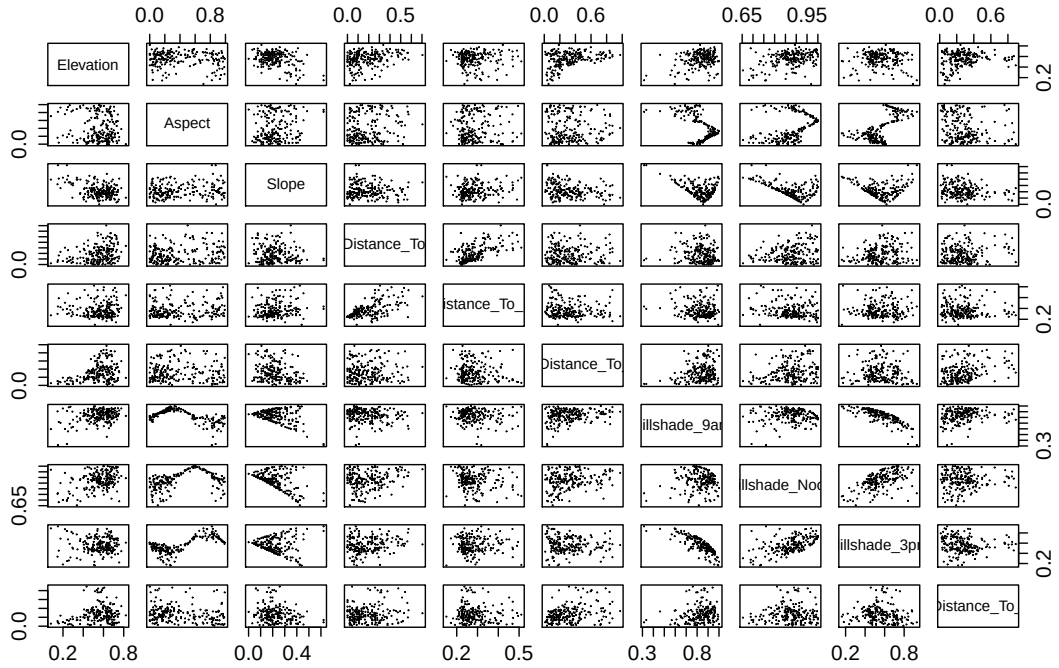


Figure 5.15: Forest Cover Type

(Forestiero et al., 2013; Ruiz et al., 2009; Lin & Lin, 2009; Hassani et al., 2012).

All experiments were conducted on a 2.5 GHz machine with 4GB memory, running on Mac OS X.

5.3 Quality Evaluation of MuDi-Stream

In this section MuDi-Stream is evaluated on different datasets with various external quality metrics which are introduced in Chapter 2. Since we know the class labels of data points in datasets the external metrics are used (Cao et al., 2006). The dataset are simulated as data streams by taking order (Forestiero et al., 2013).

Intuitively, the quality metrics evaluate the clustering results with respect to the true cluster (class) labels that are known for our datasets. Since the weights of data points fade out gradually, the quality metrics are measured by only the points arriving in a pre-defined horizon *h* (or window) from current time with different stream speeds. The stream speed is the number of arriving data points in each time unit (Amini et al., 2014). By horizon (or window) we mean, how many time steps from the current time we consider when running

the clustering algorithms. Thus, for example, if the horizon is 10, it means that the last 10 blocks of data are clustered. The small datasets are evaluated on one horizon since we got almost similar results for the other horizon and stream speed.

For each dataset we measured the following metrics:

- Purity
- Normalized Mutual Information (NMI)
- Rand Index (RI)
- Adjusted Rand Index (ARI)
- Jaccard Index (JI)
- Folkes and Mallow index (FM)
- F-Measure

The mentioned metrics are described in details in Chapter 2. These metrics are selected based on metrics which have been used for evaluating data stream clustering results in the literature (Kremer et al., 2011; Jain, 2010; Ruiz et al., 2009; Bolanos, 2014; X. Zhang et al., 2013; Hawwash, 2013). The definition of a good clustering is far from being an easy task (Han et al., 2011). In this respect we used different evaluation measures in order to perform a really accurate evaluation of our results. In this thesis, for quality evaluation of each metrics different horizons and stream speeds are considered based on the size of datasets, and yet the results are measured over different time units. Stream speed, horizon, time units are determined based on the characteristic of the dataset.

5.3.1 Evolving Data Stream (EDS)

Figures 5.16a, 5.16b, and 5.16c depict the core-mini-clusters which are detected by MuDi-Stream at different time units on EDS. In these figures, circles denote the core-

mini-clusters. It can be seen that MuDi-Stream accurately captures the shape of each cluster as the data stream evolves.

Purity (Figure 5.17), NMI (Figure 5.18), Rand Index (Figure 5.19), Adjusted Rand Index (Figure 5.20), Jaccard Index (Figure 5.21), FM (Figure 5.22), and F-Measure (Figure 5.23) show the quality metrics' results of MuDi-Stream and DenStream on EDS data stream.

MuDi-Stream achieves higher values compared to DenStream. The results are computed at time units 5, 10, 15, 20, 25, and 30 at horizon set to 2 and stream speed 1000 points per time unit. For horizon 5 with stream speed 2000, the time units are 5, 10, and 15.

MuDi-Stream's purity values are always higher than 96% and for the other criteria (NMI, RI, ARI, Jaccard Index, FM, and F-Measure), it has values more than 0.95. In this dataset, DenStream also gets good results for evolving data stream. However, in terms of efficiency, MuDi-Stream finished the clustering process significantly faster than DenStream.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.125$, $Minpts = 5$, $\alpha = 0.2$, and $gridGraunality = 30$. The DenStream's parameters are $\lambda = 0.25$, $\varepsilon = 16$, $\mu = 10$, and $\beta = 0.2$.

5.3.2 Multi-density Dataset (MDS1)

The clustering quality results on MDS1 including purity, NMI, Rand Index, Adjusted Rand Index, Jaccard index, FM, and F-measure are shown in Figures 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, and 5.30 respectively.

The results are computed at different time units (4, 6, 8, 10, and 12) with the horizon set to 5 and stream speed 1000 points per time unit. Moreover, the results are measured in horizon 1 with stream speed 2000 in 10, 15, 20, and 25 time units.

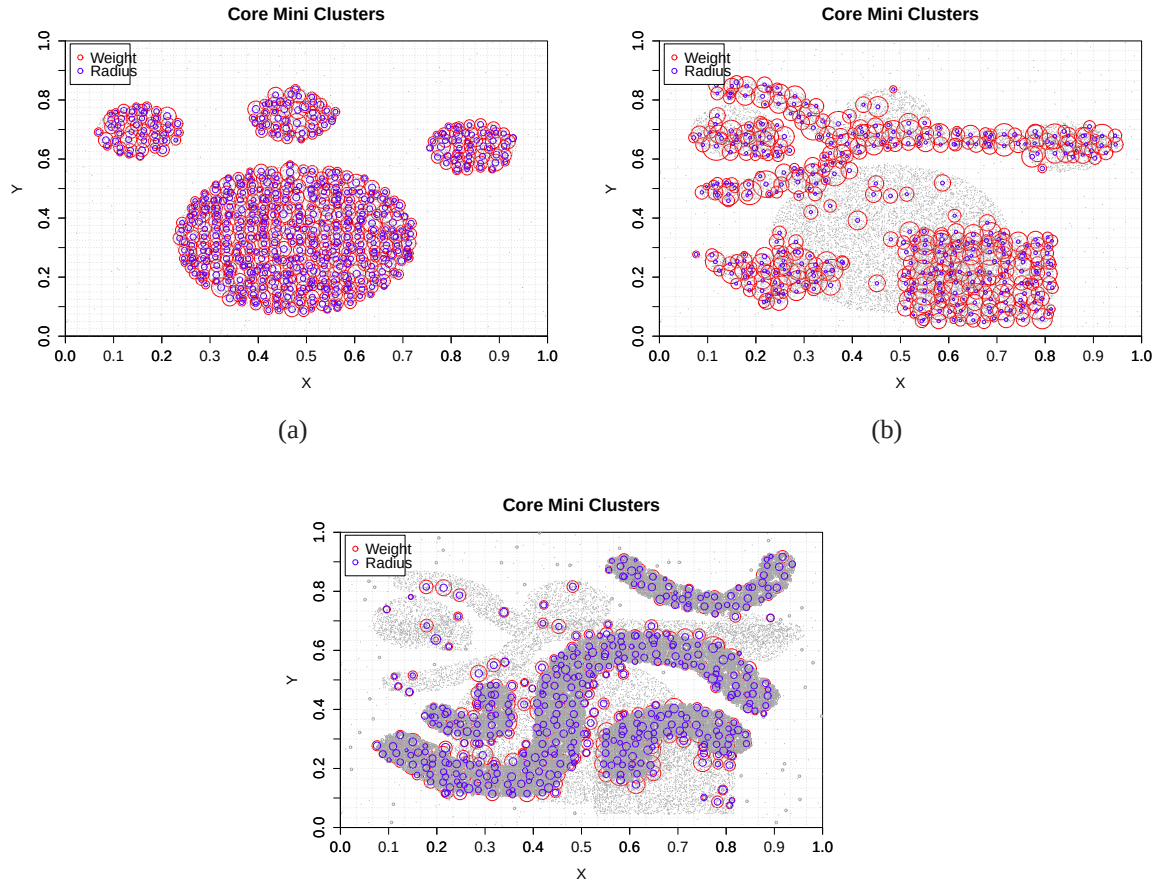


Figure 5.16: Core-mini-clusters for evolving data stream (EDS)

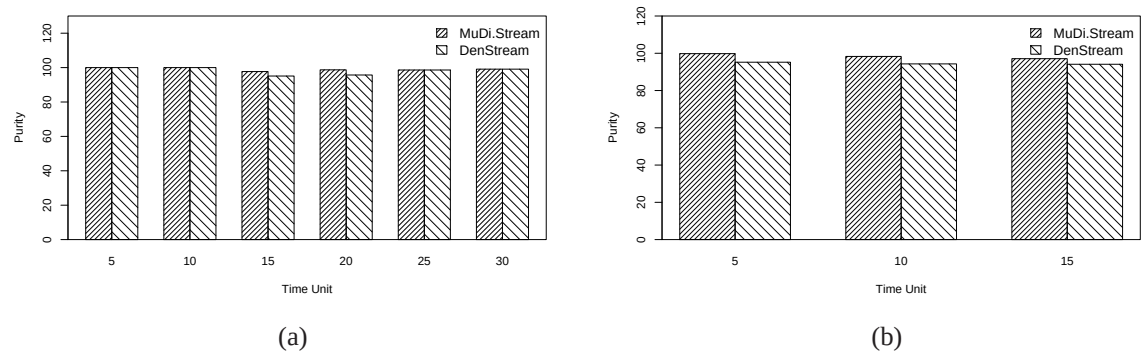
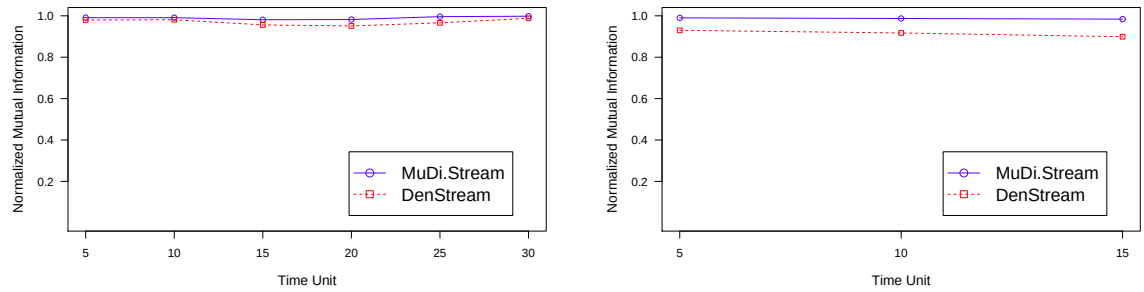


Figure 5.17: Cluster Purity of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000

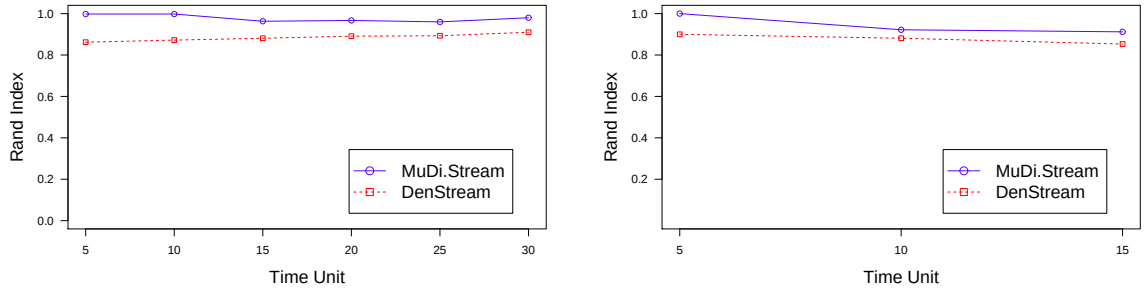
It can be observed that, MuDi-Stream outperforms DenStream when we have variety in densities. Purity, NMI, RI, ARI, Jaccard Index, FM and F-Measure have high quality results.

MuDi-Stream performed extremely well in clustering with maximum value 1 while DenStream has values below 0.7. DenStream quality decreases specifically when a sharp



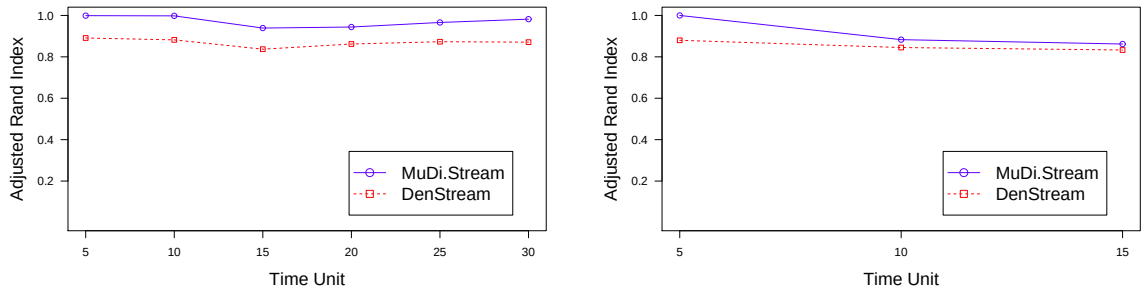
(a) (b)

Figure 5.18: Cluster Normalized Mutual Information of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000



(a) (b)

Figure 5.19: Cluster Rand Index of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000

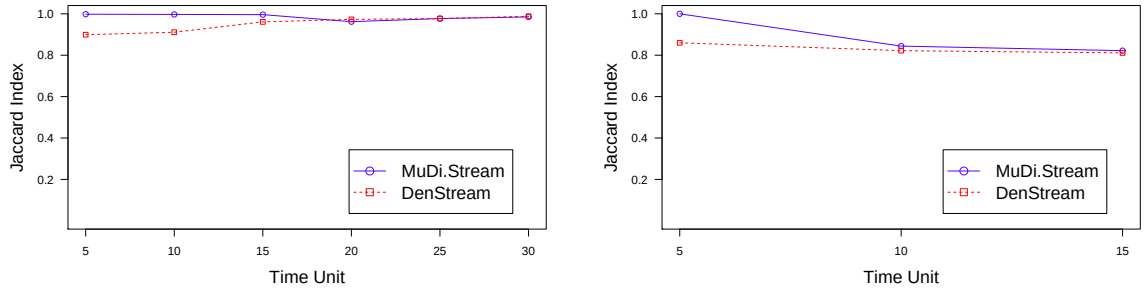


(a) (b)

Figure 5.20: Cluster Adjusted Rand Index of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000

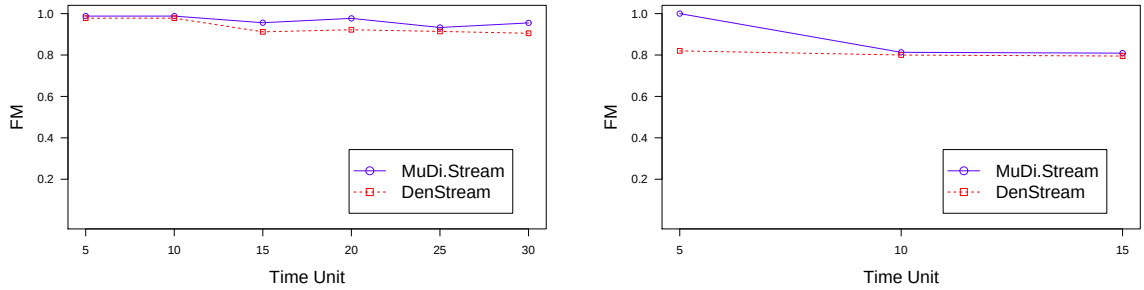
change in density happens since it uses the same parameters for all clusters. It cannot cluster the density variations.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.5$, $Minpts = 5$, $\alpha = 0.2$, $gridGraunality = 20$, and DenStream's values are chosen to be the same as (Cao et al., 2006). Figures 5.31a, and 5.31b depict the core-mini-clusters and final clusters of



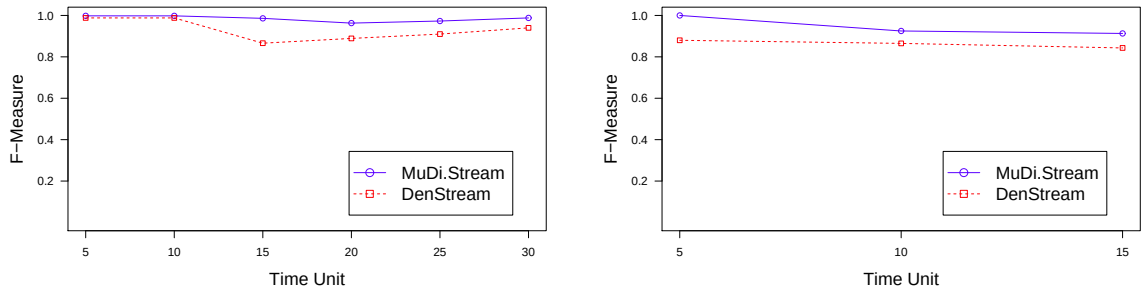
(a) (b)

Figure 5.21: Cluster Jaccard Index of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000



(a) (b)

Figure 5.22: Cluster FM of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000



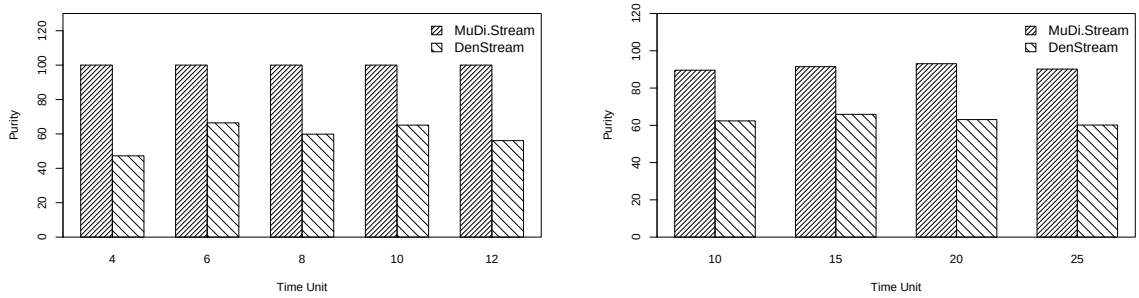
(a) (b)

Figure 5.23: Cluster F-Measure of MuDi-Stream for EDS with (a) horizon = 2 and stream speed = 1000, (b) horizon = 5 and stream speed = 2000

the dataset accordingly.

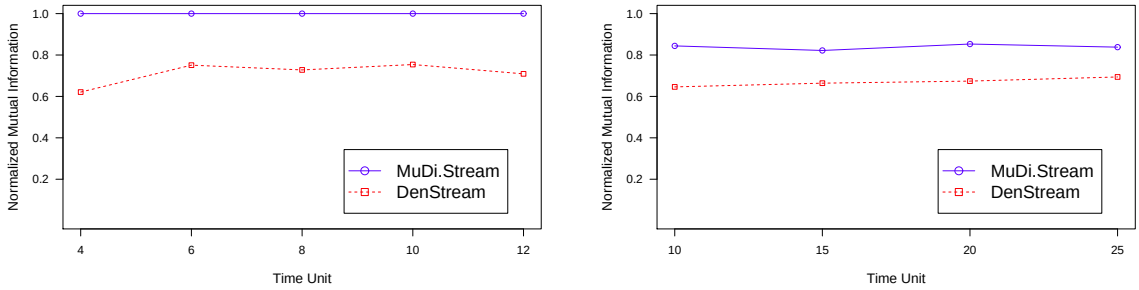
5.3.3 Multi-density Dataset - House (MDS2)

MuDi-Stream is also evaluated on a multi-density dataset called House (MDS2) at time units 2, 4, 8, and 10 with the horizon set to 5 and stream speed 1000. MuDi-Stream



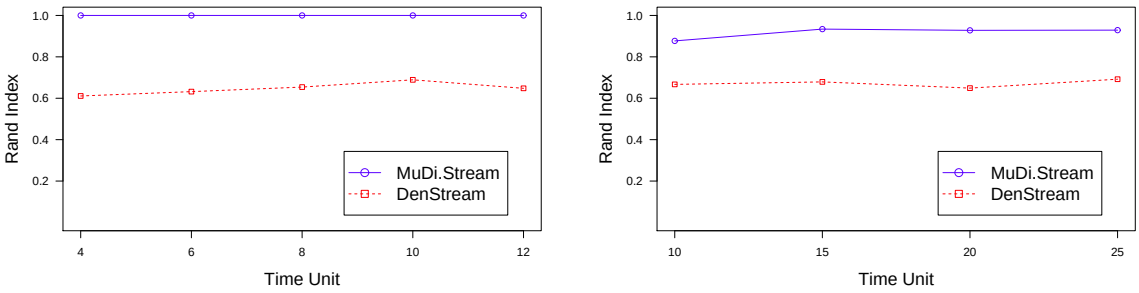
(a) (b)

Figure 5.24: Cluster Purity of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500



(a) (b)

Figure 5.25: Cluster Normalized Mutual Information of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500

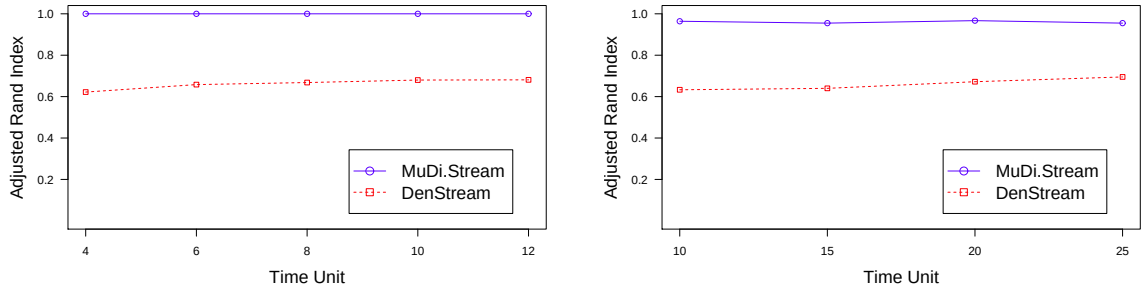


(a) (b)

Figure 5.26: Cluster Rand Index of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500

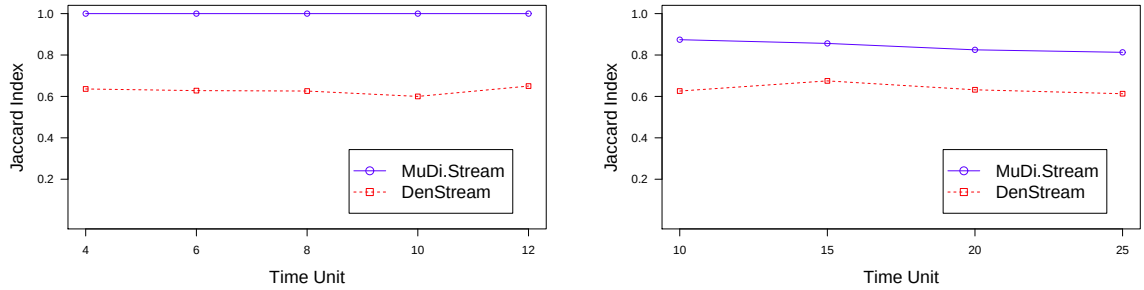
achieves high purity results for MDS2 compare to DenStream. For instance, while MuDi-Stream has purity values 99, 100, 100, and 100, DenStream has 84.1, 72.3, 79.11, and 67.4 values over different time units (Figure 5.32a).

DenStream clustering purity is high in the first time unit in which the small moon shaped cluster is detected; however, when time passes and the other clusters appear with



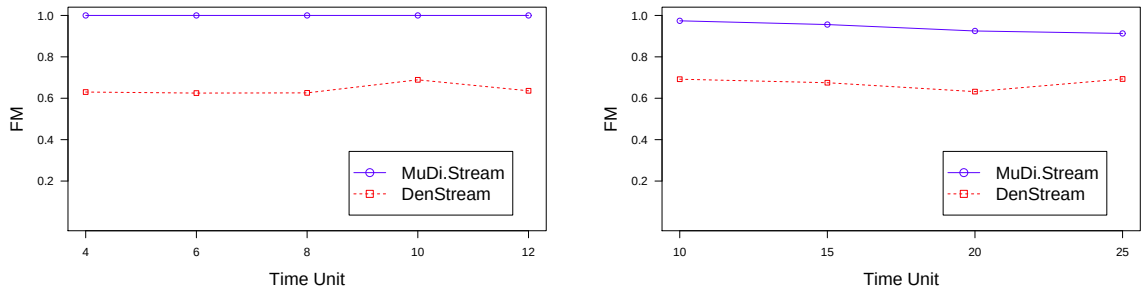
(a) (b)

Figure 5.27: Cluster Adjusted Rand Index of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500



(a) (b)

Figure 5.28: Cluster Jaccard Index of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500

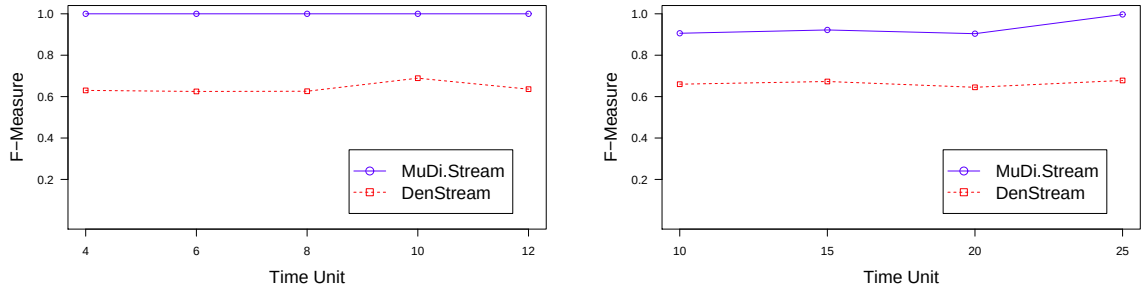


(a) (b)

Figure 5.29: Cluster FM of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500

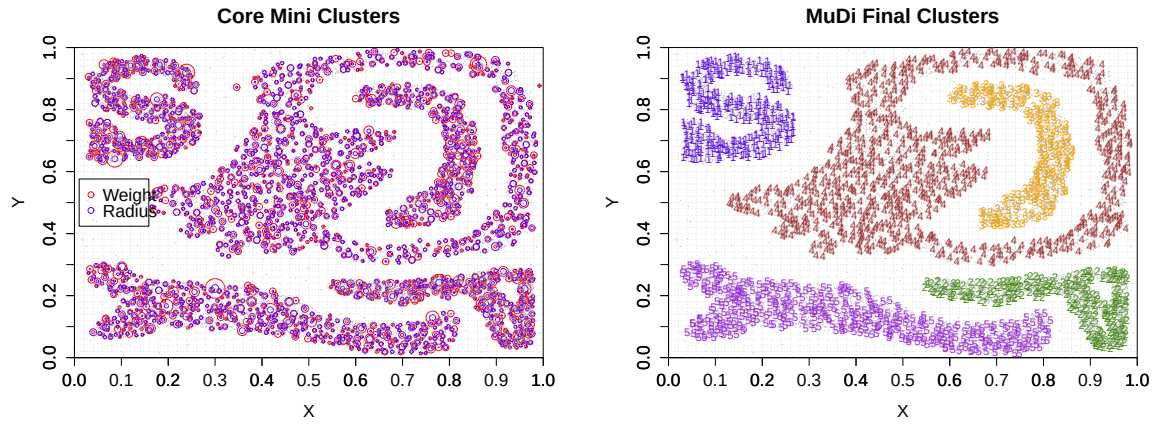
various densities, the clustering quality is decreased. We measured the other metrics, NMI (Figure 5.32b), Rand Index (Figure 5.33a), Adjusted Rand Index (Figure 5.33b), Jaccard Index (Figure 5.34a), FM (Figure 5.34b), and F-Measure (Figure 5.35)). The results prove that MuDi-Stream has much better quality results on a dataset with multi-density clusters.

The core-mini-clusters which are formed from MDS2 data points and final clustering



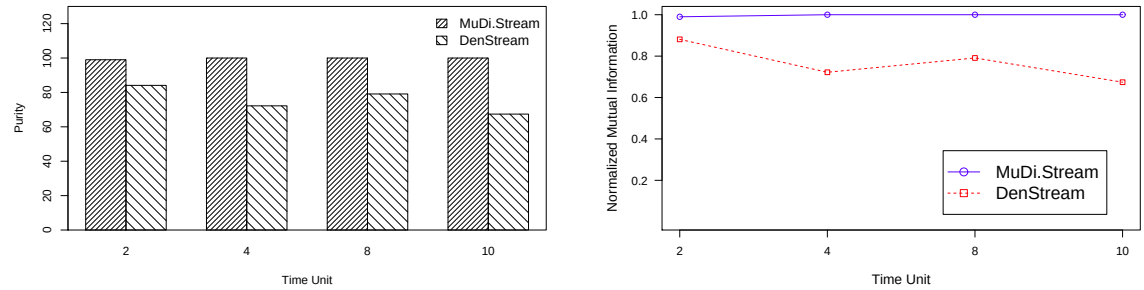
(a) (b)

Figure 5.30: Cluster F-Measure of MuDi-Stream for MDS1 with (a) horizon = 5 and stream speed = 1000, (b) horizon = 1 and stream speed = 500



(a) (b)

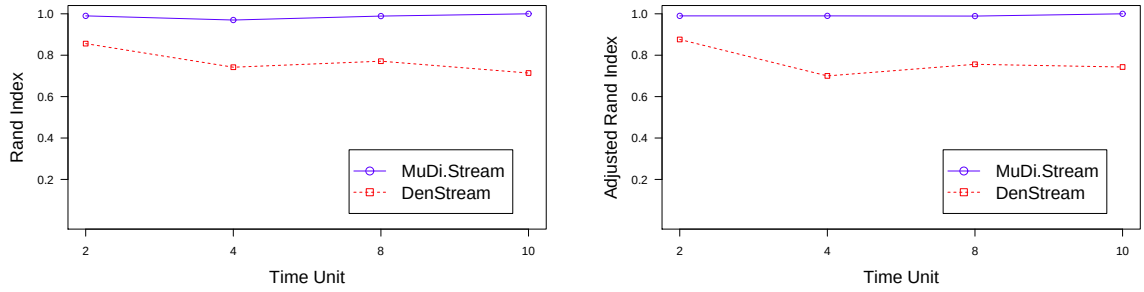
Figure 5.31: (a) Core-mini-clusters, (b) Final clusters in MDS1



(a) (b)

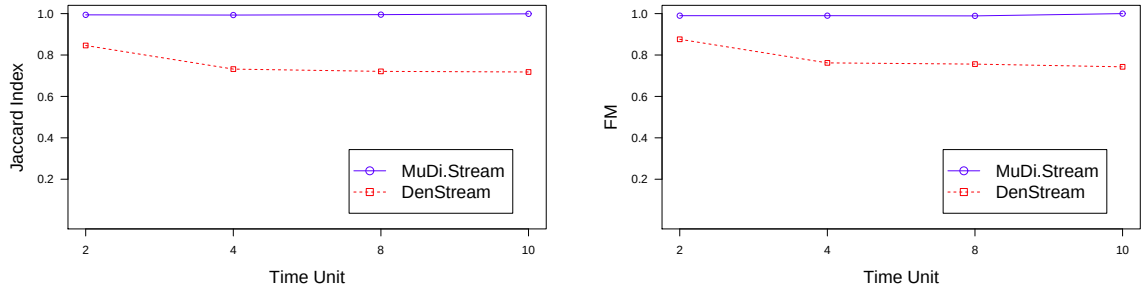
Figure 5.32: Cluster (a) Purity and (b) Normalized Mutual Information of MuDi-Stream for MDS2 with horizon = 2 and stream speed = 1000

results are shown in Figures 5.36a, and 5.36b respectively. The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.5$, $Minpts = 4$, $\alpha = 0.25$, $gridGraunality = 20$, and DenStream's values are chosen to be the same as the ones in the work presented in (Cao et al., 2006).



(a) (b)

Figure 5.33: Cluster (a) Rand Index and (b) Adjusted Rand Index of MuDi-Stream for MDS2 with horizon = 2 and stream speed = 1000



(a) (b)

Figure 5.34: Cluster (a) Jaccard Index and (b) FM of MuDi-Stream for MDS2 with horizon = 2 and stream speed = 1000

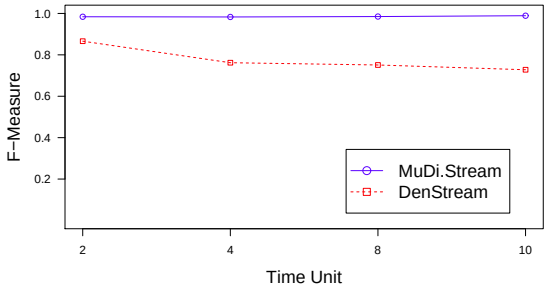
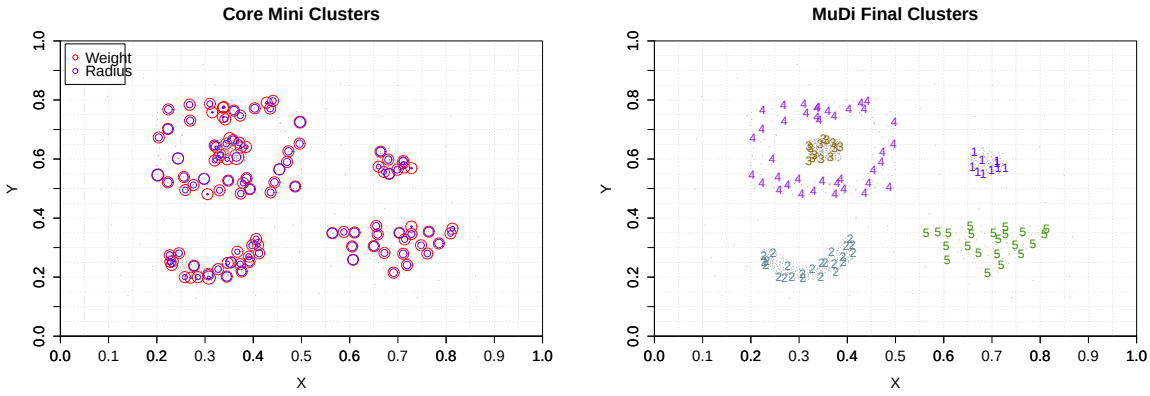


Figure 5.35: Cluster F-Measure of MuDi-Stream for MDS2 with (a) horizon = 2 and stream speed = 1000

5.3.4 Multi-density Dataset - 5Circles (MDS3)

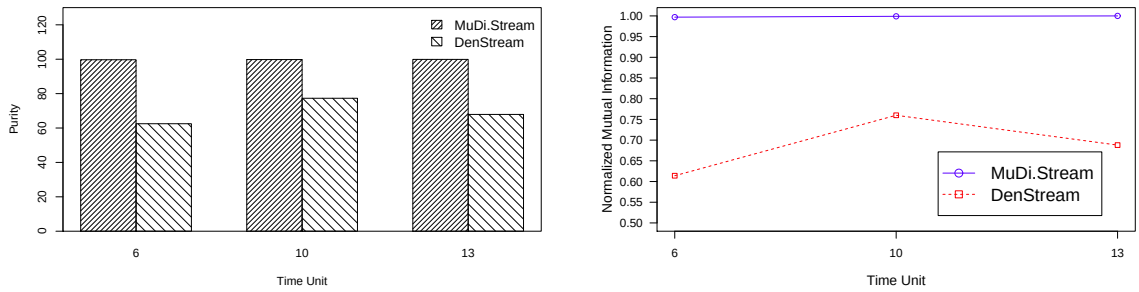
The results are computed at different time units, 6, 10, and 13 with the horizon set to 1 and stream speed 1000 points per time unit.

It can be observed from the evaluation results, which are depicted in Figures 5.37, 5.38, 5.39, and 5.40, that MuDi-Stream outperforms DenStream. MuDi-Stream has val-



(a) (b)

Figure 5.36: Core-mini-clusters and final clusters for MDS2

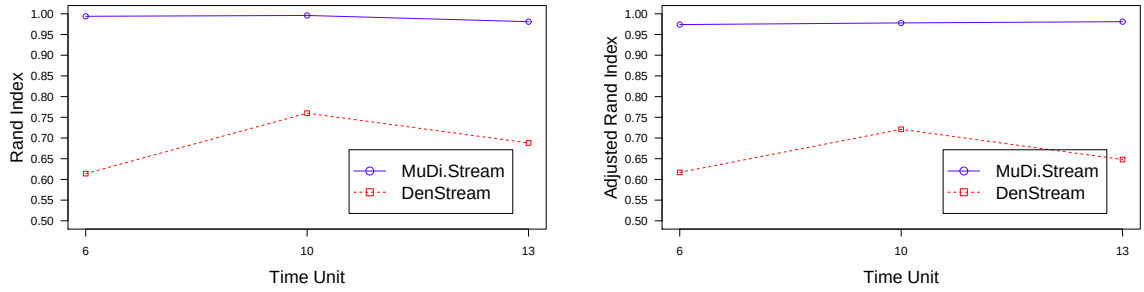


(a) (b)

Figure 5.37: Cluster Purity and Normalized Mutual Information of MuDi-Stream for MDS3 with (a) horizon = 1 and stream speed = 1000

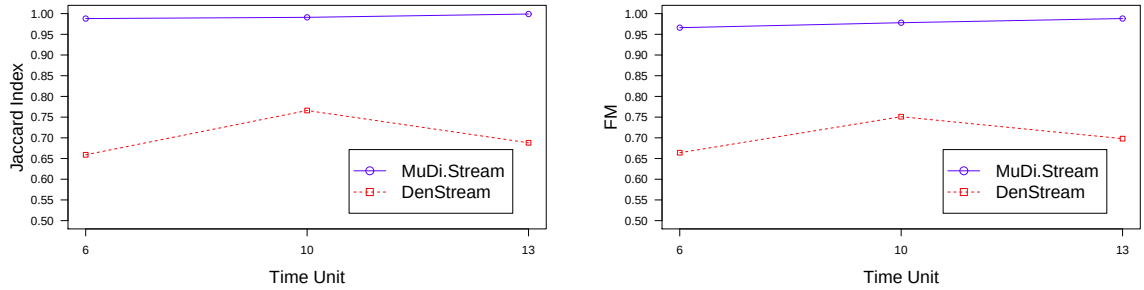
ues almost equal to 1 for most of the metrics; however, DenStream gets low quality results with highest value 0.68%. The dataset has two sparse clusters with three dense ones for which it is difficult to handle. DenStream quality values is low since with high ε values, it can detect the sparse grids correctly while the other three dense clusters are considered as one. If DenStream detects the small clusters precisely, the sparse clusters are detected as noise. Therefore, DenStream clustering results are low for this dataset.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.125$, $Minpts = 4$, $\alpha = 0.04$, $gridGraunality = 30$, and DenStream's values are chosen to be the same as its research paper (Cao et al., 2006). Figures 5.41a, and 5.41b depict the core-mini-clusters and the final clusters of the dataset accordingly.



(a) (b)

Figure 5.38: Cluster Rand Index and Adjusted Rand Index of MuDi-Stream for MDS3 with (a) horizon = 1 and stream speed = 1000



(a) (b)

Figure 5.39: Cluster Jaccard Index and FM of MuDi-Stream for MDS3 with (a) horizon = 1 and stream speed = 1000

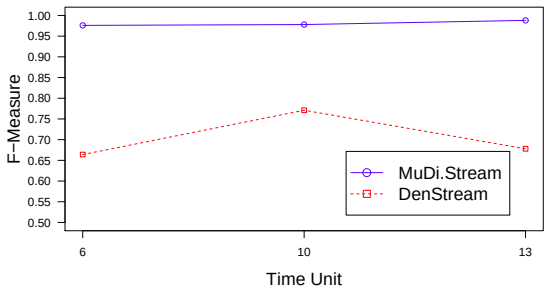
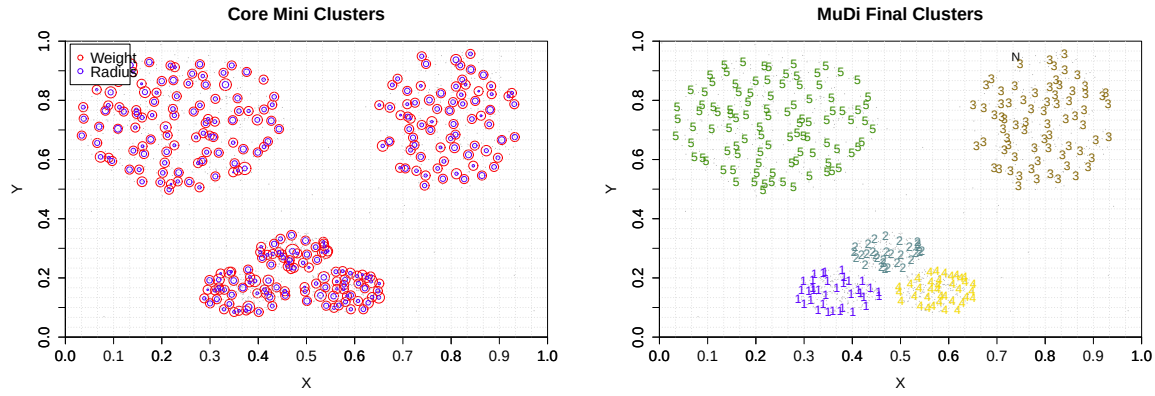


Figure 5.40: Cluster F-Measure of MuDi-Stream for MDS3 with (a) horizon = 1 and stream speed = 1000

5.3.5 Evolving Multi-density Dataset (EMDS)

Evaluation with seven different metrics on EMDS shows that MuDi-Stream performs quite well in detecting correct clusters. DenStream become confused when there is a change in the density of clusters. In Figures 5.42, 5.43, 5.44, 5.45, 5.46, 5.47, and 5.48, the values of external validity criteria computed from clustering results on dataset EMDS



(a) (b)

Figure 5.41: Core-mini-clusters and final clusters for MDS3

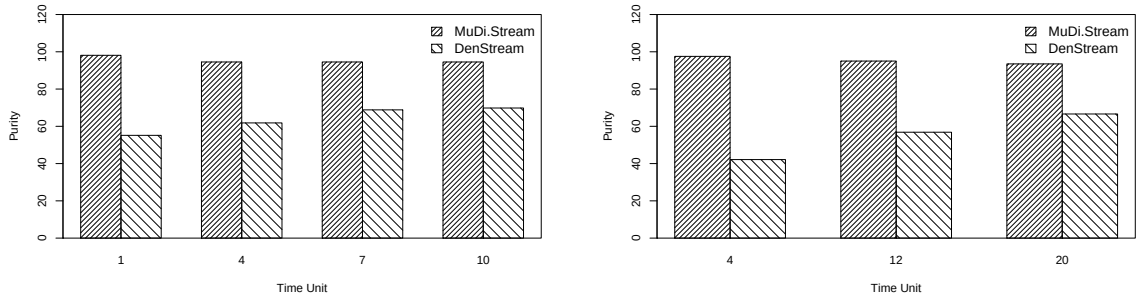
validated the mentioned claim. MuDi-Stream achieved almost maximum values of 1 while DenStream attained values of approximately 0.6.

The final clustering results obtained by MuDi-Stream on the evolving multi density data stream (EMDS) are shown in Figure 5.49. In Figure 5.49a points indicate the raw data while circles in Figure 5.49b denote the core mini clusters. The results are computed at different time units with horizon set to 2 at time units 4, 12, 20 and stream speed 680, and horizon 1 with stream speed 1360 at time units 1, 4, 7, and 10. The parameters of MuDi-Stream adopt the following settings: $\lambda = 1$, $Minpts = 3$, $\alpha = 0.03$ $gridGraunality = 20$, and DenStream as in (Cao et al., 2006).

DenStream has the ability to detect arbitrary shape clusters in evolving data stream; however, when the evolving data has changes in its density, DenStream cannot detect properly. Therefore, the quality results decrease compared to MuDi-Stream. MuDi-Stream can adjust its parameters as data evolves and density changes. Thus, it outperforms DenStream for evolving multi-density data stream.

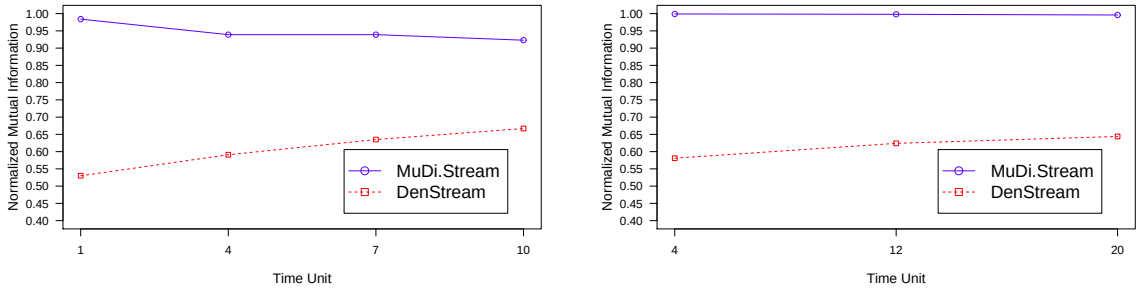
5.3.6 Multi-density CylinderCube (MDS4)

The clustering results of MuDi-Stream are shown for two perspectives in Figures 5.50a, 5.50b. It can be observed that, MuDi-Steam can detect clusters precisely. Since in MDS4, the density distribution changes are not high, hence, DenStream also can get



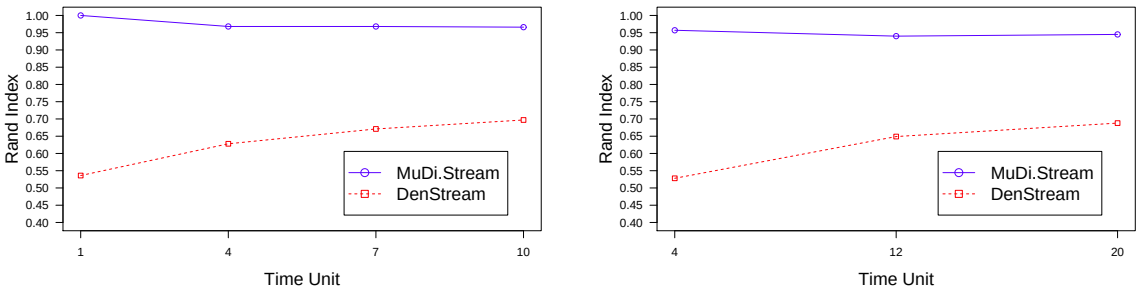
(a) (b)

Figure 5.42: Cluster Purity of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680



(a) (b)

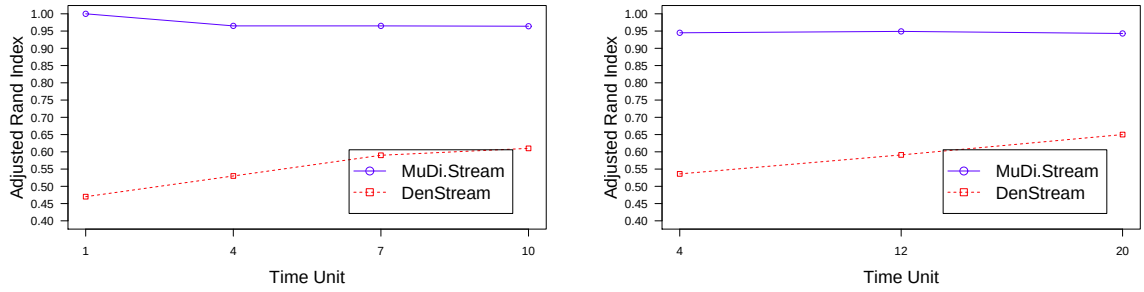
Figure 5.43: Cluster Normalized Mutual Information of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680



(a) (b)

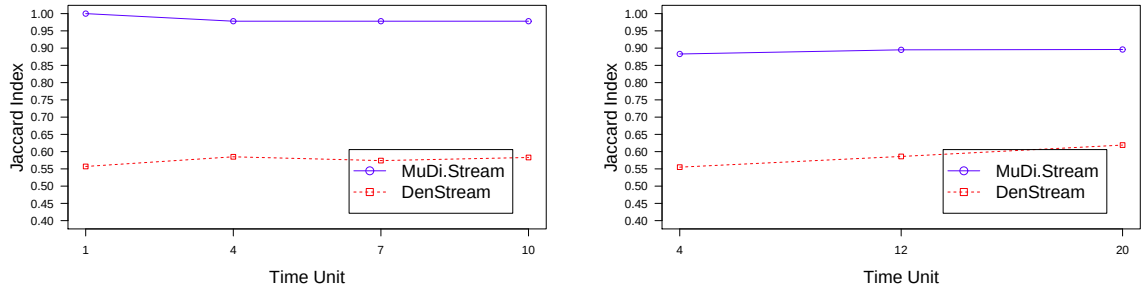
Figure 5.44: Cluster Rand Index of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680

good results such as purity above 87.9% while MuDi-Stream has values almost 100% in time unit 5. The quality comparisons of the algorithms are shown in Figures 5.51, 5.52, 5.53, 5.54. The results are computed at time units 5, and 10 with horizon set to 10 at time stream speed 1000. The parameters of MuDi-Stream adopt the following settings: $\lambda = 1$, $Minpts = 3$, $\alpha = 0.03$, $gridGraunality = 20$, and DenStream's as in (Cao et al., 2006).



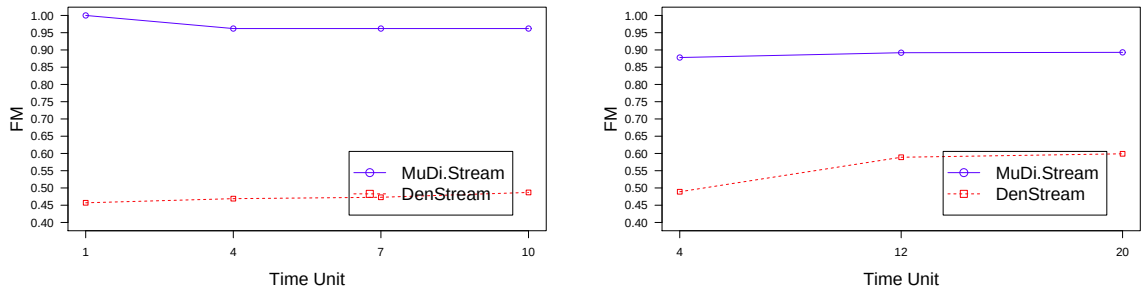
(a) (b)

Figure 5.45: Cluster Adjusted Rand Index of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680



(a) (b)

Figure 5.46: Cluster Jaccard Index of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680

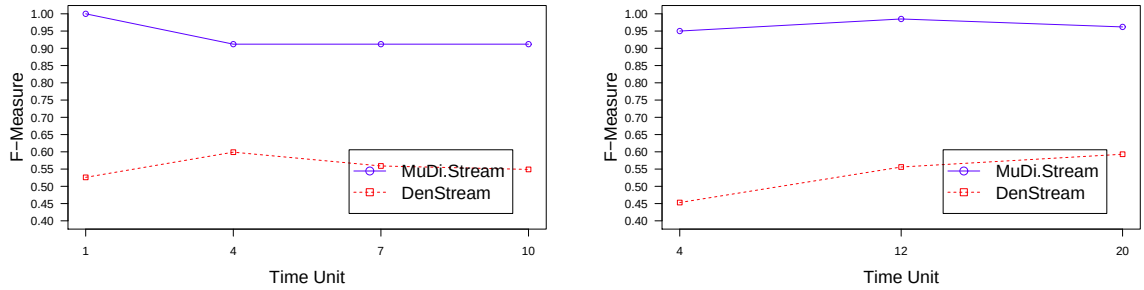


(a) (b)

Figure 5.47: Cluster FM of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680

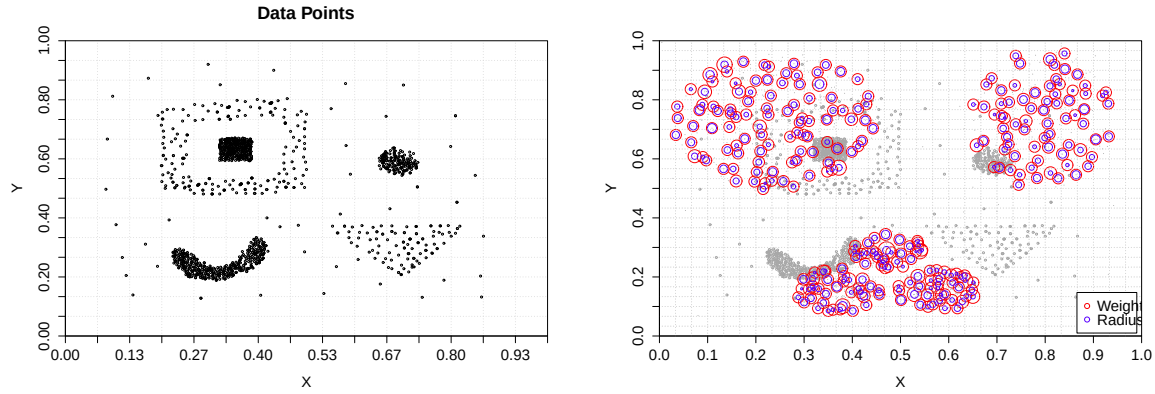
5.3.7 Gaussian Multi-density Dataset (GMDS)

This dataset is used to evaluate the ability of MuDi-Stream in Gaussian distributions. For GMDS, both algorithms gave close quality values in purity, NMI, RI, ARI, JI, FM, and F-Measure. The quality values are approximately equal to 1 and for purity almost equals to 100%. However, in term of efficiency MuDi-Stream process remarkably faster



(a) (b)

Figure 5.48: Cluster F-Measure of MuDi-Stream for EMDS with (a) horizon = 1 and stream speed = 1360, (b) horizon = 2 and stream speed = 680



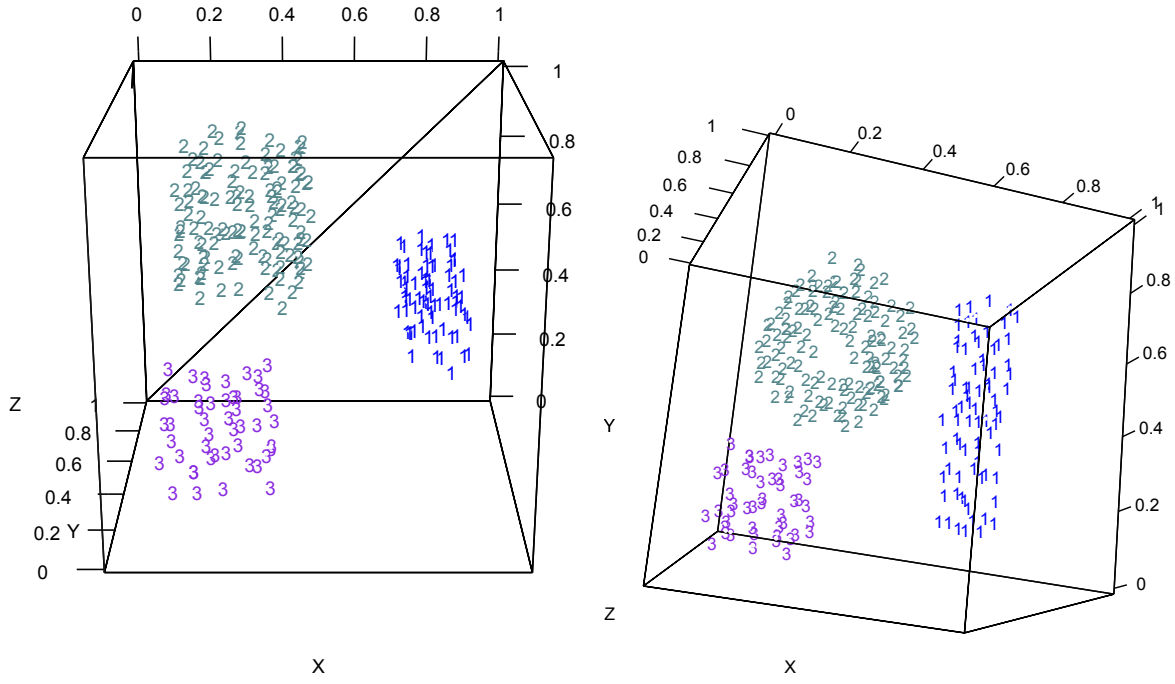
(a) (b)

Figure 5.49: EMDS dataset (a) data points at t = 6 (b) Core-mini-clusters at t = 12

than DenStream. The results are displayed in Figures 5.56, 5.57, 5.58, 5.59, 5.60, 5.61, and 5.62. The results confirm that MuDi-Stream has the ability to cluster the data in Gaussian distributions.

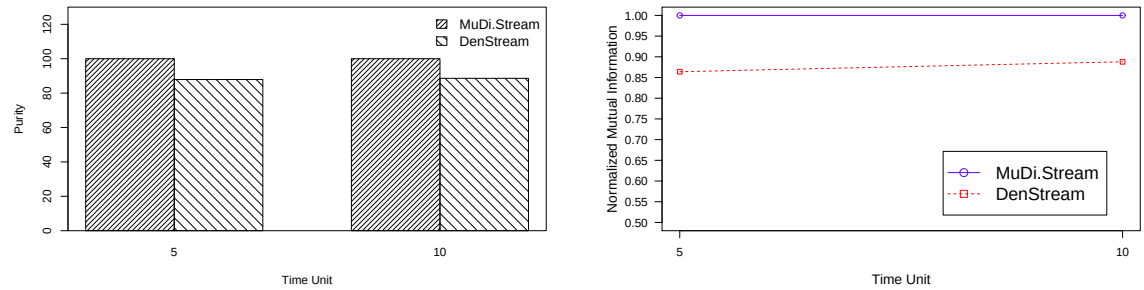
The results are computed at 1) time units 1, 2, 3, …, 10 at horizon set to 1 with stream speed 1000, and 2) horizon 5 with stream speed 500 and time units 4, 8, 12, 16, and 20.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.5$, $Minpts = 3$, $\alpha = 0.05$, $gridGraunality = 30$, and DenStream's values are chosen to be the same as (Cao et al., 2006). Figures 5.55a, and 5.55b depict the core-mini-clusters and final clusters of GMDS respectively.



(a) (b)

Figure 5.50: Multi-density CylinderCube - Final Clusters

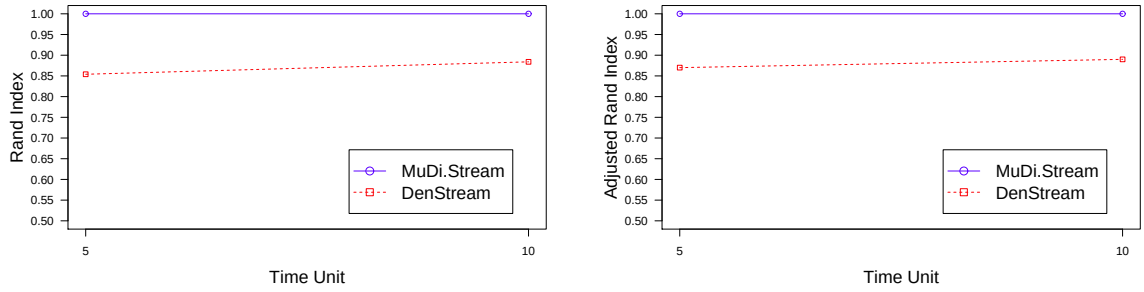


(a) (b)

Figure 5.51: Cluster Purity and Normalized Mutual Information of MuDi-Stream for MDS4 with horizon = 10 and stream speed = 1000

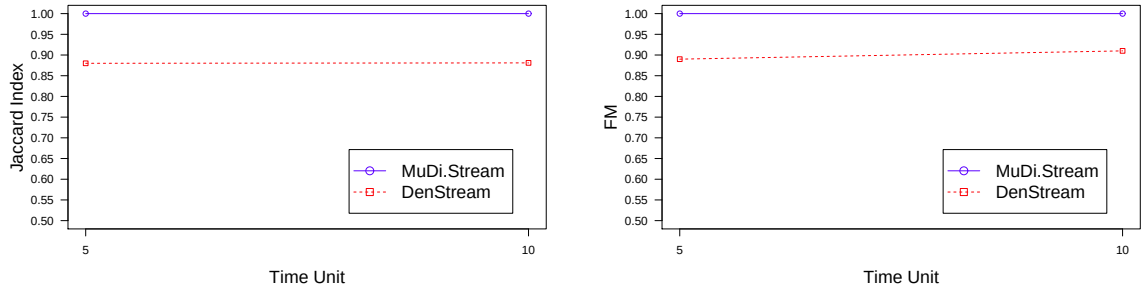
5.3.8 Network Intrusion Detection Dataset

Figures 5.63, 5.64, 5.65, 5.66, 5.67, 5.68, and 5.69 show the quality results for the Network Intrusion Detection dataset on seven metrics. For the evaluation purposes, we performed the measurements at time units where some attacks exist (8, 14, 19, 22 and 43, 51, 86, 100). The results have been computed by setting the horizon to 1 and 2, whereas the stream speed is 1000. We can clearly see the high clustering quality achieved by MuDi-Stream on this dataset. For example, Rand Index, for all the time units almost reaches 100% when the horizon is set to 1. Analogous results are obtained when horizon



(a) (b)

Figure 5.52: Cluster Rand Index and Adjusted Rand Index for MDS4 with horizon = 10 and stream speed = 1000



(a) (b)

Figure 5.53: Cluster Jaccard Index and FM for MDS4 with horizon = 10 and stream speed = 1000

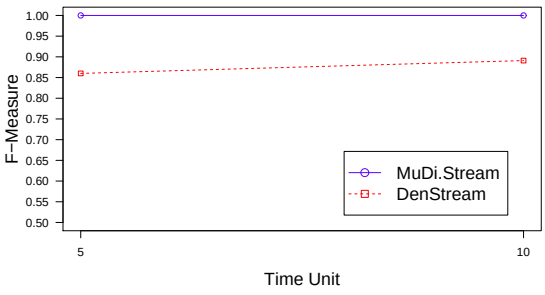
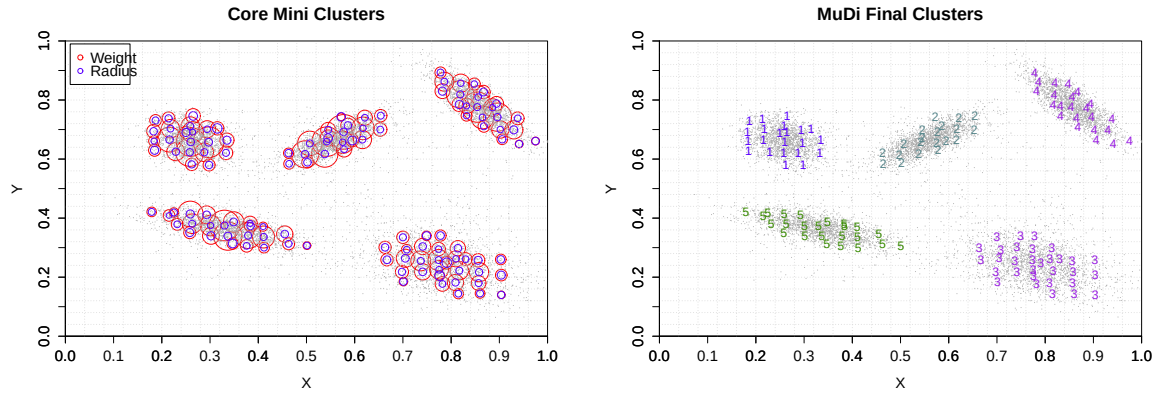


Figure 5.54: Cluster F-Measure for MDS4 with horizon = 10 and stream speed = 1000

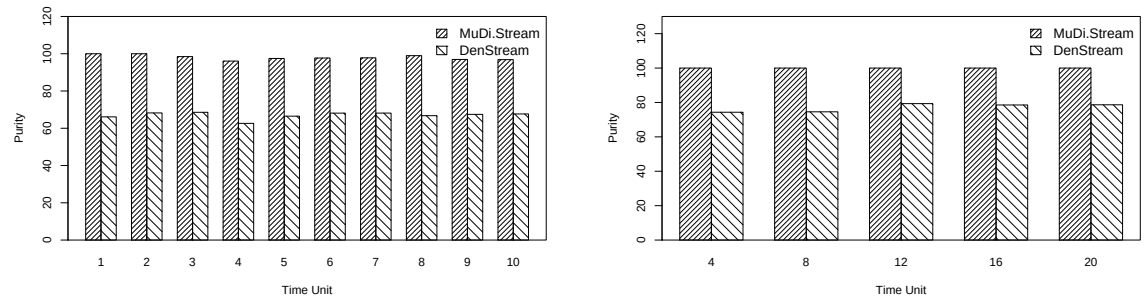
value 2 is used. On this dataset MuDi-Stream outperforms DenStream on others quality metrics as well.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.25$, $Minpts = 3$, $\alpha = 0.15$ $gridGraunality = 30$, and DenStream as in (Cao et al., 2006).



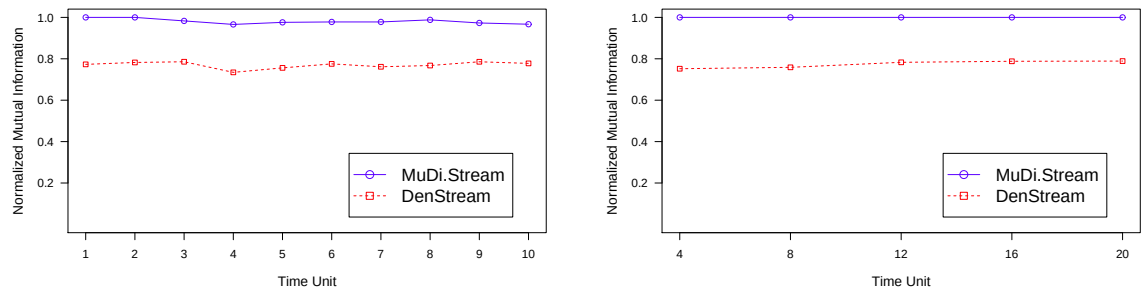
(a) (b)

Figure 5.55: GMDS - Core-mini-clusters and final clusters



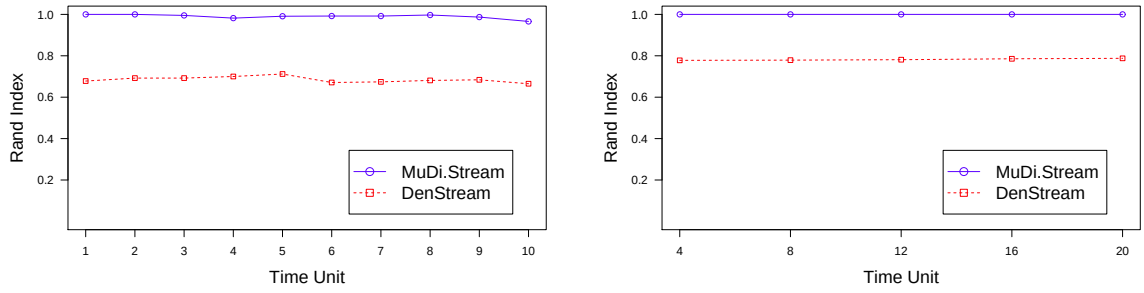
(a) (b)

Figure 5.56: Cluster Purity for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



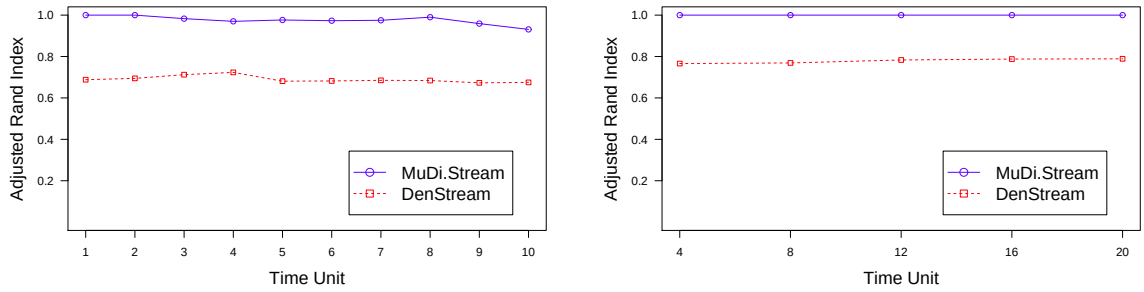
(a) (b)

Figure 5.57: Cluster Normalized Mutual Information for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



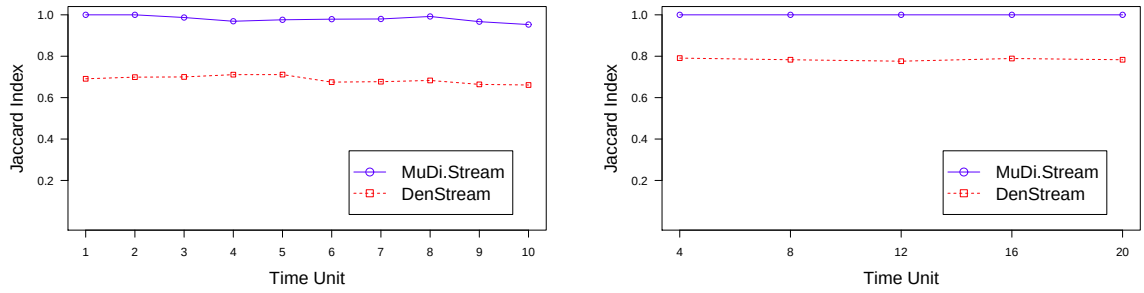
(a) (b)

Figure 5.58: Cluster Rand Index for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



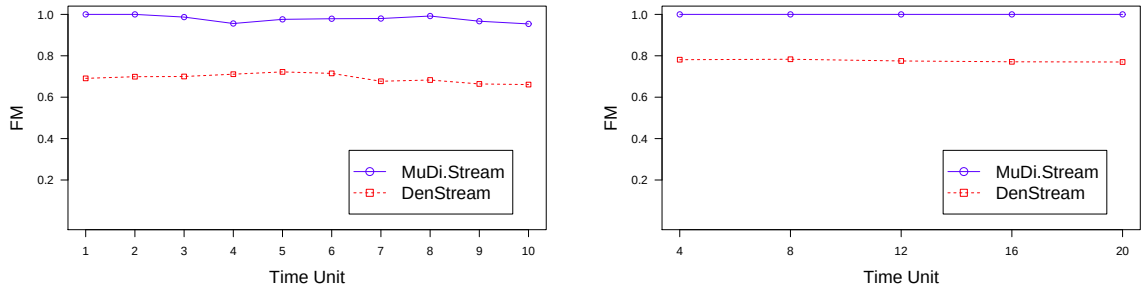
(a) (b)

Figure 5.59: Cluster Adjusted Rand Index for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



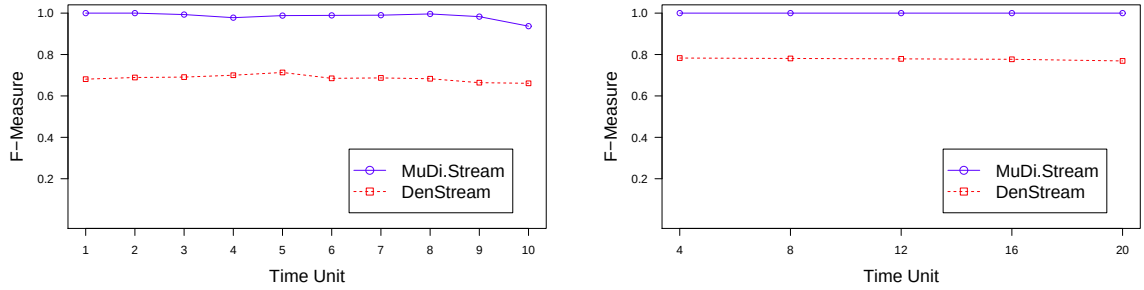
(a) (b)

Figure 5.60: Cluster Jaccard Index for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



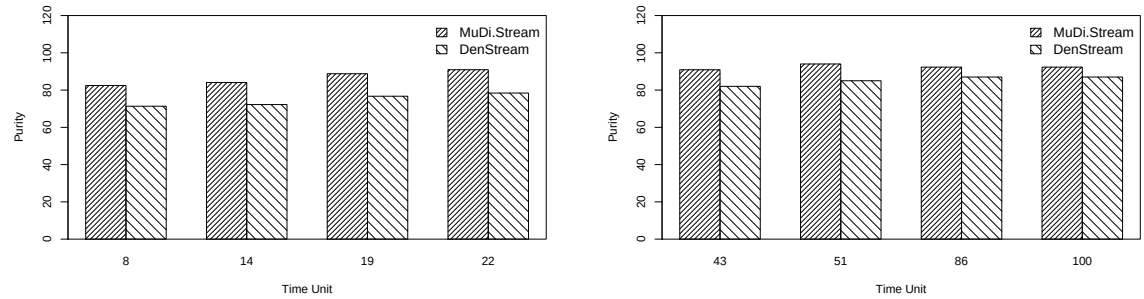
(a) (b)

Figure 5.61: Cluster FM for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



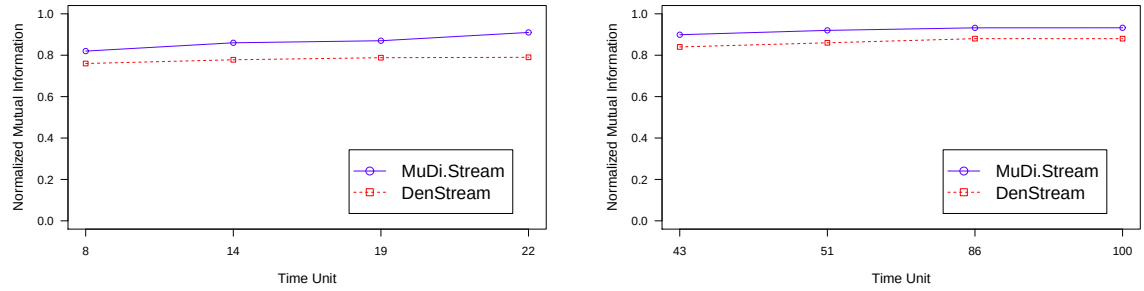
(a) (b)

Figure 5.62: Cluster F-Measure for GMDS with (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 500



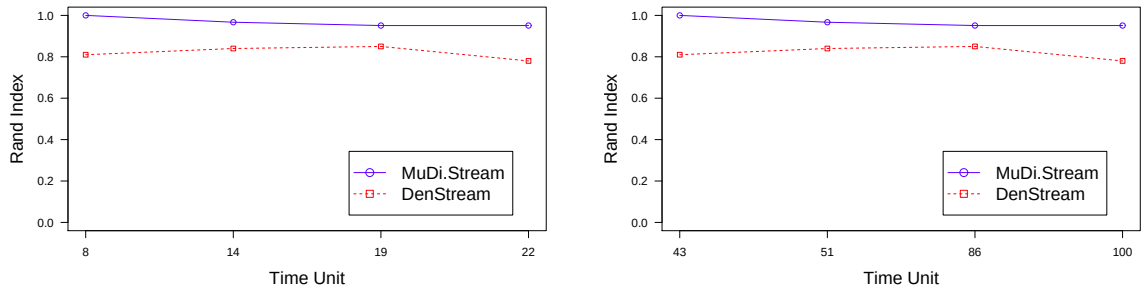
(a) (b)

Figure 5.63: Clustering Purity on Network Intrusion Detection (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000



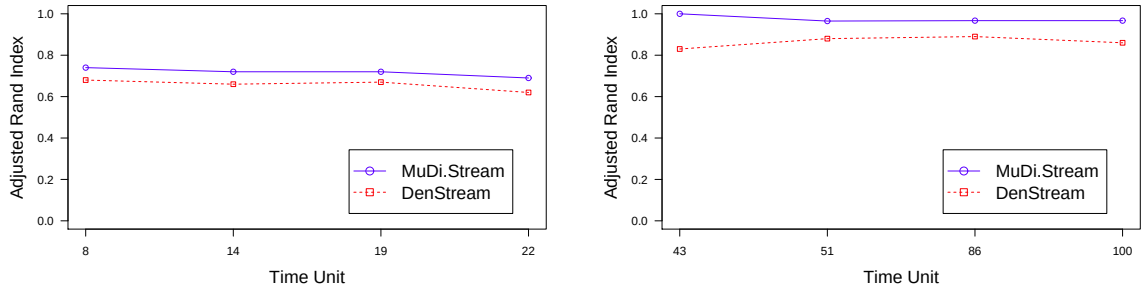
(a) (b)

Figure 5.64: Clustering Normalized Mutual Information on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000



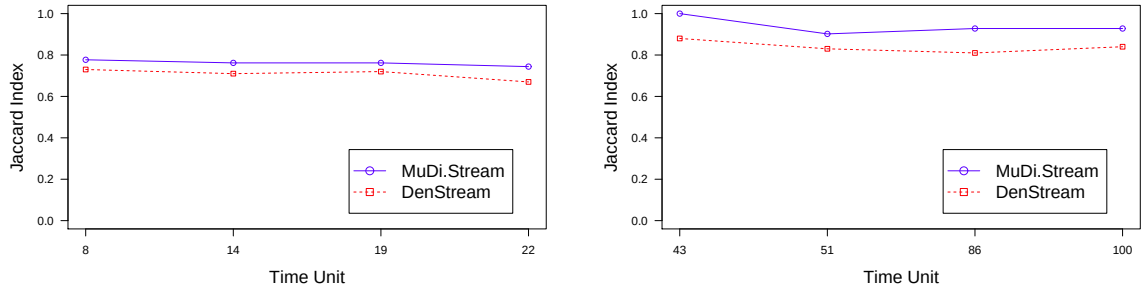
(a) (b)

Figure 5.65: Clustering Rand Index on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000



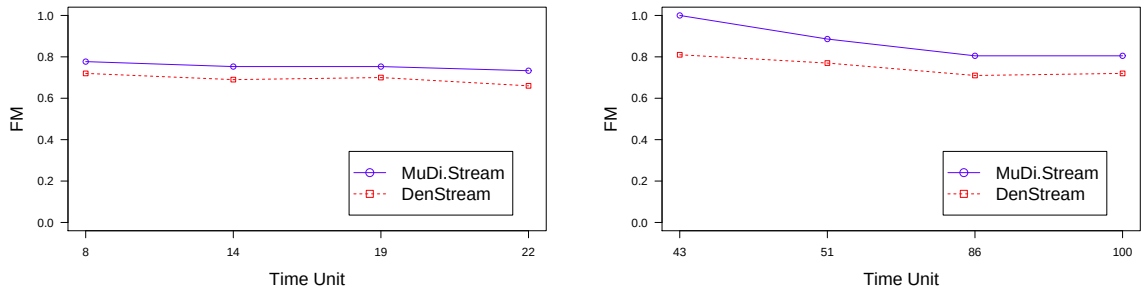
(a) (b)

Figure 5.66: Clustering Adjusted Rand Index on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000



(a) (b)

Figure 5.67: Clustering Jaccard Index on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000

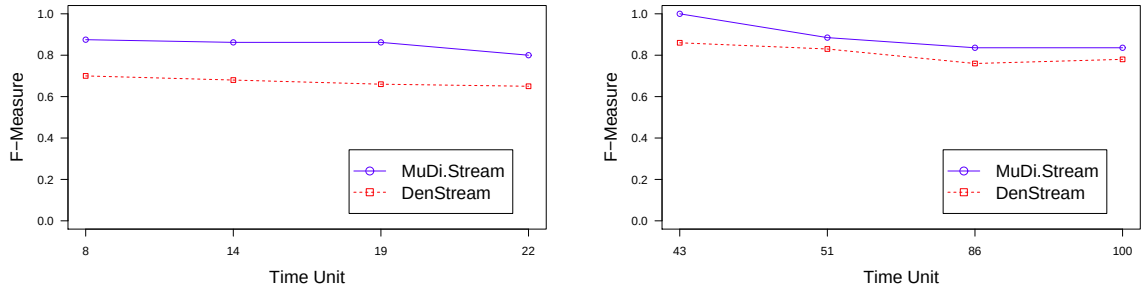


(a) (b)

Figure 5.68: Clustering FM on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000

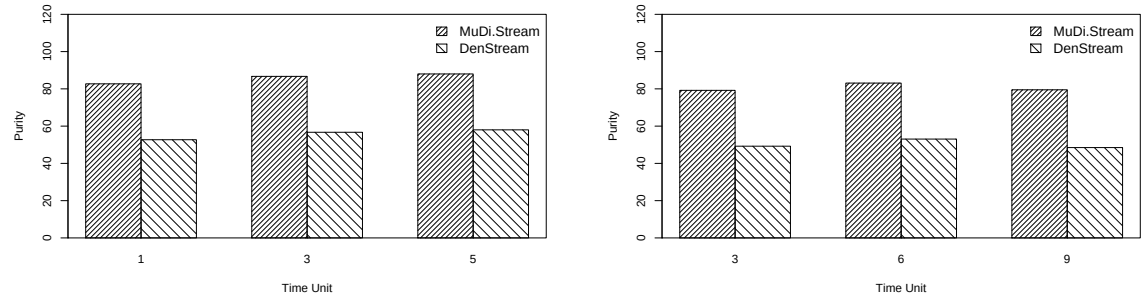
5.3.9 LandSat Satellite Data

The quality comparisons are performed on the horizons 1 and 3 with stream speeds 1000 and 500 respectively. The time units in horizon 1 are 1, 3, and 5 and in horizon 3 are 3, 6, and 9. We evaluated the algorithms on different time units as it is depicted in Figures 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, and 5.76. It can be seen that MuDi-Stream



(a) (b)

Figure 5.69: Clustering F-Measure on Network Intrusion Detection Dataset (a) horizon = 1 and stream speed = 1000, (b) horizon = 2 and stream speed = 1000

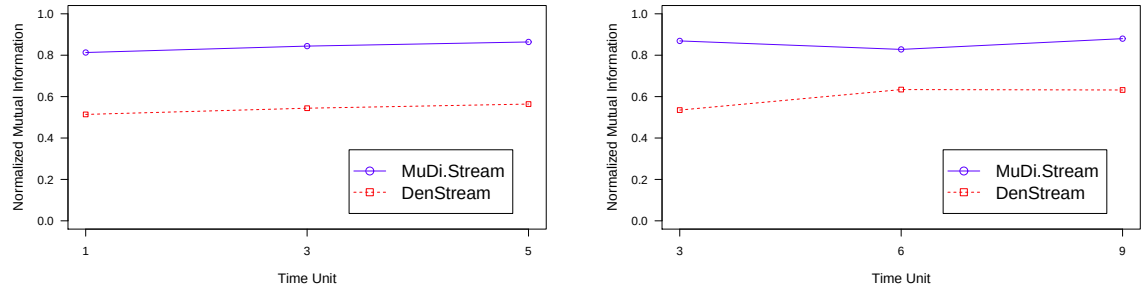


(a) (b)

Figure 5.70: Clustering Purity on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500

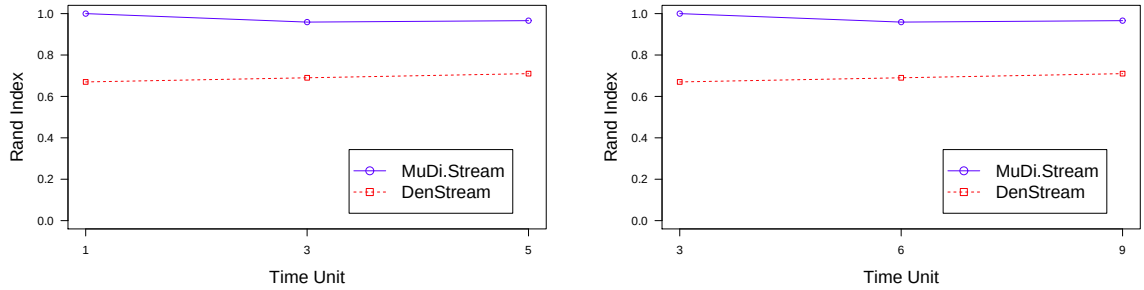
clearly outperforms DenStream for most of the quality metrics and the values are almost 1. For instance, when the Adjusted Rand Index values of MuDi-Stream in horizon 3 and time unit 3 is equal to 1, DenStream's value is 0.59 which is quite low compared to MuDi-Stream.

The reason for the difference is that MuDi-Stream detects clusters with different densities precisely. The quality of DenStream deteriorates greatly because in DenStream a global set of parameters is applied which is not sufficient for the data with a range of densities and it may detect different clusters as one or sparse cluster as noise. The parameters of MuDi-Stream adopt the following settings: $\lambda = 0.5$, $Minpts = 5$, $\alpha = 0.01$, $gridGraunality = 20$, and DenStream's as in (Cao et al., 2006).



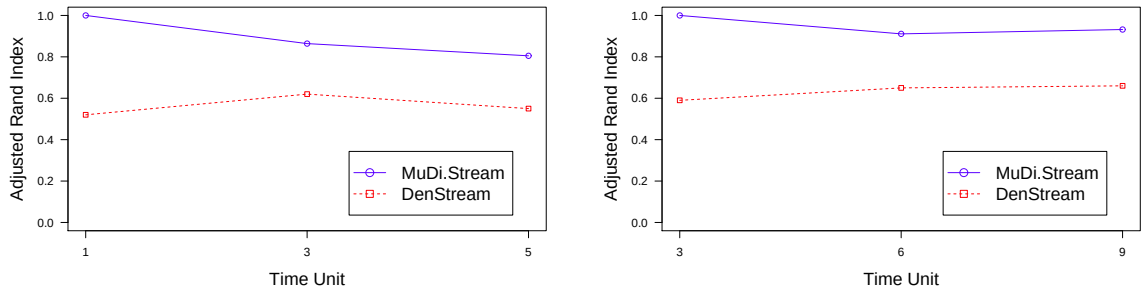
(a) (b)

Figure 5.71: Clustering Normalized Mutual Information on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500



(a) (b)

Figure 5.72: Clustering Rand Index on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500

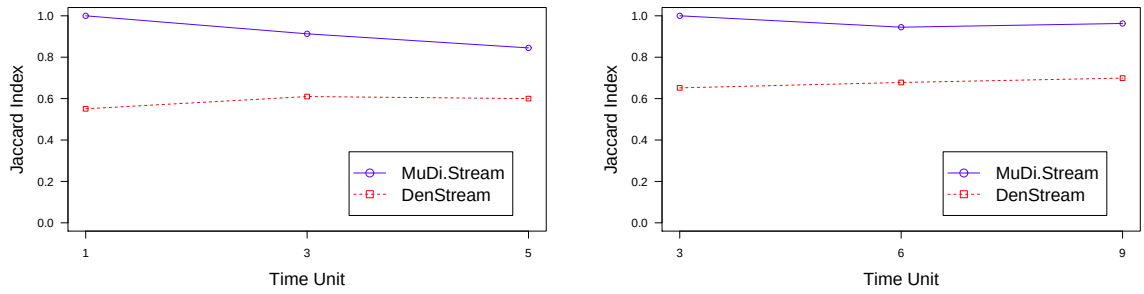


(a) (b)

Figure 5.73: Clustering Adjusted Rand Index on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500

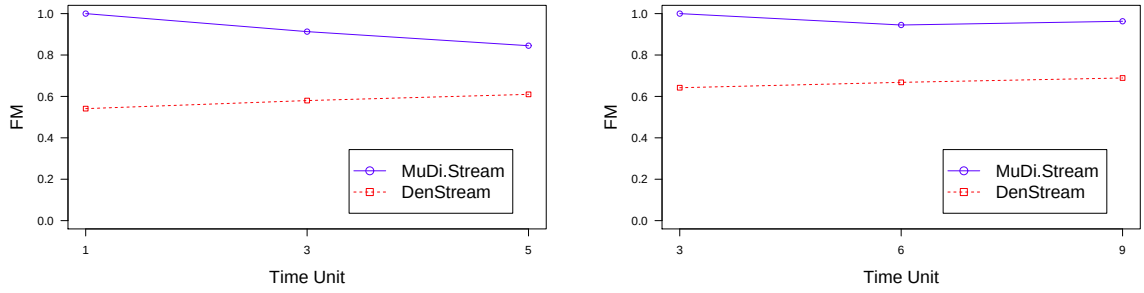
5.3.10 Forest Cover Type

The evaluations on Forest dataset are reported in Figures 5.77, 5.78, 5.77, 5.79, 5.80, 5.81, 5.82, and 5.83, for Purity, NMI, RI, ARI, JI, FM, and F-Measure metrics respectively. The results show that MuDi-Stream overcomes DenStream on this dataset as well. The results are computed at 1) time units 20, 40, 80, and 100 with the horizon set



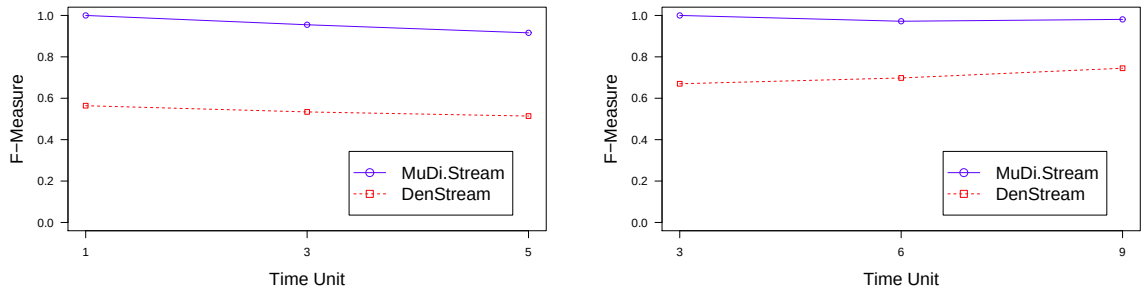
(a) (b)

Figure 5.74: Clustering Jaccard Index on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500



(a) (b)

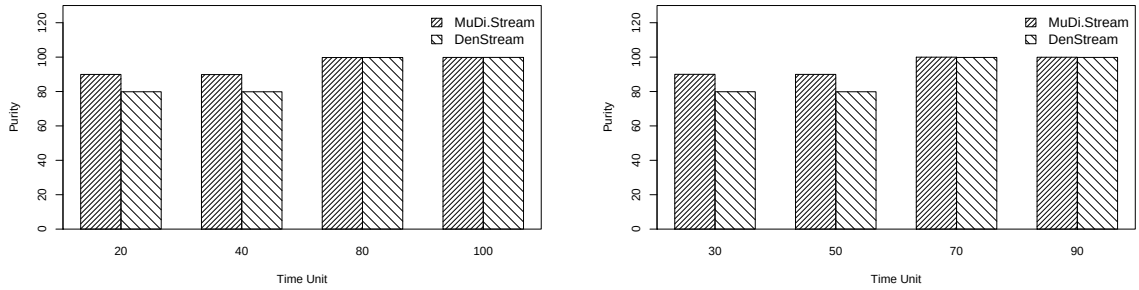
Figure 5.75: Clustering FM on LandSat (a) horizon = 1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500



(a) (b)

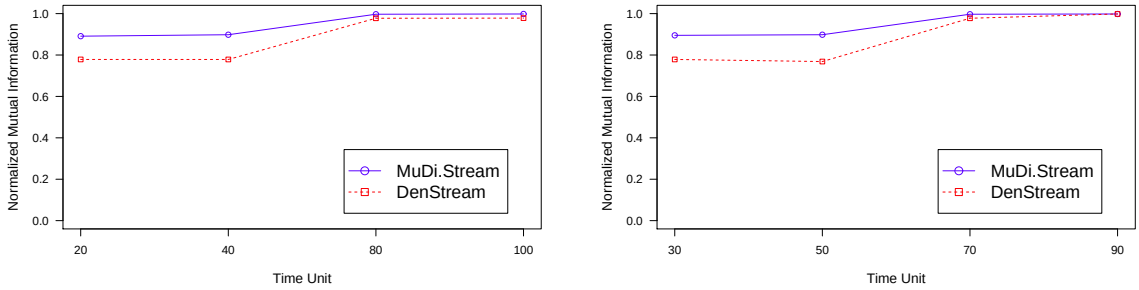
Figure 5.76: Clustering F-Measure on LandSat (a) horizon =1 and stream speed = 1000, (b) horizon = 3 and stream speed = 500

to 1 and stream speed 1000 and 2) horizon 5 with same stream speed in 30, 50, 70, and 90 time units. The degraded quality values during the initial clustering results were found to be due to the presence of all seven class types during the initial portion of the data combined with a lack of prior knowledge of cluster distributions. However, examining the stream processing shows that this initial poor results does not happen again later with



(a) (b)

Figure 5.77: Clustering Purity on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000

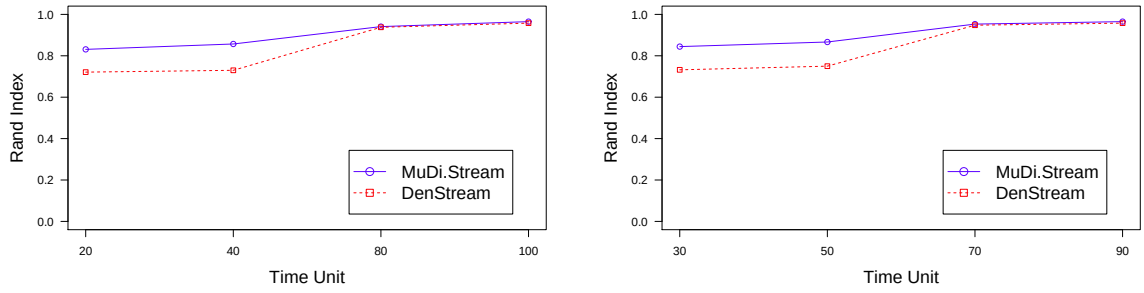


(a) (b)

Figure 5.78: Clustering Normalized Mutual Information on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000

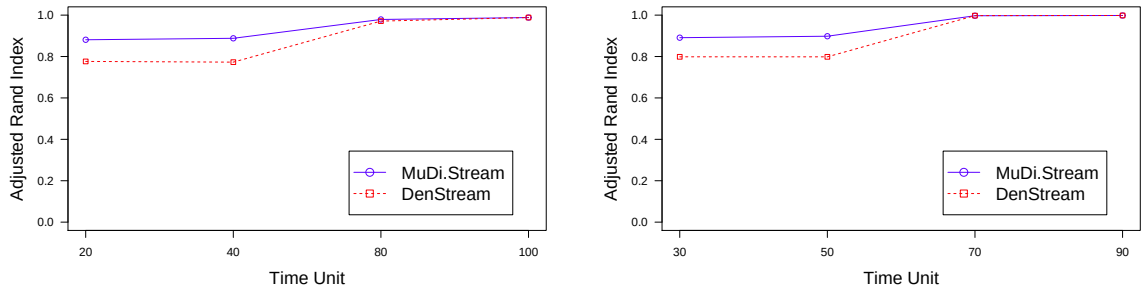
the simultaneously reappearance of all seven classes. For example, as it is depicted in Figure 5.83, F-Measure values in horizon 5 is 0.891, and 0.898 in time units 30, and 50 respectively; however, in the 70, and 90 time units the values increase to 0.979 and 0.999 in the order given.

The parameters of MuDi-Stream adopt the following settings: $\lambda = 1$, $Minpts = 5$, $\alpha = 0.04$, $gridGraunality = 25$, and DenStream's as in (Cao et al., 2006).



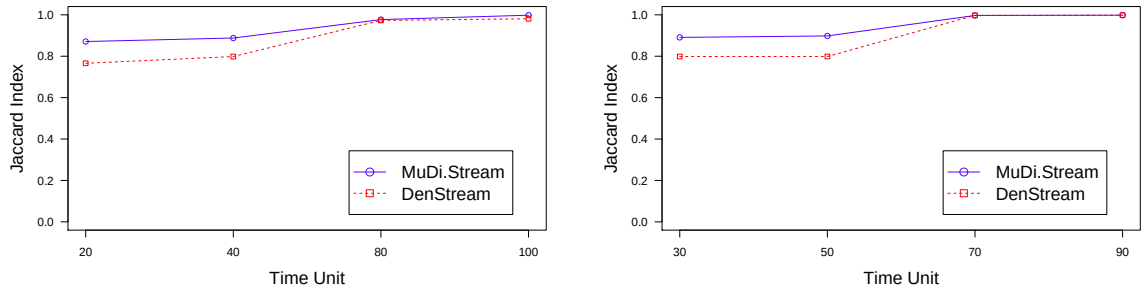
(a) (b)

Figure 5.79: Clustering Rand Index on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000



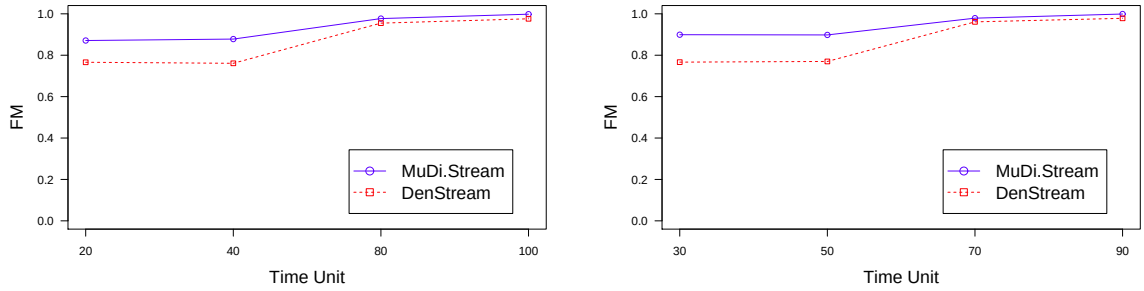
(a) (b)

Figure 5.80: Clustering Adjusted Rand Index on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000



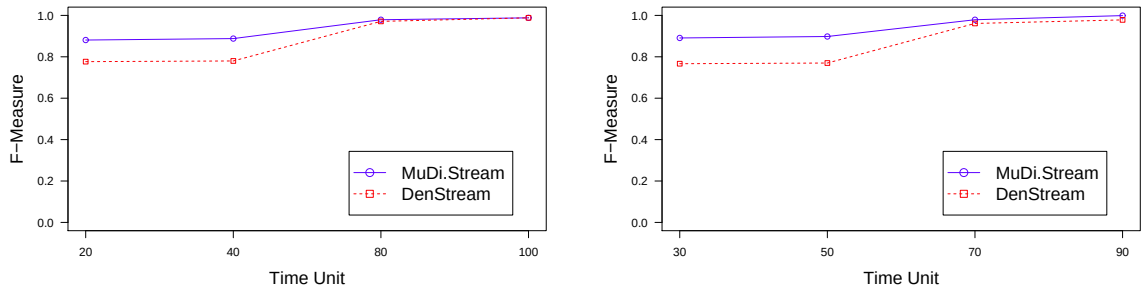
(a) (b)

Figure 5.81: Clustering Jaccard Index on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000



(a) (b)

Figure 5.82: Clustering FM on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000



(a) (b)

Figure 5.83: Clustering F-Measure on Forest cover type (a) horizon = 1 and stream speed = 1000, (b) horizon = 5 and stream speed = 1000

5.4 Quality Comparison of MuDi-Stream with a Grid-based Method

In this section, we compare the clustering result of MuDi-Stream with D-Stream (Tu & Chen, 2009). D-Stream is a grid-based method which cluster data stream in limited time. The main reason behind the low execution time of D-Stream is its synopsis storing method which is based on grid.

We compare the proposed algorithm with D-Stream to show that it performs better than D-Stream in terms of the quality of final clustering results. D-Stream forms final clusters by merging dense grids whereas our method forms core-mini-clusters from the grid and final clusters from the core-mini-clusters. We utilize the grid method to keep the outliers; however, final clusters are not generated according to the grid structure.

Figure 5.84 shows the precision (cf. Equation 2.17) comparison of MuDi-Stream versus D-Stream on Network Intrusion Detection Dataset. The values of the precision metrics are on different time units, 100, 150, 200, 250, and 300. The figure shows that in terms of precision MuDi-Stream outperforms D-Stream in most time units.

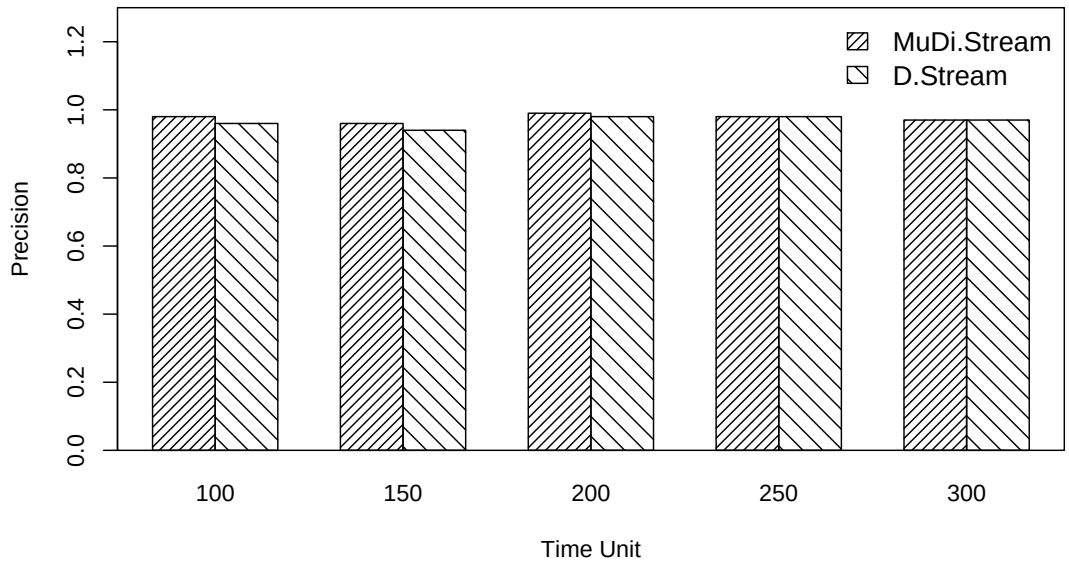


Figure 5.84: Precision of MuDi-Stream Compared to D-Stream on Network Intrusion Detection Dataset

5.5 Quality Comparison of MuDi-Stream with a Multi-density Method

DSCLU (Namadchian & Esfandani, 2012) is a multi density-based clustering algorithm for data streams. It has the ability to cluster multi-density data. However, the difference between this method and MuDi-Stream is that DSCLU clusters the data in the offline phase using a multi-density algorithm and yet it has high computation time. Our method is faster than DSCLU since it has three list of micro clusters including dense, transitional and sporadic. Searching in all these three lists is a time consuming task. MuDi-Stream has only one list for searching and a grid structure which is not as time consuming as DSCLU. Hence, the computation time is lower.

Figure 5.85 depicts the purity comparison of MuDi-Stream versus DSCLU on Network Intrusion Detection Dataset. The values of the purity metrics are on different time units, 100, 150, 350, and 470. The figure shows that in terms of purity MuDi-Stream is almost better than DSCLU in most time units. The difference is not notable; however, MuDi-Stream is a more efficient method which finishes its online phase much faster than DSCLU.

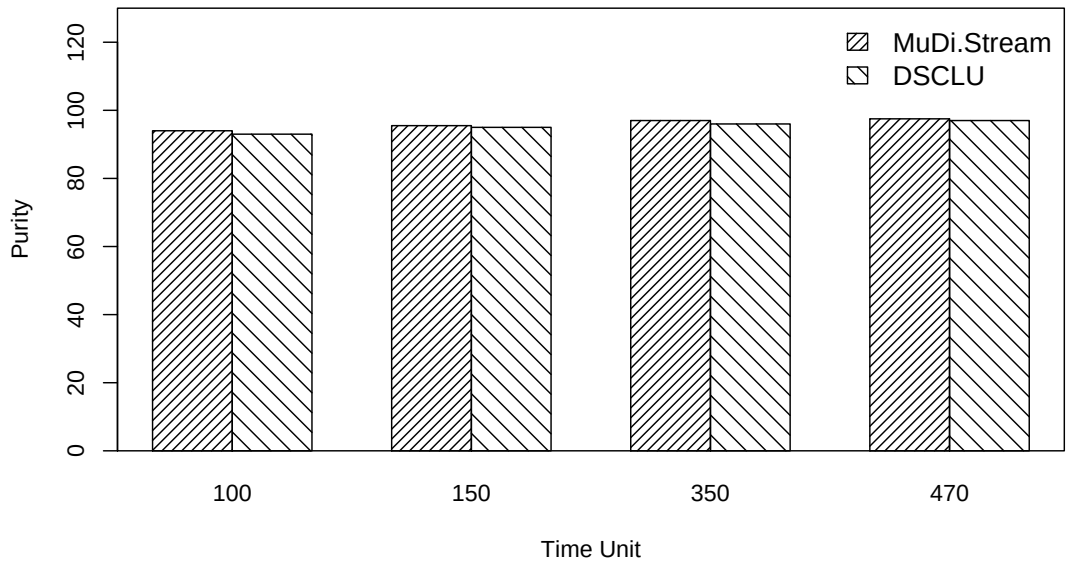


Figure 5.85: Purity of MuDi-Stream Compared to DSCLU on Network Intrusion Detection Dataset

5.6 Complexity Analysis

Streaming algorithms are required to have a fixed and small memory usage and a short computing time for the whole process.

In this section, we discuss about time and space complexity of MuDi-Stream Algorithm. The overall complexity of MuDi-Stream depends on its components complexities.

5.6.1 Space Complexity

The potential data which participate in the final clusters are kept in core-mini-clusters and the outliers are maintained in the grid list. For core-mini-clusters, we need $O(mc)$ space where mc is the number of core-mini-clusters. However, the core-mini-clusters are pruned frequently, and so the space complexity is $O(r_{mc})$ in which $r_{mc} < mc$. The space complexity for the grid is N. While the process continues the scattered grids are removed. Therefore, although N is exponential to the number of dimensions, the space complexity is $O(\log N)$ (according to Appendix A on Page 177).

Our experimental results also showed that for small, and even big datasets no matter the data is dense or scattered the memory consumption is not high. This is due to the pruning processes that are performed regularly on the core-mini-clusters as well as the grid list.

$$SpaceComplexity(MuDi-Stream) = SC(coreminiclusters) + SC(grid) =$$

$$O(r_{mc}) + O(\log N)$$

5.6.2 Time Complexity

In order to determine the time complexity of MuDi-Stream, we have to measure the time complexity of its online phase. Therefore, the complexity of each components of the online phase is calculated.

One of the main components of the online phase of MuDi-Stream is MM-Component which has two important tasks, i.e. merging or mapping. The first step in merging task is

finding the nearest core-mini-cluster in core-mini-cluster list for the new arrival data point from the data stream. In fact, the algorithm performs a linear search on core-mini-cluster list. So, the time complexity is $O(r_{mc})$ since the core mini clusters are pruned and the number is less than *mc* where *mc* is the number of core-mini-clusters.

If the data point cannot be merged into an existing core-mini-cluster, it will be mapped to the grid. In MuDi-Stream, we maintain a grid list which includes the grids that are under consideration for clustering analysis. The grid list is implemented as a tree, which allows for fast look up, update, and deletion. The key of the tree is the grid coordinates, while the associated data for each grid entry is the grid's synopsis. Since the space complexity is $O(\log N)$, the time complexity to search, and update in the tree structure is $O(\log\log N)$ which is very small. The time complexity of FCM-Component is almost zero since only a grid to core mini cluster conversion is performed.

In the PGCM-component, both grid and core mini clusters are pruned. Therefore, the entire list of core mini clusters have to examines which lead to $O(mc)$ as well as grid list is $O(logN)$.

In the existing methods such as DenStream when a new data point arrives, it takes time to search in two lists of micro clusters including potentials and outliers in order to find the suitable micro-cluster. However, MuDi-Stream only searches in potential list and if it cannot find the suitable core-mini-cluster, the data point is mapped to the grid, which keeps the outlier buffer. In fact, time complexity of clustering algorithm is decreased using the grid-based clustering.

The overall time complexity of MuDi-Stream is as follows:

$TimeComplexity(MuDi - Stream) = T(MM - Component) + T(PGCM - Component) = T_{search}(cmc) + T_{map}(g) + T_{pruning}(cmc) + T_{pruning}(g)$

$O(MuDi - Stream) = O(r_{mc}) + O(\log\log N) + O(1) + O(mc) + O(\log N)$

5.7 Scalability Evaluation

The following experiments are designed to evaluate the scalability of MuDi-Stream. The first part is used to evaluate the execution time and the second part is used to study the memory usage.

5.7.1 Execution Time

We use both Network Intrusion Detection and LandSat datasets to test the efficiency of MuDi-Stream against DenStream. Figure 5.86a shows the execution time for the Network Intrusion Detection dataset. We can see that both the execution time of MuDi-Stream and DenStream grow linearly as the stream proceeds, and MuDi-Stream is more efficient than DenStream. In addition, MuDi-Stream takes less than 3 seconds to process 20,000 data points. Thus, MuDi-Stream can comfortably handle high speed data streams. Furthermore, Figure 5.86b shows that MuDi-Stream is more efficient than DenStream for the LandSat dataset as well.

DenStream keeps two lists for micro-clusters: potential and outlier. When a new data point arrives, DenStream searches in two lists while in MuDi-Stream we only need to check a list of core-mini-clusters and then the grid list. Since the grid list is kept in a tree structure it is much more faster to find a grid cell to map the new data point. Therefore, it makes MuDi-Stream much faster than DenStream.

Then, the execution time of MuDi-Stream is evaluated on data streams with various dimensionality and different number of natural clusters (classes). Synthetic datasets are used for these evaluations because any combination of the number of natural clusters and dimensions can be obtained during the generation of datasets. Similar to the experiments performed in (Cao et al., 2006), the data points of each synthetic dataset follows a series of Gaussian distributions. We adopt the following notations to characterize the synthetic datasets: “B” indicates the number of data points in the dataset (times 1000), whereas

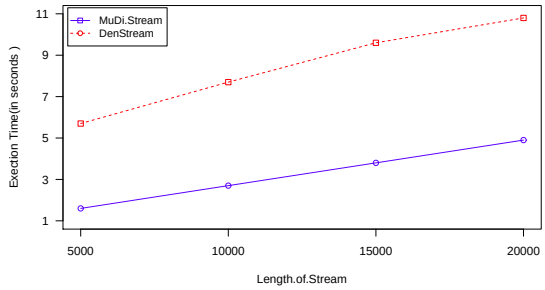
"C" and "D" indicate the number of natural clusters, and the dimensionality of each point, respectively. For example, B400C5D20 means the dataset contains 400,000 data points of 20-dimensions, belonging to 5 different clusters (classes).

1. The first series of datasets are generated by varying the number of natural clusters from 5 to 25, while keeping the size and dimensionality of the data streams fixed. Figure 5.87a shows that the execution time of MuDi-Stream is not dependent to the number of natural clusters. The execution time does not notably change since the search for core-micro-clusters is not dependant on the number of classes. For example, when the number of clusters increases from 5 to 25 for dataset series B200D40, the execution time only increases by 10 milliseconds (ms).

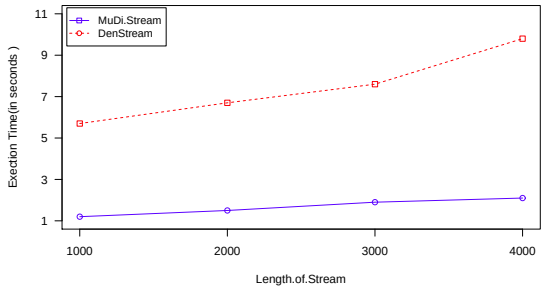
   We repeated the experiment for three different dimensions, i.e. 10, 20, 30, and 40. The results are similar when the number of classes are increased, i.e. increasing the number of classes does not notably change the execution time. The difference for the execution time of different numbers of dimensions are not high. The main cause for the small change is that the number of tree levels increase when we have a change in the dimensions of the dataset. So, it takes a bit longer to find map the data points into the grid list.

2. The second series of datasets are generated by varying the dimensionality from 10 to 40, while keeping the stream size and the number of natural clusters fixed. Figure 5.87b shows that the execution time grows linearly with respect to the dimensionality. Once more, this is because the time needed to map the new data point is a bit longer because the tree structure will have higher number of levels.

   Similarly, we repeated the experiment for three different settings, i.e. number of data points in the dataset and the number of classes. Figure 5.87b depicts that when the number of data points are doubled the time taken to process the whole data is

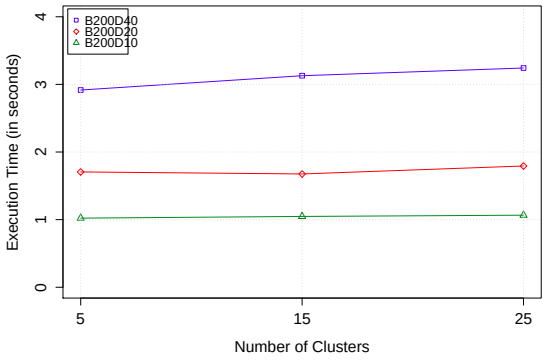


(a) Execution time for increasing stream lengths on Network Intrusion Detection dataset

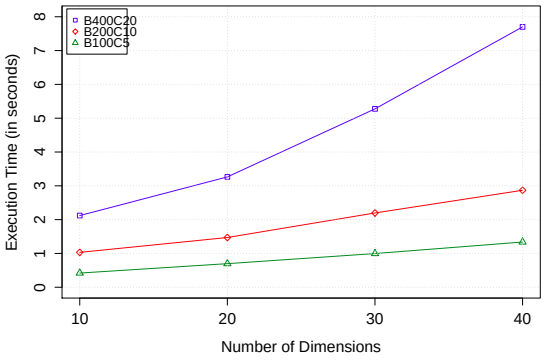


(b) Execution time for increasing stream lengths on LandSat dataset

Figure 5.86: Execution Time on two different datasets



(a) Execution time vs. number of clusters



(b) Execution time vs. dimensionality

Figure 5.87: Execution Time Changes

also doubled. So, the required time to process the same number of data points is constant and it is independent of the number of clusters.

5.7.2 Memory Usage

One common feature for the algorithms applied to data stream is their limited upper bounds for the memory usage. Since the memory usage may fluctuate in the progress of data streams, the maximum memory usage is used as the measurement. The entity used for the evaluation in MuDi-Stream is the core-mini-cluster.

For the comparison of memory usage, the stream length ranges from 10000 to 30000 for four datasets including Network Intrusion Detection, LandSat, EDS, and EMDS. As it is shown in Figure 5.88, for real datasets, the memory usage of MuDi-Stream is limited.

The number of core-mini-clusters is less than 30. Moreover, Figure 5.88 shows that the memory usage of MuDi-Stream is bounded as the streams proceed for synthetic datasets. For example, for EDS data stream, when the stream length changes from 15000 to 30000, the memory usage only increases by 5.

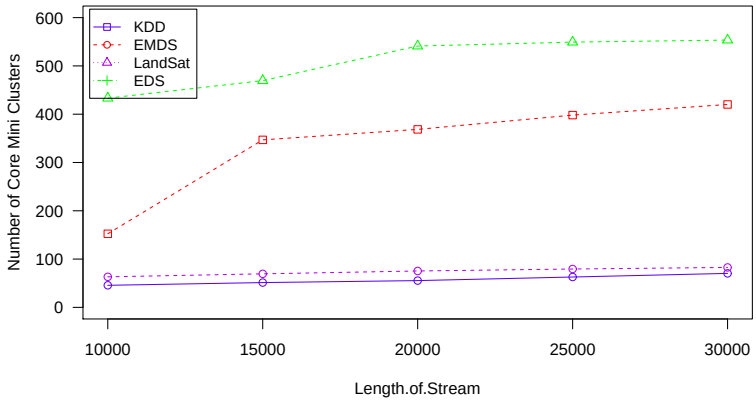


Figure 5.88: Memory Usage

5.8 Sensitivity Evaluation

The prominent parameters of MuDi-Stream include λ, α, and *gridGranularity*. We test clustering quality using seven different metrics over different ranges of parameters on different datasets. The results are shown for LandSat dataset.

5.8.1 Outlier Threshold: α

One of the important parameters of MuDi-Stream is the outlier threshold. Figure 5.89 depicts the clustering quality when α varys from 0.01 to 0.3. If α ranges between 0.03 to 0.2, the clustering quality is very good. When the values of α are too small it leads to small pruning times. Therefore, pruning happens more frequently and more core-mini-clusters are pruned. Hence, the quality is reduced.

This implies that since α and N has direct effect on each other. When a dimension is so high the bigger values of α is more preferable to decrease the effect of high values

of N. However, in the datasets with lower dimension, better result can get using smaller values of α.

The important note is that the α which is used in algorithm is multiple by a coefficient. Otherwise, since N is a big number, α has no effect on it.

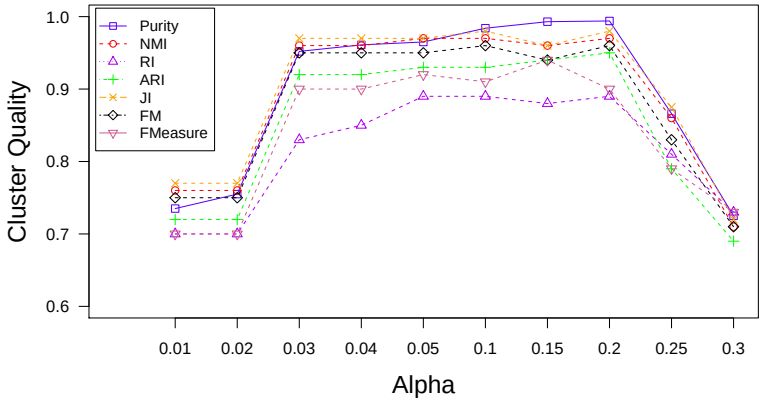


Figure 5.89: Clustering Quality vs. α

5.8.2 Density Threshold: λ

λ is another parameter of MuDiStream. It controls the importance of historical data to the current clusters. We test the clustering quality by varying it from 0.031 to 32. Figure 5.90 shows the results. When it is set to a relatively small or high value, the clustering quality becomes poor. For example, when $\lambda = 0.031$, the NMI is about 0.69. Further, when $\lambda = 32$, the points decay soon after their arrival and only a small number of recent points participate in the final clustering. So, the result is also not very good. It can been seen that if λ ranges from 0.125 to 8, the clustering quality is quite good and stable, and always above 90%.

5.8.3 Grid Granularity: *gridGranularity*

gridGranularity is a parameter which affects the clustering quality of MuDi-Stream. We varied the value of *gridGranularity* parameter of MuDi-Stream from 5 to 40 increasing by 5 in order to investigate the sensitivity of the algorithm. The results on sen-

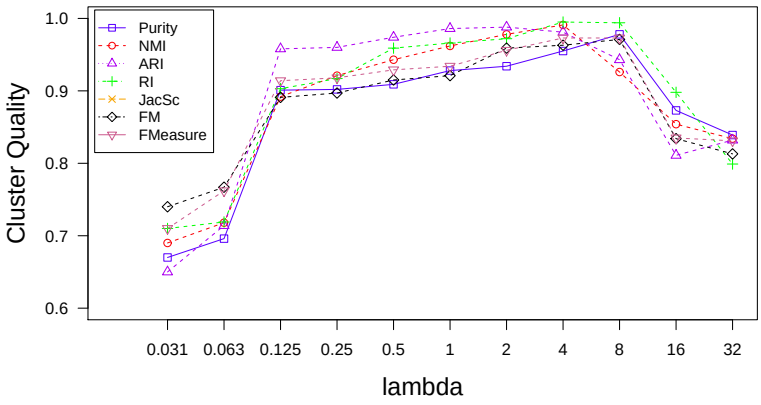


Figure 5.90: Clustering Quality vs. λ

sitivity analysis are shown in Figure 5.91. It can be observed that the best range for *gridGranularity* is between 20 to 30.

According to the sensitivity results for *gridGranularity*, we concluded that for the datasets with denser clusters, higher *gridGranularity* values achieve better results.

The *gridGranularity* is an important parameter in MuDi-Stream in terms of the algorithm's execution time. It has a significant impact on execution time. This is due to the change in the number of nodes of the tree structure used for the grid. Higher grid granularity causes higher number of children for each node in the tree.

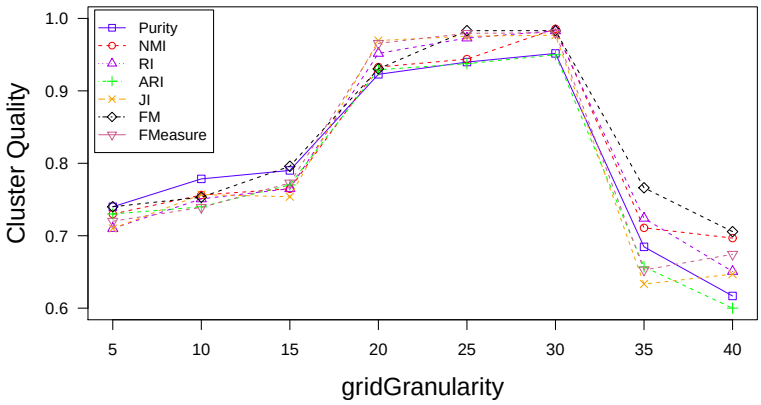


Figure 5.91: Clustering Quality vs. *gridGranularity*

5.9 Summary

In this chapter, we introduced the real and synthetic datasets which are used for the evaluation of MuDi-Stream. The real and synthetic datasets are chosen from the reviewed literature which are most used. They have variety in size, number of clusters, and differences in their densities. A wide spectrum of experiments have been conducted in this section as well.

Seven well-known evaluation metrics including Purity, Normalized Mutual Information, Rand Index, Adjusted Rand Index, Jaccard Index, FM, F-Measure are selected to show the high quality of the proposed algorithm. The metrics are calculated on selected time units, stream speeds and horizons. Evaluations on several datasets using the quality metrics show that MuDi-Stream achieves much better results compared to its competitive methods.

We also measure the scalability of the proposed methods with different numbers of dimensions and clusters. The scalability results show that MuDi-Stream is scalable with respect to the length and dimensionality of the data streams. MuDi-Stream shows linear scalability on both the number of clusters and the number of data dimensions. In terms of efficiency, MuDi-Stream finished the clustering process significantly faster than DenStream.

Furthermore, by varying important parameters of MuDi-Stream, the clustering quality of MuDi-Stream is measured. The sensitivity analysis determined the best range for prominent parameters of the proposed algorithm, therefore it can be applicable for any researcher who wants to perform MuDi-Stream on a new dataset.

The complexity analysis of MuDi-Stream shows that the time complexity of the computation is less than the existing methods. It is worth to note that, despite the lower computation time of MuDi-Stream, the quality obtained is much higher than that obtained

by DenStream specifically in multi-density datasets.

It is proved that using grid-based clustering in the online phase to map a new data point while it might be noise and forming final clusters from real data using a density-based clustering algorithm improve the quality and efficiency of the proposed algorithm. We conclude that MuDi-Stream is an effective and efficient clustering algorithm in the clustering evolving data stream with various densities.

CHAPTER 6

CONCLUSION

6.1 Overview

This study proposed an adaptive density-based clustering method for evolving data streams. The method not only can overcome the challenges to cluster evolving data stream but also can cluster multi-density data stream. Furthermore, it has low computation time and high quality with acceptable memory usage. The summary of finding, limitations of the research and some recommendations are given in the following sections.

6.2 Summary of Results

We summarize the results by answering research questions from Chapter 1 as follows:

Question 1: which method is more appropriate for summarizing data stream?

In data stream, data points arrive continuously over time with high speed, and the size of a stream is (potentially) unbounded. Therefore, there is not enough time to process and memory to keep. The synopsis model of data stream should not be only compact, but also does not grow with the number of data points processed. Therefore, in this thesis we proposed a hybrid method for data stream synopsis. A micro cluster method is used to keep summary information about arbitrary shape clusters while grid based method maintains the outliers. These methods have two aforementioned characteristics. Our new proposed core-mini-cluster concept is a vector which keeps summary information of data points as well as some information for multi-density data. The grid-based method, also keeps some information about data points inside each grid cell in grid synopsis. When the data is mapped into the grid, the information related to that cell is updated and it is

not needed to update the whole grid. This method of updating synopsis data, makes our computation time significantly faster.

Question 2: How to handle evolving data stream?

There are different methods for clustering evolving data stream. One of the well-known methods is fading window model which is explained in Section 2.2.1. For handling evolving data stream, a weight coefficient is considered for each data point. The coefficient decreases exponentially over time. In this method, we have more emphasis on the recent data. Since the coefficient is related to the arrival time of the data point, the recent data has more weight compared to the old data.

Question 3: What are the reasons of high computation time?

According to the comprehensive review of the existing methods, the high computation time is because of searching to find suitable place to add new arrival data to existing synopsis data. However, in this thesis we decrease this exhaustive search with mapping into the grids. The existing method keeps an extra list for outliers, which leads to high computation time. However, we introduce grid-based method in which when a new data point arrives and it cannot be located in any existing synopsis, it is mapped to the grid.

Question 4: How to lower the computation time?

In this thesis, a hybrid method consisting of micro-clustering and grid-based technique is used to decrease the computation time. 1) Micro-clustering method is used to form real clusters. 2) Grid-based method is used to keep outliers because of its fast processing time. Low computational complexity of grid data structure has been the main motivation for using it as synopsis.

Question 5: What issues impede the clustering quality in multi-density environments?

Based on the literature, the main reason which decreases the clustering quality in multi-density environments is using a similar radius for different distributions of data. In

the existing methods of density-based clustering for data stream, a small value of radius will detect the sparse clusters as noise and a big number may not detect the dense clusters properly. In fact, the existing methods do not adopt the clustering parameters according to the density distribution of data.

Question 6: How to increase the clustering quality in multi-density data?

One of the well known frameworks for clustering data streams is the online-offline framework. In this thesis, not only we consider to improve quality in online phase but also in the offline phase. In the online phase, we keep summary information about data as well as information for multi-density clustering. In the offline phase, we also propose a new multi-density clustering algorithm which uses the online synopsis information and yet statistical analysis to form final clusters considering density distribution of data. The radius parameters of clustering are adjusted according to the density distribution of data. This leads to high quality clustering specially in multi-density data.

6.3 Achievement of Objectives

The objectives of this research are as follows:

1. To propose and develop a new density-based algorithm for clustering evolving data stream.

2. To improve the quality of clustering for multi-density data. The clustering quality has to be high for normal distribution data as well as for the data with dissimilar density distributions.

3. To reduce the computation time. The computation time has to be low enough to cope with the speed of arriving data stream.

4. To evaluate the capability of the proposed methods in improving the quality.

A multi-density based clustering algorithm, MuDi-Stream, is proposed to cluster evolving data stream in order to achieve the first objective. The proposed algorithm was designed, implemented and its improvement in quality is shown by a comprehensive evaluation using different datasets with different metrics.

The methods and results are summarized as follows:

- A hybrid method was introduced for the online phase consist of grid- and micro-clustering. The method can handle multi-density data with low computation time (cf. Section 4.7.1).

- A pruning method was proposed in the online phase to discard both the grid cells which are scattered and the core-mini-clusters with low density. The pruning is performed frequently (cf. Section 4.7.3).

- A new density-based clustering algorithm is also proposed in the offline. The algorithm has the ability to cluster multi-density data from synopsis information received from the online phase (cf. Section 4.8.1).

The proposed method has high quality for multi-density data stream due to the consideration of the density distribution of data. This has been proved in the experimental evaluation explained in Chapter 5. This has answered the second objective. The radius of clustering is updated by density distribution of data using information kept in the online phase as well as some statistical information.

The proposed method reduced the computation time to achieve the third objective. This is discussed in Section 5.6. Three strategies caused the low computation time: 1) Using the grid-based method which maps the outliers, 2,3) pruning the grid and the core-mini-clusters frequently.

An extensive evaluation was performed to show the superiority of the method to the existing methods. This has been done to achieve the forth objective. The proposed method's evaluation is not biased to any dataset, size, parameter, or quality metrics. We claim the unbiased evaluations due to the following analysis:

- The proposed method was evaluated on various datasets with different numbers of data points, sizes and densities.
- Different synthetic datasets with different density distribution are used to measure the quality of clustering in the multi-density environments.
- Seven well-known evaluation metrics in the literature were used to evaluate clustering quality.
- The range of algorithm's parameters were determined in the sensitivity analysis of the method.

6.4 Contributions

There are a number of density-based clustering algorithms for data stream. However, they have high computation time due to their high merging time. Furthermore, they do not have the ability to cluster multi-density data which leads to their low quality results. On the other hand, there are some multi-density clustering algorithms for static datasets which are not applicable for data stream since they have some problems such as requiring whole data for clustering or high execution time. Yet only a few approaches have been proposed so far which tackle both the stream and the multi-density aspects of the data simultaneously. Therefore, a new density-based clustering was presented in this thesis to overcome the aforementioned problems. Furthermore, according to data stream characteristics, the challenges in clustering data streams had to be considered in the pro-

posed algorithm. The specific research contributions of this thesis can be summarized as follows:

- a new multi density-based clustering for evolving data streams (MuDi-Stream)
- a new synopsis structure for multi density data
- a new hybrid method for online phase of algorithm using grid and micro clustering method
- a new multi density-based clustering for offline phase (M-DBSCAN)
- an extensive evaluation

A new multi density-based clustering for evolving data streams (MuDi-Stream): MuDi-Stream consists of four main components: MM-component, FCM-component, PGCM-component and FFC-component. The first three components are used in online phase while the last component is related to offline phase. They perform four important tasks including merging or mapping, forming core mini clusters, pruning grids and core mini clusters, and forming final clusters.

MuDi-Stream offers a novel approach for clustering data stream from a multi density environment. The approach complies with all data streams clustering requirements. It requires only one scan of data. It updates new data observations received from data sources to the previously mined models. The system spends a small amount of processing time per data point.

The proposed method can overcome challenges in clustering data stream as follows:

- Handling evolving data streams: using fading window model
- Handling noisy data: map them into grids, mark as scattered grids and pruned them

- Clustering in limited time: using a hybrid method for clustering allow us to perform clustering in limited time.

- Clustering in limited memory: by pruning grids and micro clusters frequently, we try to keep the memory bounded.

A new synopsis structure for multi density data: Some new concepts are introduced including mini core distance, core mini clusters, and core-neighboring in order to handle multi-density data stream.

A new hybrid method for online phase of algorithm using grid and micro clustering method: A hybrid method based on the micro-clustering and grid-based clustering is applied to capture summary information of the data points. Grid-based method is used for mapping outliers and forming new mini clusters with different radius. This is conducive to improve the quality in multi-density data as well as a dramatic decrease in merging time. In fact, MuDi-Stream replaces the exhaustive search of assigning a point to the appropriate outlier micro-cluster with a grid mapping task.

A new multi density-based clustering for offline phase (M-DBSCAN): A density-based clustering algorithm called M-DBSCAN is proposed for the offline phase which forms final clusters with various densities from the synopsis data.

An extensive evaluation: A comprehensive set of experiments were conducted on both synthetic and real-world datasets using various kinds of evaluation metrics. The results have proved that the proposed method can clusters data stream in multi-density environments in comparison with the existing methods. Our algorithm has better clustering quality, efficiency, and scalability than existing methods. We performed a comprehensive evaluation on ten different dataset with various numbers of size, cluster, dimension and density. Seven different metrics consist of purity, normalized mutual information, Rand Index, Adjusted Rand Index, FM, F-measure is evaluated the quality of proposed method.

The evaluation proved that our proposed method has higher quality compare to existing method.

6.5 Limitations of Current Study

- We observed that, the clustering algorithms cannot deal with high dimensional data. The number of grids will be increased as the space dimensionality grows. They have low performance on very high dimensional data. Therefore, further research may involve handling both the high dimensional data in density-based data stream clustering and at the same time handling the other challenges.

- The method uses fading window model for clustering evolving data streams. The window model is chosen based on the best and most used method in the literature. Nevertheless, more analysis are need to check whether other window models can increase the clustering quality in case of having multi-density clusters in our data.

- In this thesis, Euclidean distance is used. However, more analysis is required to determine if other kinds of distances such as mahalanobis distance can increase the quality.

6.6 Recommendations and Future Directions

The researcher strongly believes that the proposed method in this thesis highly improves not only the quality of density-based clustering algorithms in multi-density environments but also has low computation time. Hopefully, the proposed method is used by other researchers in other real applications with multi-density distributions or in applications with requirements of limited time. Apart from the above improvements, some important refinements are discussed as follows:

- In real applications, we have other different kinds of data such as categorical or uncertain. Extending the proposed method to be applicable for other types of data

is required for clustering evolving data stream.

- Different applications produce infinite streams of data distributed from various resources such as sensor networks. Therefore, data should also be processed in a distributed fashion. So, the method can be adapted to be applicable in distributed environments.

- High dimensional data are collected in many scientific projects, humanity research, or business processes. This is to better understand the phenomena that the researchers or managers are interested in. Therefore, extending the proposed method to be applicable for high dimensional data is a further research topic.

- With the emergence of big data, which is evolving and changing, data stream is a specific approach to deal with it. Therefore, extending proposed algorithms to meet the requirements for big data is another interesting issue for research.

We hope that the research presented in this thesis will inspire more research from the researchers and practitioners in the field.

Appendices

APPENDIX A

OUTLIER THRESHOLD FORMULA

Definition 25 ***(Outlier Weight Threshold function (OWT)).*** *If the last updated time of a grid g is t_p then at current time t_c, Outlier Weight Threshold (OWT) is defined as follows ($t_c > t_p$):*

$$OWT(t_p, t_c) = \frac{\alpha}{N} \sum_{i=0}^{t_c - t_p} 2^{-\lambda i} = \frac{\alpha(1 - 2^{-\lambda(t_c - t_p + 1)})}{N(1 - 2^{-\lambda})} \tag{A.1}$$

Lemma 3 *The outlier weight threshold function has the following attributes:*

1. *if $t_1 \leq t_2 \leq t_3$, then*

$$2^{-\lambda(t_3 - t_2)} owt(t_1, t_2) + owt(t_2 + 1, t_3) = owt(t_1, t_3)$$

2. *if $t_1 \leq t_2$ then*

$$owt(t_1, t) \geq owt(t_2, t), t > t_1, t_2$$

Proof.

1.

$$2^{-\lambda(t_3 - t_2)} owt(t_1, t_2) + owt(t_2 + 1, t_3) = \frac{\alpha}{N} \sum_{i=0}^{t_2 - t_1} 2^{-\lambda(t_3 - t_2 + i)} + \frac{\alpha}{N} \sum_{i=0}^{t_3 - t_2 - 1} 2^{-\lambda i}$$

$$= \frac{\alpha}{N} \left(\sum_{i=t_3 - t_2}^{t_3 - t_1} 2^{-\lambda i} + \sum_{i=0}^{t_3 - t_2 - 1} 2^{-\lambda i} \right) = \frac{\alpha}{N} \sum_{i=0}^{t_3 - t_1} 2^{-\lambda i} = owt(t_1, t_3).$$

2. $\delta t = t_2 - t_1$

$$owt(t_1, t) = \frac{\alpha}{N} \sum_{i=0}^{t - t_1} \lambda^i = \frac{\alpha}{N} \sum_{i=0}^{t - t_2 + \delta t} 2^{-\lambda i}$$

$$\frac{\alpha}{N}(\sum_{i=0}^{t-t_2} 2^{-\lambda i} + \sum_{i=t-t_2+1}^{t-t_2+\delta t} 2^{-\lambda i}) = \frac{\alpha}{N}\sum_{i=0}^{t-t_2} 2^{-\lambda i} + \frac{\alpha}{N}\sum_{i=t-t_2+1}^{t-t_2+\delta t} 2^{-\lambda i}$$

$$= owt(t_2,t) + \frac{\alpha}{N}\sum_{i=t-t_2+1}^{t-t_2+\delta t} 2^{-\lambda i} \geq owt(t_2,t).$$

The design of density threshold function is explained as follows. Assume a grid g has been deleted at time steps $t_1, t_2, \ldots, t_k = t$. We need to guarantee that g cannot become a core mini cluster even if it has not been deleted. therefore we should have: $w_g(t) \leq \frac{\alpha(1-2^{-\lambda(t+1)})}{N(1-2^{-\lambda})}$

we also have:

$w_g(t) \leq \sum_{i=1}^{k} owt(t_{i-1},t_i)2^{-\lambda(t-t_i)}$ therefore, we need

$$\sum_{i=1}^{k} owt(t_{i-1},t_i)2^{-\lambda(t-t_i)} \leq \frac{\alpha(1-2^{-\lambda(t+1)})}{N(1-2^{-\lambda})} \tag{A.2}$$

$$\frac{\alpha(1-2^{-\lambda(t+1)})}{N(1-2^{-\lambda})} = \frac{\alpha}{N}(1+2^{-\lambda}+\ldots+2^{-\lambda t})$$
$$= \frac{\alpha}{N}(1+2^{-\lambda}+\ldots+2^{-\lambda(t_i-t_{i-1})})2^{-\lambda(t-t_i)}$$
$$= \sum_{i=1}^{k} \frac{\alpha(1-2^{-\lambda(t_i-t_{i-1}+1)})}{N(1-2^{-\lambda})}2^{-\lambda(t-t_i)}$$

(A.3)

from A.2 and A.3, we have

$owt(t_{i-1},t_i) = \frac{\alpha(1-2^{-\lambda(t_i-t_{i-1}+1)})}{N(1-2^{-\lambda})}$

Theorem. The size of grid list at most $L = \frac{1}{\lambda}\log\frac{\alpha}{N+\alpha} N$ is the total number of grid and λ is the decay factor.

Proof. $t - t_p = L$ the density of each grid g at time t equals

$w_g(t) = w_g(t-t_p)2^{-\lambda(t-t_p)} < w_g(t-t_p)2^{-\lambda L} < \frac{2^{-\lambda L}}{1-2^{-\lambda}}$

(because the maximum density is $\frac{1}{1-2^{-\lambda}}$)

Since $N2^{-\lambda L}+\alpha 2^{-\lambda(t-t_p+1)} < N2^{-\lambda L}+\alpha 2^{-\lambda(t-t_p+1)} < N2^{-\lambda L}+\alpha 2^{-\lambda L}=\alpha$

Then $N2^{-\lambda L} < \alpha-\alpha 2^{-\lambda(t-t_p+1)}$

Therefore, we have

$$w_g(t) \leq \frac{2^{-\lambda L}}{1-\lambda} = \frac{N2^{-\lambda L}}{N(1-\lambda)} < \frac{\alpha-\alpha 2^{-\lambda(t-t_p+1)}}{N(1-\lambda)} = \frac{\alpha(1-2^{-\lambda(t-t_p+1)})}{N(1-\lambda)} = owt(t_p,t)$$

Proposition 2. Assume the last time a grid g is deleted as scattered grid is t_k and the last updated of grid g is t_p. If at current time t, we have $w_g(t) < owt(t_p,t)$, then we also have $w_g(t) < owt(0,t)$

Proof. Suppose the grid g has been deleted before for the periods of $(0,t_1),(t_1+1,t_2),\ldots,(t_{k-1}+1,t_m)$, then the density value $w_g(t_i)$, $i=1\ldots k$ satisfies:

$$w_g(t_i) < owt(t_{i-1}+1,t_i) \tag{A.4}$$

if the previous data is not deleted, we have density threshold function as follows:

$$w_g(t) = \sum_{i=1}^{k} w_g(t_i)2^{-\lambda(t-t_i)} + w_g(t) < \sum_{i=1}^{k} owt(t_{i-1}+1,t)2^{-\lambda(t-t_i)} + owt(t_p,t) \tag{A.5}$$

Because $t_p \geq t_m+1$, according to attribute 2 of Definition 25, we have:

The last equalities are based on the attribute 1 of Definition 25.

Definition 26 ***(Pruning time).*** *We check the cmc' weight as well as grids' in a specific time called t_{pt}. t_{pt} is the minimum time for a cmc in timestamp t_1 to be converted to an outlier in t_2 ($t_2 > t_1$), which is formally defined as follows:*

Lemma 4

$$t_{pt} = \frac{1}{\lambda}\log_2^{\frac{\alpha}{\alpha-1+2^{-\lambda}}}$$

Proof.

$$w_{cmc}(t_2) < w_{cmc}(t_1) = \frac{\alpha}{N(1-2^{-\lambda})} \tag{A.6}$$

$$2^{-\lambda(t_2-t_1)} * w_{cmc}(t_1) + 1 < w_{cmc}(t_1)$$

$$2^{-\lambda(t_2-t_1)} < \frac{w_{cmc}(t_1)-1}{w_{cmc}(t_1)}, \quad t_{pt} = t_2 - t_1$$

$$2^{-\lambda t_{pt}} < \frac{w_{cmc}(t_1)-1}{w_{cmc}(t_1)}$$

$$t_{pt} > \frac{1}{\lambda} \log_2^{\frac{w_{cmc}(t_1)}{w_{cmc}(t_1)-1}} \tag{A.7}$$

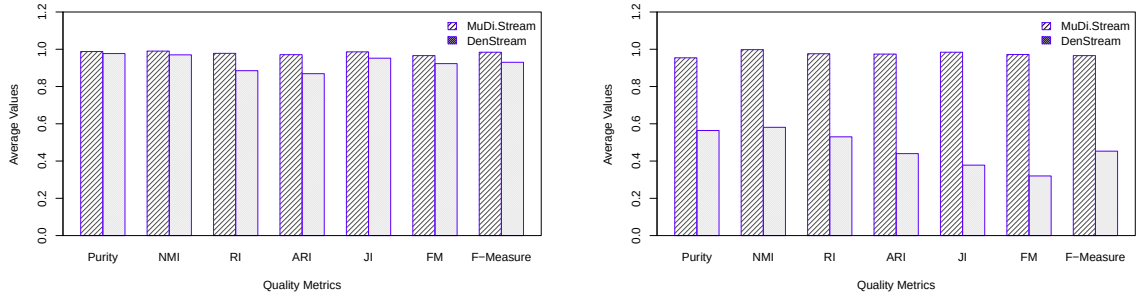
From Equations A.6, and A.7 we have:

$$t_{pt} = \left\lceil \frac{1}{\lambda} \log_2^{\frac{\alpha}{\alpha-1+2^{-\lambda}}} \right\rceil \tag{A.8}$$

APPENDIX B

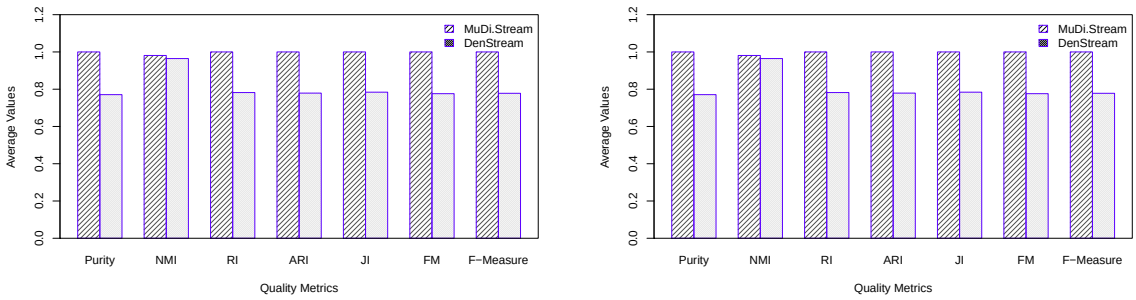
AVERGAE QUALITY COMPARISON

Since each dataset is evaluated on different horizon stream speed. In this Section, we also evaluated quality metrics on different dataset by measuring the average values on the whole data stream. As it is shown in Figures B.1, B.2, B.3, B.4, B.5. MuDi-Stream is outperform DenStream on all quality metrics on whole data stream.



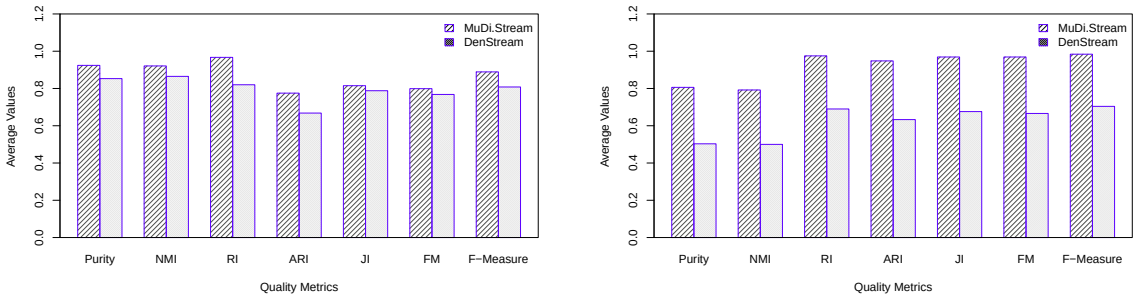
(a) (b)

Figure B.1: Average of Quality Metrics on a) EDS and b) EMDS



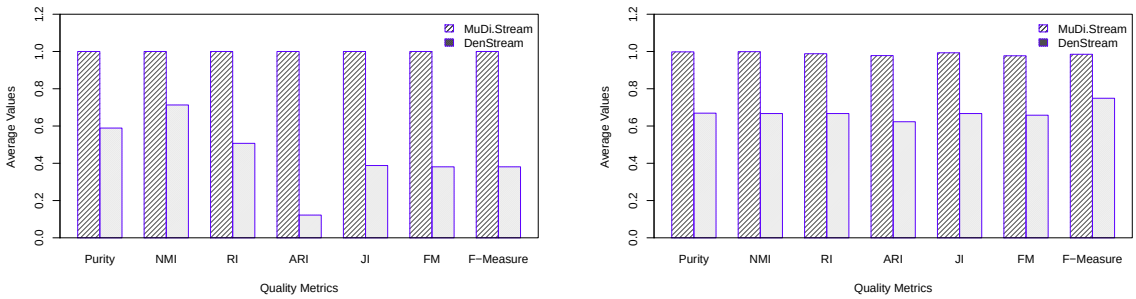
(a) (b)

Figure B.2: Average of Quality Metrics on a) Forest and b) GMDS



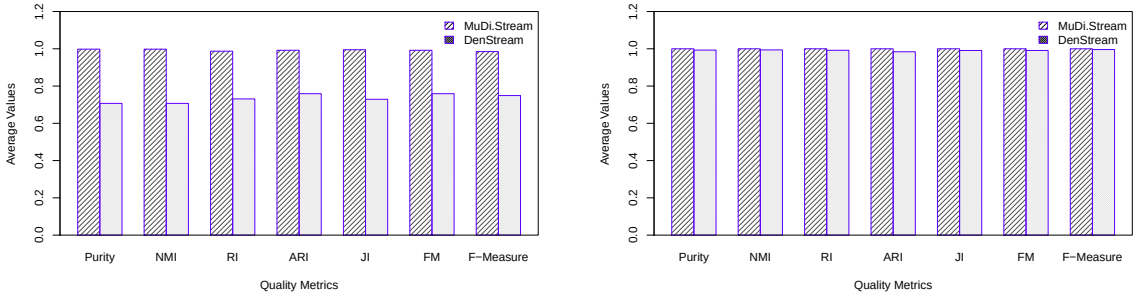
(a) (b)

Figure B.3: Average of Quality Metrics on a) KDD and b) LandSat



(a) (b)

Figure B.4: Average of Quality Metrics on a) MDS1 and b) MDS2



(a) (b)

Figure B.5: Average of Quality Metrics on a) MDS3 and b) MDS4

APPENDIX C

INTERNAL QUALITY EVALUATION

In terms of internal quality evaluation, we have evaluated the final clustering results of MuDi-Stream on one of the datasets, i.e. Gaussian Multi-density Dataset (GMDS). Other results on this dataset are presented in Section 5.3.7.

Figure C.1 illustrates the silhouette plot of the clustering results. The dataset includes five clusters. The average Silhouette values are reported in the plot. MuDi-Stream produced compact and well-separated clusters since Silhouette value is close to 0.9.

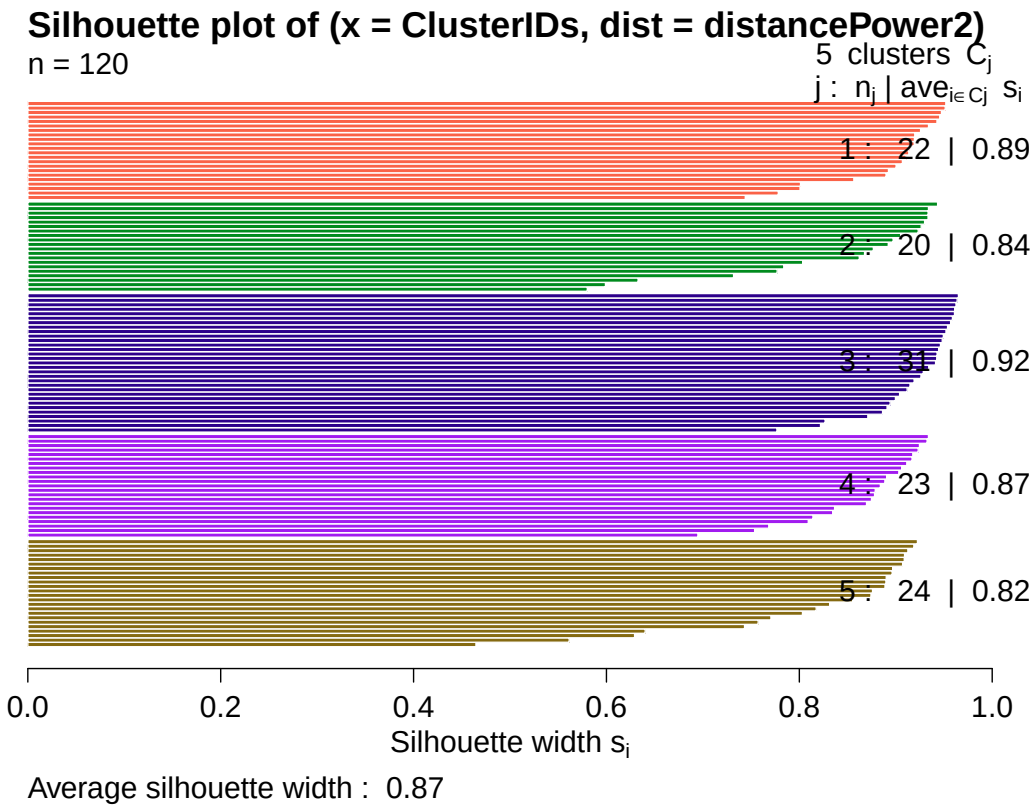


Figure C.1: Silhouette Plot for Gaussian Multi-density Dataset (GMDS)

APPENDIX D

THE DETAILS OF THE EXPERIMENTAL RESULTS

D.1 House (MDS2) Dataset Results

The results of the execution of MuDi-Stream on a multi-density dataset which we called it House (MDS2) is presented in Section 5.3.3 on page 131. The details of the execution are presented in Table D.1.

The parameters of MuDi-Stream for this execution adopt the following settings. We measured the quality metrics with the horizon set to 5 and stream speed 1000.

- $\lambda = 16$,
- $Minpts = 5$,
- $\alpha = 0.9$,
- $gridGraunality = 30$

In Table D.1 the following values are identified:

- **Point Number**: It shows the number of data points arrived at this time unit. The dataset contains 12131 points and we measured the quality metrics for each 2000 points as well as at the end of the data stream.
- **CMC Count**: It represents the number of core-mini-clusters generated by the online phase of MuDi-Stream.
- t_{online}: The time elapsed (seconds) from the execution's commencement until this data point arrived.

- **Quality Metrics**: The values of several quality metrics including Purity, Entropy, NMI, RI, ARI, Jaccard Score, Precision, Recall, FM, and F-Measure are measured and reported.

Point Number	CMC Count	t_{online}	Purity	Entropy	NMI	RI	ARI	Jaccard Score	Precision	Recall	FM	F-Measure
2000	642	0.04	95.794	2.225	0.961	0.861	0.658	0.77	1	0.77	0.77	0.87
4000	1009	0.08	95.837	2.197	0.971	0.967	0.911	0.935	1	0.935	0.935	0.966
6000	1129	0.12	99.734	2.205	0.996	0.998	0.996	0.997	0.999	0.998	0.997	0.998
8000	1163	0.16	99.656	2.196	0.998	0.999	0.998	0.998	0.999	0.999	0.998	0.999
10000	1184	0.2	99.747	2.188	1	1	1	1	1	1	1	1
12131	1223	0.25	94.849	2.187	0.966	0.968	0.913	0.937	1	0.937	0.937	0.967

Table D.1: The details of an execution of MuDi-Stream on MDS2 dataset

The number of classes in this dataset are five. MuDi-Stream could find the correct number of clusters. For example, in time unit 10 for which 10000 points were read, MuDi-Stream could find all 5 classes as five clusters. These five clusters are generated by MDBSCAN using 1184 core-mini-clusters. The details are presented in Table D.2. The table shows that, for example, 174 core-mini-clusters which all have class ID 1 are clustered in Cluster 3.

Class Number	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
1	0	0	174	0	0
2	0	0	0	0	240
3	0	166	0	0	0
4	445	0	0	0	0
5	0	0	0	159	0

Table D.2: Number of core-mini-clusters for the classes and clusters by an execution of MuDi-Stream on MDS2 dataset

D.2 CylinderCube (MDS4) Dataset Results

The results of the execution of MuDi-Stream on a multi-density dataset which we called it CylinderCube (MDS4) is presented in Section 5.3.6 on page 138. The details of the execution are presented in Table D.3.

In Table D.3 the following values are identified:

- **ID**: It is a sequential number which shows different executions of MuDi-Stream.
- **Point Number**: It shows the number of data points arrived at this time unit. The dataset contains 10000 points and we measured the quality metrics for each 5000 points.
- Parameters' Settings: The values of MuDi-Stream's parameters: λ, *minPts*, α, and Grid Granularity.
- **CMC Count**: It represents the number of core-mini-clusters generated by the on-line phase of MuDi-Stream.
- t_{online}: The time elapsed (seconds) from the execution's commencement for each execution ID until this data point arrived.
- **CMCs in each cluster**: The dataset contains three classes. For each execution we identified the number of core-mini-clusters which are put in each cluster.

ID	Point Number	λ	*minPts*	α	Grid Granularity	CMC Count	t_{online}	CMCs in each cluster
1	5000	0.0625	4	0.6	4	28	0.514123821	11-9-8
	10000	0.0625	4	0.6	4	31	0.644977329	11-12-8
2	5000	0.0625	4	0.6	8	57	0.161896245	30-20-7
	10000	0.0625	4	0.6	8	68	0.262233437	36-24-8
3	5000	0.0625	4	0.5	4	31	0.086273589	11-12-8
	10000	0.0625	4	0.5	4	36	0.171721328	14-14-8
4	5000	0.0625	4	0.5	8	64	0.088338516	35-21-8
	10000	0.0625	4	0.5	8	72	0.177076073	39-25-8
5	5000	0.0625	4	0.4	8	74	0.086969645	28-38-8
	10000	0.0625	4	0.4	8	88	0.17596015	31-47-10
6	5000	0.0625	4	0.2	4	55	0.085826253	23-20-12
	10000	0.0625	4	0.2	4	56	0.170563748	23-21-12
7	5000	0.0625	5	0.6	4	28	0.090228373	11-9-8
	10000	0.0625	5	0.6	4	31	0.179543361	11-12-8
8	5000	0.0625	5	0.6	8	57	0.086583283	30-20-7
	10000	0.0625	5	0.6	8	68	0.174242344	36-24-8
9	5000	0.0625	5	0.5	8	64	0.0860792	35-21-8
	10000	0.0625	5	0.5	8	72	0.169994769	39-25-8
10	5000	0.0625	5	0.4	8	74	0.086496956	28-38-8
	10000	0.0625	5	0.4	8	88	0.175808622	31-47-10
11	5000	0.0625	5	0.3	8	84	0.086029698	29-45-10
	10000	0.0625	5	0.3	8	103	0.170605403	36-53-14
12	5000	0.0625	5	0.2	8	115	0.086670817	38-61-16
	10000	0.0625	5	0.2	8	131	0.172985759	39-66-26
13	5000	0.0625	5	0.2	12	212	0.08885256	67-112-33
	10000	0.0625	5	0.2	12	256	0.178885035	77-130-49
14	5000	0.0625	5	0.1	4	83	0.084934902	28-35-20
	10000	0.0625	5	0.1	4	92	0.169996278	31-39-22

15	5000	0.0625	5	0.1	8	213	0.086874564	110-69-34
	10000	0.0625	5	0.1	8	239	0.174666438	117-82-40
16	5000	0.0625	6	0.6	8	57	0.086084634	30-20-7
	10000	0.0625	6	0.6	8	68	0.171038554	36-24-8
17	5000	0.0625	6	0.5	8	64	0.085819915	35-21-8
	10000	0.0625	6	0.5	8	72	0.169890934	39-25-8
18	5000	0.0625	6	0.4	8	74	0.085379823	28-38-8
	10000	0.0625	6	0.4	8	88	0.173411362	31-47-10
19	5000	0.0625	6	0.3	8	84	0.08520928	29-45-10
	10000	0.0625	6	0.3	8	103	0.17202589	36-53-14
20	5000	0.0625	6	0.2	8	115	0.086599884	38-61-16
	10000	0.0625	6	0.2	8	131	0.176605797	39-66-26
21	5000	0.0625	6	0.2	12	212	0.088005882	67-112-33
	10000	0.0625	6	0.2	12	256	0.178103254	77-130-49
22	5000	0.0625	6	0.1	4	83	0.085681065	28-35-20
	10000	0.0625	6	0.1	4	92	0.169976054	31-39-22
23	5000	0.0625	6	0.1	8	213	0.090166797	110-69-34
	10000	0.0625	6	0.1	8	239	0.177025666	117-82-40
24	5000	0.0625	6	0.1	12	393	0.09482126	110-222-61
	10000	0.0625	6	0.1	12	448	0.190156867	124-243-81
25	5000	0.0625	6	0.1	16	510	0.098255964	276-175-59
	10000	0.0625	6	0.1	16	638	0.20819608	328-216-94
26	5000	0.0625	7	0.5	8	64	0.091571588	35-21-8
	10000	0.0625	7	0.5	8	72	0.176798376	39-25-8
27	5000	0.0625	7	0.4	8	74	0.088172501	28-38-8
	10000	0.0625	7	0.4	8	88	0.173376349	31-47-10
28	5000	0.0625	7	0.3	8	84	0.084173044	29-45-10
	10000	0.0625	7	0.3	8	103	0.16997696	36-53-14
29	5000	0.0625	7	0.2	8	115	0.086550685	38-61-16
	10000	0.0625	7	0.2	8	131	0.173306623	39-66-26
30	5000	0.0625	7	0.2	12	212	0.088254906	67-112-33
	10000	0.0625	7	0.2	12	256	0.177710553	77-130-49
31	5000	0.0625	7	0.1	8	213	0.087976906	110-69-34
	10000	0.0625	7	0.1	8	239	0.176155746	117-82-40
32	5000	0.0625	7	0.1	12	393	0.09548019	110-222-61
	10000	0.0625	7	0.1	12	448	0.19158158	124-243-81
33	5000	0.0625	7	0.1	16	510	0.097541495	276-175-59
	10000	0.0625	7	0.1	16	638	0.196910366	328-216-94
34	5000	0.125	4	0.5	12	176	0.086638822	55-99-22
	10000	0.125	4	0.5	12	219	0.176162084	64-115-40
35	5000	0.125	4	0.4	4	54	0.084928263	23-19-12
	10000	0.125	4	0.4	4	54	0.168181581	23-19-12
36	5000	0.125	4	0.4	8	122	0.086144097	39-65-18
	10000	0.125	4	0.4	8	134	0.171552293	41-71-22
37	5000	0.125	4	0.4	12	221	0.088466498	72-118-31
	10000	0.125	4	0.4	12	252	0.1792717	83-128-41
38	5000	0.125	4	0.3	4	68	0.082977735	24-29-15
	10000	0.125	4	0.3	4	74	0.167900563	24-32-18
39	5000	0.125	5	0.6	8	81	0.085724531	28-45-8
	10000	0.125	5	0.6	8	95	0.170517566	33-51-11
40	5000	0.125	5	0.6	12	147	0.087834433	48-89-10
	10000	0.125	5	0.6	12	175	0.175407168	51-95-29
41	5000	0.125	5	0.5	4	43	0.082415093	14-18-11
	10000	0.125	5	0.5	4	47	0.167004081	16-20-11
42	5000	0.125	5	0.4	8	122	0.087056578	39-65-18
	10000	0.125	5	0.4	8	134	0.172259822	41-71-22
43	5000	0.125	5	0.4	12	221	0.08904363	72-118-31
	10000	0.125	5	0.4	12	252	0.178731699	83-128-41
44	5000	0.125	5	0.3	4	68	0.084240355	24-29-15
	10000	0.125	5	0.3	4	74	0.1675471	24-32-18
45	5000	0.125	5	0.3	8	147	0.086107876	80-44-23
	10000	0.125	5	0.3	8	162	0.171980312	89-48-25
46	5000	0.125	5	0.2	4	85	0.084114183	28-35-22
	10000	0.125	5	0.2	4	91	0.167988701	30-38-23
47	5000	0.125	5	0.2	8	219	0.086269967	111-73-35
	10000	0.125	5	0.2	8	243	0.174279471	118-83-42
48	5000	0.125	5	0.2	12	407	0.092426113	113-228-66
	10000	0.125	5	0.2	12	459	0.189531441	129-250-80
49	5000	0.125	5	0.1	4	215	0.08728266	84-82-49
	10000	0.125	5	0.1	4	228	0.174775103	90-85-53
50	5000	0.125	5	0.1	8	472	0.094939583	240-164-68

	10000	0.125	5	0.1	8	514	0.194200393	255-177-82
51	5000	0.125	6	0.6	8	81	0.087633102	28-45-8
	10000	0.125	6	0.6	8	95	0.172604829	33-51-11
52	5000	0.125	6	0.5	4	43	0.083342062	14-18-11
	10000	0.125	6	0.5	4	47	0.167383501	16-20-11
53	5000	0.125	6	0.5	8	89	0.086207786	47-31-11
	10000	0.125	6	0.5	8	99	0.17048889	52-35-12
54	5000	0.125	6	0.4	8	122	0.086366557	39-65-18
	10000	0.125	6	0.4	8	134	0.172012005	41-71-22
55	5000	0.125	6	0.4	12	221	0.090925034	72-118-31
	10000	0.125	6	0.4	12	252	0.179368593	83-128-41
56	5000	0.125	6	0.3	4	68	0.083804792	24-29-15
	10000	0.125	6	0.3	4	74	0.166878814	24-32-18
57	5000	0.125	6	0.3	8	147	0.08614953	80-44-23
	10000	0.125	6	0.3	8	162	0.173624465	89-48-25
58	5000	0.125	6	0.2	4	85	0.084294084	28-35-22
	10000	0.125	6	0.2	4	91	0.170782587	30-38-23
59	5000	0.125	6	0.2	8	219	0.086477335	111-73-35
	10000	0.125	6	0.2	8	243	0.174639272	118-83-42
60	5000	0.125	6	0.2	12	407	0.092587299	113-228-66
	10000	0.125	6	0.2	12	459	0.188793731	129-250-80
61	5000	0.125	6	0.2	16	531	0.097675816	290-181-60
	10000	0.125	6	0.2	16	637	0.202092153	342-214-81
62	5000	0.125	6	0.1	4	215	0.087063822	84-82-49
	10000	0.125	6	0.1	4	228	0.173368802	90-85-53
63	5000	0.125	6	0.1	8	472	0.09466913	240-164-68
	10000	0.125	6	0.1	8	514	0.192974899	255-177-82
64	5000	0.125	6	0.1	16	1037	0.12316338	583-340-114
	10000	0.125	6	0.1	16	1277	0.239770367	700-395-182
65	5000	0.125	7	0.6	8	81	0.085758338	28-45-8
	10000	0.125	7	0.6	8	95	0.170241075	33-51-11
66	5000	0.125	7	0.5	4	43	0.084842538	14-18-11
	10000	0.125	7	0.5	4	47	0.167008004	16-20-11
67	5000	0.125	7	0.5	8	89	0.086061996	47-31-11
	10000	0.125	7	0.5	8	99	0.169859844	52-35-12
68	5000	0.125	7	0.4	8	122	0.08596148	39-65-18
	10000	0.125	7	0.4	8	134	0.171417671	41-71-22
69	5000	0.125	7	0.4	12	221	0.0889736	72-118-31
	10000	0.125	7	0.4	12	252	0.177928184	83-128-41
70	5000	0.125	7	0.3	4	68	0.083795736	24-29-15
	10000	0.125	7	0.3	4	74	0.166523542	24-32-18
71	5000	0.125	7	0.3	8	147	0.086989869	80-44-23
	10000	0.125	7	0.3	8	162	0.173320808	89-48-25
72	5000	0.125	7	0.3	12	272	0.090517542	86-142-44
	10000	0.125	7	0.3	12	320	0.180516816	95-163-62
73	5000	0.125	7	0.2	8	219	0.08579154	111-73-35
	10000	0.125	7	0.2	8	243	0.174158732	118-83-42
74	5000	0.125	7	0.2	12	407	0.092998715	113-228-66
	10000	0.125	7	0.2	12	459	0.189975457	129-250-80
75	5000	0.125	7	0.2	16	531	0.098496233	290-181-60
	10000	0.125	7	0.2	16	637	0.20413082	342-214-81
76	5000	0.125	7	0.1	4	215	0.086932519	84-82-49
	10000	0.125	7	0.1	4	228	0.173107706	90-85-53
77	5000	0.125	7	0.1	8	472	0.094368189	240-164-68
	10000	0.125	7	0.1	8	514	0.193562593	255-177-82
78	5000	0.125	7	0.1	12	728	0.107345752	411-206-111
	10000	0.125	7	0.1	12	840	0.213078137	470-220-150
79	5000	0.125	7	0.1	16	1037	0.124592319	583-340-114
	10000	0.125	7	0.1	16	1277	0.242503884	700-395-182
80	5000	0.125	7	0.1	20	1129	0.127442348	633-383-113
	10000	0.125	7	0.1	20	1473	0.256610952	809-474-190
81	5000	0.25	4	0.6	4	66	0.083639984	25-27-14
	10000	0.25	4	0.6	4	68	0.167997756	24-29-15
82	5000	0.25	4	0.5	4	70	0.083218909	24-30-16
	10000	0.25	4	0.5	4	73	0.166723666	25-31-17
83	5000	0.25	4	0.4	8	218	0.08837655	113-70-35
	10000	0.25	4	0.4	8	235	0.175340157	118-76-41
84	5000	0.25	5	0.6	4	66	0.083231284	25-27-14
	10000	0.25	5	0.6	4	68	0.166730005	24-29-15
85	5000	0.25	5	0.6	8	142	0.088199064	78-43-21
	10000	0.25	5	0.6	8	152	0.175111359	85-43-24
86	5000	0.25	5	0.5	4	70	0.082618838	24-30-16

	10000	0.25	5	0.5	4	73	0.167273025	25-31-17
87	5000	0.25	5	0.5	8	151	0.08530889	83-45-23
	10000	0.25	5	0.5	8	164	0.173682723	89-47-28
88	5000	0.25	5	0.5	12	281	0.08975206	93-145-43
	10000	0.25	5	0.5	12	324	0.179950853	105-164-55
89	5000	0.25	5	0.4	8	218	0.087789157	113-70-35
	10000	0.25	5	0.4	8	235	0.174583732	118-76-41
90	5000	0.25	5	0.2	4	229	0.087224706	84-93-52
	10000	0.25	5	0.2	4	253	0.172335281	90-107-56
91	5000	0.25	5	0.1	4	229	0.085602888	84-93-52
	10000	0.25	5	0.1	4	253	0.170678752	90-107-56
92	5000	0.25	6	0.6	4	66	0.084344491	25-27-14
	10000	0.25	6	0.6	4	68	0.16706626	24-29-15
93	5000	0.25	6	0.6	8	142	0.08727783	78-43-21
	10000	0.25	6	0.6	8	152	0.172602415	85-43-24
94	5000	0.25	6	0.6	12	264	0.088137185	87-141-36
	10000	0.25	6	0.6	12	287	0.181272636	88-153-46
95	5000	0.25	6	0.5	4	70	0.084939128	24-30-16
	10000	0.25	6	0.5	4	73	0.169548943	25-31-17
96	5000	0.25	6	0.5	8	151	0.085908657	83-45-23
	10000	0.25	6	0.5	8	164	0.173470525	89-47-28
97	5000	0.25	6	0.5	12	281	0.089629814	93-145-43
	10000	0.25	6	0.5	12	324	0.181961147	105-164-55
98	5000	0.25	6	0.4	4	86	0.085087938	27-38-21
	10000	0.25	6	0.4	4	90	0.17021723	28-39-23
99	5000	0.25	6	0.4	8	218	0.086478241	113-70-35
	10000	0.25	6	0.4	8	235	0.176239055	118-76-41
100	5000	0.25	6	0.4	12	409	0.112061792	115-232-62
	10000	0.25	6	0.4	12	447	0.221507786	128-242-77
101	5000	0.25	6	0.4	16	540	0.10134898	304-192-44
	10000	0.25	6	0.4	16	622	0.208691108	336-213-73
102	5000	0.25	6	0.3	4	217	0.087731504	84-88-45
	10000	0.25	6	0.3	4	222	0.17461452	85-90-47
103	5000	0.25	6	0.3	8	467	0.097380309	247-164-56
	10000	0.25	6	0.3	8	483	0.194825214	251-171-61
104	5000	0.25	6	0.2	4	229	0.087470408	84-93-52
	10000	0.25	6	0.2	4	253	0.174577092	90-107-56
105	5000	0.25	6	0.2	16	1115	0.124036017	632-363-120
	10000	0.25	6	0.2	16	1382	0.248676029	759-433-190
106	5000	0.25	6	0.1	4	229	0.086065618	84-93-52
	10000	0.25	6	0.1	4	253	0.173092916	90-107-56
107	5000	0.25	7	0.6	4	66	0.083749855	25-27-14
	10000	0.25	7	0.6	4	68	0.16794765	24-29-15
108	5000	0.25	7	0.6	8	142	0.088547092	78-43-21
	10000	0.25	7	0.6	8	152	0.175293673	85-43-24
109	5000	0.25	7	0.6	12	264	0.091464735	87-141-36
	10000	0.25	7	0.6	12	287	0.186701938	88-153-46
110	5000	0.25	7	0.5	4	70	0.083624892	24-30-16
	10000	0.25	7	0.5	4	73	0.168619258	25-31-17
111	5000	0.25	7	0.5	8	151	0.086878185	83-45-23
	10000	0.25	7	0.5	8	164	0.175105924	89-47-28
112	5000	0.25	7	0.5	12	281	0.089410672	93-145-43
	10000	0.25	7	0.5	12	324	0.181451328	105-164-55
113	5000	0.25	7	0.4	4	86	0.08740702	27-38-21
	10000	0.25	7	0.4	4	90	0.173518216	28-39-23
114	5000	0.25	7	0.4	8	218	0.090637978	113-70-35
	10000	0.25	7	0.4	8	235	0.181227661	118-76-41
115	5000	0.25	7	0.4	12	409	0.096043434	115-232-62
	10000	0.25	7	0.4	12	447	0.192570728	128-242-77
116	5000	0.25	7	0.3	4	217	0.088641872	84-88-45
	10000	0.25	7	0.3	4	222	0.174535437	85-90-47
117	5000	0.25	7	0.3	8	467	0.095573763	247-164-56
	10000	0.25	7	0.3	8	483	0.191728879	251-171-61
118	5000	0.25	7	0.2	4	229	0.086894185	84-93-52
	10000	0.25	7	0.2	4	253	0.173983663	90-107-56
119	5000	0.25	7	0.2	8	486	0.096517634	248-171-67
	10000	0.25	7	0.2	8	543	0.192880119	271-195-77
120	5000	0.25	7	0.2	16	1115	0.122211058	632-363-120
	10000	0.25	7	0.2	16	1382	0.245486727	759-433-190
121	5000	0.25	7	0.2	20	1207	0.127275427	678-413-116
	10000	0.25	7	0.2	20	1606	0.259383407	890-520-196
122	5000	0.25	7	0.1	4	229	0.086159491	84-93-52

	10000	0.25	7	0.1	4	253	0.172653729	90-107-56
123	5000	0.25	7	0.1	8	486	0.095688162	248-171-67
	10000	0.25	7	0.1	8	543	0.191626554	271-195-77
124	5000	0.25	7	0.1	20	1207	0.128119388	678-413-116
	10000	0.25	7	0.1	20	1621	0.259699138	898-523-200
125	5000	0.5	5	0.6	8	234	0.087608653	119-76-39
	10000	0.5	5	0.6	8	237	0.175264998	120-75-42
126	5000	0.5	5	0.6	12	430	0.094359436	123-240-67
	10000	0.5	5	0.6	12	439	0.190697473	130-235-74
127	5000	0.5	5	0.4	4	241	0.087241609	89-101-51
	10000	0.5	5	0.4	4	249	0.172795295	92-102-55
128	5000	0.5	5	0.3	4	243	0.086949119	89-103-51
	10000	0.5	5	0.3	4	254	0.172541744	93-105-56
129	5000	0.5	5	0.2	4	243	0.101063735	89-103-51
	10000	0.5	5	0.2	4	254	0.209261296	93-105-56
130	5000	0.5	5	0.1	4	243	0.085349337	89-103-51
	10000	0.5	5	0.1	4	254	0.170963091	93-105-56
131	5000	0.5	6	0.6	4	90	0.08415614	28-39-23
	10000	0.5	6	0.6	4	88	0.168122419	28-38-22
132	5000	0.5	6	0.6	8	234	0.087706753	119-76-39
	10000	0.5	6	0.6	8	237	0.176028668	120-75-42
133	5000	0.5	6	0.6	12	430	0.094053364	123-240-67
	10000	0.5	6	0.6	12	439	0.191415865	130-235-74
134	5000	0.5	6	0.6	16	589	0.10089108	335-211-43
	10000	0.5	6	0.6	16	654	0.212018659	365-222-67
135	5000	0.5	6	0.5	4	218	0.086253666	83-91-44
	10000	0.5	6	0.5	4	216	0.172477752	82-91-43
136	5000	0.5	6	0.5	8	482	0.095639262	255-168-59
	10000	0.5	6	0.5	8	472	0.192091999	245-167-60
137	5000	0.5	6	0.4	4	241	0.086459525	89-101-51
	10000	0.5	6	0.4	4	249	0.171954049	92-102-55
138	5000	0.5	6	0.3	4	243	0.086498464	89-103-51
	10000	0.5	6	0.3	4	254	0.172923881	93-105-56
139	5000	0.5	6	0.2	4	243	0.092021338	89-103-51
	10000	0.5	6	0.2	4	254	0.180364383	93-105-56
140	5000	0.5	6	0.1	4	243	0.087741465	89-103-51
	10000	0.5	6	0.1	4	254	0.174748539	93-105-56
141	5000	0.5	7	0.6	4	90	0.095052173	28-39-23
	10000	0.5	7	0.6	4	88	0.192676071	28-38-22
142	5000	0.5	7	0.6	8	234	0.098959266	119-76-39
	10000	0.5	7	0.6	8	237	0.187123315	120-75-42
143	5000	0.5	7	0.6	12	430	0.098619689	123-240-67
	10000	0.5	7	0.6	12	439	0.205991398	130-235-74
144	5000	0.5	7	0.6	16	589	0.136769688	335-211-43
	10000	0.5	7	0.6	16	654	0.274735591	365-222-67
145	5000	0.5	7	0.5	4	218	0.08835059	83-91-44
	10000	0.5	7	0.5	4	216	0.188833272	82-91-43
146	5000	0.5	7	0.5	8	482	0.097317828	255-168-59
	10000	0.5	7	0.5	8	472	0.209919019	245-167-60
147	5000	0.5	7	0.5	12	734	0.109291148	415-219-100
	10000	0.5	7	0.5	12	786	0.214625098	437-224-125
148	5000	0.5	7	0.4	4	241	0.086756542	89-101-51
	10000	0.5	7	0.4	4	249	0.193272519	92-102-55
149	5000	0.5	7	0.4	8	530	0.098163901	278-177-75
	10000	0.5	7	0.4	8	591	0.19647631	300-201-90
150	5000	0.5	7	0.4	12	840	0.124773427	468-246-126
	10000	0.5	7	0.4	12	1008	0.252059117	541-288-179
151	5000	0.5	7	0.4	16	1204	0.160883854	695-394-115
	10000	0.5	7	0.4	16	1542	0.324570043	866-489-187
152	5000	0.5	7	0.3	4	243	0.086897807	89-103-51
	10000	0.5	7	0.3	4	254	0.173437925	93-105-56
153	5000	0.5	7	0.3	8	538	0.098195293	283-179-76
	10000	0.5	7	0.3	8	611	0.204147422	310-206-95
154	5000	0.5	7	0.3	12	856	0.119006662	475-252-129
	10000	0.5	7	0.3	12	1047	0.228941044	558-300-189
155	5000	0.5	7	0.3	20	1395	0.136331105	786-480-129
	10000	0.5	7	0.3	20	1975	0.284011013	1078-656-241
156	5000	0.5	7	0.2	4	243	0.087295337	89-103-51
	10000	0.5	7	0.2	4	254	0.173751844	93-105-56
157	5000	0.5	7	0.2	8	538	0.096786277	283-179-76
	10000	0.5	7	0.2	8	611	0.195504369	310-206-95
158	5000	0.5	7	0.2	12	856	0.112944087	475-252-129

	10000	0.5	7	0.2	12	1047	0.225770152	558-300-189
159	5000	0.5	7	0.2	20	1395	0.134417705	786-480-129
	10000	0.5	7	0.2	20	1975	0.280273558	1078-656-241
160	5000	0.5	7	0.1	4	243	0.088326142	89-103-51
	10000	0.5	7	0.1	4	254	0.176526414	93-105-56
161	5000	0.5	7	0.1	8	538	0.097454262	283-179-76
	10000	0.5	7	0.1	8	611	0.195054619	310-206-95
162	5000	0.5	7	0.1	12	856	0.109099777	475-252-129
	10000	0.5	7	0.1	12	1047	0.219154897	558-300-189
163	5000	0.5	7	0.1	20	1395	0.135179865	786-480-129
	10000	0.5	7	0.1	20	1975	0.279595007	1078-656-241
164	5000	1	6	0.6	4	265	0.086178809	100-113-52
	10000	1	6	0.6	4	291	0.173166867	114-122-55
165	5000	1	6	0.5	4	268	0.087887258	101-115-52
	10000	1	6	0.5	4	323	0.175854504	123-137-63
166	5000	1	6	0.5	12	1005	0.114629896	551-305-149
	10000	1	6	0.5	12	1411	0.234505572	750-413-248
167	5000	1	6	0.4	4	268	0.086787632	101-115-52
	10000	1	6	0.4	4	323	0.173510972	123-137-63
168	5000	1	6	0.4	12	1005	0.113720736	551-305-149
	10000	1	6	0.4	12	1411	0.230108281	750-413-248
169	5000	1	6	0.3	4	268	0.086328524	101-115-52
	10000	1	6	0.3	4	323	0.173017152	123-137-63
170	5000	1	6	0.3	12	1005	0.112880095	551-305-149
	10000	1	6	0.3	12	1411	0.228113079	750-413-248
171	5000	1	6	0.2	4	268	0.085847081	101-115-52
	10000	1	6	0.2	4	323	0.173421324	123-137-63
172	5000	1	6	0.2	12	1005	0.115412582	551-305-149
	10000	1	6	0.2	12	1411	0.251255301	750-413-248
173	5000	1	6	0.1	4	268	0.086490314	101-115-52
	10000	1	6	0.1	4	323	0.173574661	123-137-63
174	5000	1	6	0.1	12	1005	0.114358234	551-305-149
	10000	1	6	0.1	12	1411	0.231115842	750-413-248
175	5000	1	7	0.6	4	265	0.086623729	100-113-52
	10000	1	7	0.6	4	291	0.175897969	114-122-55
176	5000	1	7	0.6	12	991	0.113860793	547-302-142
	10000	1	7	0.6	12	1261	0.23887449	673-375-213
177	5000	1	7	0.5	4	268	0.087077405	101-115-52
	10000	1	7	0.5	4	323	0.174831246	123-137-63
178	5000	1	7	0.5	8	617	0.102895638	314-209-94
	10000	1	7	0.5	8	792	0.210363938	385-267-140
179	5000	1	7	0.5	12	1005	0.113977306	551-305-149
	10000	1	7	0.5	12	1411	0.235019013	750-413-248
180	5000	1	7	0.4	4	268	0.086415154	101-115-52
	10000	1	7	0.4	4	323	0.172513672	123-137-63
181	5000	1	7	0.4	8	617	0.097513424	314-209-94
	10000	1	7	0.4	8	792	0.203306479	385-267-140
182	5000	1	7	0.4	12	1005	0.112150837	551-305-149
	10000	1	7	0.4	12	1411	0.229029787	750-413-248
183	5000	1	7	0.3	4	268	0.08593703	101-115-52
	10000	1	7	0.3	4	323	0.172867436	123-137-63
184	5000	1	7	0.3	8	617	0.097127362	314-209-94
	10000	1	7	0.3	8	792	0.201568451	385-267-140
185	5000	1	7	0.3	12	1005	0.12538225	551-305-149
	10000	1	7	0.3	12	1411	0.259726606	750-413-248
186	5000	1	7	0.2	4	268	0.093274602	101-115-52
	10000	1	7	0.2	4	323	0.180393661	123-137-63
187	5000	1	7	0.2	8	617	0.098552075	314-209-94
	10000	1	7	0.2	8	792	0.207476782	385-267-140
188	5000	1	7	0.2	12	1005	0.114306921	551-305-149
	10000	1	7	0.2	12	1411	0.232183169	750-413-248
189	5000	1	7	0.1	4	268	0.087854053	101-115-52
	10000	1	7	0.1	4	323	0.175426787	123-137-63
190	5000	1	7	0.1	8	617	0.098309995	314-209-94
	10000	1	7	0.1	8	792	0.200909825	385-267-140
191	5000	1	7	0.1	12	1005	0.112740342	551-305-149
	10000	1	7	0.1	12	1411	0.230741855	750-413-248

Table D.3: The details of an execution of MuDi-Stream on MDS4 dataset

APPENDIX E

JAVA CODE

We have implemented our proposed method, MuDi-Stream, in Java. The method and all its required concepts are implemented in different inter-related Java classes.

For the sake of clarity and brevity only major classes are presented here. The major classes are as follows.

- *MuDi*
- *CoreMiniCluster*
- *MDBSCAN*
- *Grid*
- *GridCharacteristicVector*
- *GridIndex*

The Java classes are explained in the following sections.

E.1 MuDi.java

The main class is called MuDi and implemented in *MuDi.java* which its code listing follows in this section. The *MuDi* class receives the data stream which is implemented as an interface called *DataGeneratorInterface.* The *DataGeneratorInterface* has the abilities to check if it still has data and provide the data points based on different requests. Other than that, *MuDi* class receives MuDi-Stream's parameters, i.e. GridGranularity, λ, and α.

When the data stream flow starts, *MuDi* will generate a list of core-mini-clusters which is implemented as *CoreMiniCluster* class (cf. Section E.2). Moreover, when a clustering request arrives this list will be used to generate final clusters using MDBScan. MDBScan is implemented in *MDBSCAN* class which is presented in Section E.3. MuDi-Stream keeps the list of outliers in a grid structure. The grid structure is implemented in *Grid* class. The *Grid* class (cf. Section E.4) uses a tree structure to dynamically store only the required grid cells which already received some data points. The indices of each grid cell are stored in an instance of *GridIndex* class and its characteristics identified as "Grid Characteristic Vector" in an instance of *GridCharacteristicVector* class. These two classes are presented in Sections E.6 and E.5 respectively.

The source code for *MuDi* class is presented as follows:

```
1  package MuDi;

3  import Common.*;
   import Data.ClusterList;
5  import Data.DataGeneratorInterface;
   import Data.Point.DataPoint;
7  import Data.R_PointArray;
   import MuDi.Grid.Grid;
9  import MuDi.Grid.GridIndex;

11 import java.util.ArrayList;
   import java.util.Iterator;
13 import java.util.TreeMap;

15 /**
    * User: Amineh Amini
17  */
   public class MuDi implements AlgorithmsInterface {
19   private int MinPts_forOffline;
     private double lambda;
21   private double alpha, alphaCounter_fromOutside;
     private int gridGranularity;
23   private double pruningTime;
     private int pruningTimeInteger;
25
     private int horizon;
27
     private double N, oneP2Lambda;
29
     private DataGeneratorInterface data;
31
     private ArrayList<CoreMiniCluster> coreMiniClusterList;
33   private Grid grid;
```

```
35    private double Weight_of_UpgradeToMiniCluster;

37    private ArrayList<Integer> offlineClustering_PointNumber_List;

39    private TreeMap<Integer, MDBSCAN> offlineResultList;

41    private R_PointArray rPT;

43    public static String datasetName;
      public static String executionName;
45
      public static String rCode_plots_static;
47    private String rCode_plots;

49    private StringBuilder analysisResultCSV;
      private StringBuilder analysisResultTable;
51
      public TimeIntervalPausible executionOnlineTimeInterval,
         pruningTimeInterval;
53
      private double sumCumulative, countCumulative;
55
      public static String outputR_path;
57
      public MuDi(DataGeneratorInterface data,
59                ArrayList<Integer> offlineClustering_PointNumber_List
                     ,
                  double lambda,
61                int gridGranularity,
                  double alpha, double alphaCounter_fromOutside,
63                int minPts_forOffline,
                  int horizon
65    ) {
        this.data = data;
67      this.offlineClustering_PointNumber_List =
           offlineClustering_PointNumber_List;
        this.lambda = lambda;
69      this.gridGranularity = gridGranularity;
        this.alpha = alpha;
71      this.alphaCounter_fromOutside = alphaCounter_fromOutside;

73      this.horizon = horizon;

75      N = Math.pow(gridGranularity, data.getDimensionCount());
        oneP2Lambda = 1 - Math.pow(2, -lambda);
77
        Weight_of_UpgradeToMiniCluster = alpha / (N * oneP2Lambda);
79      System.out.printf("\nWeight_of_UpgradeToMiniCluster=%2.5f\n",
           Weight_of_UpgradeToMiniCluster);
        setPruningTime(1000.0 *
81              (1.0 / lambda) * GeneralFunctions.log2(alpha / (alpha -
                    N * oneP2Lambda)));
        System.out.printf("Pruning time: %1.3f\n", this.pruningTime);
83
        this.MinPts_forOffline = minPts_forOffline;
85
        offlineResultList = new TreeMap<Integer, MDBSCAN>();
87      rPT = new R_PointArray(2, datasetName, "");
        rCode_plots = MuDi.rCode_plots_static.replace("{DatasetName}",
           datasetName);
89      analysisResultCSV = new StringBuilder();
```

```
    analysisResultTable = new StringBuilder();

    executionOnlineTimeInterval = new TimeIntervalPausible(
        TimeIntervalPausible.PAUSED);
    pruningTimeInterval = new TimeIntervalPausible(
        TimeIntervalPausible.PAUSED);

    sumCumulative = 0;
    countCumulative = 0;
  }

  public void setPruningTime(double pruningTime) {
    this.pruningTime = pruningTime;
    this.pruningTimeInteger = (int) pruningTime;
  }

  public void start(int dataRecordLimit) {
    generateMicroClusters(dataRecordLimit);
  }

  private void generateMicroClusters(int dataRecordLimit) {
    CONSTANTS.executionID++;

    coreMiniClusterList = new ArrayList<CoreMiniCluster>();

    grid = new Grid(data.getDimensionCount(), gridGranularity);

    int pointsCount = 0;
    long lastTS = 0;

    data.setDataRecordLimit(dataRecordLimit);

    executionOnlineTimeInterval.Resume();
    while (data.hasData()) {
      DataPoint dp = data.nextDataPoint();
      rPT.addPoint(dp);

      pointsCount++;

      CoreMiniCluster mc = searchInCoreMiniClusterList(dp);
      if (mc != null) {
        mc.addDataPoint(dp);
      } else {
        GridIndex gi = grid.addPoint(dp);
        CheckAndUpgradeGridCell(gi, dp.getTimestamp());
      }

      if ((pruningTimeInteger > 0) && (pointsCount %
          pruningTimeInteger == 0))
        PruneCoreMiniClusterList(dp.getTimestamp());

      lastTS = dp.getTimestamp();

      if (offlineClustering_PointNumber_List.contains(pointsCount))
          {
        executionOnlineTimeInterval.Pause();
        doOfflineClustering(pointsCount, GeneralFunctions.
            getTimestampArray(lastTS, horizon));
        executionOnlineTimeInterval.Resume();
      }

```

```
        }
147     executionOnlineTimeInterval.Pause();
        if (offlineClustering_PointNumber_List.contains(CONSTANTS.
            END_FILE_CLUSTERING_POINT_NUMBER)) {
149       doOfflineClustering(pointsCount, GeneralFunctions.
              getTimestampArray(lastTS, horizon));
        }
151   }

153   public String getAnalysisResultTable() {
        return analysisResultTable.toString();
155   }

157   private void doOfflineClustering(int pointNumber, ArrayList<Long>
          timestampArray) {
      MDBSCAN mDBSCAN = null;
159     ArrayList<CoreMiniCluster> CMC_List = getMicroClusters(
          timestampArray);
        int CMC_List_size = CMC_List.size();
161     countCumulative++;
        if (CMC_List_size > 1) {
163       mDBSCAN = new MDBSCAN(CMC_List, MinPts_forOffline,
              gridGranularity);
          mDBSCAN.doClustering();
165
          if (data.getDimensionCount() <= 3) {
167         Logger l_execParam = new Logger();
            l_execParam.startFile(String.format(outputR_path + "/MuDi_%
                s_%d_p%d__L%s_M%d_Al%s_G%s.r",
169                   executionName,
                      CONSTANTS.executionID,
171                   pointNumber,
                      GeneralFunctions.numberLeftPadding((int) (lambda *
                          10000), 4),
173                   MinPts_forOffline,
                      GeneralFunctions.numberLeftPadding((int) (
                          alphaCounter_fromOutside * 10), 2),
175                   GeneralFunctions.numberLeftPadding(gridGranularity,
                          2)));

177         R_PointArray rCMC = new R_PointArray(data.getDimensionCount
                (), "mc",
                      rCode_plots.replace("{gGran}", "" + gridGranularity
                          ));
179
            String shoa = String.format("shoa<-array(,dim=c(%d,1));\n",
                CMC_List_size);
181         String vazn = String.format("vazn<-array(,dim=c(%d,1));\n",
                CMC_List_size);
            String clusNum = String.format("clusNum<-array(,dim=c(%d,1)
                );\n", CMC_List_size);
183         String clusColor = String.format("clusCol<-array(,dim=c(%d
                ,1));\n", CMC_List_size);

185         for (int zx = 0; zx < CMC_List_size; zx++) {
              rCMC.addPoint(CMC_List.get(zx));
187           vazn = vazn + "vazn[" + (zx + 1) + "]<-" + CMC_List.get(
                  zx).getWeight() + ";\n";
              shoa = shoa + "shoa[" + (zx + 1) + "]<-" + CMC_List.get(
                  zx).getMCD() + ";\n";
189           int clusterID_MC = CMC_List.get(zx).
```

```
                getCalculatedClusterID();
            clusNum = clusNum + "clusNum[" + (zx + 1) + "]<-\"" + (
                clusterID_MC != ClusterList.NOISE ? clusterID_MC : "N"
                ) + "\";\n";
            clusColor = clusColor + "clusCol[" + (zx + 1) + "]<-\"" +
                CONSTANTS.getClusterColor(clusterID_MC) + "\";\n";
        }

        l_execParam.logFile(rPT.toString());
        l_execParam.logFile(vazn);
        l_execParam.logFile(shoa);
        l_execParam.logFile(clusNum);
        l_execParam.logFile(clusColor);
        l_execParam.logFile(rCMC.toString());
        l_execParam.endFile();
    }

    AnalyzeResultClusters arc = new AnalyzeResultClusters(mDBSCAN
        .getSCC_List());
    arc.printResultClusters(/*printCircleCommand=*/false);
    arc.dumpResults(/*useBestActual=*/true);
    SimplifiedContingency aSC = arc.
        calculateSimplifiedContingency();

    Statistics sts = new Statistics(arc.getClusterPointCountList(
        false));

    System.out.println();

    double avg =
            AnalyzeResultClusters.getAverage9(
                    aSC.getAverage7(),
                    arc.getNMI_bestActual(),
                    arc.getPurityBestActual(1.0));
    sumCumulative += avg;
    String s = String.format(CONSTANTS.analysisResultCSVFormat,
            "MuDi", CONSTANTS.executionID, pointNumber, horizon,
            lambda, MinPts_forOffline,
            alpha, alphaCounter_fromOutside, gridGranularity,
            0.0, 0.0, 0.0, 0.0,
            pruningTimeInteger,
            CMC_List_size, arc.getNoiseCount(),
            arc.getActualClassCount(false, false), arc.
                getCalculatedClusterCount(false),
            getExecutionTimeIntervalPausible().untilNowMPT(),
                getPruningTimeIntervalPausible().untilNowMPT(),
            arc.getPurityBestActual(100.0), arc.
                getPurityBestActual(1.0), arc.getEntropyOfClusters
                (), arc.getNMI_bestActual(),
            aSC.getTruePositive(), aSC.getTrueNegative(), aSC.
                getFalsePositive(), aSC.getFalseNegative(),
            aSC.getRandIndex(), aSC.getAdjustedRandIndex(), aSC.
                getJaccardScore(),
            aSC.getPrecision(), aSC.getRecall(), aSC.getFM(), aSC
                .getFMeasure(),
            arc.getClusterPointCountListString(true),sts.
                getStdDev(),
            AnalyzeResultClusters.getTotalQuality(CMC_List_size,
                arc.getNoiseCount(),
                    arc.getActualClassCount(false, false), arc.
                        getCalculatedClusterCount(false),
```

```
                        arc.getPurityBestActual(1.0),
                        arc.getNMI_bestActual()),
                avg, sumCumulative / countCumulative
        );

        analysisResultCSV.append(s);
        analysisResultTable.append("==========\n")
                .append(s).append("\n").append(arc.
                    getClassClusterTableBestActual())
                .append("\n");
      } else {
        String s = String.format(CONSTANTS.analysisResultCSVFormat,
                "MuDi", CONSTANTS.executionID, pointNumber, horizon,
                lambda, MinPts_forOffline,
                alpha, alphaCounter_fromOutside, gridGranularity,
                0.0, 0.0, 0.0, 0.0,
                pruningTimeInteger,
                CMC_List_size, -1,
                -1, -1,
                getExecutionTimeIntervalPausible().untilNowMPT(),
                    getPruningTimeIntervalPausible().untilNowMPT(),
                -1.0, -1.0, -1.0, -1.0,
                0.0, 0.0, 0.0, 0.0,
                -1.0, -1.0, -1.0,
                -1.0, -1.0, -1.0, -1.0,
                "0-0",-1.0,
                -1.0, -1.0, sumCumulative / countCumulative
        );

        analysisResultCSV.append(s);
      }
      offlineResultList.put(pointNumber, mDBSCAN);
    }

    private CoreMiniCluster searchInCoreMiniClusterList(DataPoint p)
        {
      for (CoreMiniCluster c : coreMiniClusterList)
        if (p.getDistance(c.getCenter()) <= c.getMCD())
          return c;
      return null;
    }

    private void CheckAndUpgradeGridCell(GridIndex gridIndex, long
        currentTimestamp) {
      if (grid.isReadyToBeUpgraded(gridIndex,
          Weight_of_UpgradeToMiniCluster)) {
        CoreMiniCluster scc1 = grid.getCell(gridIndex).
            getSmallCircleCluster(currentTimestamp);
        coreMiniClusterList.add(scc1);
        grid.deleteCell(gridIndex);
      }
    }

    private boolean PruneCoreMiniClusterList(long currentTimestamp) {
      boolean anyDeleted = false;

      pruningTimeInterval.Resume();

      grid.updateAllWeights(currentTimestamp);
      grid.removeLightCells(lambda, alpha, currentTimestamp);
```

```
289       for (int i = coreMiniClusterList.size() - 1; i >= 0; i--) {
            CoreMiniCluster aCMC = coreMiniClusterList.get(i);
291         if (aCMC.getWeight() < (alpha / (N * oneP2Lambda))) {
              coreMiniClusterList.remove(i);
293           anyDeleted = true;
            }
295       }

297       pruningTimeInterval.Pause();

299       return anyDeleted;
        }
301
        public ArrayList<CoreMiniCluster> getMicroClusters() {
303       return coreMiniClusterList;
        }
305
        public ArrayList<CoreMiniCluster> getMicroClusters(ArrayList<Long
           > timestampList) {
307       if (timestampList==null)
            return coreMiniClusterList;
309       ArrayList<CoreMiniCluster> timestampMCList = new ArrayList<
            CoreMiniCluster>();
          Iterator<CoreMiniCluster> it = coreMiniClusterList.iterator();
311       while (it.hasNext()) {
            CoreMiniCluster cmc = it.next();
313         if (cmc.containsAnyTimestamp(timestampList)) {
              timestampMCList.add(cmc);
315         }
          }
317       return timestampMCList;
        }
319
        public TreeMap<Integer, MDBSCAN> getOfflineResultList() {
321       return offlineResultList;
        }
323
        public String getAnalysisResultCSV() {
325       return analysisResultCSV.toString();
        }
327
        public TimeIntervalPausible getExecutionTimeIntervalPausible() {
329       return executionOnlineTimeInterval;
        }
331
        public TimeIntervalPausible getPruningTimeIntervalPausible() {
333       return pruningTimeInterval;
        }
335   }
```

E.2 CoreMiniCluster.java

CoreMiniCluster class which is inherited from another class called *SmallCircleCluster* is a class which is similar to the concept of MicroCluster implemented for DenStream algorithm (cf. Section 2.5.1).

CoreMiniCluster class has the extra ability (in addition to other methods defined in *SmallCircleCluster*) to add a data point to the core-mini-cluster if its distance is less than the *mcd* of the core-mini-cluster. *SmallCircleCluster* class is not represented here and it has several methods to manage a small circle-shaped cluster such as defining its class ID, cluster ID, center, and radius.

The source code for *CoreMiniCluster* class is presented as follows:

```
1  package MuDi;

3  import Common.SmallCircleCluster;
   import Data.Point.DataPoint;
5
   /**
7   * User: Amineh Amini
    */
9  public class CoreMiniCluster extends SmallCircleCluster {
     public static double LAMBDA;
11   private double mcd;

13   public CoreMiniCluster() {
       super();
15   }

17   public double getMCD() {
       return mcd;
19   }

21   public void setMCD(double radius) {
       this.mcd = radius;
23   }

25   @Override
     public boolean addDataPoint(DataPoint dataPoint) {
27     if (pointCanBeAdded(dataPoint)) {
         setWeight(1 + getWeight() * Math.pow(2, -LAMBDA * (dataPoint.
             getTimestamp() - getTimestamp_of_lastPoint())));
29       setTimestamp_of_lastPoint(dataPoint.getTimestamp());
         addClassID(dataPoint);
31
         if (getNumber_of_points() == 0)
33         setTimestamp_of_creation(dataPoint.getTimestamp());
         addTimestamp(dataPoint);
35
         incNumber_of_points();
```

```
37          return true;
          } else
39          return false;
      }
41
      public void updateWeight5(long currentTimestamp) {
43      setWeight(getWeight() * Math.pow(2, -LAMBDA * (currentTimestamp
            - getTimestamp_of_creation())));
      }
45
      @Override
47    public boolean pointCanBeAdded(DataPoint dataPoint) {
        double dist = getCenter().getDistance(dataPoint);
49      return (dist <= mcd);
      }
51
      @Override
53    public CoreMiniCluster clone() {
        return new CoreMiniCluster();
55    }

57    @Override
      public String toString() {
59      return String.format("CMC. Center: %s; Radius=%2.2f; Weight=%2.2
            f; (%d) Points' Class=%d (Cluster=%d)",
                getCenter().toString(), getMCD(), getWeight(),
                   getNumber_of_points(),
61              getActualClassID(), getCalculatedClusterID());
      }
63
  }
```

E.3 MDBSCAN.java

MDBSCAN class is an adapted implementation of DBScan algorithm. The major difference is that for its density-reachability checking purposes it does not use the ε value. Instead, it uses the mean and standard deviation values of the distances of the core-mini-clusters.

The source code for *MDBSCAN* class is presented as follows:

```
package MuDi;

import Common.AverageVariance;
import Common.SmallCircleCluster_Interface;
import Data.ClusterList;

import java.util.ArrayList;
import java.util.Iterator;

/**
 * User: Amineh Amini
 */
public class MDBSCAN {
  private int minPoints;

  private ArrayList<CoreMiniCluster> mcList;

  private double GRANULARITY;

  public MDBSCAN(ArrayList<CoreMiniCluster> mcList, int minPoints,
     int gridGranularity) {
    this.mcList = mcList;
    this.minPoints = minPoints;
    this.GRANULARITY = gridGranularity;
  }

  public void doClustering() {
    for (CoreMiniCluster scc : mcList) {
      scc.setCalculatedClusterID(ClusterList.NO_CLUSTER);
      scc.setAsUnvisited();
    }

    byte currentClusterId = 1;
    for (int i = 0; i < mcList.size(); i++) {
      CoreMiniCluster p = mcList.get(i);
      p.setAsVisited();
      if (p.getCalculatedClusterID() == ClusterList.NO_CLUSTER
              ||
              p.getCalculatedClusterID() == ClusterList.NOISE
              ) {
        if (expandCluster(p, currentClusterId)) {
          if (currentClusterId==ClusterList.MAX_CLUSTER_NUMBER) {
            System.out.println("Exceeded number of clusters");
            return;
          }
            else
```

```
46            currentClusterId++;
          }
48          if (false) {
            System.out.println("------------");
50            for (CoreMiniCluster ggg : mcList)
              if (ggg.getCalculatedClusterID() == currentClusterId -
                  1)
52              System.out.printf("%s\n", ggg.getCenter().
                    getCSV_ofCenter());
          }
54        }
      }
56  }

58  private boolean expandCluster(CoreMiniCluster cmc_p, byte
      currentClusterId) {
      ArrayList<CoreMiniCluster> neighbors =
          getNeighborCoreMiniClusterList(cmc_p);
60    if (neighbors.size() < minPoints) {
        cmc_p.setCalculatedClusterID(ClusterList.NOISE);
62      return false;
      }
64    ArrayList<CoreMiniCluster> N_Core = getMinPtsNearestNeighbors(
          cmc_p, neighbors);
      AverageVariance N_Core_AV = getAverageVariance_ofDistance(cmc_p
          , N_Core);
66    ArrayList<CoreMiniCluster> seeds = new ArrayList<
          CoreMiniCluster>();
      seeds.addAll(N_Core);
68
      cmc_p.setCalculatedClusterID(currentClusterId);
70
      seeds.remove(cmc_p);
72    int N_Core_size = seeds.size();
      while (seeds.size() > 0) {
74      CoreMiniCluster cmc_q = seeds.get(0);
        if (!cmc_q.isVisited()) {
76        cmc_q.setAsVisited();
          ArrayList<CoreMiniCluster> cmc_q_neighbors =
              getNeighborCoreMiniClusterList(cmc_q);
78        if (cmc_q_neighbors.size() >= minPoints) {
            ArrayList<CoreMiniCluster> N_Sh_Q =
                getMinPtsNearestNeighbors(cmc_q, cmc_q_neighbors);
80          AverageVariance N_Sh_Q_AV = getAverageVariance_ofDistance
                (cmc_q, N_Sh_Q);
            if ((N_Sh_Q_AV.mean >= (N_Core_AV.mean - N_Core_AV.
                getStdDev())) ||
82                (N_Sh_Q_AV.mean <= (N_Core_AV.mean + N_Core_AV.
                      getStdDev()))) {
              for (CoreMiniCluster a_N_Sh_Q : N_Sh_Q) {
84              seeds.add(a_N_Sh_Q);

86              N_Core_AV = updateAverageVariance(
                        N_Core_AV, N_Core_size,
88                      cmc_q.getCenter().getDistance(a_N_Sh_Q.
                            getCenter()));
                N_Core_size++;
90            }
            }
92        }
        }
```

```
94        if (cmc_q.getCalculatedClusterID() == ClusterList.NO_CLUSTER)
            cmc_q.setCalculatedClusterID(currentClusterId);
96        seeds.remove(cmc_q);
        }
98      return true;
    }
100
    private ArrayList<CoreMiniCluster> getNeighborCoreMiniClusterList
        (
102           CoreMiniCluster main) {
      ArrayList<CoreMiniCluster> resultList = new ArrayList<
          CoreMiniCluster>();
104     double distance;
      for (CoreMiniCluster c : mcList) {
106       distance = main.getCenter().getDistance(c.getCenter());
        if (distance > 0 &&
108               distance <= (1.0 / GRANULARITY + 1.0 / GRANULARITY))
          resultList.add(c);
110     }
      return resultList;
112   }

114   private AverageVariance getAverageVariance_ofDistance(
          CoreMiniCluster scc, ArrayList<CoreMiniCluster> neighbors
              ) {
116     double sum = 0.0, sumSqr = 0.0, count = 0;
      Iterator<CoreMiniCluster> it = neighbors.iterator();
118     CoreMiniCluster aSCC;
      double distance;
120     while (it.hasNext()) {
        aSCC = it.next();
122       distance = aSCC.getCenter().getDistance(scc.getCenter());
        if (distance > 0) {
124         count++;
          sum += distance;
126         sumSqr += distance * distance;
        }
128     }
      return new AverageVariance(sum / count, (sumSqr - (sum * sum) /
          count) / (count - 1));
130   }

132   private ArrayList<CoreMiniCluster> getMinPtsNearestNeighbors(
          CoreMiniCluster scc, ArrayList<CoreMiniCluster> neighbors
              ) {
134     ArrayList<CoreMiniCluster> minPtsNearestNeighbors =
            new ArrayList<CoreMiniCluster>();
136
      double[] distances = new double[neighbors.size()];
138     for (int i = 0; i < neighbors.size(); i++)
        distances[i] = scc.getCenter().getDistance(neighbors.get(i).
            getCenter());
140
      double previousMinDistance = 0.0, minDist;
142     do {
        minDist = Double.MAX_VALUE;
144       boolean minChanged = false;
        for (int i = 0; i < neighbors.size(); i++)
146         if (distances[i] < minDist && distances[i] >
              previousMinDistance) {
            minDist = distances[i];
```

```
148             minChanged = true;
              }
150         if (!minChanged)
              break;
152
            for (int i = 0; i < neighbors.size(); i++)
154           if (distances[i] - minDist < 0.0000001f && distances[i] >
                 previousMinDistance)
              minPtsNearestNeighbors.add(neighbors.get(i));
156
            previousMinDistance = minDist;
158       } while (minPtsNearestNeighbors.size() < minPoints);

160       return minPtsNearestNeighbors;
      }
162
      private AverageVariance updateAverageVariance(AverageVariance
         lastValues, int lastCount, double newDistance) {
164       AverageVariance newValues = new AverageVariance();
          newValues.mean = lastValues.mean + (newDistance - lastValues.
             mean) / lastCount;
166       newValues.variance = lastValues.variance + (newDistance -
             lastValues.mean) * (newDistance - newValues.mean);
          return newValues;
168   }

170   public ArrayList<CoreMiniCluster> getMcList() {
        return mcList;
172   }

174   public ArrayList<SmallCircleCluster_Interface> getSCC_List() {
        ArrayList<SmallCircleCluster_Interface> result = new ArrayList<
           SmallCircleCluster_Interface>(mcList.size());
176     Iterator<CoreMiniCluster> it = mcList.iterator();
        while (it.hasNext()) {
178       CoreMiniCluster aCMC = it.next();
          result.add(aCMC);
180     }
        return result;
182   }

184   private void printPoints_CSV(ArrayList<CoreMiniCluster> cmcs) {
        for (CoreMiniCluster c : cmcs)
186       System.out.printf("%s\n", c.getCenter().getCSV_ofCenter());
      }
188 }
```

E.4 Grid.java

Grid class manages the grid structure of MuDi-Stream which is used to keep the outliers. The class has different methods to add newly arrived data points, to check if grid cells are ready to be converted to core-mini-clusters, to remove the grid cells which their weights are below a specified threshold, and to update the weights of the grid cells according to the current timestamp.

The source code for *Grid* class is presented as follows:

```
package MuDi.Grid;

import Data.Point.DataPoint;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.TreeMap;

/**
 * User: Amineh Amini
 */
public class Grid {
  private Array_of_Array grid;

  private int gridGranularity;
  private double len;

  private int cells, totalPoints;

  private TreeMap<GridIndex, GridCharacteristicVector> gridList;

  public Grid(int dimensionCount, int gridGranularity) {
    Array_of_Array.DIMENSION_COUNT = dimensionCount;
    Array_of_Array.DIMENSION_SIZE = gridGranularity;
    grid = new Array_of_Array(0);
    this.gridGranularity = gridGranularity;
    len = 1.0 / gridGranularity;
    cells = 0;
    totalPoints = 0;
    gridList = new TreeMap<GridIndex, GridCharacteristicVector>();
  }

  public GridIndex addPoint(DataPoint p) {
    int[] indices = new int[Array_of_Array.DIMENSION_COUNT];
    for (int i = 0; i < indices.length; i++)
      indices[i] = Math.min(gridGranularity - 1, (int) (p.
          getDimensionValue(i) / len));

    GridIndex gridIndex = new GridIndex(Array_of_Array.
        DIMENSION_COUNT, indices);
    if (grid.getCellValue(gridIndex) == null) {
      GridCharacteristicVector aGCV = new GridCharacteristicVector(
          p);
      grid.setCellValue(gridIndex, aGCV);
```

```
42          gridList.put(gridIndex, aGCV);
            cells++;
44        } else
            grid.getCellValue(gridIndex).addPoint(p);
46
          totalPoints++;
48
          return gridIndex;
50      }

52      public GridCharacteristicVector getCell(GridIndex gridIndex) {
          return grid.getCellValue(gridIndex);
54      }

56      public boolean isReadyToBeUpgraded(GridIndex gridIndex, double
            thresholdWeight) {
          return (getCell(gridIndex) != null) && (getCell(gridIndex).
              isReadyToBeUpgraded(thresholdWeight));
58      }

60      public void deleteCell(GridIndex gridIndex) {
          GridCharacteristicVector aGCV = getCell(gridIndex);
62        if (aGCV != null) {
            gridList.remove(gridIndex);
64          totalPoints -= aGCV.getNumberOfDataPoints();
            cells--;
66        }
          grid.setCellValue(gridIndex, null);
68      }

70      public void removeLightCells(double lambda, double alpha, long
            currentTimestamp) {
          GridIndex gi;
72        GridCharacteristicVector cell;
          double OWT; /* Outlier Weight Threshold */
74
          ArrayList<GridIndex> toBeDeleted = new ArrayList<GridIndex>();
76
          Iterator<GridIndex> itGI = gridList.keySet().iterator();
78        while (itGI.hasNext()) {
            gi = itGI.next();
80          cell = gridList.get(gi);
            if (cell != null) {
82            OWT = (alpha * (1 - Math.pow(2, -lambda * (currentTimestamp
                  - cell.getLastTimestamp() + 1)))) /
                    (Math.pow(gridGranularity, Array_of_Array.
                        DIMENSION_COUNT) * (1 - Math.pow(2, -lambda *
                        cell.getLastTimestamp())));
84            if (cell.getWeight() <= OWT)
                toBeDeleted.add(gi);
86          }
          }
88
          itGI = toBeDeleted.iterator();
90        while (itGI.hasNext()) {
            deleteCell(itGI.next());
92        }
        }
94
        public void removeLightCells_old(double lambda, double alpha,
            long currentTimestamp) {
```

```
96      GridCharacteristicVector cell;
        double OWT; /* Outlier Weight Threshold */
98
        int[] r = new int[Array_of_Array.DIMENSION_COUNT];
100
        GridIndex g = new GridIndex(Array_of_Array.DIMENSION_COUNT, r);
102     do {
          cell = grid.getCellValue(g);
104       if (cell != null) {
            OWT = (alpha * (1 - Math.pow(2, -lambda * (currentTimestamp
                - cell.getLastTimestamp() + 1)))) /
106               (Math.pow(gridGranularity, Array_of_Array.
                    DIMENSION_COUNT) * (1 - Math.pow(2, -lambda *
                    cell.getLastTimestamp())));
            if (cell.getWeight() <= OWT)
108           deleteCell(g);
          }
110       g = GridIndex.next(g, gridGranularity);
        } while (g != null);
112   }

114   public void updateAllWeights(long currentTimestamp) {
        GridIndex gi;
116     GridCharacteristicVector cell;

118     Iterator<GridIndex> itGI = gridList.keySet().iterator();
        while (itGI.hasNext()) {
120       gi = itGI.next();
          cell = gridList.get(gi);
122       if (cell != null)
            cell.updateWeight(currentTimestamp);
124     }
      }
126
      public void updateAllWeights_old(long currentTimestamp) {
128     GridCharacteristicVector cell;

130     int[] r = new int[Array_of_Array.DIMENSION_COUNT];

132     GridIndex g = new GridIndex(Array_of_Array.DIMENSION_COUNT, r);
        do {
134       cell = grid.getCellValue(g);
          if (cell != null)
136         cell.updateWeight(currentTimestamp);
          g = GridIndex.next(g, gridGranularity);
138     } while (g != null);
      }
140
      public String toString() {
142     return String.format("Cells=%d, Points=%d", cells, totalPoints)
          ;
      }
144 }
```

E.5 GridCharacteristicVector.java

GridCharacteristicVector class stores various characteristics of a grid cell. The attributes are the weight, the last data point's timestamp, and the class ID of the data points. Moreover, the class has several methods to add a data point, calculate the grid cell's center, and to check if the grid cell is ready to be converted to a core-mini-cluster.

The source code for *GridCharacteristicVector* class is presented as follows:

```
package MuDi.Grid;

import Data.ClusterList;
import Data.Point.Center;
import Data.Point.DataPoint;
import Data.Point.Point;
import MuDi.CoreMiniCluster;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

/**
 * User: Amineh Amini
 */
public class GridCharacteristicVector {
  public static double LAMBDA;
  public static final int INITIAL_CELL_CAPACITY = 5;

  protected ArrayList<DataPoint> points;
  protected double weight;
  protected long lastTimestamp;
  protected byte classID;
  private HashMap<Byte, Integer> MixedClassIDs;

  public GridCharacteristicVector(DataPoint p) {
    points = new ArrayList<DataPoint>(INITIAL_CELL_CAPACITY);
    weight = 1;
    lastTimestamp = 0;
    classID = ClusterList.NO_CLUSTER;
    addPoint(p);
  }

  public double getWeight() {
    return weight;
  }

  public long getLastTimestamp() {
    return lastTimestamp;
  }

  public int getNumberOfDataPoints() {
    return points.size();
  }

  public void addPoint(DataPoint p) {
```

```
      points.add(p);
48    if (getNumberOfDataPoints() == 1)
        weight = 1;
50    else
        weight = 1 + weight * Math.pow(2, -LAMBDA * (p.getTimestamp()
            - lastTimestamp));
52    lastTimestamp = p.getTimestamp();
      addClassID(p);
54  }

56  public void updateWeight(long currentTimestamp) {
      weight = weight * Math.pow(2, -LAMBDA * (currentTimestamp -
          lastTimestamp));
58  }

60  private void addClassID(DataPoint p) {
      if (MixedClassIDs == null)
62      MixedClassIDs = new HashMap<Byte, Integer>();
//    else      p = p;
64    if (classID == ClusterList.NO_CLUSTER) {
        classID = p.getClassID();
66      MixedClassIDs.put(classID,
                1 + (MixedClassIDs.containsKey(classID) ?
                    MixedClassIDs.get(classID) : 0)
68      );

70    } else if (classID != p.getClassID()) {
        MixedClassIDs.put(p.getClassID(),
72              1 + (MixedClassIDs.containsKey(p.getClassID()) ?
                    MixedClassIDs.get(p.getClassID()) : 0)
        );
74      classID = ClusterList.MIXED_CLUSTER;
      } else
76      MixedClassIDs.put(classID,
                1 + (MixedClassIDs.containsKey(classID) ?
                    MixedClassIDs.get(classID) : 0)
78      );
    }
80
    public Center calculateCenter(long currentTimestamp) {
82    Center center = new Center();
      double wSum = 0;
84    double[] dimensionValues = new double[DataPoint.DIMENSION_COUNT
          ];

86    for (DataPoint p : points) {
        double w = p.getWeight(currentTimestamp);
88      for (int i = 0; i < DataPoint.DIMENSION_COUNT; i++)
          dimensionValues[i] += p.getDimensionValue(i) * w;
90
        wSum += w;
92    }
      for (int j = 0; j < DataPoint.DIMENSION_COUNT; j++)
94      center.setDimensionValue(j, dimensionValues[j] / wSum);

96    return center;
    }
98
    public boolean isReadyToBeUpgraded(double thresholdWeight) {
100   return              (getNumberOfDataPoints() > 1) &&
                      (weight >= thresholdWeight);
```

```
102     }
104     public double calculateMCD(Point mainPoint) {
          double mcd = 0;
106
          for (DataPoint p : points)
108         mcd = Math.max(mainPoint.getDistance(p), mcd);
110       return mcd;
        }
112
        public CoreMiniCluster getSmallCircleCluster(long
            currentTimestamp) {
114       CoreMiniCluster c = new CoreMiniCluster();
          c.setWeight(getWeight());
116       c.setCenter(calculateCenter(currentTimestamp));
          c.setMCD(calculateMCD(c.getCenter()));
118       c.setTimestamp_of_lastPoint(getLastTimestamp());
          c.setNumber_of_points(getNumberOfDataPoints());
120       c.setActualClassID(classID);
          c.setMixedClassIDs(MixedClassIDs);
122
          HashMap<Long, Integer> tsList=new HashMap<Long, Integer>();
124       Iterator<DataPoint> it=points.iterator();
          while (it.hasNext()) {
126         DataPoint dp=it.next();
            long ts = dp.getTimestamp();
128
            if (tsList.containsKey(ts))
130           tsList.put(ts, 1 + tsList.get(ts));
            else
132           tsList.put(ts, 1);
          }
134       c.setTimestampList_of_points(tsList);

136       return c;
        }
138   }
```

E.6 GridIndex.java

GridCharacteristicVector class stores the location for each grid cell according to the data point's dimensions.

The source code for *GridIndex* class is presented as follows:

```
package MuDi.Grid;

/**
 * User: Amineh Amini
 */
public class GridIndex implements Comparable<GridIndex> {
  protected int[] dimensionIndex;
  private int dimensionCount;

  public GridIndex(int dimensionCount, int[] dimensionIndex) {
    this.dimensionCount = dimensionCount;
    this.dimensionIndex = new int[dimensionCount];
    if (dimensionCount != dimensionIndex.length)
      throw new UnsupportedOperationException("Invalid dimensions")
          ;
    for (int i = 0; i < dimensionCount; i++)
      this.dimensionIndex[i] = dimensionIndex[i];
  }

  public int getIndex(int dimensionNumber) {
    return dimensionIndex[dimensionNumber];
  }

  public void setIndex(int dimensionNumber, int value) {
    dimensionIndex[dimensionNumber] = value;
  }

  public int getDimensionCount() {
    return this.dimensionCount;
  }

  public static GridIndex next(GridIndex gridIndex, int
     maxPerDimension) {
    int d = gridIndex.getDimensionCount() - 1;
    while (d >= 0) {
      gridIndex.setIndex(d, gridIndex.getIndex(d) + 1);
      if (gridIndex.getIndex(d) >= maxPerDimension) {
        gridIndex.setIndex(d, 0);
        if (d == 0)
          return null;
        d--;
      } else
        break;
    }
    if (gridIndex.getIndex(0) > maxPerDimension)
      return null;
    else
      return gridIndex;
  }

```

```
50    public String toString() {
        StringBuilder s = new StringBuilder();
52      for (int i = 0; i < dimensionIndex.length; i++)
          s.append(String.format("d%d=%d%s", i + 1, dimensionIndex[i],
54                  i == dimensionIndex.length - 1 ? "" : ", "
          ));
56      return s.toString();
      }
58
      @Override
60    public int compareTo(GridIndex o) {
        if (this.getDimensionCount() != o.getDimensionCount())
62        throw new UnsupportedOperationException("GridIndex: Different
              Dimension Count.");

64      for (int i = 0; i < getDimensionCount(); i++)
          if (this.getIndex(i) != o.getIndex(i))
66          return this.getIndex(i) < o.getIndex(i) ? 1 : -1;

68      return 0;
      }
70  }
```

REFERENCES

Ackermann, M. R., Lammersen, C., Märtens, M., Raupach, C., Sohler, C., & Swierkot, K. (2010). Streamkm++: A clustering algorithm for data streams. In *Proceedings of the 12th workshop on algorithm engineering and experiments (alenex '10)* (pp. 173–187). Society for Industrial and Applied Mathematics.

Aggarwal, C. C. (Ed.). (2007). *Data streams – models and algorithms*. Springer.

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on very large data bases* (pp. 81–92). VLDB Endowment.

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of the thirtieth international conference on very large data bases - volume 30* (pp. 852–863). VLDB Endowment.

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2005). On high dimensional projected clustering of data streams. *Data Mining and Knowledge Discovery*, *10*(3), 251–273. doi: 10.1007/s10618-005-0645-7

Aggarwal, C. C., & Reddy, C. K. (Eds.). (2013). *Data clustering: Algorithms and applications*. CRC Press.

Aggarwal, C. C., & Yu, P. S. (2007). A survey of synopsis construction in data streams. In C. C. Aggarwal (Ed.), *Data streams - models and algorithms* (Vol. 31, p. 169-207). Springer. doi: http://dx.doi.org/10.1007/978-0-387-47534-9_9

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998, June). Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Record*, *27*(2), 94–105.

Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, *12*(4), 461-486.

Amini, A., & Ying Wah, T. (2012). DENGRIS-Stream: A density-grid based clustering algorithm for evolving data streams over sliding window. In *International conference on data mining and computer engineering (icdmce)* (pp. 206–210). Bangkok, Thailand.

Amini, A., Ying Wah, T., & Saboohi, H. (2014, January). On density-based data streams clustering algorithms: A survey. *Journal of Computer Science and Technology*, *29*(1), 116-141. doi: 10.1007/s11390-013-1416-3

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999, June). Optics: ordering points to identify the clustering structure. *ACM SIGMOD Record*, *28*(2), 49–60. doi: 10.1145/304181.304187

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 1–16). New York, NY, USA: ACM. doi: 10.1145/543613.543615

Babcock, B., Datar, M., & Motwani, R. (2002). Sampling from a moving window over streaming data. In *Proceedings of the thirteenth annual acm-siam symposium on discrete algorithms* (pp. 633–634). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Babcock, B., Datar, M., Motwani, R., & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 234–243). New York, NY, USA: ACM. doi: 10.1145/773153.773176

Bache, K., & Lichman, M. (2013). *UCI machine learning repository.*

Barbará, D. (2002, January). Requirements for clustering data streams. *SIGKDD Explorations*, *3*(2), 23–27. doi: 10.1145/507515.507519

Bhatnagar, V., Kaur, S., & Chakravarthy, S. (2013). Clustering data streams using grid-based synopsis. *Knowledge and Information Systems*, 1-26. doi: 10.1007/s10115-013-0659-1

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). *Moa massive online analysis.*

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., & Seidl, T. (2010). Moa: Massive online analysis, a framework for stream classification and clustering. In *Journal of machine learning research (jmlr)* (Vol. 11, pp. 44–50).

Bohm, C., Kailing, K., Kriegel, H.-P., & Kroger, P. (2004). Density connected clustering with local subspace preferences. In *Proceedings of the fourth ieee international conference on data mining (icdm)* (pp. 27–34).

Bolanos, M. (2014). *Moving generator - infrastructure for evolving streams in r* (Bachelor thesis). School of Computer Science and Engineering, Southern Methodist University.

Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., & Dougherty, E. R. (2007, March). Model-based evaluation of clustering validation measures. *Pattern Recogn.*, *40*, 807–824. doi: http://dx.doi.org/10.1016/j.patcog.2006.06.026

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006, April). Density-based clustering over an evolving data stream with noise. In *Proc. the 2006 siam conference on data mining* (pp. 328–339).

Carmelo, C., Alfredo, F., Rosalba, G., Giuseppe, P., & Alfredo, P. (2013). Enhancing density-based clustering: Parameter reduction and outlier detection. *Information Systems*, *38*(3), 317 - 330.

Chaovalit, P. (2009). *Clustering transient data streams by example and by variable* (PhD Thesis). University of Maryland at Baltimore County, Catonsville, MD, USA.

Charikar, M., O'Callaghan, L., & Panigrahy, R. (2003). Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual acm symposium on theory of computing* (pp. 30–39). New York, NY, USA: ACM. doi: 10.1145/780542.780548

Chen, X., Liu, S., Chen, T., Zhang, Z., & Zhang, H. (2012). An improved semi-supervised clustering algorithm for multi-density datasets with fewer constraints. *Procedia Engineering*, *29*, 4325–4329.

Chen, Y., & Tu, L. (2007, August). Density-based clustering for real-time stream data. In *Proc. of the 13th acm sigkdd international conference on knowledge discovery and data mining* (pp. 133–142). New York, NY, USA: ACM. doi: 10.1145/1281192.1281210

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2006). Map-reduce for machine learning on multicore. In *Nips* (Vol. 6, pp. 281–288).

Dang, X. H., Lee, V., Ng, W. K., Ciptadi, A., & Ong, K. L. (2009). An EM-based algorithm for clustering data streams in sliding windows. In *Proceedings of the 14th international conference on database systems for advanced applications* (pp. 230–235). Berlin, Heidelberg: Springer-Verlag.

De Francisci Morales, G. (2013). Samoa: a platform for mining big data streams. In *Proceedings of the 22nd international conference on world wide web companion* (pp. 777–778).

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, series b*, *39*(1), 1–38.

Dongen, S. V., & Dongen, S. V. (2000). *Performance criteria for graph clustering and markov cluster experiments* (Tech. Rep.). -: National Research Institute for Mathematics and Computer Science in the.

Duan, L., Xu, L., Guo, F., Lee, J., & Yan, B. (2007). A local-density based spatial clustering algorithm

with noise. *Information System, 32*(7), 978-986.

Esfandani, G., & Abolhassani, H. (2010). Msdbscan: multi-density scale-independent clustering algorithm based on dbscan. In *Advanced data mining and applications* (pp. 202–213). Springer.

Esfandani, G., Sayyadi, M., & Namadchian, A. (2012). Gdclu: a new grid-density based clustring algorithm. In *Software engineering, artificial intelligence, networking and parallel & distributed computing (snpd), 2012 13th acis international conference on* (pp. 102–107).

Ester, M. (2013). Density-based clustering. *Data Clustering: Algorithms and Applications, 31*, 111.

Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., & Xu, X. (1998). Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24rd international conference on very large data bases* (pp. 323–333). San Francisco, CA, USA: Morgan Kaufmann Publishers.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd international conference on knowledge discovery and data mining (kdd)* (pp. 226–231). Portland, Oregon: AAAI Press.

Forestiero, A., Pizzuti, C., & Spezzano, G. (2013). A single pass algorithm for clustering evolving data streams based on swarm intelligence. *Data Mining and Knowledge Discovery*, *26*(1), 1–26.

Fowlkes, E. B., & Mallows, C. L. (1983). A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, *78*(383), 553–569.

Frank, A., & Asuncion, A. (2010). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Gama, J., & (Eds), M. G. (2007). *Learning from data streams – processing techniques in sensor networks*. Springer.

Gama, J. a., Rodrigues, P. P., & Lopes, L. (2011, January). Clustering distributed sensor data streams using local processing and reduced communication. *Intelligent Data Analysis, 15*(1), 3–28.

Gao, J., Li, J., Zhang, Z., & Tan, P.-N. (2005). An incremental data stream clustering algorithm based on dense units detection. In *Proceedings of the 9th pacific-asia conference on advances in knowledge discovery and data mining* (pp. 420–425). Springer Netherlands. doi: 10.1007/11430919_49

Garofalakis, M., Gehrke, J., & Rastogi, R. (2002). Querying and mining data streams: You only get one look. In *Proceedings of the 2002 acm sigmod international conference on management of data* (pp. 635–635). ACM. doi: 10.1145/564691.564794

Garofalakis, M. N. (2009). Wavelets on streams. In *Encyclopedia of database systems* (p. 3446-3451). Springer US.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., & Strauss, M. J. (2003, March). One-pass wavelet decompositions of data streams. *IEEE Trans. on Knowl. and Data Eng., 15*(3), 541–554. doi: 10.1109/TKDE.2003.1198389

Guha, S., Meyerson, A., Mishra, N., Motwani, R., & O'Callaghan, L. (2003, June). Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, *15*(3), 515–528. doi: 10.1109/TKDE.2003.1198387

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). Clustering data streams. In *Proceedings of the 41st annual symposium on foundations of computer science* (p. 359). Washington, DC, USA: IEEE Computer Society.

Hahsler, M., & Dunham, M. H. (2011, April). Temporal structure learning for clustering massive data streams in real-time. In *Proc. of the SIAM conference on data mining* (pp. 664–675). SIAM.

Han, J., & Kamber, M. (2006). *Data mining: Concepts and techniques (2nd edition)*. Morgan Kaufmann.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques (3rd edition)* (3rd ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hassani, M., Spaus, P., Gaber, M. M., & Seidl, T. (2012). Density-based projected clustering of data streams. In *Scalable uncertainty management* (pp. 311–324). Springer.

Hawwash, B. (2013). *Stream-dashboard : a big data stream clustering framework with applications to social media streams.* (PhD Thesis). University of Louisville.

Hershberger, J., Shrivastava, N., & Suri, S. (2009, February). Summarizing spatial data streams using clusterhulls. *Journal of Experimental Algorithmics*, *13*, 4:2.4–4:2.28.

Hinneburg, A., & Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In *Kdd* (pp. 58–65).

Holmes, G., Donkin, A., & Witten, I. H. (1994, August). WEKA: a machine learning workbench. In *Proceedings of the second australian and new zealand conference on intelligent information systems* (pp. 357–361).

Huang, T.-q., Yu, Y.-q., Li, K., & Zeng, W.-f. (2009). Reckon the parameter of dbscan for multi-density data sets with constraints. In *International conference on artificial intelligence and computational intelligence, 2009. aici'09* (Vol. 4, pp. 375–379).

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*(1), 193–218.

Hubert, L. J., & Levin, J. R. (1976). A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, *83*(6), 1072–1080.

Isaksson, C., Dunham, M. H., & Hahsler, M. (2012). Sostream: Self organizing density-based clustering over data stream. In P. Perner (Ed.), *Machine learning and data mining in pattern recognition* (Vol. 7376, p. 264-278). Springer Berlin Heidelberg.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, *37*, 547–579.

Jain, A. K. (2010, June). Data clustering: 50 years beyond k-means. *Pattern Recognition Letter*, *31*(8), 651–666. doi: 10.1016/j.patrec.2009.09.011

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Jia, C., Tan, C., & Yong, A. (2008). A grid and density-based clustering algorithm for processing data stream. In *Proceedings of the second international conference on genetic and evolutionary computing* (pp. 517–521). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/WGEC.2008.32

Jin, W., Tung, A. K., Han, J., & Wang, W. (2006). Ranking outliers using symmetric neighborhood relationship. In *Advances in knowledge discovery and data mining* (pp. 577–593). Springer.

Karypis, G., Han, E.-H. S., & Kumar, V. (1999, August). Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, *32*(8), 68–75.

Kaufman, L., & Rousseeuw, P. J. (2005). *Finding groups in data: An introduction to cluster analysis, 1st edition* (1st ed.). Wiley.

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis, 2nd edition* (2nd ed., Vol. 344). John Wiley & Sons.

Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2005). Selecting features for intrusion detection: a feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust.*

Kennedy, J. F., Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann Pub.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59-69. doi: 10.1007/BF00337288

Kranen, P., Assent, I., Baldauf, C., & Seidl, T. (2011). The clustree: indexing micro-clusters for anytime stream mining. *Knowledge Information System.*, *29*(2), 249-272.

Kranen, P., Kremer, H., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2010). Clustering performance on evolving data streams: Assessing algorithms and evaluation measures within moa. In *Proceeding of ieee international conference on data mining (icdm 2010), sydney, australia* (pp. 1400–1403).

Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 868–876). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/2020408.2020555

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(3), 231-240.

Lee, C.-H. (2012). Mining spatio-temporal information on microblogging streams using a density-based online clustering method. *Expert Systems with Applications*, *39*(10), 9623–9641. doi: http://dx.doi.org/10.1016/j.eswa.2012.02.136

Li, X., Ye, Y., Li, M. J., & Ng, M. K. (2010). On cluster tree for nested and multi-density data clustering. *Pattern Recognition*, *43*(9), 3130 - 3143.

Li, Y., Li, D., Wang, S., & Zhai, Y. (2014). Incremental entropy-based clustering on categorical data streams with concept drift. *Knowledge-Based Systems*, *59*(0), 33 - 47.

Lin, J., & Lin, H. (2009). A Density-Based Clustering over Evolving Heterogeneous Data Stream. In Luo, Q and Yi, J and Bin, C (Ed.), *2nd International Colloquium on Computing, Communication, Control and Management (CCCM 2009)* (p. 275-277). IEEE.

Li-xiong, L., Jing, K., Yun-fei, G., & Hai, H. (2009). A three-step clustering algorithm over an evolving data stream. In *Proceedings of ieee international conference on intelligent computing and intelligent systems (icis)* (pp. 160–164).

Lloyd, S. P. (1982, March). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. doi: 10.1109/TIT.1982.1056489

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam & J. Neyman (Eds.), *Proc. of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, p. 281-297). University of California Press.

Manning, C. D., Raghavan, P., & Schtze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Masud, M. M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *Icdm'08. eighth ieee international conference on data mining.* (pp. 929–934).

Matysiak, M. (2012, November). *Data stream mining: Basic methods and techniques* (Tech. Rep.). -: RWTH Aachen University.

Meesuksabai, W., Kangkachit, T., & Waiyamai, K. (2011). Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In J. Tang, I. King, L. Chen, & J. Wang (Eds.), *Advanced data mining and applications* (Vol. 7121, p. 27-40). Springer Berlin Heidelberg.

Meilă, M. (2005). Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international*

conference on machine learning (pp. 577–584). New York, NY, USA: ACM.

Mete, M., Kockara, S., & Aydin, K. (2011). Fast density-based lesion detection in dermoscopy images. *Computerized Medical Imaging and Graphics, 35*(2), 128–136. doi: 10.1016/j.compmedimag.2010.07.007

Michalak, S., DuBois, A., DuBois, D., Wiel, S. V., & Hogden, J. (2012). Developing systems for real-time streaming analysis. *Journal of Computational and Graphical Statistics, 21*(3), 561–580.

Milligan, G. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika, 46*(2), 187-199. doi: 10.1007/BF02293899

Mitra, S., & Nandy, J. (2011). Kddclus: A simple method for multi-density clustering. In *Proceeding of international workshop on soft computing applications and knowledge discovery (scakd'2011), moscow* (pp. 72–76).

Namadchian, A., & Esfandani, G. (2012). Dsclu: a new data stream clustring algorithm for multi density environments. In *Software engineering, artificial intelligence, networking and parallel & distributed computing (snpd), 2012 13th acis international conference on* (pp. 83–88).

Ng, W., & Dash, M. (2010). Discovery of frequent patterns in transactional data streams. In *Transactions on large-scale data- and knowledge-centered systems ii* (Vol. 6380, pp. 1–30). Springer Berlin / Heidelberg.

Ntoutsi, I., Zimek, A., Palpanas, T., Kröger, P., & Kriegel, H.-P. (2012). Density-based projected clustering over high dimensional data streams. In *Sdm* (p. 987-998).

OĆallaghan, L., Meyerson, A., Motwani, R., Mishra, N., & Guha, S. (2002). Streaming-data algorithms for high-quality clustering. In *International conference on data engineering* (pp. 685–694). Los Alamitos, CA, USA: IEEE Computer Society. doi: 10.1109/ICDE.2002.994785

Papapetrou, O., & Chen, L. (2011). Xstreamcluster: An efficient algorithm for streaming xml data clustering. In J. X. Yu, M.-H. Kim, & R. Unland (Eds.), *Dasfaa (1)* (Vol. 6587, p. 496-510). Springer.

Plant, C., Teipel, S. J., Oswald, A., Bohm, C., Meindl, T., Miranda, J. M., ... Ewers, M. (2010, March). Automated detection of brain atrophy patterns based on MRI for the prediction of alzheimer's disease. *NeuroImage, 50*(1), 162–174. doi: 10.1016/j.neuroimage.2009.11.046

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association, 66*(336), 846–850.

Read, J., Bifet, A., Holmes, G., & Pfahringer, B. (2012). Scalable and efficient multi-label classification for evolving data streams. *Machine Learning, 88*(1-2), 243-272. doi: 10.1007/s10994-012-5279-6

Ren, J., Cai, B., & Hu, C. (2011). Clustering over data streams based on grid density and index tree. *Journal of Convergence Information Technology, 6*(1), 83–93.

Ren, J., Ma, R., & Ren, J. (2009). Density-based data streams clustering over sliding windows. In *Proceedings of the 6th international conference on fuzzy systems and knowledge discovery (fskd)* (pp. 248–252). Piscataway, NJ, USA: IEEE Press. doi: 10.1109/FSKD.2009.553

Rijsbergen, C. J. V. (1979). *Information retrieval* (2nd ed.). Newton, MA, USA: Butterworth-Heinemann.

Rohlf, F. J. (1974). Methods of comparing classifications. *Annual Review of Ecology and Systematics, 5*, 101–113.

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning(emnlp-conll)* (pp. 410–420).

Rosset, S., & Inger, A. (2000, January). Kdd-cup 99: knowledge discovery in a charitable organization's

donor database. *SIGKDD Explorations*, *1*(2), 85–90. doi: 10.1145/846183.846204

Rousseeuw, P. (1987, November). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics.*, *20*(1), 53–65.

Ruiz, C., Menasalvas, E., & Spiliopoulou, M. (2009). C-DenStream: Using domain knowledge on a data stream. In *Proceedings of the 12th international conference on discovery science* (pp. 287–301). Berlin, Heidelberg: Springer-Verlag.

Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2007). C-dbscan: Density-based clustering with constraints. In *Proceedings of the 11th international conference on rough sets, fuzzy sets, data mining and granular computing* (pp. 216–223). Berlin, Heidelberg: Springer-Verlag. doi: http://dx.doi.org/10.1007/978-3-540-72530-5_25

Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2010, November). Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery*, *21*, 345–370. doi: 10.1007/s10618-009-0157-y

Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998, June). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, *2*(2), 169–194. doi: 10.1023/A:1009745219419

Santos, J. M., & Embrechts, M. (2009). On the use of the adjusted rand index as a metric for evaluating supervised classification. In C. Alippi, M. M. Polycarpou, C. G. Panayiotou, & G. Ellinas (Eds.), *Icann (2)* (Vol. 5769, p. 175-184). Springer.

Severien, A. L. (2013). *Scalable distributed real-time clustering for big data streams* (Master's Thesis). Polytechnic University of Catalonia.

Sheikholeslami, G., Chatterjee, S., & Zhang, A. (2000, February). Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, *8*(3-4), 289–304. doi: 10.1007/s007780050009

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. P. L. F. d., & Gama, J. a. (2013, July). Data stream clustering: A survey. *ACM Computing Survey*, *46*(1), 13:1–13:31.

Song, M. J., & Zhang, L. (2008). Comparison of cluster representations from partial second- to full fourth-order cross moments for data stream clustering. In *Proceedings of the 2008 eighth ieee international conference on data mining* (pp. 560–569). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/ICDM.2008.143

Strehl, A. (2002). *Relationship-based clustering and cluster ensembles for high-dimensional data mining* (PhD Thesis). The University of Texas at Austin.

Strehl, A., & Ghosh, J. (2003, March). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617. doi: 10.1162/153244303321897735

Tasoulis, D. K., Ross, G., & Adams, N. M. (2007). Visualising the cluster structure of data streams. In *Proceedings of the 7th international conference on intelligent data analysis* (pp. 81–92). Berlin, Heidelberg: Springer-Verlag.

Tu, L., & Chen, Y. (2009). Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery Data*, *3*(3), 1–27.

Vitter, J. S. (1985, March). Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, *11*(1), 37–57.

Wan, L., Ng, W. K., Dang, X. H., Yu, P. S., & Zhang, K. (2009). Density-based clustering of data streams at multiple resolutions. *ACM Transactions Knowledge Discovery Data*, *3*(3), 1–28.

Wang, W., Yang, J., & Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data

mining. In *Proceedings of the 23rd international conference on very large data bases* (pp. 186–195). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Weka Group Project. (2008). *WEKA*. University of Waikato. (`http://www.cs.waikato.ac.nz/ml/weka/`)

Wu, J., Xiong, H., & Chen, J. (2009). Adapting the right measures for k-means clustering. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 877–886). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/1557019.1557115

Xiaoyun, C., Yufang, M., Yan, Z., & Ping, W. (2008). Gmdbscan: multi-density dbscan cluster based on grid. In *e-business engineering, 2008. icebe'08. ieee international conference on* (pp. 780–783).

Xiong, Z., Chen, R., Zhang, Y., & Zhang, X. (2012). Multi-density dbscan algorithm based on density levels partitioning. *Journal of Information and Computational Science*, *9*(10), 2739–2749.

Yang, C., & Zhou, J. (2006). Hclustream: A novel approach for clustering evolving heterogeneous data stream. In *Data mining workshops, 2006. icdm workshops 2006. sixth ieee international conference on* (p. 682-688).

Yang, D., Rundensteiner, E. A., & Ward, M. O. (2011, October). Summarization and matching of density-based clusters in streaming environments. *Proceedings of the VLDB Endowment*, *5*(2), 121–132.

Yang, Y., Liu, Z., Zhang, J.-p., & Yang, J. (2012). Dynamic density-based clustering algorithm over uncertain data streams. In *9th international conference on fuzzy systems and knowledge discovery (fskd)* (pp. 2664–2670). doi: 10.1109/FSKD.2012.6233800

Yu, Y., Wang, Q., Wang, X., Wang, H., & He, J. (2013). Online clustering for trajectory data stream of moving objects. *Computer Science and Information Systems*, *10*(3), 1293–1317.

Yu, Y.-Q., Huang, T.-Q., Guo, G.-D., & Li, K. (2008). Semi-supervised clustering algorithm for multi-density and complex shape dataset. In *Pattern recognition, 2008. ccpr'08. chinese conference on* (pp. 1–6).

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: an efficient data clustering method for very large databases. In J. Widom (Ed.), *Proceedings of the 1996 acm sigmod international conference on management of data* (pp. 103–114). ACM Press.

Zhang, X., Furtlehner, C., Germain-Renaud, C., & Sebag, M. (2013). Data stream clustering with affinity propagation. *IEEE Transactions on Knowledge and Data Engineering*, *99*(PrePrints), 1. doi: 10.1109/TKDE.2013.146

Zhao, Y., & Karypis, G. (2004, June). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, *55*(3), 311–331. doi: 10.1023/B:MACH.0000027785.44527.d6

Zhou, A., Cao, F., Qian, W., & Jin, C. (2008, May). Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, *15*(2), 181–214. doi: 10.1007/s10115-007-0070-x

Zliobaite, I., Bifet, A., Read, J., Pfahringer, B., & Holmes, G. (2014). Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, *In press*, 1-28. doi: 10.1007/s10994-014-5441-4

AWARDS

- Nominated for The Best Paper Award: International Conference on Data Mining and Applications (ICDMA), Hong Kong, March 2011
- Winner of The Best Paper Presentation Award for Post Graduate Research Excellence Symposium (PGReS), University Of Malaya, September 2011
- 1st Runner up of 3 Minutes Thesis Competition, Faculty Level University of Malaya, July 2013

PUBLICATIONS

Journal Papers

1. **Amineh Amini**, Teh Ying Wah, Hadi Saboohi. "*On Density-based Data Streams Clustering Algorithms: A Survey*". Journal of Computer Science and Technology, Vol. 29, No. 1, January 2014, pp. 116-141.

2. **Amineh Amini**, Hadi Saboohi, Teh Ying Wah, Tutut Herawan. "*A Fast Density-based Clustering Algorithm for Real-Time Internet of Things Stream*". The Scientific World Journal. 2014.

3. **Amineh Amini**, Hadi Saboohi, Teh Ying wah, Tutut Herawan. "*MuDi-Stream: A Multi Density Clustering Algorithm for Evolving Data Stream*". Journal of Network and Computer Applications. Under Review.

4. Shahaboddin Shamshirband, **Amineh Amini**, Nor Badrul Anuar, Miss Laiha Mat Kiah, Teh Ying Wah, Steven Furnell. "*D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks*". Measurement Journal. Vol. 55, September 2014, pp. 212–226.

5. **Amineh Amini**, Teh Ying Wah. "*LeaDen-Stream: A Leader Density-based Clustering Algorithm over Evolving Data Stream*". Journal of Computer and Communications, Vol. 1, No. 5, October 2013, pp. 26-31.

6. Mahmoud Reza Saybani, Teh Ying Wah, **Amineh Amini**, Saeed Reza Aghabozorgi. "*Anomaly Detection and Prediction of Sensors Faults in a Refinery using Data Mining Techniques and Fuzzy Logic*", Journal of Scientific Research and Essays, Vol. 6, No. 27, November 2011, pp 5685-5695.

7. Mahmoud Reza Saybani, Teh Ying Wah, **Amineh Amini**, Saeed Reza Aghabozorgi, and Adel Lahsasna. "*Applications of Support Vector Machines in Oil Refineries: A Survey*", International Journal of Physical Sciences, Vol. 6, No. 27, November 2011, pp 6295-6302.

8. Hadi Saboohi, **Amineh Amini**, and Hassan Abolhassani, "*An Ontology Web Language based Framework for Information Inference from Heterogeneous Data Sources*", Irandoc Scientific Communication Journal, Vol. 4, No. 3, May 2005.

Book Chapter

- **Amineh Amini**, and Teh Ying Wah. "*A Comparative Study of Density-based Clustering Algorithms on Data Streams: Micro-clustering Approaches*", in Ao, Sio Iong; Castillo, Oscar; Huang, Xu (Eds.), Intelligent Control and Innovative Computing, Springer, Vol. 110, February 2012, pp. 275-287.

Conference Papers

1. **Amineh Amini**, Hadi Saboohi, Teh Ying Wah. "*A Hybrid Density-based Clustering Algorithm over Evolving Data Stream with Noise*". 7th Iran Data Mining Conference (IDMC'13), Tehran, Iran, December 2013, pp. 1–9.

2. **Amineh Amini**, Hadi Saboohi, Teh Ying Wah, and Tutut Herawan. "*DMM-Stream: A Density Mini-Micro Clustering Algorithm for Evolving Data Streams*". The First International Conference on Advanced Data and Information Engineering (DaEng'13), Kuala Lumpur, Malaysia, December 2013, pp. 675–682.

3. **Amineh Amini**, Hadi Saboohi, Teh Ying Wah. "*A Multi Density-based Clustering Algorithm for Data Stream with Noise*". IEEE 13th International Conference on Data Mining Workshops (ICDMW'13), Dallas, Texas, December 2013, pp. 1105–1112.

4. **Amineh Amini**, Teh Ying Wah. "*A Detection Approach for Tsunami Warning System Using Mining Data Streams*". International Congress on Natural Sciences and Engineering, Bali, Indonesia. August 2013.

5. **Amineh Amini**, Teh Ying Wah. "*Requirements for Clustering Evolving Data Stream*". 2nd International Conference on Soft Computing and its Applications (ICSCA'2013), Phnom Penh, Cambodia, September 2013, pp. 47-50.

6. **Amineh Amini**, Teh Ying Wah. "*On Density-based Clustering Algorithms over Evolving Data Streams: A Summarization Paradigm*". International Conference on Information Technology and Management Innovation (ICITMI'12), Guangzhou, China, November 2012, pp. 263-266.

7. **Amineh Amini**, Teh Ying Wah. "*Adaptive Density-based Clustering Algorithms for Data Stream Mining*". Third International Conference on Theoretical and Mathematical Foundations of Computer Science (ICTMF'12), Bali, Indonesia, December 2012, pp. 620-624.

8. **Amineh Amini**, Teh Ying Wah, "*DENGRIS-Stream: A Density-Grid based Clustering Algorithm for Evolving Data Streams over Sliding Window*". International Conference on Data Mining and Computer Engineering (ICDMCE'12), Bangkok, Thailand, December 2012, pp. 206-211.

9. **Amineh Amini**, Teh Ying Wah, Mahmoud Reza Saybani, and Saeed Reza Aghabozorgi. "*A Study of Density-Grid based Clustering Algorithms on Data Streams*". The 8th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'11), Shanghai, China, July 2011, pp. 1652-1656.

10. **Amineh Amini**, and Teh Ying Wah, "*Density Micro-Clustering Algorithms on Data Streams: A Review*". The International Conference on Data Mining and

Applications (ICDMA'11) in the International MultiConference of Engineers and Computer Scientists (IMECS), Hong Kong, March 2011, pp. 410-414.

11. Hadi Saboohi, **Amineh Amini**, Tutut Herawan, and Sameem Abdul Kareem, "*Failure Recovery of Composite Semantic Services using Expiration Times*", in Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng'13), Kuala Lumpur, Malaysia, December 2013, pp. 683–690.

12. Saeed Reza Aghabozorgi, Teh Ying Wah, **Amineh Amini**, and Mahmoud Reza Saybani. "*A New Approach to Present Prototypes in Clustering of Time Series*". The 7th International Conference of Data Mining, Las Vegas, USA, July 2011.

13. Hadi Saboohi, **Amineh Amini**, and Hassan Abolhassani, "*Failure Recovery of Composite Semantic Web Services using Replacement Subgraphs*", in Proceedings of the International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, May 2008, pp. 489-493.

14. **Amineh Amini**, Hadi Saboohi, and Nasser Nemat bakhsh, "*A RDF-based Data Integration Framework*", National Electrical Engineering Conference (NEEC), Najafabad, Iran, March 2008 (Persian).

15. Hadi Saboohi, **Amineh Amini**, and Hassan Abolhassani, "*Electronic City Services Management Using Failure Recoverable Composites of Autonomic Distributed Software Components*", The First International e-City Conference, Tehran, Iran, February 2008 (Persian).

Google Scholar Profile

Citation indices	All	Since 2009
Citations	89	88
h-index	5	5
i10-index	3	3

Title	Cited by	Year
A study of density-grid based clustering algorithms on data streams A Amini, TY Wah, MR Saybani, SR Aghabozorgi Sahaf Yazdi Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International ...	23	2011
Failure recovery of composite semantic web services using subgraph replacement H Saboohi, A Amini, H Abolhassani Computer and Communication Engineering, 2008. ICCCE 2008. International ...	19	2008
Density Micro-Clustering Algorithms on Data Streams: A Review A Amini, TY Wah Proceedings of the International MultiConference of Engineers and Computer ...	11	2011
A new approach to present prototypes in clustering of time series SR Aghabozorgi, TY Wah, A Amini, MR Saybani	8	2011
DENGRIS-Stream: A Density-Grid based Clustering Algorithm for Evolving Data Streams over Sliding Window A Amini, TY Wah, MR Saybani, S Aghabozorgi PSRCentre	5	2012
A comparative study of density-based clustering algorithms on data streams: Micro-clustering approaches A Amini, TY Wah Intelligent Control and Innovative Computing, 275-287	5	2012
On Density-Based Data Streams Clustering Algorithms: A Survey A Amini, TY Wah, H Saboohi Journal of Computer Science and Technology 29 (1), 116-141	4	2014
D-FICCA: A Density-based Fuzzy Imperialist Competitive Clustering Algorithm for Intrusion Detection in Wireless Sensor Networks S Shamshirband, A Amini, NB Anuar, MLM Kiah, TY Wah, S Furnell Measurement	2	2014