Density-based Algorithms for Active and Anytime Clustering

Mai Thai Son

München 2014

Density-based Algorithms for Active and Anytime Clustering

Mai Thai Son

Dissertation
an der Fakultät Für Mathematik, Informatik und Statistik,
Institut Für Informatik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Mai Thai Son
aus Vietnam

München, den 09 Sep 2013

Erstgutachter: Prof. Dr. Christian Böhm
Zweitgutachter: Prof. Dr. Xiaowei Xu
Tag der mündlichen Prüfung: 26 Sep 2014

Abstract

Data intensive applications like biology, medicine, and neuroscience require effective and efficient data mining technologies. Advanced data acquisition methods produce a constantly increasing volume and complexity. As a consequence, the need of new data mining technologies to deal with complex data has emerged during the last decades. In this thesis, we focus on the data mining task of clustering in which objects are separated in different groups (clusters) such that objects inside a cluster are more similar than objects in different clusters. Particularly, we consider density-based clustering algorithms and their applications in biomedicine.

The core idea of the density-based clustering algorithm DBSCAN is that each object within a cluster must have a certain number of other objects inside its neighborhood. Compared with other clustering algorithms, DBSCAN has many attractive benefits, e.g., it can detect clusters with arbitrary shape and is robust to outliers, etc. Thus, DBSCAN has attracted a lot of research interest during the last decades with many extensions and applications. In the first part of this thesis, we aim at developing new algorithms based on the DBSCAN paradigm to deal with the new challenges of complex data, particularly expensive distance measures and incomplete availability of the distance matrix.

Like many other clustering algorithms, DBSCAN suffers from poor performance when facing expensive distance measures for complex data. To tackle this problem, we propose a new algorithm based on the DBSCAN paradigm, called Anytime Density-based Clustering (A-DBSCAN), that works in an anytime scheme: in contrast to the original batch scheme of DBSCAN, the algorithm A-DBSCAN first produces a quick approximation of the clustering result and then continuously refines the result during the further run. Experts can interrupt the algorithm, examine the results, and choose between (1) stopping the algorithm at any time whenever they are satisfied with the result to save runtime and (2) continuing the algorithm to achieve better results. Such kind of anytime scheme has been proven in the literature as a very useful technique when dealing with

time consuming problems. We also introduced an extended version of A-DBSCAN called A-DBSCAN-XS which is more efficient and effective than A-DBSCAN when dealing with expensive distance measures.

Since DBSCAN relies on the cardinality of the neighborhood of objects, it requires the full distance matrix to perform. For complex data, these distances are usually expensive, time consuming or even impossible to acquire due to high cost, high time complexity, noisy and missing data, etc. Motivated by these potential difficulties of acquiring the distances among objects, we propose another approach for DBSCAN, called Active Density-based Clustering (Act-DBSCAN). Given a budget limitation B, Act-DBSCAN is only allowed to use up to B pairwise distances ideally to produce the same result as if it has the entire distance matrix at hand. The general idea of Act-DBSCAN is that it actively selects the most promising pairs of objects to calculate the distances between them and tries to approximate as much as possible the desired clustering result with each distance calculation. This scheme provides an efficient way to reduce the total cost needed to perform the clustering. Thus it limits the potential weakness of DBSCAN when dealing with the distance sparseness problem of complex data.

As a fundamental data clustering algorithm, density-based clustering has many applications in diverse fields. In the second part of this thesis, we focus on an application of density-based clustering in neuroscience: the segmentation of the white matter fiber tracts in human brain acquired from Diffusion Tensor Imaging (DTI). We propose a model to evaluate the similarity between two fibers as a combination of structural similarity and connectivity-related similarity of fiber tracts. Various distance measure techniques from fields like time-sequence mining are adapted to calculate the structural similarity of fibers. Density-based clustering is used as the segmentation algorithm. We show how A-DBSCAN and A-DBSCAN-XS are used as novel solutions for the segmentation of massive fiber datasets and provide unique features to assist experts during the fiber segmentation process.

Zusammenfassung

Datenintensive Anwendungen wie Biologie, Medizin und Neurowissenschaften erfordern effektive und effiziente Data-Mining-Technologien. Erweiterte Methoden der Datenerfassung erzeugen stetig wachsende Datenmengen und Komplexität. In den letzten Jahrzehnten hat sich daher ein Bedarf an neuen Data-Mining-Technologien für komplexe Daten ergeben. In dieser Arbeit konzentrieren wir uns auf die Data-Mining-Aufgabe des Clusterings, in der Objekte in verschiedenen Gruppen (Cluster) getrennt werden, so dass Objekte in einem Cluster untereinander viel ähnlicher sind als Objekte in verschiedenen Clustern. Insbesondere betrachten wir dichtebasierte Clustering-Algorithmen und ihre Anwendungen in der Biomedizin.

Der Kerngedanke des dichtebasierten Clustering-Algorithmus DBSCAN ist, dass jedes Objekt in einem Cluster eine bestimmte Anzahl von anderen Objekten in seiner Nachbarschaft haben muss. Im Vergleich mit anderen Clustering-Algorithmen hat DBSCAN viele attraktive Vorteile, zum Beispiel kann es Cluster mit beliebiger Form erkennen und ist robust gegenüber Ausreißern. So hat DBSCAN in den letzten Jahrzehnten großes Forschungsinteresse mit vielen Erweiterungen und Anwendungen auf sich gezogen. Im ersten Teil dieser Arbeit wollen wir auf die Entwicklung neuer Algorithmen eingehen, die auf dem DBSCAN Paradigma basieren, um mit den neuen Herausforderungen der komplexen Daten, insbesondere teurer Abstandsmaße und unvollständiger Verfügbarkeit der Distanzmatrix umzugehen.

Wie viele andere Clustering-Algorithmen leidet DBSCAN an schlechter Performanz, wenn es teuren Abstandsmaßen für komplexe Daten gegenüber steht. Um dieses Problem zu lösen, schlagen wir einen neuen Algorithmus vor, der auf dem DBSCAN Paradigma basiert, genannt Anytime Density-based Clustering (A-DBSCAN), der mit einem Anytime Schema funktioniert. Im Gegensatz zu dem ursprünglichen Schema DBSCAN, erzeugt der Algorithmus A-DBSCAN zuerst eine schnelle Annäherung des Clusterings-Ergebnisses und verfeinert dann kon-

tinuierlich das Ergebnis im weiteren Verlauf. Experten können den Algorithmus unterbrechen, die Ergebnisse prüfen und wählen zwischen (1) Anhalten des Algorithmus zu jeder Zeit, wann immer sie mit dem Ergebnis zufrieden sind, um Laufzeit sparen und (2) Fortsetzen des Algorithmus, um bessere Ergebnisse zu erzielen. Eine solche Art eines "Anytime Schemas" ist in der Literatur als eine sehr nützliche Technik erprobt, wenn zeitaufwendige Problemen anfallen. Wir stellen auch eine erweiterte Version von A-DBSCAN als A-DBSCAN-XS vor, die effizienter und effektiver als A-DBSCAN beim Umgang mit teuren Abstandsmaßen ist.

Da DBSCAN auf der Kardinalität der Nachbarschaftsobjekte beruht, ist es notwendig, die volle Distanzmatrix auszurechen. Für komplexe Daten sind diese Distanzen in der Regel teuer, zeitaufwendig oder sogar unmöglich zu errechnen, aufgrund der hohen Kosten, einer hohen Zeitkomplexität oder verrauschten und fehlende Daten. Motiviert durch diese möglichen Schwierigkeiten der Berechnung von Entfernungen zwischen Objekten, schlagen wir einen anderen Ansatz für DBSCAN vor, namentlich Active Density-based Clustering (Act-DBSCAN). Bei einer Budgetbegrenzung B, darf Act-DBSCAN nur bis zu B ideale paarweise Distanzen verwenden, um das gleiche Ergebnis zu produzieren, wie wenn es die gesamte Distanzmatrix zur Hand hätte. Die allgemeine Idee von Act-DBSCAN ist, dass es aktiv die erfolgversprechendsten Paare von Objekten wählt, um die Abstände zwischen ihnen zu berechnen, und versucht, sich so viel wie möglich dem gewünschten Clustering mit jeder Abstandsberechnung zu nähern. Dieses Schema bietet eine effiziente Möglichkeit, die Gesamtkosten der Durchführung des Clusterings zu reduzieren. So schränkt sie die potenzielle Schwäche des DBSCAN beim Umgang mit dem Distance Sparseness Problem von komplexen Daten ein.

Als fundamentaler Clustering-Algorithmus, hat dichte-basiertes Clustering viele Anwendungen in den unterschiedlichen Bereichen. Im zweiten Teil dieser Arbeit konzentrieren wir uns auf eine Anwendung des dichte-basierten Clusterings in den Neurowissenschaften: Die Segmentierung der weißen Substanz bei Faserbahnen im menschlichen Gehirn, die vom Diffusion Tensor Imaging (DTI) erfasst werden. Wir schlagen ein Modell vor, um die Ähnlichkeit zwischen zwei Fasern als einer Kombination von struktureller und konnektivitätsbezogener Ähnlichkeit von Faserbahnen zu beurteilen. Verschiedene Abstandsmaße aus Bereichen wie dem Time-Sequence Mining werden angepasst, um die strukturelle Ähnlichkeit von Fasern zu berechnen. Dichte-basiertes Clustering wird als Segmentierungsalgorithmus verwendet. Wir zeigen, wie A-DBSCAN und A-DBSCAN-XS als neuartige Lösungen für die Segmentierung von sehr großen Faserdatensätzen verwendet

werden, und bieten innovative Funktionen, um Experten während des Fasersegmentierungsprozesses zu unterstützen.

Contents

Abstract v

Zusammenfassung vii

Table of Contents x

I Introduction 1

1 Introduction 3

1.1 Density-based Clustering 4

1.2 Challenges of Complex Data 4

1.3 Contributions and Thesis Outline 6

2 Density-based Clustering Algorithms 9

2.1 Introduction 10

2.2 The Algorithm DBSCAN 13

2.3 The Algorithm OPTICS 16

2.4 Other Algorithms 18

2.5 Applications of DBSCAN 20

2.6 Extensions of DBSCAN 23

2.6.1 Parameter Optimization 23

2.6.2 Clustering with Varying Densities 25

2.6.3 Speeding up the Algorithm 28

2.6.4 Parallel and Distributed Clustering . . . 34
2.6.5 Incremental Clustering . . . 37
2.6.6 Subspace Clustering . . . 38
2.6.7 Semi-supervised Clustering . . . 40
2.6.8 Clustering Complex Data . . . 42
2.6.9 Other Algorithms . . . 47
2.7 Conclusions . . . 48

3 Preliminaries **51**
3.1 Cluster Validation . . . 51
3.1.1 Information Theoretic Validation Techniques . . . 52
3.1.2 Other Validation Techniques . . . 53
3.2 Lower bounding Distance . . . 54
3.2.1 Piecewise Aggregate Approximation . . . 54
3.2.2 Convergent Bounds on the Euclidean Distance . . . 55
3.3 Discrete Wavelet Transform . . . 56
3.3.1 Haar Wavelet Transform . . . 56
3.3.2 Applications of Haar Wavelet Transform . . . 57

II Density-based Clustering of Complex Data **59**

4 Anytime and Active Clustering **61**
4.1 Anytime Clustering . . . 61
4.1.1 Anytime Algorithms . . . 62
4.1.2 Applications of Anytime Algorithms . . . 66
4.1.3 Anytime Clustering . . . 71
4.1.4 Conclusions . . . 75
4.2 Active Clustering . . . 76
4.2.1 Active Learning . . . 76
4.2.2 Applications of Active Algorithms . . . 77

4.2.3 Active Clustering 78
4.2.4 Conclusions 82

5 Anytime Density-based Clustering 85
5.1 Introduction 86
5.2 Backgrounds 88
5.2.1 Anytime Clustering 88
5.2.2 The Algorithm DBSCAN 89
5.3 Anytime Density-based Clustering 90
5.4 Extended A-DBSCAN 97
5.4.1 The Xseedlist 97
5.4.2 The Algorithm A-DBSCAN-XS 98
5.5 Similarity Measure and Lower bounding 102
5.6 Experiments 103
5.6.1 Evaluation Methodology 103
5.6.2 Performance Evaluation 105
5.6.3 Parameter Analysis 109
5.7 Discussions 112
5.8 Conclusions 115

6 Active Density-based Clustering 117
6.1 Introduction 118
6.2 Backgrounds 121
6.2.1 Density-based Clustering 121
6.2.2 Active Clustering 123
6.3 Active Density-based Clustering 123
6.3.1 The Algorithm Act-DBSCAN 123
6.3.2 Monotonicity Property 127
6.3.3 Edge Probability 129
6.3.4 Core Object Probability 134

6.3.5 Edge Score and Comparison . . . 134
6.3.6 Reduction Property . . . 135
6.3.7 Algorithm Analysis . . . 136
6.4 Similarity Measures . . . 137
6.5 Experiments . . . 138
6.5.1 Algorithms and Comparison Criteria . . . 138
6.5.2 Performance Analysis . . . 139
6.5.3 Parameter Analysis . . . 140
6.5.4 Comparison with Spectral Clustering . . . 143
6.5.5 How many similarities do other algorithms use? . . . 145
6.5.6 Runtime Analysis . . . 146
6.6 Discussions . . . 147
6.7 Conclusions . . . 150

III Application for Fiber Clustering 151

7 Background on Fiber Segmentation 153
7.1 Diffuse Tensor Imaging . . . 153
7.2 Fiber Similarity Measures . . . 155
7.3 Fiber Segmentation Algorithms . . . 156
7.4 Conclusions . . . 157

8 Advantage Fiber Similarity Measure Techniques 159
8.1 Introduction . . . 160
8.2 Fiber Similarity Measure . . . 161
8.2.1 Shape Similarity of Two Fibers . . . 162
8.2.2 Lower Bounding Distance for WLCS . . . 166
8.2.3 Connection Similarity of Two Fibers . . . 167
8.2.4 Unified Fiber Similarity Measure . . . 168
8.2.5 Other Important Characteristics of Fiber Similarity . . . 169

8.3 Fiber Segmentation . . . 170
8.4 Empirical Evaluation . . . 172
8.4.1 Effectiveness of The Shape Similarity Measures . . . 172
8.4.2 Effectiveness of the Unified Similarity Measure SIM . . . 175
8.4.3 Characteristics of SIM . . . 176
8.4.4 Efficiency of the Unified Similarity Measure . . . 180
8.5 Discussions . . . 181
8.6 Conclusions . . . 183

9 Advantage Fiber Clustering Techniques 185
9.1 Introduction . . . 186
9.2 Fiber Similarity Measure . . . 187
9.3 Empirical Evaluation . . . 190
9.4 Conclusions . . . 196

IV Summary 197

10 Summary 199
10.1 Conclusions . . . 199
10.2 Future Works . . . 202

Bibliography 203

List of Figures 228

List of Tables 235

Publications 239

Acknowledgements 241

Curriculum Vitae 243

Part I

Introduction

Chapter 1

Introduction

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [87]. The KDD process consists of several steps as illustrated in Figure 1.1.

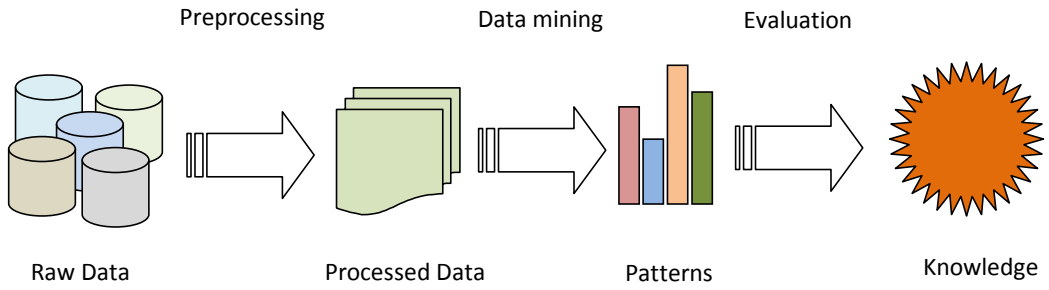

Figure 1.1: Knowledge Discovery in Databases (KDD) process.

In the beginning of the KDD process, raw data are collected from different data sources. These data are usually not in a good form as they may contain noise, missing entries, inconsistencies, etc. From these data, most relevant data are then selected and preprocessed, e.g., noise removal, by the KDD process in order to increase data quality to support the subsequent processes. In the next step, data mining, as the key component of the KDD process, is performed to extract previously unknown and useful patterns from the data using automatic or semi-automatic algorithms. At the end of the KDD process, these patterns are evaluated in order to extract useful knowledge from data.

In this thesis, we focus on data clustering [121], a central task of the data

mining step, in which objects are separated into different groups (clusters) so that the ones inside a cluster are more similar than those in different clusters. In particular, we aim at the density-based clustering algorithm DBSCAN [83], a fundamental data clustering technique proposed in the literature, its extensions and its applications in various fields.

1.1 Density-based Clustering

In density-based clustering, clusters are regarded as areas of high object density in the data space separated by areas of lower object density. The algorithm DBSCAN [83] formalizes a density notion for clustering using two parameters: ϵ denoting a volume and μ denoting a minimal number of objects. An object belongs to a cluster if it has at least μ objects inside its ϵ-neighborhood. Compared with other clustering algorithms, DBSCAN has several attractive benefits, e.g., it can detect clusters with arbitrary shapes and is robust to outliers. Moreover, the total number of clusters does not need to be specified beforehand. Thus, DBSCAN has attracted much research interest during the last decades with many extensions and applications in various fields, e.g., [45, 160, 228, 157, 16, 272, 32].

Among many different extensions of DBSCAN, density-based clustering algorithms for complex data have become an emerging research topic with many proposed techniques in the literature recently, e.g., [84, 228, 272, 284, 86, 206, 157, 158]. However, the rapid growth of advanced data acquisition methods in many fields, e.g., medicine, biology and environment, has continuously produced a large amount of data with increasing volume and complexity, e.g., stream, time-series, graph or uncertain data. As a consequence, many challenges have been constantly arisen in order to provide efficient and effective data mining algorithms to extract knowledge from these data, in particular density-based clustering algorithms.

In the following Section, we address some challenges of complex data for the density-based clustering algorithm DBSCAN.

1.2 Challenges of Complex Data

Since DBSCAN relies on the pairwise similarities (dissimilarities or distances) among objects to operate, it suffers from two important problems including expensive similarity measures and similarity sparseness as described below:

- **Expensive Simmilarity Measures.** For complex data, there exist many effective (dis)similarity measures between objects such as Dynamic Time Warping [132] and Longest Common Subsequence [261]. However, most of these similarity measures have high time complexity (usually quadratic time complexity) which obviously becomes a bottle neck in many applications, e.g., the clustering of start light curves in [300]. A star light curve is the measurement of light intensity of a celestial object or region as a function of time. A light curve can be used to estimate the rotation period of a planet or comet nucleus in planetology or to discover supernovas in astronomy, etc. For these tasks, clustering is commonly used to support the analysis of the digital star light curves. In [300], Dynamic Time Warping (DTW) is used as an effective similarity measure for the clustering process. However, it consumes around 127 days to cluster a mere 9236 curves due to the quadratic time complexity of DTW [300]. Obviously, such the runtime bottle neck is undesired, especially in real time and interactive systems [303]. Thus, it poses an important challenge: how to improve the performance of clustering algorithms when coping with these expensive similarity measures. Among various existing approaches for density-based clustering algorithms, only some of them are designed to handle this problem, e.g., [45, 44].

- **Similarity Sparseness.** In many applications, obtaining all similarities among objects is a nontrivial process since they may be difficult or even unavailable to obtain. For example, in transportation monitoring and control systems, GPS is usually used to collect the positions of vehicles, people, airplanes, etc. Then, clustering algorithms are used to discover common or unusual movement patterns as in [201, 93]. However, in many cases, the GPS signals may be very noisy or may be lost due to bad weather, obstacles, etc. Thus, measuring the similarities among moving object trajectories becomes very hard or even infeasible. Another example comes from the task of clustering of photos acquired from a wearable camera in [275], which plays an important role in a variety of applications, e.g., improving life quality for Alzheimer's patients or summarizing personal memories. Since the photos are collected in an arbitrary manner, assessing the similarities among these photo is out of the capability of existing automatic image processing techniques. Consequently, human annotators must be involved to rate the similarities among photos. It therefore makes the clustering an expensive process in terms of both time and money. The potential difficulties of acquiring the similarities among all objects raise another important challenge:

how to perform clustering without accessing the whole similarity matrix in order to reduce the potential costs or to cope with the unavailability of pairwise similarities. Though there are many density-based clustering algorithms proposed in the literature [155, 9], to the best of our knowledge, none of them are designed to deal with this challenge.

Recently, interactive exploring of data has become a significant feature in many data mining algorithms, especially for complex data, e.g., [234, 33], since it allows domain experts to be involved into the clustering process to improve the performance and outcome. However, throughout the literature review, all the existing extensions of DBSCAN only work in a batch scheme. They produce a single result at the end and do not allow user interaction during their runtime. Providing an interactive extension of DBSCAN, therefore, is another challenge and is extremely useful for many applications, e.g., the segmentation of white matter structure in human brain [55], characters recognition [145] or image clustering [33].

1.3 Contributions and Thesis Outline

In this thesis, we aim at providing efficient and effective density-based clustering algorithms for complex data and their applications for the task of segmentation of white matter structure in human brain in the neuroscience field. In summary, this thesis is organized as follows.

Part 1: Introduction. In this first part of the thesis, we briefly introduce our research focuses and present some backgrounds of our works.

- In Chapter 1, we describe about the KDD process and explain our research focuses.
- In Chapter 2, we present a literature survey on density-based clustering algorithms and their applications in the literature. In particular, we focus on algorithms that follow the paradigm of the density-based clustering algorithm DBSCAN. This work provides a comprehensive review about the evolvement of density-based clustering algorithms during the last decades and thus could significantly contribute to the evolvement of the clustering field. We note that, although there exist several surveys about density-based clustering in the literature, they aim at general density-based clustering algorithms and only cover a small set of existing algorithms.

- In Chapter 3, we illustrate some related backgrounds, e.g., cluster validation techniques, lower bounding functions, and the Haar wavelet transformation technique.

Part 2: Density-based Clustering of Complex Data. In this part of the thesis, we focus on the development of efficient and effective density-based clustering algorithms for complex data.

- In Chapter 4, a literature survey about anytime and active clustering, two recent emerging researches in data clustering, is presented. Based on of our knowledge, there is no other survey about these clustering techniques proposed in the literature so far.
- In Chapter 5, we propose a new approach for density-based clustering called anytime density-based clustering. In contrast to other clustering algorithms, our proposed algorithms, called A-DBSCAN and A-DBSCAN-XS, exploit a sequence of lower bounding distances of the similarity measure to become anytime algorithms. As anytime algorithms, they can be interrupted at anytime to provide an intermediate result and then resumed to search for better results. This anytime scheme provides a very useful way to cope with high time complexity of the similarity measures for complex data and allows user interaction during the clustering process. As far as we know, A-DBSCAN and A-DBSCAN-XS are the first anytime algorithms for density-based clustering proposed in the literature.
- In Chapter 6, we aim at dealing with data sparseness problem described above. Our proposed algorithm, named active density-based clustering (Act-DBSCAN), is capable to provide a desired clustering result without the availability of the full similarity matrix. By actively choosing which pairwise similarities are most important for constructing the clusters, Act-DBSCAN can only use a small number of pairwise similarities to produce the same result as if it had the entire distance matrix at hand. Thus, unlike other algorithms, Act-DBSCAN is able to work quite well under a limited *budget* constraint, e.g., time or money. It can also be able to cope with the unavailability of pairwise similarities. To the best of our knowledge, Act-DBSCAN is the first active algorithm for density-based clustering proposed in the literature.

Part 3: Application for Fiber Clustering. In this part of the thesis, an application of density-based clustering is presented. In particular, we focus on the

problem of segmenting the white matter structure in human brain using Diffusion Tensor Imaging (DTI) technique.

- In Chapter 7, some backgrounds about Diffusion Tensor Imaging (DTI) is introduced. Moreover, several literature researches about fiber similarity measures and fiber clustering techniques are involved.
- In Chapter 8, we focus on developing an efficient and effective similarity model for density-based fiber clustering algorithms. In contrast to other model, our proposed similarity model combines both the shape and the connectivity similarity of fibers to enhance the efficacy. We also propose several novel techniques to measure the shape similarity of fibers. Compared with existing fiber similarity measures, our proposed model provides a more efficient and effective similarity measure for fibers, especially when dealing with noisy and spurious fibers.
- In Chapter 9, we show how anytime density-based clustering algorithms like A-DBSCAN and A-DBSCAN-XS can be used as a novel solution for the segmentation of massive fiber datasets and for providing unique features to assist experts during the fiber segmentation process.

Part 4: Summary. This last part contains the summarization of this thesis as well as some future researches.

- In Chapter 10, we sum up all our contributions in this thesis and discuss some future researches.

Chapter 2

Density-based Clustering Algorithms

Density-based clustering is one of the most common techniques for data clustering and constantly attracts numerous research efforts in many fields. During the last decades, many density-based clustering algorithms have been proposed in the literature including applications of density-based clustering, extensions for complex data and complex tasks, enhancements of existing techniques, etc. Therefore, comprehensive reviews for density-based clustering algorithms are necessary to draw deep insights into the research field and thus could significantly contribute to the development of the field.

Though there are many surveys on density-based clustering algorithms proposed in the literature, most of them are generic surveys that focus on many different kinds of density-based algorithms and only cover small sets of existing techniques. Currently, density-based clustering algorithms have evolved very far from the reach of any existing surveys or text books with hundreds of algorithms proposed in the literature during the last decades.

In this Chapter, we provide a comprehensive literature review on density-based clustering algorithms. In contrast to other works, our survey particularly focuses on the density-based clustering algorithms that follow the paradigm of DBSCAN [83]. Moreover, our survey covers a wide variety of proposed algorithms in the literature including extensions and applications of these algorithms in many different fields such as physic, medicine and transportation.

Publications. Parts of the material presented in this Chapter have been published in [181]. The detailed information are described as follows:

- Son T. Mai. Density-based Clustering: A Comprehensive Survey. Technical Report, University of Munich, 2013.

  In this work, S.T.M. did the major part including the literature review, experiments and paper writing.

2.1 Introduction

Clustering is the task of assigning unlabeled objects into groups called clusters such that the similarity of objects within a group is maximized, and the similarity of objects between different groups is minimized. It plays a vital role for statistical data analysis in many fields including data mining, machine learning, pattern recognition, image analysis, information retrieval, etc. [9]. During the past decades, thousands of clustering algorithms have been proposed in the literature from many different fields [106, 9]. These clustering algorithms can be roughly classified into different groups including: hierarchical clustering algorithms such as Single-Link, Average-Link and Complete-Link methods, etc. [106], partitional clustering algorithms such as k-Means, k-Medoids, EM clustering, k-Harmonic Means, etc. [121], density-based clustering algorithms such as DBSCAN, DENCLUE, OPTICS, etc. [106], grid-based clustering algorithms such as STING, WaveCluster, etc. [106], spectral clustering algorithms [263], and many other clustering algorithms such as Affinity Propagation (AP) [92].

Data clustering surveys. Since there are a vast amount of clustering algorithms proposed in the literature in term of both diversity and quantity, many research efforts are spent to summarize these clustering techniques in order to provide more comprehensive reviews about the field.

Metaheuristic clustering algorithms are the main research topic in [67] and [117]. The surveys proposed by Kriegel et al. [156] and Parson et al. [212] focus on the clustering of high-dimensional data including subspace clustering, projected clustering, pattern-based clustering and correlation clustering algorithms. Moise et al. [193] proposed an interesting work aims at experimental evaluation and analysis of subspace and projected clustering algorithms. Luxburg [263] provided a comprehensive tutorial about spectral clustering algorithms and their nature and characteristics. Filippone et al. [88] provided a survey about kernel and spectral methods for clustering. Another work proposed by Kriegel et al. [155] briefly focused on major density-based clustering algorithms proposed in the lit-

erature. An interesting tutorial from Müller et al. [199] concentrated on multiple clusterings, an emerging research field in data clustering. In [70], Davidson et al. provided a great survey about clustering with instance level constraints. A survey of clustering methods for wireless and mobile networks is provided in [2, 289]. The work of Liao [171] focused on clustering algorithms for time-series data. The comparative study of twelve model-based document clustering algorithms is the main focus of [295]. There exist in the literature many other interesting surveys about generic data clustering techniques, e.g., [121, 29, 9, 281, 122]. Among these works, one of the most interesting works proposed recently is the text book from Aggarwal and Reddy [9] which provides very comprehensive studies about different approaches for data clustering including semi-supervised clustering algorithms, cluster ensembles, alternative clusterings, interactive clustering, clustering high-dimensional data, big data, stream data, biological data, etc. Interested readers please refer to these surveys for more details.

However, it is important to note that, the data clustering field has evolved very far beyond the capability of any text books or surveys proposed in the literature. Therefore, more and more research efforts are still constantly required in order to provide more systematic and comprehensive surveys about the field.

Density-based clustering. Many clustering algorithms, e.g., k-Means, implic-

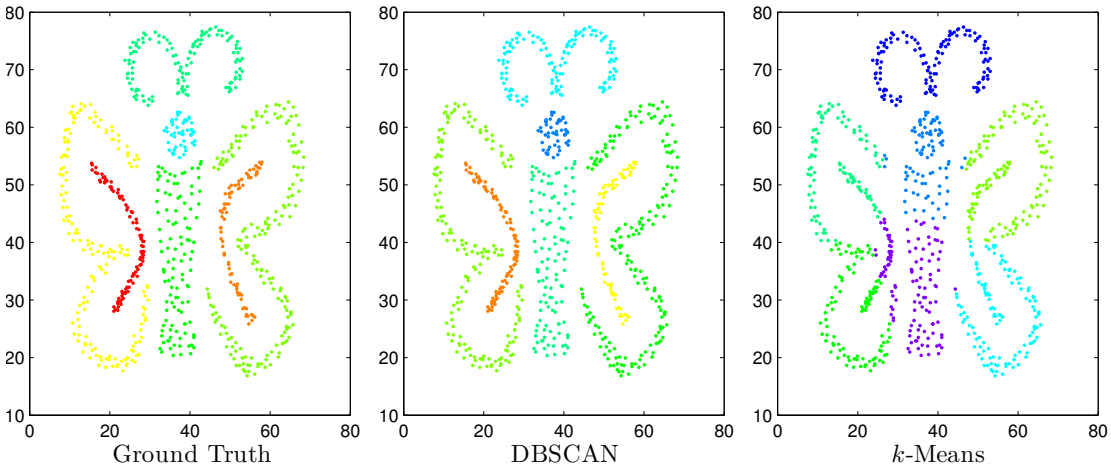

Figure 2.1: The clustering results on a toy example. Due to its ability of detecting clusters with arbitrary shapes, DBSCAN can group data exactly as the ground truth. The traditional algorithm k-Means however cannot group data correctly since it can detect spherical clusters only.

itly or explicitly assume that data are generated from a probability distribution of a given type, e.g., from a mixture of k Gaussian distributions. These clustering algorithms thus are limited to spherical clusters and are unable to deal with data which contain nonspherical shape clusters [9]. In density-based clustering, clusters are regarded as areas of high object density in the data space separated by areas of lower object density. This notion thus helps density-based clustering algorithms to be able to detect clusters with arbitrary shapes by following dense connected areas. Figure 2.1 shows the clustering results of the density-based clustering algorithm DBSCAN [83] and the traditional clustering algorithm k-Means [121] on a toy example. Due to its ability of detecting clusters with arbitrary shapes, DBSCAN can group data exactly as the ground truth. However, the traditional algorithm k-Means cannot group data correctly since it can only detect spherical clusters.

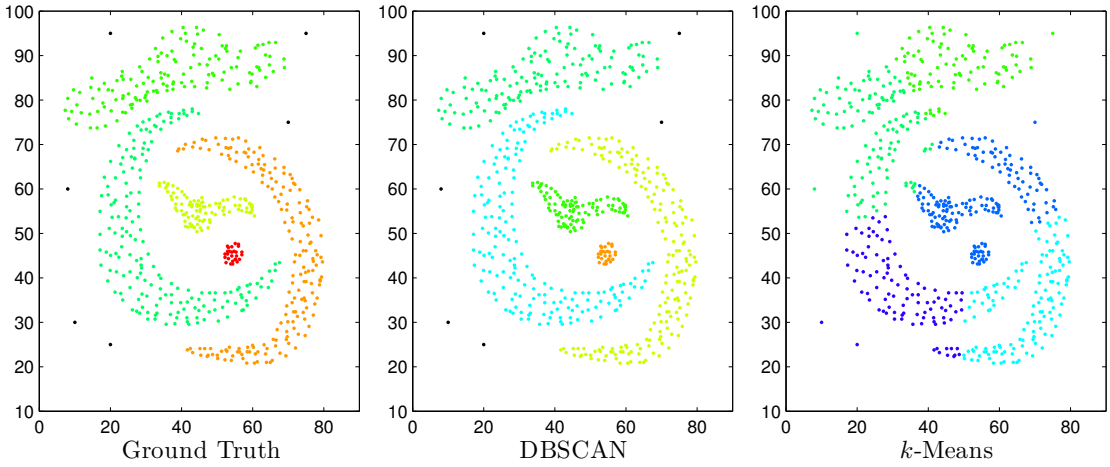

Figure 2.2: The clustering results on a toy example. Outliers are drawn in black. While density-based clustering algorithm DBSCAN can detect those outliers, traditional clustering algorithm k-Means is unable to classify those outliers.

In contrast to many other traditional clustering algorithms, density-based clustering algorithms have capability to deal with outliers. In density-based clustering, outliers are regarded as objects which belong to sparse areas and thus cause an intuition that they are generated from different mechanisms compared with other objects. Figure 2.2 shows the clustering result acquired by the algorithm DBSCAN [83] where outliers are represented by black dots. Traditional clustering algorithm k-Means, however, is unable to classify these outliers.

Another advantage of density-based clustering compared with other traditional clustering techniques is that density-based clustering algorithms do not need the

number of clusters k to be specified beforehand. It is a significant advantage when dealing with complex datasets where determining the number of clusters beforehand is a non-trivial task.

Since the first density-based clustering algorithm DBSCAN [83] was proposed, density-based clustering algorithms have attracted considerable research efforts due to their many attractive benefits, e.g., robustness again noise, the ability to detect arbitrarily-shaped clusters described above. There exist in the literature many density-based clustering algorithms follow different density notions, e.g., the cardinality of neighborhood of an object [83], the influence of an object in its neighborhood [111], and different research directions, e.g., subspace clustering [160], network clustering [284], data stream clustering [61]. Among them, the density-based notion of DBSCAN is perhaps one of the most successful paradigms. In the literature, there exist many algorithms that have been proposed based on the DBSCAN paradigm, e.g., GDBSCAN [82], SUBCLU [160].

Contents. Though there are many surveys on density-based clustering algorithms proposed in the literature, e.g., [155, 9]. Most of them are generic surveys that focus on various kinds of density-based algorithms and only cover small sets of existing techniques. In this Chapter, we provide a comprehensive literature survey for density-based clustering algorithms. In particular, we focus on algorithms which closely follow the paradigm of DBSCAN [83]. The rest of this Chapter is organized as follows. In Section 2.2, we briefly describe the density-based clustering algorithm DBSCAN, a fundamental data clustering technique. In Section 2.3, we briefly describe the algorithm OPTICS, a hierarchical extension of DBSCAN. Other related algorithms are described in Section 2.4. Section 2.5 focuses on extensions of DBSCAN proposed in the literature. Finally, conclusions and discussions are given out in Section 2.7.

2.2 The Algorithm DBSCAN

In [83], Ester et al. proposed the first and perhaps the most well-known density-based clustering algorithm called DBSCAN. DBSCAN formalizes a density notion for clustering by measuring the cardinality of the neighborhood of each object. An object belongs to a cluster if it has enough objects inside its neighborhood. During the past decades, DBSCAN has attracted many research efforts, and thus many extensions of DBSCAN have been proposed in the literature, e.g., [45, 44, 160, 38, 228, 82, 282, 157, 16, 165, 272, 69, 32].

Given a set of objects O which contains N objects, a distance function $d : O \times O \to R$ and two parameters $\epsilon \in R^+$ and $\mu \in N^+$.

Definition 1 *(ϵ-neighborhood) The ϵ-neighborhood of $p \in O$, denoted as $N_\epsilon(p)$, is defined by $N_\epsilon(p) = \{q \in O | d(p,q) \leq \epsilon\}$.*

Each object in O is classified either as core object, border object or noise object depending on its neighborhood. An object p is a core object if it has more than μ objects inside its ϵ-neighborhood. If p has less than μ objects inside its ϵ-neighborhood and none of its neighbors are core objects, then p is classified as noise object or outlier. Otherwise, p is called a border object.

Definition 2 *(Core object property) An object $p \in O$ is a:*

1. *Core object, denoted as $core(p)$, iff $|N_\epsilon(p)| \geq \mu$.*
2. *Border object, denoted as $border(p)$, iff $|N_\epsilon(p)| < \mu$ and $\exists q \in N_\epsilon(p) : |N_\epsilon(q)| \geq \mu$*
3. *Noise object, denoted as $noise(p)$, iff it is not a core object or a border object.*

An object q is density-reachable from object $p \in O$ if p is a core object and q lies inside the ϵ-neighborhood of p. Note that, object q is density-reachable from object p does not mean that object p is also density-reachable from object q.

Definition 3 *(Directly density-reachable) An object $q \in O$ is directly density-reachable from object $p \in O$, denoted as $p \triangleright q$, iff $|N_\epsilon(p)| \geq \mu$ and $q \in N_\epsilon(p)$.*

Two objects p and q are density-connected if there exists a chain of density-reachable core objects x_i so that p is density-reachable from x_i and q is density-reachable from x_i. Note that, p and q are not necessary core objects.

Definition 4 *(Density-connected) Two object p and $q \in O$ are density-connected, denoted as $p \bowtie q$, iff there exists a sequence $(x_1, \ldots, x_m)$ of objects so that $\forall x_i : |N_\epsilon(x_i)| \geq \mu$ and $p \triangleleft x_1 \triangleleft \cdots \triangleright x_m \triangleright q$.*

A cluster is defined as a maximal set of density-connected objects and is composed of core objects and border objects. In DBSCAN, a border object could belong to several clusters depending on the order of objects. A noise object does not belong to any clusters and is called outlier.

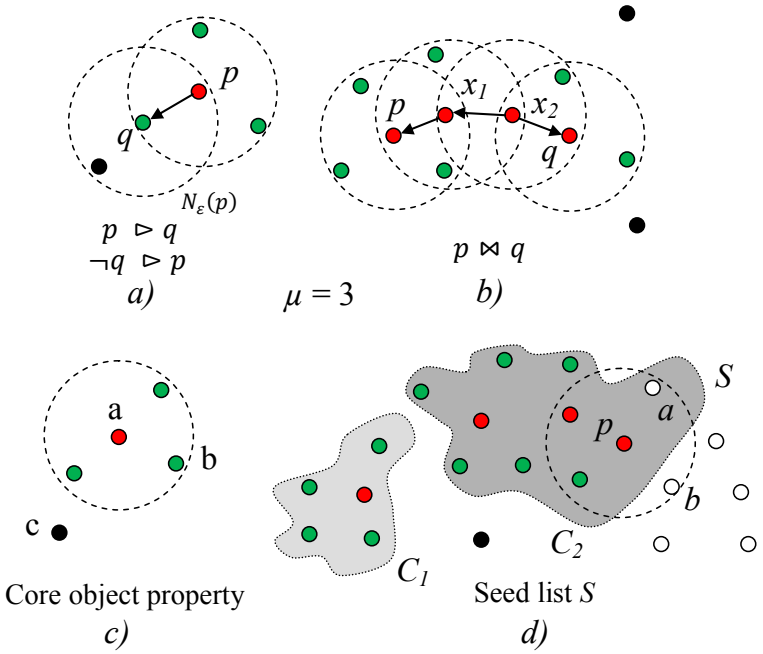

Figure 2.3: The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object; (d) the seed list S for cluster expansion. DBSCAN is currently constructing the cluster C_2. Object p is extracted from S and examined. Object a and b which lie inside the ϵ-neighborhood of p are not processed and thus are inserted into S.

Definition 5 *(Cluster) A subset $C \subseteq O$ is called a cluster iff the two following conditions hold:*

1. *Maximality:* $\forall p \in O, \forall q \in O \setminus C : \neg p \bowtie q$
2. *Connectivity:* $\forall p, q \in C : p \bowtie q$

DBSCAN uses a data structure called the seed list S which contains a set of seed objects for cluster expansion. To construct a cluster, DBSCAN randomly selects an unprocessed object and puts it into the empty seedlist S as an initialization. Then, it continuously extracts an object p from S and performs the ϵ-range query on p to find objects which are directly-reachable from p and inserts them into S if they are not processed so far. When the seed list S is empty, the cluster construction is complete and the construction for a new cluster begins.

The whole expansion process is repeated until all objects are labeled. Interested readers please refer to [83] for more details.

The time complexity of DBSCAN is $O(N^2)$ where N is the total number of objects. When an index structure such as R-Tree [26] is used to speed up the range-query processing, the time complexity of DBSCAN becomes $O(NlogN)$. It is important to note that, the time complexity of similarity measures among objects is still not considered here. Assuming that the similarity measure among objects has time complexity ψ, then the final complexity of DBSCAN is $O(\psi N^2)$ or $O(\psi NlogN)$ iff an index structure is provided.

Figure 2.3 demonstrates some notions of DBSCAN including the directly density-rearchable notion (a), the density-connected notion (b), the core property of objects (c) and the cluster expansion process (d).

2.3 The Algorithm OPTICS

One major drawback of DBSCAN is that it only uses a single density threshold ϵ to extract clusters from data. Besides the difficulty of parameter selections, in real life applications, the intrinsic clustering structures usually cannot be characterized by a global density. They can only be revealed with many different local density thresholds instead. One simple approach is to repeatedly running DBSCAN with different parameter sets to find the intrinsic clustering structures. However, it obviously results in significant performance degradation while it does not guarantee a proper solution. OPTICS [16] is thus provided to cope with this problem. In contrast to DBSCAN, OPTICS does not produce explicit clustering results. Instead, it produces an ordering of objects in a dataset which encapsulates the information of many clusters in this dataset w.r.t. arbitrary values of ϵ that are smaller than a predefined threshold ϵ^*. The outcome of OPTICS is a so called reachability plot which can be graphically visualized to support interactive analysis of the cluster structure as show in Figure 2.4. OPTICS is based on the concepts of core-distance and rechability-distance of an object p to operate.

Given a set of objects O which contains N objects, a distance function $d : O \times O \rightarrow R$ and two parameters $\epsilon^* \in R^+$ and $\mu \in N^+$. The core-distance of an object p is defined as:

$$core\text{-}dist_{\epsilon^*,\mu}(p) = \begin{cases} \text{UNDEFINED if } |N_{\epsilon^*}(p)| < \mu \\ k\text{-dist}(p) \text{ otherwise} \end{cases}$$

where k-dist(p) is the distance between p and its k-th nearest neighbor. The reachability-distance of an object p w.r.t. object o is defined as:

$$reach\text{-}dist_{\epsilon^*,\mu}(p,o) = \begin{cases} \text{UNDEFINED if } |N_{\epsilon^*}(o)| < \mu \\ max(core\text{-}dist_{\epsilon^*,\mu}(o), d(o,p)) \text{ otherwise} \end{cases}$$

OPTICS works by creating an ordering of objects and additionally storing for each object its core-dist and reach-dist w.r.t. the previous object. The reachability plot can be constructed from this information in order to provide an interactive way for extracting clusters. For readability, interested readers please refer to [16] for more details.

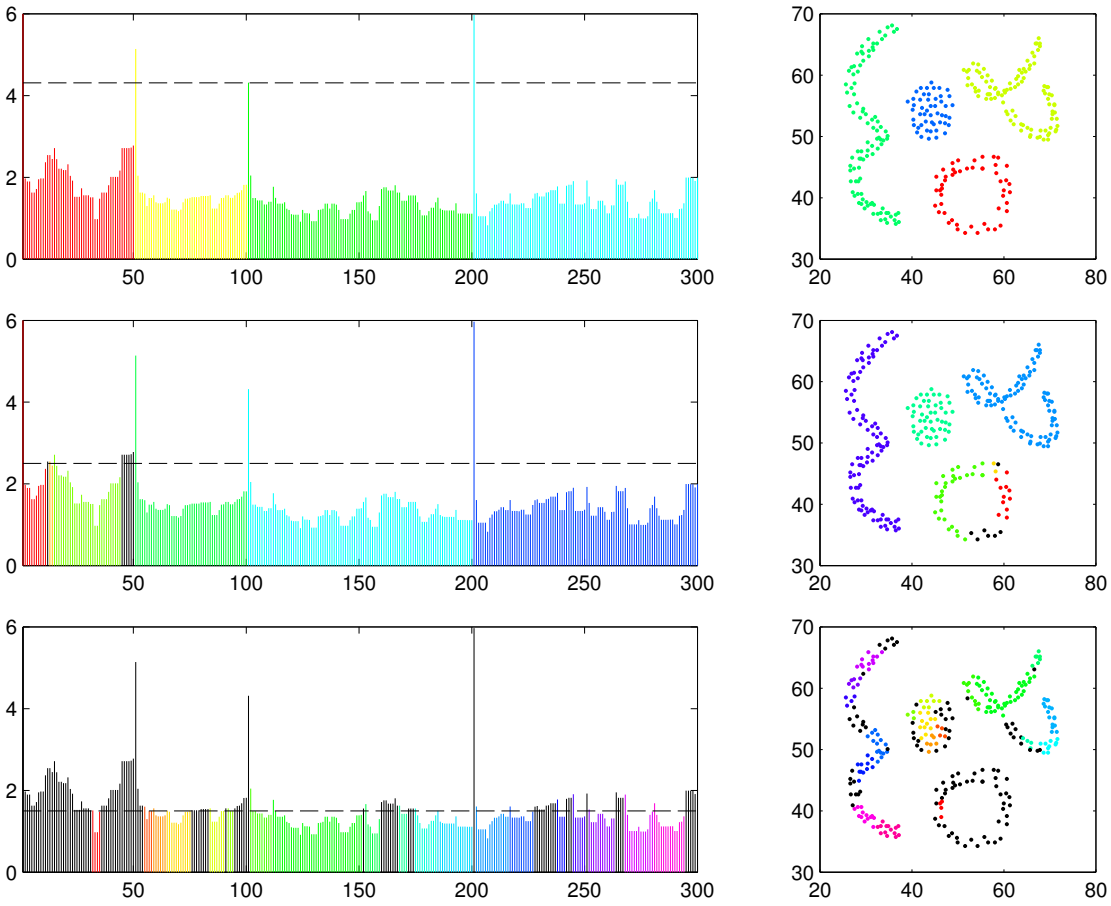

Figure 2.4: The reachability plots (left) and clustering results (right) of OPTICS w.r.t. different extract thresholds (the dotted horizon lines in the reachability plots). Outliers are drawn in black. The parameter ϵ^* is set to 6. From the top to the bottom, the threshold value ϵ is set to 3, 2.5 and 1.5 respectively.

The time complexity of OPTICS is similar to that of DBSCAN however with higher constant factor (around 1.6 times slower than DBSCAN as reported by the authors). Similar to DBSCAN, an index structure could be employed in order to reduce the query time, thus makes OPTICS $O(NlogN)$ time complexity algorithm.

There exist some algorithms like DeLi-Clu [7] which also try to produce a visualization structure similar to the reachability plot of OPTICS however with different algorithmic schemes.

2.4 Other Algorithms

There exist many different density-based clustering algorithms with different algorithmic schemes like DENCLUE [111], WaveCluster [239], STING [270], etc. In this Section, we briefly describe some of them as examples.

The algorithm DENCLUE. While the density notion of DBSCAN relies on the cardinality of the neighborhood of objects, the density notion of DENCLUE [111, 112] is based on the influence of an object into its neighborhood. In DENCLUE, the density at each point p is modeled by the sum of the influence functions, which are typically Gaussian functions or square wave functions, of all other objects with respected to object p. An object p is called density-attractor iff p is local maximum of the overall density function. An object q is density-attracted to a density-attractor p iff q can be reached from p through a sequence of objects that lie within an ϵ-circle from each other in the direction of the gradient. An arbitrary shape cluster of a given set of density-attractors X is defined as the set of objects that are density-attracted to one of the density-attractors x of X and the density function at x exceeds a predefined threshold ξ. Moreover, all pairs of density-attractors of X need to be connected by a path of objects P whose density must exceed the threshold ξ. The main advantage of DENCLUE is that it can robustly cluster datasets with large amount of noise and it allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional datasets.

Together with DBSCAN, DENCLUE is one of the most well-known density-based clustering algorithms. However, to the best of our knowledge, there are not many algorithms that follow the notion of DENCLUE proposed in the literature. In [110], the authors proposed an improve version of DENCLUE called DENCLUE 2.0 which aims at improving the hill-climbing process to determine density-attractors in the original version of DENCLUE. Klusch et al. [144] pro-

posed an algorithm called KDEC for distributed data clustering based on sampling density estimates. In KDEC, each data source transmits an estimate of the probability density function of its local data to a helper site. The helper then builds the overall density estimate. Based on the overall density estimate, each data source executes a density based clustering algorithm that follows the scheme of DENCLUE to cluster data.

Density estimation algorithms. There exist in the literature many clustering algorithms which are based on density estimation like DENCLUE. For example, the local density-based clustering algorithm proposed by Pamudurthy [211] et al. generally has the same idea with DENCLUE: using Kernel Density Estimation to calculate the density function and then determining clusters based on a predefined density threshold. However, while DENCLUE uses the fixed predefined kernel width σ, Pamudurthy et al. proposed to use different kernel widths for each object using the average distances from the centroid C of k-nearest neighbors to the k-nearest neighbors themselves. To cluster the data, the cluster boundaries are extracted from the estimated density of the data, and the objects are labeled following the contour tests instead of the density-attractor scheme of DENCLUE. Though the proposed algorithm outperforms DBSCAN in clustering the overlapping clusters and clusters with different densities [211], the performance comparison with DENCLUE was unfortunately not performed.

The algorithm WaveCluster. WaveCluster [239] applies wavelet transform on the spatial data feature space which helps in detecting arbitrary shape clusters at different scales due to the multi-resolution property of wavelet transform. Outliers are automatically removed from the transformed data feature by applying low-pass filters usually used in the wavelet transform. Concretely, the first step of WaveCluster is to quantilize the feature space into units by dividing each dimension of the feature space into equal intervals to form the unit cells. Objects are assigned into the cells based on their feature values. Then the wavelet transform is applied on the quantized feature space. Connected components in the transformed feature space at different levels are then formed by finding dense regions. Labels are assigned to objects and stored in a lookup table for finally determining the class label of objects in the original feature space. WaveCluster has $O(N)$ time complexity in general where N is the total number of objects. Thus, it is very fast compared with DBSCAN.

The algorithm STING. STING (Statistical INformation Grid) [270] divides

data space into rectangular cells at different level of resolutions. Each cell at the higher level is partitioned into cells of the next lower levels. Thus, at the end, they form a hierarchical structure which allows an incremental update when there are new objects arrive. Each cell stores statistical information of data inside it including the number of contained objects, the mean, standard deviation, minimum and maximum value of the attribute in this cell, and the type of distribution that the attribute value in this cell follows. The region query processing is processed in a top-down scheme: starting from the root node, following the most relevant cells based on statistical information of them until the lowest level is reached. The time complexity for a region query is $O(K)$ where $K \ll N$ is the number of grid cells at the lowest level and N is number of data objects. Though it is more efficient than index structures of DBSCAN and thus can be employed as the range query process of DBSCAN in order to acceleration the performance, STING may cause the loss of information in query processing.

Other algorithms. There exist in the literature many algorithms which are capable of detecting arbitrary shape clusters like CURE [99], CHAMELEON [129], etc. However, we classified them as distance-based clustering algorithms instead of density-based clustering algorithms. An algorithm is called density-based if it is based on some local criteria to form clusters [123].

2.5 Applications of DBSCAN

During the past decades, DBSCAN has become one of the most successful data clustering techniques and has been widely applied in many fields, e.g., neuroscience [238], trajectories clustering [166], aircraft monitoring [93], biomedical images segmentation [56]. In this Section, we briefly describe some of them as examples.

Lee et al. [166] proposed a trajectory clustering algorithm called TRACLUS for discovering common sub-trajectories in trajectory databases. TRACLUS consists of two phases: partition and group phases. In the first phase, each trajectory is divided into segments using Minimum Description Length (MDL) principle. In the second phase, these segments are grouped into group using DBSCAN algorithm on segment objects. The proposed algorithm would be a useful tool to detect similar common movement patterns of animal immigrants or movement patterns of hurricanes as demonstrated in the paper.

In [93], DBSCAN is used as principal components of two trajectory clustering

algorithms that are specifically designed for clustering of aircraft trajectories into flight patterns. This information can be used to enhance the monitoring and control of aircrafts, e.g., detecting unusual movement of an aircraft if it does not follow common movement patterns. In the first algorithm, DBSCAN is used to group all the turning points acquired from all trajectories into a finite number of way-points where it has been determined that aircrafts usually turn. These way-points are then further used by other clustering algorithm for grouping trajectories. In the second algorithm, DBSCAN is directly used to group trajectories represented by the first five Principal Components of them.

Segmenting the white matter fiber tracts acquired from Diffuse Tensor Imaging [194] plays an important role in neuroscience to study the structure of the brain and onset and progression of neurodegenrative and mental diseases. In [238], Shao et al. proposed to use DBSCAN to group the white matter fiber tracts in human brain into anatomical meaningful bundles and to reject noisy and spurious fibers to enhance the clarity.

The clustering is an essential part of Automated Diffraction Tomography (ADT) data processing, delivering the lattice basis vectors for single-crystal electron-diffraction data [230]. In [230], Schlitt et al. proposed to use DBSCAN for grouping electron ADT data since it is robust to noisy data, can detect arbitrary shape clusters and is easily to implement. The acquired clusters can then be used to determine the unit-cell basis vectors, usually as three shortest non-coplanar vectors within clusters, when a sufficient number of clusters are found [230].

DBSCAN is used to identify clusters of prophage genes (clusters of phage-like genes within a bacterial genome) in PHAge Search Tool (PHAST), a web server designed to rapidly and accurately identify, annotate and graphically display prophage sequences within bacterial genomes or plasmids [299].

Tramacere et al. [250] proposed a slightly extended version of DBSCAN called γ-ray DBSCAN for the detection of sources in γ-ray astrophysical images obtained from the *Fermi*-LAT data where each object is regarded as the arrival direction of photon. In this case, the robustness to outlier property of DBSCAN provides a useful solution for the noisy background rejection. γ-ray DBSCAN uses the angular distance between two photons as a similarity measure and follows the density-notion of DBSCAN exactly with only a minor change in the cluster expansion process.

Another interesting application of DBSCAN comes from Celebi et al. [56] where the authors focused on the problem of identification of homogenous color

regions in biomedical images, in particular, the segmentation of pigmented skin lesion images. First, the image is split into sub-regions following a top-down process to acquire the homogeneity criterion in each region. Then, GDBSCAN [228], an extension of DBSCAN for clustering spatial data, is used to segment the image by grouping similar sub-regions. Last, a post processing procedure is conducted to reduce the total number of grouped regions in order to enhance the clarity of the results.

The algorithm P-DBSCAN [139] is a variation of DBSCAN for the analysis of places and events using a collection of geo-tagged photos. P-DBSCAN extends the notion of DBSCAN by considering the number of peoples (owners of photos) into the definition of neighborhood and core photos. A notion of adaptive density for optimizing search for dense areas and faster convergence of the algorithm towards clusters with high density is also proposed.

Other applications. Huang et al. [118] proposed to use Self Organizing Map (SOM) and DBSCAN-based models for landslide hazard and spatial correlations analysis. In [189], the authors proposed an improved Storm Cell Identification and Tracking (SCIT) algorithm based on DBSCAN Clustering and JPDA Tracking Methods. Francis et al. [89] used an slightly adapted version of DBSCAN, in which the shared border objects between two clusters are randomly assigned to one of those clusters, for the simulation of DNA damage clustering after proton irradiation. The work of Kumar et al. [162] focused on DBSCAN algorithm for privacy preversing clustering. In [246], the authors proposed the NETwork-DBSCAN (NET-DBSCAN) for clustering dynamic linear networks. In [127], DBSCAN is used to discover moving clusters in spatio-temporal data with many applications such as discovering the moving groups of migrating animals. Xu et al. [283] presented an application of DBCLASD, a variant of DBSCAN, to cluster earthquake data.

Note that, there exist many other applications which are built upon other generalized density-based clustering paradigms instead of the paradigm of DBSCAN, e.g., the Density-based Hierarchical Clustering (DHC) method for clustering time series gene expression data [125], the density-based hierarchical clustering technique to identify coherent patterns from gene expression data [229]. However, in this Section, we mainly focus on applications of the density-based clustering algorithms that closely follow the DBSCAN paradigm.

2.6 Extensions of DBSCAN

Among various density-based clustering algorithms, DBSCAN perhaps is the most successful one with many extensions proposed in the literature, e.g., [45, 44, 160, 38, 228, 82, 282, 157, 16, 165, 272, 69, 267, 265, 84, 228, 272]. In this Section, we briefly summarize some of these works. Note that, we only focus on the algorithms which closely follow the DBSCAN paradigm. These extensions of DBSCAN can be roughly classified into different groups as the following.

2.6.1 Parameter Optimization

The algorithm DBSCAN requires two parameters μ, that describes the cardinality threshold, and ϵ, that describes the radius of neighborhood, to be set. These parameters play an important role on the performance of the algorithm DBSCAN, especially the parameter ϵ.

In [83], the authors suggested choosing $\mu = 4$ for 2-dimensional data. For the parameter ϵ, the authors suggested using a sorted k-dist graph, which contains the distances from every point p to its k-th nearest neighbor in ascending order, and an estimated percentage of noise to derive the value of ϵ. However, for many datasets, ϵ may not be easy to pick, especially when the percentage of noise is small or unknown.

Lee et al. [166] suggested another method for choosing the parameters μ and ϵ based on information theory. The proposed technique is generally based on an observation that $|N_\epsilon(p)|$ tends to be uniform in the worst clustering. Thus, if ϵ is too small, $|N_\epsilon(p)|$ tends to become 1; if ϵ is too large, $|N_\epsilon(p)|$ contains the whole data. Therefore, the entropy becomes maximal. In a good clustering, the entropy should be smaller since $|N_\epsilon(p)|$ tends to be skewed. The entropy $H(O)$ of a dataset O with N objects is defined as follows:

$$H(O) = \sum_{i=1}^{N} p(x_i) log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{N} p(x_i) log_2 p(x_i)$$

where

$$p(x_i) = \frac{|N_\epsilon(x_i)|}{\sum_{j=1}^{N} |N_\epsilon(x_j)|}$$

The parameter ϵ can be chosen as ϵ^* that minimizes $H(O)$ by using Simulated Annealing algorithm [68]. Then, the parameter μ can be chosen as $avg(N_{\epsilon^*}(p)) +$

$1 \sim 3$. Though, the proposed heuristic works quite well on the particular problem of clustering line segment data, its performance on other kinds of data, however, remains unknown.

The algorithm OPTICS [16] proposed a different approach for the parameter setting problem of DBSCAN. Given a predefined value ϵ^*, OPTICS produces a reachability plot and an ordering of objects which allow the algorithm to quickly produce the clustering results for any value of ϵ that $\epsilon \leq \epsilon^*$. Thus, users do not need to set a specific value for ϵ beforehand. However, OPTICS still has two parameters to set including μ and ϵ^*. Under the assumption of a random distribution of the objects, the authors suggested choosing ϵ as the radius r of a d-dimensional hypersphere R in S where S contains exactly k ($k = \mu$) points.

$$r = \sqrt[d]{\frac{Volume_O \times k \times \Gamma(\frac{d}{2}+1)}{N \times \sqrt{\pi^d}}}$$

where Γ denotes the Gamma-function. This technique, however, is only applicable for vector data and not for other kinds of data, e.g., time series. For the parameter μ, the authors suggested choosing μ between 10 and 20. An automatic technique to explore the reachability-plot and extract the clusters was also developed based on the steeps of the reachability plot.

The algorithm Automatic Eps Calculation (AEC) proposed by Gorawski et al. [96, 97] iteratively and randomly chooses a fixed number of sets of points and calculates three coefficients: distance between the points, number of points located in a stripe between the points and density of the stripe. Then the algorithm chooses the best possible result, which is the minimal distance between clusters as the value for ϵ. The calculated result in the previous step has an influence on the sets of points created in the next iteration. The algorithm AEC, however, suffers from several major drawbacks. It can only robustly estimate ϵ for simple datasets with small amount of noise. It requires a parameter that is hard to set. It has high runtime, even higher than runtime of clustering algorithm. Moreover, AEC is only designed for 2-dimensional datasets and mainly aims at finding parameter for DBRS [272], a variant of DBSCAN, and not for DBSCAN itself.

Summarization. Though there are several techniques to determine the parameters of DBSCAN proposed in the literature, they are only designed to deal with specific datasets [166, 96] or are still hard to select parameter [83, 16] or require user interaction [83, 16]. There are no automatic techniques which are applicable for many kinds of data proposed in the literature so far.

2.6.2 Clustering with Varying Densities

One major drawback of DBSCAN is that it is unable to detect clusters with varying densities due to the single density threshold usage. Figure 2.5 shows an example of a dataset with clusters of varying densities. DBSCAN cannot detect all the clusters exactly. Several approaches have been proposed in order to cope with this problem of DBSCAN, e.g., [16, 80].

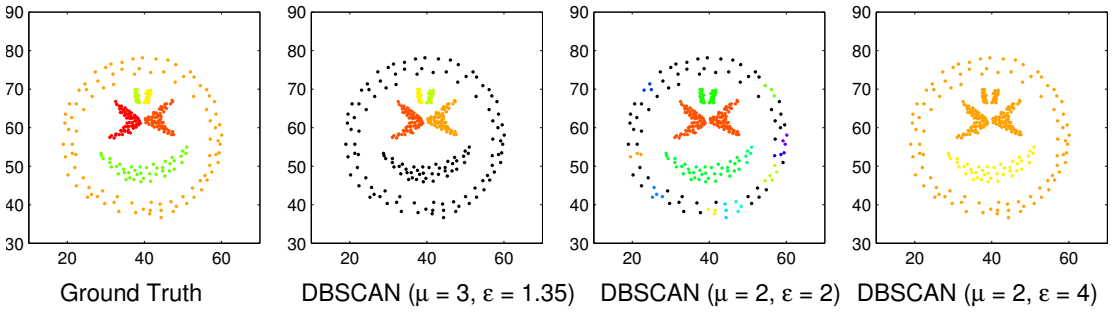

Figure 2.5: The clustering results on a dataset with clusters of varying densities. Outliers are drawn in black. DBSCAN cannot detect all the clusters exactly.

OPTICS [16] perhaps is the first algorithm which is able to deal with this problem of DBSCAN by producing the reachability plot to extract clusters. However, it only visuals the cluster structures without providing any method for determining clusters with varying densities. Thus, how to extract clusters with varying densities from the reachability plot of OPTICS remains an open research.

An approach based on Shared Nearest Neighbors (SNN) was proposed in [80] to cope with clusters with varying densities. Ertoz et al. defined the similarity between two objects p and q as the size of the intersection of the nearest neighbor sets of p and q based on an SNN graph, that is constructed from the dataset by connecting two objects p and q if p and q lie in the k nearest neighbor sets of each other, as follows:

$$similarity(p, q) = |NN(p) \cap NN(q)|$$

where $NN(p)$ and $NN(q)$ are sets of nearest objects of p and q w.r.t. the SNN graph respectively. The use of SNN graph can help to remove a lot of noise since they usually end up having most of their links broken. It also keeps the links in a region of any density, as long as the region has relatively uniform density. This

useful property helps to detect clusters with varying densities. An object p is called core objects if it has more than μ object q around it so that the $similarity(p, q)$ larger than a predefined threshold ϵ. The algorithm DBSCAN is then can be used to cluster the data.

Another approach came from Khani et al. [135] with an algorithm called Algorithm for Clustering Spatial Data with different densities (ACSD). The general idea of ACSD is to construct a graph on the data by adding edges between objects so that objects in a cluster lie in a connected component correspond to the cluster, whereas objects in different clusters are almost disconnected. In the beginning, ACSD creates a preliminary graph and iteratively improves it by sending feedbacks from each point to its neighborhood points. The neighborhoods and the feedback to be sent are determined by investigating the received feedbacks. After a stable graph is created, the clusters are formed by post-processing the constructed graph. The core and border objects are determined by calculating angles between edges. Then, the clustering algorithm DBSCAN is performed to group the data. ACSD may not perform well on high dimensional data due to its core and border object calculation scheme. Moreover, it has three parameters to set in comparison with two parameters of DBSCAN.

The algorithm LDBSCAN [77], a variant of DBSCAN, exploits the Local Outlier Factor (LOF) [48] of objects to discover clusters with different densities. An object p is called core object if its LOF score, denoted as $LOF(p)$, is below a predefined threshold $LOFUP$. A point q is directly density-reachable from a point p if q is inside $N_\mu(p)$ and $LRD(p)/(1 + pct) < LRD(q) < LRD(p) * (1 + pct)$, where $N_\mu(p)$ contains the μ-distance neighborhood of p, $LRD(p)$ denotes the Local Reachability Distance [48] of p and pct is a predefined parameter to control the fluctuation of local density. Based on these definitions, the algorithm DBSCAN can be used to cluster the data. One major drawback of LDBSCAN is that it has three parameters which are hard to set, especially $LOFUP$ and pct, though the authors has introduced a heuristic to choose them. Moreover, the comparison with existing techniques like SNN [80] or OPTICS [16] was unfortunately not conducted. Thus, it is difficult to assess the performance of LDBSCAN.

In [256], a grid-based algorithm called GRIDBSCAN was introduced as another solution for varying density problem. The algorithm works by first dividing data space into grids so that the density of object in each group is homogeneous. Then, it merges the cells with roughly similar densities and estimates the parameter ϵ for DBSCAN in each produced group of cells. Last, DBSCAN algorithm is performed on the objects in each cell group. Generally, the strategy here is quite

common: clustering different clusters with different density thresholds ϵ. Many other proposed algorithms follow this strategy to tackle with the varying density clusters. Similar to [77], there was no comparison with other techniques like SNN or OPTICS. Thus, the performance of algorithm is somewhat hard to evaluate.

Others. In [42], an algorithm called Density Differentiated Spatial Clustering (DDSC) was proposed to find clusters in which the densities within clusters are homogeneous. DDSC introduces an additional parameter α to measure the homogeneousness of a core object. A core object is homogeneous if its density is neither more than $\alpha_1 = (1+\alpha/(2\alpha)$ nor less than $\alpha_2 = 2/(1+\alpha)$ times the density of any of its neighbors. A cardinality test of a currently processed object p is proposed to guarantee that the number of already processed objects present in the neighborhood of p object should be within a certain minimum limit $\beta_{min} = 2/(1+d)(1+\alpha)$ and maximum limit $\beta_{max} = \alpha/(1+\alpha)$ where d is the dimensionality of data. The clustering process of DDSC is similar to that of DBSCAN. During the cluster expansion, the homogeneous test and cardinality test are conducted to ensure the proper cluster expansion. Assuming that object p is currently being processed during the cluster expansion, all object $q \in N_\epsilon(p)$ will be examined in an ascending order according to the distance $d(p,q)$ to impose growing of cluster in contiguous regions. One interesting property of DDSC is that the algorithm is able to find density based natural clusters that may not be separated by any sparse region. Besides an additional parameter α which is a challenging to choose, DDSC may suffer from the inherent sparseness of high dimensional data.

The algorithm Locally Scaled Density-based Clustering (LSDBC) [30] groups objects by connecting dense regions of space until the density falls below a threshold determined by the center of the cluster. Instead of using a single value of ϵ for the whole dataset, LSDBC computes for each object p a local density value ϵ_p based on its k-nearest neighbor distance. The smaller the value of ϵ_p, the more dense the area p lies. During the clustering, objects in denser area will be examined first. Given an object p as a center object of a cluster, LSDBC expands the cluster in the way that is similar to DBSCAN except that an object q will be inserted into the Seedlist S if $\epsilon_q \leq 2^{\alpha/N}\epsilon_p$ where α is used to determine the boundary of the current cluster expansion based on its density. It is still unknown how LSDBC perform when there exist several high density clusters that are close together and how the algorithm deals with outliers. There are several other algorithms proposed in the literature following the same approach with LSDBC: using different density thresholds to tackle with varying density clusters. The algorithm proposed

in [85] has similar approach with LSDBC: using k-nearest neighbor to calculate local density of each object, sorting each object according to the estimated density, performing clustering with varying value of ϵ for each object. However, it is still unknown how these algorithms perform when there exist several high density clusters that are close together and how the algorithm deals with outliers.

Summarization. There exist in the literature many other algorithms which are trying to cope with the varying densities of clusters besides those described above. However, most of these works suffer from the lack of comparisons and discussions with existing techniques. The used datasets are also simple and sometimes not clear enough. Thus it is hard to evaluate the quality of results. Moreover, they often have many parameters to measure the densities of clusters and to control the clustering process, which are somehow difficult to set.

2.6.3 Speeding up the Algorithm

During the past decades, many research efforts are being spent in order to speed up DBSCAN, e.g., [83, 34, 45, 44, 296, 272, 69, 253, 260, 48, 47, 297, 298]. Due to a vast amount of proposed algorithms in the literature, we only select some major algorithms as examples. Generally, these approaches can be roughly classified as the following.

Speeding up the neighborhood query. Since DBSCAN relies on the cardinality of the neighborhood of each object, speeding up the neighborhood query will significantly reduce the runtime of the algorithm.

In the original algorithm DBSCAN [83], any indexing structure such as R-Tree [103] and R^*-Tree [26] can be used to speed up the ϵ-range query process thus reducing the time complexity of DBSCAN to $O(NlogN)$ where N is the total number of objects.

When the data is large enough so that it cannot fit into the main memory, indexing approach like R-Tree and R^*-Tree may cause serious performance degenerations for the range query. And so is the performance of DBSCAN. In [34], the authors proposed a schema to transform the ϵ-range query of DBSCAN into a representation using the similarity join, a database primitive prevalent in multimedia database systems, as a basic operation to speed up the query processing while ensuring the correctness of the result of the algorithm. The same scheme can also be applied to OPTICS. Experiments on large datasets show the performance

acceleration of the proposed algorithms by factor of up to 33 times for the use of X-Tree [27] and 54 times for the use of R^*-Tree [26].

Brecheisen et al. [45, 46] approaches the problem from a different view point with [83] and [34] to cope with complex data and complex distance measures. Instead of using index structure, which may not be available, to reduce the query time, their algorithm relies on a concept of multi-step query processing which is based on *lower bounding* (LP) approximate distance functions to reduce to number of exact distance calculations and consequently to speed up DBSCAN (and also OPTICS). The general idea of the algorithm is that an object p only need to have μ neighbors under the true distance to be a true core object. Thus, in order to determine the core property of an object p, the ϵ-range query is performed with the lower bounding distances in order to approximate the neighborhood of p and to reduce unnecessary true distance calculations due to the filtering property of lower bounding distances. Then the true distances $d(p, q)$ between p and objects inside approximate neighborhood of p are calculated until p is surely determined as a true core object. This process is called μ-range query by the authors and is a key point of their proposed algorithm, since it helps to significantly reduce the total number of true expensive distance calculations and thus to significantly improve the performance of the clustering algorithm. After p is determined as a core object, the unprocessed true distances $d(p, q)$ are only calculated when they are necessary to expand a cluster. Inside the proposed algorithm, a special data structure called the *Xseedlist* is maintained in order to determine which true distances need to be calculated during the cluster expansion process. Figure 2.6 shows the detail and operation of the data structure Xseedlist. To be concrete, the lower bounding distance acts as a guideline for the cluster expansion. Conducted experiments on real datasets have reported the speed up factor of more than one order of magnitudes. However, since the Xseedlist requires to be continuously resorted during execution, it incurs a significant operation cost which may overwhelm the distance calculation reduction benefits, especially when the true distance function is cheap. Moreover, it is required (though not explicitly stated) that the lower bounding distance function must be significant faster than the true distance function.

Changing the cluster notion. Several techniques allow the changes in the density notion of DBSCAN in order to allow more efficient clustering scheme to enhance the performance, e.g., [296, 272].

The algorithm FDC [296] extends the notion of DBSCAN by introducing a new

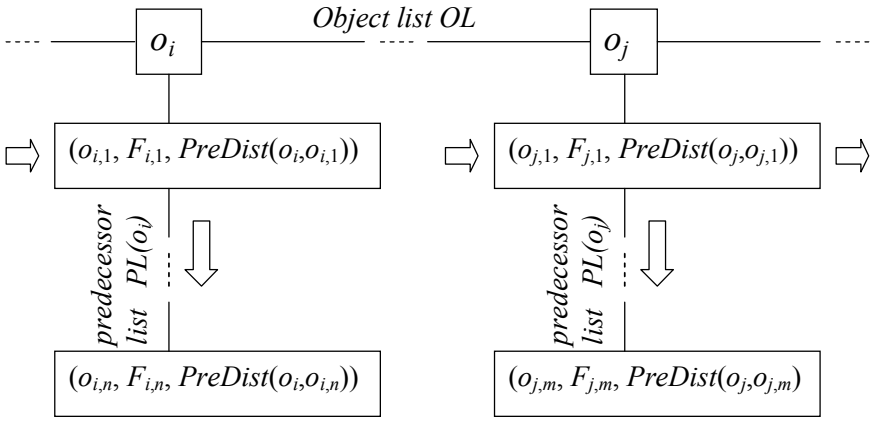

Figure 2.6: The data structure Xseedlist. It consists of an ordered object list OL. Each object o_i in OL is associated with a predecessor list $PL(o_i)$ and is sorted by the first element of $PL(o_i)$. Each entry of $PL(o_i)$ contains 3 items: (1) PreID: ID of a neighbor object $o_{i,k}$ $(1 \leq k \leq n)$; (2) PreFlag: indicates that the distance between o_i and $o_{i,k}$ is the LB or the true distance; (3) PreDist: the distance between o_i and $o_{i,k}$. The Xseedlist operates by using a LB distance as a guideline to extend the clusters. For every object p, the Xseedlist determines all of its neighbors under the LB distance, sorts them and then updates the distance between p and its neighbors with the true distances until the core property of p is determined. The others will be updated only when they are necessary to determine the density connectivity of objects during the cluster expansion.

notion called *density-linked* to replace the *density-connected* notion of DBSCAN. Two objects p and q are density-linked together if there exist a sequence of object $p = o_1, \ldots, o_n = q$ so that o_i is directly density-reachable from o_{i+1} and vice versa for all i. Given an object p, it is called noise object iff $N_\epsilon(p) = 1$ or p does not belong the the neighborhood of any core object. It is core object iff $N_\epsilon(p) \geq \mu$. In this case, all object inside $N_\epsilon(p)$ will belong to the same cluster with p. Based on these definition, the authors proposed a multi-stage algorithm to cluster data based on k-d tree. Due to its grid-based scheme of k-d tree, FDC may face difficulties when dealing with high dimensional data.

Wang et al. [272] proposed a random sampling method called Density-Based clustering with Random Sampling (DBRS) in order to speed up DBSCAN and also to tackle with varying densities problem. DBRS randomly selects an unprocessed object p and finds $N_\epsilon(p)$ like DBSCAN. If p is a core object, DBRS checks if $N_\epsilon(p)$ intersects with any existing clusters stored in a cluster list L. If the intersections

are found, DBRS merges all the corresponding clusters together with $N_\epsilon(p)$ to obtain a new cluster. In contrast to DBSCAN, DBRS does not examine the neighbors of p to expand cluster. It randomly picks up another object and repeats the whole process. This scheme allows DBRS to significantly reduce the number of region query thus improving the efficiency remarkably compared with DBSCAN. However, the clustering results of DBSCAN and DBRS are clearly not equivalent. One major drawback of DBRS is that it may miss joining certain clusters as pointed out by the authors.

Hybrid clustering techniques. Taking the face that k-Means cannot be able to deal with outlier though it is fast and DBSCAN can produce arbitrary shape clusters but it is slow, Dash et al. [69] proposed an algorithm called BRIDGES that merges DBSCAN and k-Means into one algorithm to overcome the limitation of both algorithms. BRIDGES generally works by first using k-Means to group data and then using DBSCAN to group objects on each k-Means cluster. Then noise are removed from the data in order to improve the results of k-Means. In order to group clusters produced by k-Means, the authors defined a *CoreDistance* of a cluster as a half of the distance between its center and its closest cluster center. An object is called core object of a cluster iff its distance to the cluster center smaller than $CoreDistance - \epsilon$. An object is ϵ-core object iff its distance to the cluster center is between $CoreDistance - \epsilon$ and $CoreDistance + \epsilon$. An object is non-core object if it is not a core or ϵ-core object. Concretely, the algorithm works in 6 steps: (1) using k-Means to group data; (2) estimating μ from ϵ and clustering result; (3) running DBSCAN for core and ϵ-core objects of each k-Means clustering; (4) running DBSCAN for all ϵ-core and non-core objects; (5) resolving the cluster conflicts by matching the results acquired from the step (3) and (4) and generating final clustering results; (6) running k-Means on data without noise objects using the same cluster centers acquired from previous steps. Since DBSCAN is performed on much smaller set of objects, the runtime reduces significantly. And by removing noise objects, the clustering quality of k-Means clearly is improved. However, the proposed algorithm requires two parameters k and ϵ which are somehow hard to set. Moreover, since two clustering results are produced, which of them is the most suitable result for the dataset?

There exist in the literature several algorithms with the same hybrid scheme, e.g., [253, 78]. The algorithm NPUST [253], for example, uses k-Means to group data, then performs DBSCAN on the acquired results and merges the closest clusters produced by DBSCAN to acquire final clustering result. In [78], CLARAN

[204] is used instead of k-Means.

Instead of using k-Means, the algorithm l-DBSCAN [260] proposes to use the *leaders clustering* method, a fast method which runs in linear time, with two different threshold values to provide two level hierarchy of prototypes. These prototypes are then used to deriving the density-based clusters. In [259] an extended version of l-DBSCAN called Rough-DBSCAN using rough set theory was also proposed.

Data summarization. Data summarization techniques are proposed to speed up clustering processes. They first summarize the data by calculating representative objects. The clustering processes are then performed on those representatives. And the clustering results for the whole dataset are derived from the results of the representatives [293].

In [48], a technique called *Data Bubbles* is first proposed in order to speed up the hierarchical clustering methods such as OPTICS. Given a set of object X which contains n objects in d-dimensional space, a data bubble of X is defined as a tuple $B = (n, M, e)$ where M is the center of X and e is average distance between all objects in X and is called *extend* of X.

$$M = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and

$$M = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2}{n(n-1)}}$$

Based on data bubbles, the core-dist and reach-dist are redefined to fit with the summarization scheme. The expected k-nearest neighbor distance inside a data bubble B is defined as $nndist(k, B) = (k/n)^{1/d}e$. Given two data bubbles $B = (n_1, M_1, e_1)$ and $C = (n_2, M_2, e_2)$, the k-distance between B and C is defined as follows:

$$dist_k(B, C) = \begin{cases} dist(M_1, M_2) - (e_1 + e_2) + nndist(k, B) + nndist(k, C) \\ \qquad\qquad \text{if } dist(M_1, M_2) - (e_1 + e_2) \geq 0 \\ max(nndist(k, B), nndist(k, C)) \text{ otherwise} \end{cases}$$

The core-distance of B is defined as:

$$core\text{-}dist_{\epsilon,\mu}(B) = \begin{cases} \text{UNDEFINED if } \sum_{X=(n,M,e)\in N_\epsilon(B)} n < \mu \\ dist(B, C) \text{ otherwise} \end{cases}$$

where C is the data bubble in $N_\epsilon(B)$ with minimal $dist(B,C)$ so that the condition $\sum_{X=(n,M,e)\in N_\epsilon(B)\wedge dist(B,X)<dist(B,C)} n \geq \mu$ holds. Note that $core\text{-}dist(B = (n,X,e)) = 0$ if $n \geq \mu$. The reachability-distance of a data bubble B w.r.t. a data bubble C is defined as:

$$reach\text{-}dist_{\epsilon,\mu}(B,C) = \begin{cases} \text{UNDEFINED if } \sum_{X=(n,M,e)\in N_\epsilon(B)} n < \mu \\ max(core\text{-}dist_{\epsilon,\mu}(C), dist(C,B)) \text{ otherwise} \end{cases}$$

Based on these new definitions, traditional OPTICS can be employed to produce a reachability plot as usual. In order to estimate the output of all objects, the authors defined the virtual reachability of all points of a data bubble B w.r.t. data bubble C as follows:

$$virtual\text{-}reachability(B,C) = \begin{cases} nndist(\mu,B) \text{ if } n \geq \mu \\ reach\text{-}dist_{\epsilon,\mu}(B,C) \text{ otherwise} \end{cases}$$

The proposed algorithm archives the speed up factor from 50 to 1700 on large datasets compared with original OPTICS. However, it is only applicable for metric space. Moreover, the performance of algorithm decreases with higher dimensional data due to the high-dimension sparseness problem.

There exist some extensions of the data bubble technique [48] in the literature. Breunig et al. [47] studied the performance of data bubble techniques with two ways to build the bubbles including the Cluster Features (CFs) [293] and a random technique. The study reveals three factors that seriously degrade the performance of algorithms including: lost objects, size distortions and structural distortions. Consequently, the authors proposed a post processing technique to solve the lost objects and the size distortion problem. In order to solve the structural distortions, a general concept of a data bubble as a more specialized kind of compressed data items, suitable for hierarchical clustering was proposed. Two efficient methods for constructing data bubbles for Euclidean distance were also proposed either by using sampling plus a nearest neighbor classification or by utilizing BIRCH [293]. In [297], the authors extended the works of [48, 47] for non-metric data space by introducing a new method for building data bubbles, a new distance measure function and a new extended clustering notion for OPTICS.

Others. The algorithm SDBSCAN [298] and its extension IDBSCAN [41] are based on an observation that some objects q in $N_\epsilon(p)$ may not need to be examined

for the cluster expansion since $N_\epsilon(q)$ may be fully covered by ϵ-neighborhood of some objects r in $N_\epsilon(p)$. The authors thus provided some random sampling schemes to select some objects as seeds for cluster expansion. Objects that are not processed after the clustering are assigned labels according to the label of their corresponding core objects. Though, this scheme can help to speed up the clustering process, it implicitly changes the clustering result in comparison with the original algorithm DBSCAN, especially with high dimensional data.

Summarization. Many research efforts have been conducted in order to speed up DBSCAN including many algorithms that are not listed here. Many algorithms slightly alternate or sacrifice the cluster notion of DBSCAN in exchange to the performance acceleration. Most techniques cannot go well with high dimensional and complex data, while the developing of model data acquisition methods are constantly producing more and more complex data in many fields.

2.6.4 Parallel and Distributed Clustering

Recent development of more powerful hardware and network infrastructures has led to the explosion of parallel and distributed data mining algorithms. Therefore, many researchers have been conducted aiming at parallel and distributed density-based clustering, in particular parallel versions of DBSCAN and OPTICS, e.g., [38, 39, 276, 14, 283, 123, 124, 109, 66, 213, 18, 214].

Graphic Processing Units (GPUs). The parallelization for GPUs structure differs considerabily from previous parallel algorithms which are mainly based on the share-nothing scheme. GPUs parallel algorithms not only share memory but also memory and fast memory by groups of processors [38].

Böhm et al. [38, 39] proposed CUDA-DClust, a density-based clustering algorithm specially dedicated to the use of GPUs under NVIDIA's CUDA architecture and programming model. CUDA-DClust is based on a new concept called *density-based chain*. A subset C is called density-based chain iff all pairs of object p and q in C are density-connected. A chain thus can be considered as a tentative cluster. The general idea of CUDA-DClust is starting many different cluster expansions at the same time via different chains from different starting points. Every chain is associated with a unique cluster ID. All collisions between all the chains will be stored in a boolean *collision matrix* $M_{m\times m}$ where m is the number of chains. Since $M$is small, transitive collisions can be determined later by a sequential algorithm. Since the detailed description is too long, interested reader please refer to [38] for

more details. The authors also proposed an extended version of CUDA-DClust called CUDA-DClust* that uses an index structure for similarity search which is particularly composed for the use in GPUs.

Recently, Welton et al. [276] proposed a parallel variant of DBSCAN called Mr. Scan. Mr. Scan combines the MRNet tree-based distribution network with GPGPU-equipped nodes into a hybrid parallel implementation scheme. Mr. Scan uses a programming paradigm that organizes processes into a multi-level tree with an arbitrary topology. On the GPGPU leaf nodes, DBSCAN is employed to group the data and the acquired results are then combined on non-leaf nodes. Mr. Scan is claimed the first parallel version of DBSCAN which incorporates the uses of GPGPUs architecture. The clustering process of Mr. Scan generally contains 4 main phases: partition, cluster, merge and sweep. In the partition step, MRNet is used to create one partition per clustering process. In the cluster step, MRNet tree of processes are first launched, one leaf process for each partition. Each leaf process uses GPGPU version of DBSCAN to group the data. And a small, constant set of representative objects is picked and prepared for the merging step. The merging step operates in a bottom-up scheme until it reaches to the root where the final merge is performed and a global ID is assigned to each cluster. The sweep phase then sends the global clusters IDs down the tree to identify the cluster ID of each object. The result is written to the output by the leaf processes. Experiments conducted on the paper are very impressive. The author used datasets with up to 6.5 billion data points (compared with 100 million at most from other works) and the parallel structures of up to 2000 nodes.

There exist in the literature some other parallel variant of DBSCAN based on the GPUs architecture, e.g., [14].

Parallel and distributed algorithms. In [283], the authors proposed PDBSCAN, a first parallel version of DBSCAN based on the share-nothing architecture. PDBSCAN relies on a distributed dR^*-Tree to partition the data among many computer nodes in the *master-slave* model. However, it replicates the entire index on each node. The slaves only cluster their local data concurrently by using DBSCAN. Message passing scheme is used for the master-slaves and slave-slave communications. The master takes responsibility for dynamic load balancing and merges the results produced by the slaves. In PDBSCAN, if a node queries the data that belong to other nodes, it must send messages to acquire the data. This makes a huge number of messages to be sent among nodes thus limits the scalability of the algorithm (experiments are conducted in [283] with 8 nodes).

The algorithm Density Based Distributed Clustering (DBDC) [123] assumes that objects are resided on different sites to be clustered. It operates on two levels: local and global level. On a local level, all sites perform a clustering independently from each other and then transmits only aggregated information about the local data to a central server including a set of pairs r and an ϵ-range value ϵ_r, where r is a representative and ϵ_r indicates the validity area of the representative. The global level is responsible for reconstructing a global clustering by using DBSCAN with two global parameters μ_{global} and ϵ_{global}. The acquired result is then sent back to all clients to relable their own objects.

Another algorithm from the same authors was proposed in [124], in which a scalable density-based distributed algorithm which allows a user-defined tradeoff between clustering quality and transmission rates between global and local sites is introduced. The authors introduced $DynRepQ(o)$, a quality criterion to measure if an object o is a suitable representative, to order all objects on the local sites for supporting an incremental clustering scheme in the server. A slightly enhanced version of DBSCAN is used to cluster representative objects in the server which takes into account the covering radius and the number of objects covered by each representative. Due to the incremental clustering scheme in the server, the clustering process can start as soon as the first representative objects arrive and allows the tradeoff between clustering quality and transmission rates.

In [44], lower bounding distance function is exploited in order to parallelize the algorithm DBSCAN. The algorithm is based on the fact that the lower bounding distance can help to produce a close approximation of the cluster structure acquired by using the expensive distance measure. The data is first partitioned by using an enumeration calculated by OPTICS so that similar objects have adjacent enumeration. At each client, efficient data clustering technique based on lower bounding distance proposed in [45] is used to cluster the local data. Then the acquired local clusters can be efficiently merged in the server by means of cluster connectivity graphs.

There exist many parallel variants of DBSCAN proposed in the literature including: MR-DBSCAN [109] and DBSCAN-MR [66] which are based on MapReduce programming platform, PDSDBSCAN [213] which is built over the Disjoint-Set data structure and the work in [18], etc. In addition, we aware some parallel variants of OPTICS proposed in the literature, e.g., POPTICS [214].

Summarization. In order to deal with massive datasets, parallel and distributed clustering algorithms have been proved a useful approaches. For density-based

clustering algorithm DBSCAN, its quadratic time complexity makes it a potential and necessary object for parallelizing. During the past decades, many parallel variants of DBSCAN have been proposed in the literature. However, only some of them concern on the clustering of large complex objects which is currently becoming an emerging research in the literature.

2.6.5 Incremental Clustering

Ester et al. [82] proposed an incremental version of DBSCAN called IncDBSCAN by us for the task of insertion, deletion and updating of objects in a data warehouse environment. Due to its special property, the changes of objects in the warehouse only affect the clustering result locally. When a new objects is inserted into the database, it may cause the merging of existing clusters. When an object is deleted from the database, it may lead to the splitting of an existing cluster. When an object changes its location, it may cause the splitting of its old cluster and the merging of other clusters at its new location. This locality helps IncDBSCAN to significantly reduce the runtime w.r.t. each change in the environment since it does not have to perform clustering on the whole database again. Only the affected areas need to be reclustered.

Kriegel et al. [152] proposed an incremental version of OPTICS called IncOPTICS. IncOPTICS is based on a key observation that the core distances of some objects may change due to an update. Consequently the reachability distances of some objects have to be updated as well. Here, the nature of density-based cluster is also exploited to reorganize the cluster ordering instead of starting from scratch. First, IncOPTICS maintains two sets of objects called *mutating* objects (objects that may change their core distance) and *moving* objects (objects that may move forward/backwards in the cluster ordering) for the reorganization process with each database update. Last, an efficient reorganization scheme is proposed to rearrange objects in mutating and moving sets. In [3], an extended version of IncOPTICS, called OnlineOPTICS, is proposed which allows bulk updating mode to handle very large sets of update operations.

There exist in the literature several incremental algorithms for some variants of DBSCAN. For example, Singh et al. [243] proposed IncSNN-DBSCAN an incremental version of SNN [80], an variant of DBSCAN described above. Kriegel et al. [153] proposed an incremental version of PreDeCon [36], a subspace clustering variant of DBSCAN described below. These algorithms also rely on the locality of each database update to reduce the runtime of the clustering processes.

There exist many density-based algorithms for data stream clustering, e.g., [206]. These algorithms can be regarded as incremental clustering algorithms and will be described in the following sections below. The algorithm DBCLASD [282] described below is also an incremental clustering algorithm.

Summarization. Incremental clustering has been proved as an efficient way to cope with dynamic data environment and is continuously attracting many research efforts. During the last decades, many incremental variants of DBSCAN have been proposed in the literature, especially for complex data such as stream data and time series data.

2.6.6 Subspace Clustering

Subspace clustering algorithms aim at coping with high dimension data. In high dimension data, meaningful clusters tend to reside in lower-dimensional subspace rather than in the full-dimensional space [156]. There exist in the literature many subspace clustering algorithms which follow the DBSCAN paradigm, e.g., [160, 36, 37, 4, 5, 6].

Subspace clustering. In contrast to the grid-based subspace clustering algorithm like CLIQUE [10], the algorithm SUBCLU (Density Connected Subspace Clustering) [160] is built upon the density-connected scheme of DBSCAN and thus can detect clusters with arbitrary shapes in subspaces of data. However, running density-based clustering on all possible subspaces results in very high running time since the total number of subspaces is exponential. SUBCLU, therefore, exploits the monotonicity property of density-connected set to prune subspaces in the process of generating all subspace clusters in a bottom up scheme. The pruning scheme of SUBCLU helps to reduce the number of examined subspaces and thus helps to improve the performance, while it still guarantees to have the same results with the naive exhaustive algorithm. However, experiments reported on 1.7 GHz CPU and 2 GB still show very high runtime of approximately a week for datasets with 50 dimensions. The runtime grows at least quadratic factors with both number of objects and dimensionality.

Projected clustering. The algorithm PreDeCon built upon the density-connected paradigm of DBSCAN like SUBCLU is able to compute all subspace preference clusters of a certain dimensionality in a single scan over the database and is linear to the number of dimensions [36]. In PreDeCon, a subspace preference clusters is

defined as density-connected set of objects associated with a certain subspace preference vector, i.e., a set of density-connected objects which has a variance smaller than a given threshold δ along one or more attributes. Given an object p, a subspace preference vector of p in d-dimensional space is denoted as $\overline{w}_p = \{w_1, \ldots, w_d\}$ where

$$w_i = \begin{cases} 1 \text{ if } VAR_{A_i}(N_\epsilon(p)) > \delta \\ \kappa \text{ otherwise} \end{cases}$$

where $\kappa \gg 1$ is a predefined constant, $VAR_{A_i}(N_\epsilon(p))$ is the variance of $N_\epsilon(p)$ along an attribute A_i in the attribute set A and is defined as:

$$VAR_{A_i}(N_\epsilon(p)) = \frac{\sum_{q \in N_\epsilon(p)} (d(\pi_{A_i}(p), \pi_{A_i}(q))^2}{|N_\epsilon(p)|}$$

where $\pi_{A_i}(p)$ is the projection of objects p onto an attribute A_i in the attribute set A. The general preference weighted similarity between two objects p and q is then defined as:

$$d_{pref}(p, q) = max\{d_p(p, q), d_q(p, q)\}$$

where $d_p(p, q)$ is the preference weighted similarity measure between p and q w.r.t. the preference weighted vector $\overline{w}_p$.

$$d_p(p, q) = \sqrt{\sum_{1}^{d} w_i \cdot d(\pi_{A_i}(p), \pi_{A_i}(q))^2}$$

Based on the preference weighted similarity measure, the density notion of DBSCAN can be straightforwardly extended as shown in [36]. For building clusters, PreDeCon performs one pass over the database to find all subspace preference clusters in a similar way with DBSCAN. One major drawback of PreDeCon is that it is only designed for vector data.

Correlation clustering. Unlike PreDeCon [36] which focuses on the clustering of axis parallel subspaces, the algorithm 4C (Computing Correlation Connected Clusters) [37] aims at discovering clusters in arbitrary oriented subspaces of data. Generally, it follows the same framework with PreDeCon. First, the authors proposed an extended dissimilarity measure function between objects called the correlation similarity measure by calculating the correlation similarity matrix using

Principal Component Analysis (PCA). Then, the density-based notion of DBSCAN is straightforwardly extended like PreDeCon. Objects are then grouped in a similar way with DBSCAN. Due to the complexity of 4C, interested reader please refer to [37] for more details. One major drawback of 4C is that it has cubic time complexity regarding the dimensionality of data. Moreover, the parameters are extremely hard to set since there is no obvious way to estimate the parameters among the correlated features of data.

Other extensions. There exist in the literature various extensions of these algorithms, e.g., Hierarchical Subspace Clustering (HiSC) [4] (an extension of PreDeCon [36]), Hierarchical Correlation Ordering (HiCO) [8] (an extension of 4C [37]), Exploring Relationships among Correlation clusters (ERiC) [5] (an extension of 4C [37] and HiCO [8]), COrrelation PArtition Clustering (COPAC) [6] (an extension of 4C [37]). There also exist in the literature some generic subspace clustering framework like FIRES [154] which can incorporate with any clustering algorithm to produce subspace clusters including DBSCAN itself.

Conclusion. Though, there are various extensions of DBSCAN for subspace clustering introduced in the literature during the past decades, most of them suffer from high runtime, e.g., SUBCLU [160], 4C [37]. Moreover, they have many difficult parameters that need to be set, e.g., 4C [37], etc. Not many algorithms are proposed to deal with complex data like graphs or trajectories.

One important question in subspace clustering is how to visualize the clustering results in subspaces. Some algorithms like HiSC [4], HiCO [8], ERiC [5] present the clustering results in hierarchical structures similar to OPTICS [16] (HiSC and HiCO) or graph that shows the relationships between subspace clusters (ERiC) which somehow allow the visualization of clustering results. There exist in the literature many visualization tools which allow the user to interact with clusters in low-dimensional subspaces, for example, HD-Eye [113] or Morpheus [198]. These techniques are, however, out of scope of this work.

2.6.7 Semi-supervised Clustering

In comparison with unconstrained clustering, constrained clustering for density-based clustering has attracted much less attention, e.g., [224, 40, 167, 294].

Data constraints. In [224], the authors proposed an algorithm called C-DBSCAN which exploits two set of instance level constraints including *must-link* (ML) and

cannot-link (CL) constraints to construct the density-based clusters. The use of constraints helps to solve many difficult cases which may not be possible to cope with using the unconstrained algorithms [9]. Generally, C-DBSCAN mainly operates in three steps: (1) It partitions the data space into dense subspaces by using the data structure KD-Tree [224] in a depth first travesal; (2) For each leaf node of the tree, the core property of each object p is determined. If there are CL constraints inside ϵ-neighborhood of p then p and all neighbors become disjoint local clusters. Otherwise, p and its neighbors becomes a single local cluster. This ensures the satisfaction of all the CL constraints; (3) Clusters which are connected by ML constraints are merged into new ones to ensure the satisfaction of ML constraints; (4) Final clusters are built in a bottom-up scheme and remaining CL constraints are enforced.

Instead of using ML and CL constraints, the algorithm HISSCLU [40] relies on a set of labeled objects to expand clusters simultaneously. The general goal is to determine a hierarchical clustering of the labeled and unlabeled objects with maximally large class pure sub-clusters of high density [40]. Inside HISSCLU, a method for cluster consistent assignment of class labels to previously unlabeled objects and a method for the determination of the overall cluster structure of the data set in a way which is consistent to original and obtained class labels are proposed. The result of HISSCLU is a hierarchical semi-supervised cluster structure that shows both cluster structure and class assignment. HISSCLU, however, suffers from the setting of four parameters, which are very hard to choose. The algorithm Semi-supervised DBSCAN (SSDBSCAN) [167] has the same approach with HISSCLU but focuses on DBSCAN instead.

In [294], the authors proposed a semi-supervised document clustering algorithm called Constrained DBSCAN (Cons-DBSCAN). Cons-DBSCAN uses instance level constraints (MLs and CLs) to guide the clustering process of DBSCAN. Given two sets of ML and CL constraints, Cons-DBSCAN first builds a collection of transitive closures (TCS) from MLs and updates the CL set. During the cluster expansion of DBSCAN, if the current examined object p belongs to a transitive closure c in TCS, all the objects inside c are then assigned to the current cluster. If object p is an object in the Seedlist S and belongs to a transitive closure c in TCS, then all the objects in c whose labels are not determined are labeled as current cluster. These steps ensure the satisfaction of ML constraints. If object p is a core object, every object q in $N_\epsilon(p)$ that (p, q) in CLs will not be added to S in order to satisfy the CL constraints.

Active learning for semi-supervised clustering. There exist many algorithms in the literature which do not focus on how to perform clustering with instant level constraints but focus on the problem of how to choose better constraints for semi-supervised clustering instead, e.g., [294, 169, 267, 265, 169]. These techniques automatically examine the data, choose the most informative objects and ask users for their opinions regarding their selections. Thus, they are called *active learning* techniques since they actively choose whatever they want to learn from users and query users for the results. For example, in [265], the authors proposed an approach based on k-nearest neighbor graph to estimate the dense regions of the data spaces, and queries users for ML and CL constraints of pairs of objects that may lie around the cluster borders where the cluster memberships are most uncertain. The acquired ML and CL constraints can help to significant improve the performance of semi-supervised clustering algorithms like [294, 224] compared with the use of randomly selected constraint sets. In [169], the authors proposed an active learning scheme for labeling objects as constraints to use with semi-supervised density-based clustering algorithms HISSCLU [40] and SSDBSCAN [167].

Conclusion. Though there exist many extension of DBSCAN (and OPTICS) proposed in the literature, semi-supervised variants of DBSCAN (and OPTICS), however, are not paid enough attention with only several proposed algorithms during the last decades.

2.6.8 Clustering Complex Data

Many density-based clustering algorithms have been developed in order to cope with the emerging complex data, e.g., spatial data [84, 228, 272, 290, 273, 274], graph data [284, 86, 100, 101], stream data [53, 268, 206], uncertain data [157, 158, 248, 104], dynamic data [165, 201, 32, 221, 222], instead of the traditional vector data. We summarize some of these approaches below.

Spatial data. The algorithm GDBSCAN [84, 228] generalizes the notion of DBSCAN in two important ways. First, any notion of a neighborhood of an object can be used as long as it is based on a predicate $NPred(p, q)$ that is symmetric and reflexive instead of the ϵ-neighborhood scheme of DBSCAN. Second, the cardinality of a neighborhood of an object can be replaced by a more general function $MinWeight(N)$ where N is a set of objects if $MinWeight$ is monotone in N instead of the simple counting scheme of DBSCAN. Based on these exten-

sions, GDBSCAN can be straightforwardly extended from the original algorithm DBSCAN. However, it not only can cluster point objects like DBSCAN but also other spatial extended kinds of objects like polygons.

In [272], the authors extend the notion of DBSCAN in order to support non-spatial attributes. For each object p, a property *prop* is defined w.r.t. one or more non-spatial attributes. A matching neighborhood of q is defined as $N^*_\epsilon(p) = \{q|d(p,q) \leq \epsilon \wedge p.prop = q.prop\}$. A threshold *minpur* is also introduced to confine the homogenerousness of a cluster w.r.t. the property *prop*. Object p is directly purity-density-reachable from p if $N^*_\epsilon(p)/N_\epsilon(p) > minpur$ or $N_\epsilon(p)/N^*_\epsilon(p) > minpur$. The cluster notion of DBRS [272] is defined similarly to that DBSCAN but with the notion of purity-density-reachability instead of the density-reachability notion of DBSCAN.

Spatial data with physical constraints. In spatial data, physical constraints such as obstacles and bridges linking clusters may significantly affect the effectiveness of the clustering. In [290], the authors proposed an extension of DBSCAN called DBCluC to cope with this problem, in particular spatial data with physical constraints including obstacles and crossings. These constraints are modeled using polygons. To perform the clustering, the reachability concept of DBSCAN is modified in the context of obstacles and crossings in order to expand clusters in a DBSCAN like scheme.

In [273, 274], the authors proposed an extension of the algorithm DBRS [272] described above called DBRS+ which can handle very large datasets with intersected obstacles and facilitators.

Dynamic data. Lai et al. [165] focused on an interesting problem of predicting density-based clusters over time for clustering in a dynamic environment. The proposed algorithm is built upon a simple formula to check wherever two objects p and q lie inside the ϵ-neighborhood of each other. As a result of neighborhood checks, one can determine when and where an object will become a core object and when and where a cluster is formed. The proposed algorithm can be used in air traffic control as stated by the authors.

In [201], the authors focused on the problem of clustering moving object trajectories using density-based clustering. First, a generalized distance function between trajectory objects which contains both the temporal and spatial aspects is proposed. Then, OPTICS is used for clustering trajectories due to its many interesting properties compared with other algorithms as experimentally proved in the

paper. The proposed algorithm is called T-OPTICS by the authors. The authors also proposed an extended version of T-OPTICS called Temporal focussing OPTIC (TF-OPTICS) to discover clusters of trajectories at determined sub-intervals.

The algorithm ST-DBSCAN [32] extends the notion of DBSCAN by incorporating an additional temporal distance measure beside the traditional spatial distance of DBSCAN. Two objects p and q are in the neighborhood of each other if they are close by both time and space. The cluster expansion procedure of DBSCAN is extended to fit with the new definition of neighborhood. In additional, the authors introduced a new concept called *density_factor* to deal with vary density clusters.

In [221], the authors proposed an algorithm called Dynamic Density Based Clustering (DDBC) for the clustering of mobile objects. The general idea of DDBC is incrementally maintaining a relationship graph to dynamically track relationships among objects. An adapted version of DBSCAN is then used to detect groups of objects that are strongly related based on the relationship graph. In [222], an extended version of [221] is proposed using Support Vector Machines (SVMs) to estimate the relationship graph.

Graph data. The algorithm SCAN (Structural Clustering Algorithm for Networks) [284] generalizes the paradigm of DBSCAN from point data to network data. The algorithm SCAN is not only able to discover clusters but also hubs connecting several clusters and outliers not belonging to any cluster. Given a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. The structure of a vertex v, denoted as $\Gamma(v)$, is defined by its neighborhood as follows.

$$\Gamma(v) = \{w \in V | (v, w) \in E\} \cup \{v\}.$$

Based on the structure of a vertex, the structural similarity between two vertexes v and w, denoted as $\sigma(v, w)$, is defined as follows.

$$\sigma(v, w) = \frac{|\Gamma(v) \cup \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

SCAN straightforwardly extends the notion of DBSCAN by using the structural similarity $\gamma(v, w)$ as the distance function. It also uses the same clustering algorithm with DBSCAN to group the vertexes. By using the *adjacency list* to represent graph, the time complexity of SCAN is thus only $O(m)$, where m is

total number of edges, which is much faster than many existing graph clustering techniques.

The algorithm DENGRAPH [86] is an incremental clustering algorithm designed to detect communities in large and dynamic social networks. DENGRAPH defines the distance between two actors p and q based on the number of interactions between them and extends the notion of DBSCAN in the similar ways with SCAN. When there are changes in the network structures including the insertion and deletion of actors and the changes of distance among objects due to the interactions of actors, DENGRAPH incrementally updates the communities to reflex the change. In [231], the authors proposed a hierarchical version of DENGRAPH called DENGRAPH-HO.

Kim et al. [137] proposed a method for clustering dynamic network under the *temporal smoothness* framework to discover a variable number of communities of arbitrary forming and dissolving. The temporal smoothness framework assumes that the structures of clusters does not change much in a short time and therefore tries to smooth clustering over time. It aims at trading between the snapshot quality and the quality of previous snapshots. This algorithm uses SCAN with an extended structural similarity function to produce clusters in a snapshot.

Beside these works, there exist in the literature several other graph clustering algorithms based on the DBSCAN paradigm, e.g., [100, 101].

Stream data. Clustering stream data has attracted many research efforts recently. In [53], the authors proposed DenStream, an algorithm for clustering an evolving data stream. In contrast to other previous techniques, DenStream is able to detect arbitrary shape clusters and is insensitive to noise. Inside Denstream, an variant of DBSCAN is proposed for partitioning micro-clusters into arbitrary shape groups based on the concepts of *core-micro-clusters*, *potential-micro-clusters* and *outlier-micro-clusters*.

The algorithm proposed by Wan et al. [268] uses a hierarchical grid of cells to maintain synopsis of streaming data in the online-phase instead of micro-clusters as in [53]. For the offline-phase, a variant of DBSCAN is proposed to partition cells into arbitrary shape clusters as in DenStream. In contrast to [53], the proposed algorithm allows users to discover clusters at multiple resolutions.

There are many other algorithms for clustering stream data proposed in the literature which follows the notion of DBSCAN or its extensions. For example, the algorithm HDDStream [206] is an extended version of PreDeCon [36] for subspace

clustering on stream data. In [108], another extension of PreDeCon for stream data was also proposed by Hassani et al.

Uncertain data. Uncertain data regard to the data that contain specific uncertainty and are observed in many fields, e.g., location-based services or sensor services where the location of objects can only be estimated with some uncertainty.

In [157], an extension of DBSCAN called Fuzzy DBSCAN (FDBSCAN) is proposed in order to cluster uncertain (fuzzy) data. Traditional distance measure among objects is replaced by *fuzzy distance functions* among objects including the *distance density function* and *distance distribution function*. Based on these two functions, the core object probability of an object can be calculated instead of the boolean values *yes* or *no* in the original DBSCAN. An object is called core object if its core object probability is larger than a predefined threshold, say 0.5 [157]. Based on the core object probability, one can determine the possibility that an object p is density-reachable from another object q. The authors called this possibility the *reachability probability*. If the reachability probability is larger than a predefined threshold (default 0.5), p is density-reachable from q. Based on these definitions, traditional cluster expansion algorithm of DBSCAN thus can be used to group the data.

In [158], the authors extend the concept of OPTICS to cope with uncertain (fuzzy) object in a similar way of FDBSCAN: replacing the original definitions of OPTICS with probabilistic versions. The proposed algorithm was named Fuzzy OPTICS (FOPTICS).

There are some other algorithms following DBSCAN paradigm proposed in the literature for clustering uncertain data, e.g., [248, 104]. Habich et al. [104] proposed an error-aware extension of the density-based algorithm DBSCAN named DBSCAN^{EA} for clustering sensor data where each data is represented by d-dimensional region (data region) in which all points within this region are equally likely to represent the object. The algorithm U-DBSCAN [248] extends DBSCAN to work with uncertain data based on a new deviation function that approximates the underlying uncertain model of objects.

Other data. The algorithm P-DBSCAN [139] extends DBSCAN to groups a collection of geo-tagged photos by considering the number of owners of a photo as an additional factor to determine core object properties.

Kailing et al. [126] proposed an variant of DBSCAN for clustering multi-represented objects which are objects that might provide several different repre-

sentations that may be used to analyze it. For example, molecules like proteins can be represented by a sequence of amino acid, a secondary structure and 3D representation [126]. The general idea of the algorithm is to combine the information of all representations at soon as possible and as late as necessary. The core object property is determined by using the local ϵ-neighborhoods of each representation and combining the results to a global neighborhood.

Conclusion. Advanced data acquisition techniques nowadays constantly produce large and complex data which consequently require special techniques to be proposed in order to analyze them, in particular, data clustering techniques. Due to its many attractive properties, DBSCAN has become one of the most used techniques for the task of clustering complex data.

2.6.9 Other Algorithms

There exists in the literature many interesting extensions of DBSCAN, e.g., [282, 177, 52, 251, 54, 60, 202]. We briefly describe some of them below.

Distribution Based Clustering of LArge Spatial Databases (DBCLASD) [282] is an variant of DBSCAN proposed in the literature which is relied on the assumption that the points inside a cluster are uniformly distributed. DBCLASD is based on an observe that the distance from an object to its nearest neighbors is smaller inside a cluster than outside that cluster. Each cluster has a probability distribution of objects to their nearest neighbors that can be exploited to define the cluster. DBCLASD has some interesting properties. First, DBCLASD is an incremental algorithm that means it can construct the clusters incrementally with each arrived data objects. Second, it is parameter free. Last, it is robust to noise. However, DBCLASD is much slower than DBSCAN (up to 3 times).

The algorithm Adaptive Density-based Clustering (ADBC) was introduced by Ma et al. [177] for the purpose of enhancing clustering quality when clustering spatial databases based on the distribution of objects. Instead of using spherical shape neighborhood like DBSCAN, the authors proposed to use dynamic ellipse which can not only adjust with different radiuses but also can rotate according to the distribution of the neighbors of an object to improve the clustering quality. In the beginning, all objects p have spherical shape neighborhood. Inspired by Newton's Universal Law of Gravitation, when there is an object q nearby, the neighborhood of p will be extended along the line $\overline{pq}$. Thus, by examining the neighbors of p, the size of eclipse can be easily determined. Then, the neighbor-

hood of an object p is defined as objects that belongs to the eclipse around p. To cluster data, the authors slightly modify the cluster expansion step of DBSCAN so that only core objects are inserted into the Seedlist. Compared with DBSCAN and OPTICS, ADBC is more robust to noise and parameter setting. However, it is not known whether ADBC can be used for higher dimensional data or varying density clusters.

In original DBSCAN [83], the border objects are assigned into clusters based on the ordering of objects. In case, two adjacent clusters shares some border objects, the performance of DBSCAN is thus limited. To tackle this problem, the algorithm from Tran et al. [251] first performs the cluster expansion procedure on the core objects only. Then, each border object is assigned to its its best density-reachablechain. This approach was also introduced in the anytime clustering algorithm A-DBSCAN [184].

In [52], the authors proposed HDBSCAN, a variant of DBSCAN, which is capable of producing a hierarchical clustering result, automatically extracting clusters from hierarchical tree, etc. HDBSCAN is based on proposed stability measure technique and density estimation in order to extract clusters.

Though they are not the main focus of this works, we would like to mention here some recent researches on clustering validation techniques for arbitrary shape clusters that may have strong impact in density-based clustering researches in the future [197, 288], since they can help to develop more efficient and effective techniques for improving the existing clustering techniques, e.g., automatically parameter finding.

2.7 Conclusions

During the recently decades, density-based clustering algorithms have attracted substantial research efforts with many extensions and applications proposed in the literature. Consequently, comprehensive reviews for density-based clustering algorithms are critical to deliver profound insights into the research field and thus significantly contribute to the development of the field.

In this Chapter, we provide a comprehensive survey about density-based clustering algorithms, their extensions and applications. In particular, we focus on the algorithms which follow DBSCAN paradigm. Our survey covers a wide variety of proposed algorithms mainly collected from many major publication venues in many different fields.

Among various approaches for density-based clustering, density-based clustering algorithms for complex data has become an emerging research due to the explosion of data acquisition techniques in many different fields that lead to the increasing complex data and consequently complex tasks to analyze these data. Thus, they are the main research focus on this thesis.

Chapter 3

Preliminaries

In this Chapter, we briefly describe some backgrounds which are used in the rest of this thesis.

3.1 Cluster Validation

Cluster validation algorithms aim at assessing the results of clustering algorithms. Generally, they are divided into two main categories: *internal measure* and *external measure*.

In external measure, clustering results are evaluated based on external benchmarks which consist of a set of pre-classified items that are often created by experts. Thus, the benchmark sets can be thought of as a gold standard or a ground truth for evaluation. These types of evaluation methods measure how close the clustering is to the predetermined benchmark classes. In contrast to external measure, there is no provided gold standard or ground truth in internal measure. A clustering result is evaluated based on the data that was clustered itself.

In this thesis, we focus on external validation techniques to assess the performance of our algorithm. There exist in the literature many external measures for clustering, e.g., DOM [76], NMI [258], AMI [258], AVI [258], Rand Index [247] and Jaccard Coefficient [247]. However, in this thesis, we employ three main information-theoretic validation techniques including DOM [76], NMI [258] and AMI [258], since these methods can robustly compare results with different numbers of clusters.

3.1.1 Information Theoretic Validation Techniques

Information theoretic validation techniques rely on the information theory to assess the performance of clustering algorithms.

Encoding Cost (DOM). Given a dataset O, a set of ground truth labels C and a set of produced labels K of a clustering algorithm, the external measure DOM [76] is defined as the coding code $Q(C, K)$ for entire dataset O.

$$Q(C, K) = H(C|K) + CL(C|K)$$

where $H(C|K)$ denotes the conditional entropy and $CL(C|K)$ is the code length for the number of clusters.

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{h(c,k)}{n}log\frac{h(c,k)}{h(k)}$$

and

$$CL(C|K) = \frac{1}{n}\sum_{k=1}^{|K|}\begin{pmatrix} h(k)+|C|-1 \\ |C|-1 \end{pmatrix}$$

where $h(c, k)$ is the number of fibers labeled with class c in C and k in K, $|C|$ and $|K|$ are the numbers of clusters in C and K respectively. The smaller the encoding code is, the better the clustering quality is.

Normalized Mutual Information (NMI). NMI [258] is one of the most popular clustering evaluation techniques which is based on the normalized mutual information between two set of clusters. Given two clusterings $U = \{U_1, \ldots, U_M\}$ and $V = \{V_1, \ldots, V_N\}$, $NMI(U, V)$ is defined as follows:

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}$$

where $H(U) = -\sum_{u\in U} p(u)log(p(u))$ and $H(V) = -\sum_{v\in V} p'(v)log(p'(v))$ are the entropy of the clustering U and V respectively. And $MI(U, V)$ is the mutual information between U and V.

$$MI(U,V) = \sum_{u\in U}\sum_{v\in V} p(u,v)log(\frac{p(u,v)}{p(u)p'(v)})$$

where $p(u, v)$ is the joint probability distribution function of U and V, $p(u)$ and $p'(v)$ are the probability function of U and V respectively. The result of NMI

lies in $[0, 1]$ where 0 means the two results are independent and 1 means the two results are identical.

Adjusted Mutual Information (AMI). AMI [258] is proposed to correct the effect of chance agreement in clustering evaluation of NMI and is based on expected mutual information.

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{max\{H(U), H(V)\} - E\{MI(U,V)\}}$$

where $H(U)$ and $H(V)$ are the entropies of clustering U and V respectively as described above and $E\{MI(U,V)\}$ is the expected mutual information of clustering U and V. Similar to NMI, AMI takes value 0 if the two clustering results are independent and 1 if the two clustering results are identical.

Adjusted Variation of Information (AVI). The Adjusted Variation of Information (AVI) [258] is given by:

$$AVI(U,V) = \frac{2MI(U,V) - 2E\{MI(U,V)\}}{H(U) + H(V) - 2E\{MI(U,V)\}}$$

where $H(U)$ and $H(V)$ are the entropies of clustering U and V respectively as described above and $E\{MI(U,V)\}$ is the expected mutual information of clustering U and V. AVI also takes value in the range of [0,1] where 0 indicates that U and V are independent and 1 indicates that U and V are identical.

3.1.2 Other Validation Techniques

There exist in the literature many other external cluster validity methods besides the information theoretic ones such as Rand Index and Jaccard Coefficient [247]. Given a dataset O and two clusterings $U = \{U_1, \ldots, U_M\}$ and $V = \{V_1, \ldots, V_N\}$. Assuming that a is the number of pairwise objects in O that are in the same set in U and in the same set in V, b is the number of pairwise objects that are in the different sets in u and in the different sets in V, c is the number of pairwise objects that are in the same sets in U and in the different sets in V, d is the number of pairwise objects that are in the different sets in U and in the same sets in V.

Rand Index (RI). The Rand Index [247], denoted as $RI(U,V)$, is defined as follows:

$$RI = \frac{a+b}{a+b+c+d}$$

$RI(U,V)$ takes value 0 if U and V are independent and 1 if U and V are identical.

Jaccard Coefficient (JC). The Jaccard Coefficient [247], denoted as $JC(U,V)$, is defined as follows:

$$RI = \frac{a}{a+c+d}$$

$JC(U,V)$ takes value 0 if U and V are independent and 1 if U and V are identical.

3.2 Lower bounding Distance

Given a set of objects O and a distance function $d : O \times O \to R$, a lower bounding (LB) distance of d is a distance function $d_{lb} : O \times O \to R$ where $\forall p, q \in O : d_{lb}(p,q) \leq d(p,q)$.

In the field of databases, LB distances are widely used to accelerate the query processing [132, 227, 216] since they are usually much faster than the original ones. There exist in the literature many different kinds of LB distances for many different kinds of distance measures such as Euclidean Distance (ED), Dynamic Time Warping (DTW) and Longest Common Subsequence (LCS) [74, 195]. The quality of LB distance is usually described by the tightness of LB (TLB) [132] which is the averaged ratio between the LB and the true distances.

Despite of their interesting properties, LB distances are however not paid enough attention in the field of data clustering with very limited number of related works such as [45, 44]. In this thesis, we will demonstrate how LB distances can be incorporated into the density-based clustering algorithm DBSCAN to make it anytime and active clustering algorithms.

In the rest of this Section, we illustrate some common lower bounding distances for the ubiquitous Euclidean distance as examples.

3.2.1 Piecewise Aggregate Approximation

Piecewise Aggregate Approximation (PAA) [133, 286] is a common data reduction technique in time series data mining. Given a time series $X = (x_1, \ldots, x_n)$, a PAA of X, denoted as $\overline{X}$, is defined as follows:

$$\overline{X} = (\overline{x}_1, \ldots, \overline{x}_N)$$

where $N \leq n$ is the dimension of $\overline{X}$ and

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

Given two n-dimensional time series X and Y, let $\overline{X}$ and $\overline{Y}$ is N-dimensional PAA representation of X and Y respectively. In [133, 286], the author proved that $d(\overline{X}, \overline{Y})$ lower bounds $d(X, Y)$ where d is the Euclidean distance function.

3.2.2 Convergent Bounds on the Euclidean Distance

In [120], the authors proposed the *MS-distance* which can provide upper and lower bounds of Euclidean distance of d-dimensional vector data in constant time assuming that the means and standard deviations of each vector are known. One interesting property of these bounds is that they can converge monotonically to the exact Euclidean distance within d refinement steps [120].

Given two d-dimensional vector $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$, let μ_x, μ_y, σ_x and σ_y be the means and standard deviations of x and y respectively. Let $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ such that $a_i = x_i - \mu_x$ and $b_i = y_i - \mu_y$. The MS-distance between x and y in its lower bound form and upper bound form, denoted as $\mathrm{MSL}(x, y, k)$ and $\mathrm{MSU}(x, y, k)$ respectively, are defined as follows:

$$MSL(x, y, k) = d((\mu_x - \mu_y)^2 + (\sigma_x - \sigma_y)^2) + \sigma_x \sigma_y \sum_{i=0}^{k} (\frac{b_i}{\sigma_y} - \frac{a_i}{\sigma_x})^2$$

$$MSU(x, y, k) = d((\mu_x - \mu_y)^2 + (\sigma_x + \sigma_y)^2) - \sigma_x \sigma_y \sum_{i=0}^{k} (\frac{b_i}{\sigma_y} + \frac{a_i}{\sigma_x})^2$$

where $a_0 = b_0 = 0$. In [120], the authors proved that $\mathrm{MSL}(x, y, k)$ increases to $d(x, y)$ with k and $\mathrm{MSU}(x, y, k)$ decreases to $d(x, y)$ with k. In case $k = d$, $\mathrm{MSL}(x, y, k)$ and $\mathrm{MSU}(x, y, k)$ are equal to $d(x, y)$.

3.3 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) [62] is a linear transformation to convert a discrete time signal whose length is an integer power of two into a discrete wavelet representation. A key advantage of wavelet transform is that it captures both frequency and location information (location in time) compared with the traditional Fourier transforms [62]. There are many kinds of DWT proposed in the literature, e.g., Haar wavelet transform, Daubechies wavelet transform. Among them, Haar wavelet transform is one of the most well-known due to its simplicity.

3.3.1 Haar Wavelet Transform

The Haar transform [216, 173] can be seen as a series of averaging and differencing operations between two adjacent values of a discrete time function $f(x)$ at a given resolution to form a smoothed, lower dimensional representation of signal. The wavelet decomposition is the combination of the coefficients at all resolutions: the first coefficient is the overall average of $f(x)$, while the other coefficients store the amount of information lost at each resolution. Due to space limitation, interested readers please refer to [216] for more details.

Resolution	Averages	Differences (Coefficients)
4	(9 5 3 7)	
2	(7 5)	(2 -2)
1	(6)	(1)

Figure 3.1: An example of Haar wavelet transformation.

Figure 3.1 shows a Haar transform for $f(x) = (9, 5, 3, 7)$ at different resolutions. Resolution 4 is the full representation of $f(x)$. The values of resolution 2 (7 5) are obtained from the averages of (9 5) and (3 7) respectively. The coefficients at resolution 2 are half of the differences of (9 5) and (3 7). The average and coefficient at level 1 are obtained by the average and half the difference of (7 5) at resolution 2. The wavelet representation of $f(x)$ is therefore (6, 1, 2, -2), which contains the overall average value of 6 and all the coefficients at all resolutions.

3.3.2 Applications of Haar Wavelet Transform

Due to its interesting property, Haar wavelet transform is widely applied in many fields, in particular the data mining and knowledge discovery field, e.g., [216, 173] to name a few.

In [216], the Haar wavelet transform is used for time series indexing by first transforming all objects into the wavelet coefficient domain and selecting some first coefficient to build an index structure for query processing.

Lemma 1 *The Haar transform preserves the Euclidean distance [216].*

Proof 1 *See Lemma 2 and 3 in [216].*

Following the Lemma 3.1 described above, it can be guaranteed that no false dismissal will occur during the query processing.

The multi-resolution property of Haar wavelet transform was exploited to improve the performance of the clustering algorithm k-Means in [174]. The proposed algorithm called I-kMeans works in multiple approximate levels w.r.t. the Haar wavelet transform on time series objects. At each level of resolution, the classical k-Means algorithm is performed on the coefficient vectors of time series at that level using the cluster centers returned at the previous level. By this way, I-kMeans is able to escape local minima thus it usually provides better results than k-Means itself. Moreover, the total cumulative runtime of I-kMeans is better than k-Means due to its faster convergency.

In this thesis, Haar wavelet transform is employed in order to construct lower bounding distances for Euclidean distances among objects.

Part II

Density-based Clustering of Complex Data

Chapter 4

Anytime and Active Clustering

In this Chapter, we briefly present some backgrounds and literature researches about anytime clustering algorithms and active clustering algorithms which are currently emerging researches in the field of data clustering.

Publications. Parts of the material presented in this Chapter have been published in [180]. The detailed information are described as follows:

- Son T. Mai. A Survey on Anytime and Active Clustering Algorithms. Technical Report, University of Munich, 2013.

  In this work, S.T.M. did the major part including the literature review, experiments and paper writing.

4.1 Anytime Clustering

Recently, anytime algorithms [303] have become an emerging research topic and attracted a lot of research efforts in many fields due to their many attractive properties when coping with complex data and complex tasks [145, 304, 71, 164, 49, 15, 219, 215, 245, 146, 136, 203, 188, 190, 13, 196, 302, 303, 161, 218, 143, 105, 115, 172, 107, 134, 143]. In this Section, we first briefly describe about anytime algorithms: their characteristics and their applications in reality. Then, we focus on applications of anytime algorithms for data clustering, one of the main tasks of exploratory data analysis.

4.1.1 Anytime Algorithms

Most algorithms work in a *batch* scheme: they run to completion and provide a single answer at the end. However, in many cases such as limited resource constraints [116], interaction and real-time systems [303], the users might need to have a *quick* result prior to completion [71, 303, 116]. To deal with this problem, anytime algorithms [305, 71, 116, 303], which work by trading execution time for quality of results, are proposed. In contrast to the *batch* algorithms, anytime algorithms are able to return a partial answer, whose quality depends on the amount of computation they were able to perform. In particular, they produce a fast approximate result which is then continuously refined during the further runs. During their executions, the algorithms can be interrupted at any time to provide a *best-so-far* result and to allow user interactions and resumed to produce better results at any time. Due to this property, anytime algorithms have been widely used during the past decades in many fields, including object recognition [145], motion planning [304, 71], image processing [164, 49], scheduling [15], multi-agent systems [219, 215, 161], surveillance [245], number partitioning [146], autonomous navigation [203], web services [136], etc.

Characteristics of anytime algorithms. According to [303], an anytime clustering algorithm should satisfy some important properties such as:

1. Measurable quality: The quality of an approximate result can be measured exactly.

2. Recognizable quality: The quality of an approximate result can be measured easily during runtime.

3. Monotonicity: The quality of result increases over time and input quality.

4. Consistency: The quality of result is correlated with computation time and input quality.

5. Diminishing returns: The quality of intermediate result improves much larger in previous stages of computation and diminishes over time.

6. Interruptibility: After spending some amount of time for the initialization step, the anytime algorithm can be stopped at any time and provide some answers.

7. Preemptability: The algorithm can be suspended and resumed again at any time in order to investigate the results with minimal overhead.

8. Low overhead: Assuming that it is not interrupted, the total cumulative runtime of an anytime algorithm should not be much longer than the runtime of the batch algorithm.

We note that, in many cases, these properties would be relaxed in order to adapt with the complexity of real-life applications. For example, it is not necessary that the quality of result must be strictly increase at some particular steps [300]. Besides, the measurement conditions are only applicable if there exists some methods to measure the quality of the *batch* algorithm [300]. In many fields such as clustering, measuring the quality of a clustering result is still an area of active researches and there is no known techniques that can handle this problem [300]. Thus, the first two conditions could also be relaxed in order to fit with these fields.

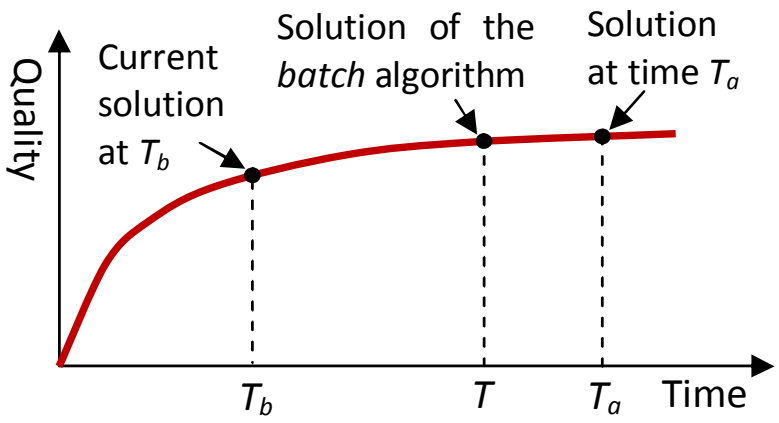

Figure 4.1: The progress of an anytime algorithm.

Performance of anytime algorithms. Figure 4.1 illustrates the progress of an anytime algorithm. Generally, the quality of the provided solutions improves over time. At the current time T_b, the algorithm is interrupted to provide an intermediate result and then resumed in order to find better solutions. If the algorithm is interrupted at the time T, it acquires the similar result with the batch algorithm. In case the algorithm is allowed to run to the time T_a, it may acquire the same or even better result than the batch algorithm.

Figure 4.2 shows the performance comparison among three different anytime algorithms A, B and C. The algorithm A clearly outperforms others. However,

the comparison between B and C is quite complicated since they dominate each other at different time periods. In this case, Zhu et al. [300] suggested choosing B since it has better quality improvement at the beginning of execution.

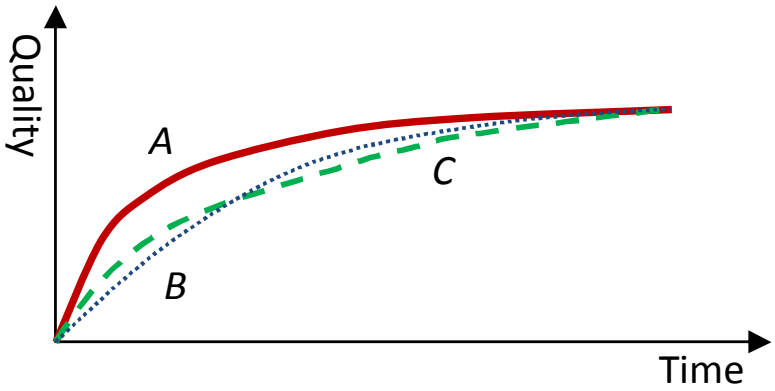

Figure 4.2: The performance comparison among three different anytime algorithms. The algorithm A clearly outperforms others.

Interruptible and contract algorithms. Zilberstein et al. [303] classified anytime algorithms into two different categories: *iterruptible* and *contract* algorithms. Interruptible algorithms can be interrupted at any time (even unexpectedly) to produce an approximate result. Contract algorithms also can trade the quality of result for execution time like interruptible algorithms. However, they require a particular time allocation in advance. If they are interrupted before their contract time, they may not provide any useful result. Obviously, interruptible algorithms are more flexible than contract algorithms and are consequently much harder to design and implement than contract algorithms. In those cases where an interruptible algorithm is required, Russell et al. [225] proposed a construction method that is capable to transform a contract algorithm into interruptible algorithm with only a small, constant penalty.

Algorithms with Anytime Features. There exist in the literature many algorithms that can have the *anytime* features though not explicitly stated as some examples described below:

Local search techniques, e.g., Simulated annealing, Hill climbing, Tabu search [1], Evolutionary algorithm, e.g., Genetic algorithm [67], Swarm intelligence algorithms, e.g., Ant Colony Optimization, Particle Swarm Optimization [94, 67]

are families of general-purpose techniques for search and optimization problems. These algorithms have many applications in reality including scheduling, image processing, network routing, partitioning, etc. They usually start with one or some initial solutions and then continuously explore the search space in order to find better solutions. One of the most important properties of these algorithms is that they are non-exhaustive search, which means that they do not guarantee to find an optimal solution, due to their heuristic scheme, but they search non-systematically until a specific stop criterion is satisfied. Since these algorithms can be, for example, interrupted at each iteration to provide an approximate result and resumed in order to search for better solutions, they could be regarded as anytime algorithms, except that they are usually not able to find optimal results (regarded as the result of the batch algorithms) and their behaviors are somehow unpredictable.

Iterative methods provide a sequence of improving approximate results to solve a problem. They have been attracted a lot of research efforts in many years, especially in solving large and sparse linear systems, one of the most common problems in scientific computing [226]. Local search techniques described above can also be considered as heuristic-based iterative search techniques. Iterative algorithms start with a simple, initial solution and then continuously refine the solution until they converge. Due to this property, iterative algorithms also could be considered as anytime algorithms.

Randomized algorithms [191] are another examples of algorithms with anytime property. These algorithms first produce an initial solution and then continuously use a randomized algorithm to try to improve that solution. Due to this scheme, these algorithms produce a sequence of improving results over time and could be interrupted and resumed at each iteration to acquire anytime property. Interested reader can find many examples of these algorithms in [191].

Besides these kinds of algorithms, there exists many other algorithms with anytime feature in the literature as examples in [302].

Anytime algorithms versus incremental algorithms. Incremental algorithms often use a local update scheme to deal with some changes of the input incrementally. The randomized incremental algorithms for constructing Voronoi diagram [21] is an example of this kind. Given a set of points in 2D, the randomized incremental algorithm works by randomly choosing each point, inserting it into the diagram and locally updating the Voronoi cells until all points are processed [21]. Similar schemes are also applied to calculate Delaunay triangulation,

convex hull, etc. of a point set in the field of computational geometry [21]. Another examples of incremental algorithms comes from the data mining field, e.g., [277, 82]. Ester et al. [82] proposed an incremental density-based clustering algorithm to efficiently deal with the insertion, deletion and updating of objects in a data warehouse environment. This algorithm exploits the local changes in the existing cluster structure to incrementally update the density-based clusters w.r.t. each operation. Widyantoro et al. [277] proposed an incremental algorithm for building cluster hierarchy.

Though incremental algorithms sometimes are called anytime algorithms in the literature, e.g., [147], they generally serve different purposes and thus are fundamentally different. While anytime algorithms aim at producing multiple approximate results of the whole input data over time, incremental algorithms focus on dealing with the change of input data efficiently.

Anytime algorithms versus approximation algorithms. Approximation algorithms are algorithms that find approximate solutions for a given problem when the optimal solution is computational expensive, especially with *NP-Hard* problems, e.g., vertex covering problem, graph coloring problem, traveling salesman problem [63]. Though anytime algorithms could be considered as approximation algorithms, they differ in the way that traditional approximation algorithms only produce a single approximate result for a given problem, while anytime algorithms produce an approximate result and continuously refine it during further runs toward the optimal result of this problem. Thus, anytime algorithms could be considered as in between approximation algorithms and exact algorithms.

4.1.2 Applications of Anytime Algorithms

During the past decades, anytime algorithms have found their applications in many fields, e.g., [145, 304, 71, 164, 49, 15, 219, 215, 245, 146, 136, 203, 188, 190, 13, 196, 302, 303, 161, 218, 143, 105, 115, 172, 107, 134, 143]. Due to the vast amount of anytime algorithms proposed in the literature as stated above, we only briefly describe some of them as examples.

Kobayashi et al. [145] proposed an anytime character recognition method which incrementally recognizes characters based on their difficulties. Easy characters, which can be detected in a short time, will be recognized earlier, while difficult characters are recognized later by accumulating the recognition results and extracted features though different time frames. The longer the algorithm

runs the more difficult characters it can recognize. Thus, the algorithm satisfies the monotonicity condition of anytime algorithms described above. Since the recognition process is split into several times, they can be interrupted. Thus, it satisfies the interruptibility condition of anytime algorithms described above. Experiments on the task of Japanese characters recognition has shown that the anytime character recognition algorithm produces the results faster than conventional batch algorithm, however it increases the total cumulative computational time at the end compared with the batch processing algorithm.

For the sensing, planning and control of robots [304, 71], the control systems usually have to cope with the changes of complex environments in real-time. The flexibility of anytime algorithms provides a practical approach for this problem. Under the time pressure, a coarse, fast sensing and planning is produced thus allows the robot to quickly response to the environment changes. However, when time is available, more accurate sensing and extended planning are conducted in order to provide the optimal operations for the robots [304]. Dean et al. [71] introduced a general frame work called *expected-driven iterative refinement* for the time-dependent planning and control of robots. Given a set of events, time-dependent planning determines how best to respond to predicted events when the time available to make such determinations varies from situation to situation [71]. The proposed framework includes a set of anytime algorithms for planning and a deliberation scheduling algorithm to allocate resources for anytime algorithms based on their performance expectations. Zilberstein et al. [304] proposed a method to construct robotic systems and to optimize their performances by using anytime algorithms for sensing and planning modules. These modules are build based on a *coarse-to-fine* grid-based scheme which allows the algorithm to have the anytime properties described above. The runtime of the algorithm increases w.r.t. the number of used grids in exchange for the better sensing and planning results. The control of anytime algorithms is conducted using offline compilation and runtime monitoring via different control *frames* which contain sensing, planning and plane execution episodes.

Kywe et al. [164] proposed an efficient scheduling method for image processing in real-time system. In their method, some basic image processing tasks including smoothing, edge detection, thinning are converted into anytime algorithms by processing the image with predefined patterns from simpler to more complex ones. The more complex the patterns, the better the acquired results and the slower the runtime. An adaptive scheduling method based on [302, 303] is also proposed to schedule the processing task under a condition that the processing time is re-

stricted. Some other conventional image processing tasks such as shape feature extraction, etc., however, are not included in this work. Another example comes from [49] in which Brooks et al. used an anytime algorithms to find the optimal transformation parameters that maximize the similarity between two images in real-time. Instead of using all the pixels to calculate the similarity between pictures, the algorithm starts with only a subset of pixels to calculate the similarity. During the runtime of algorithm, more pixels are incrementally considered based on the desired accuracy, acquired by an offline training procedure, and and the current magnitude of the gradient of similarity function. The higher the number of used pixels, the transformation parameters are further optimized.

Given a set of web services, *QoS-aware* web service composition (WCS) aims at finding a sequence of web services with the optimal accumulated Quality of Service (QoS) value in response to a user request. Due to its global optimization problem, QoS service is an intractable problem, especially when the number of web service is large or when real-time scenarios, e.g., e-business, are required. Therefore, Kil et al. [136] proposed an anytime algorithm for the QoS-aware WCS problem based on the *beam stack search* technique which explores a fixed number of candidate states in each level. Their proposed algorithm can identify high quality results much earlier compared with optimal algorithms, demonstrated by experimental analysis on six real WCS problems acquired from the Web Service Challenge 2009 competition.

Rahwan [219, 218] proposed an anytime algorithm for optimal coalition structure generation. Coalition structure generation is a central problem of coalition formation problem, which involves the creation coherent groupings of distinct, autonomous, agents in order to efficiently achieve their individual or collective goals, in the field of multi-agent systems. The purpose of coalition structure generation involves the partition of agents into exhaustive and disjoint coalitions in order to maximize the goals. Due to exponential number of possible solutions that needs to be examined, coalition structure generation has been proved a NP-complete problem and thus is a very challenging problem. In this scenario, anytime algorithms become a useful solution, since the agents usually do not have enough time to run the algorithm to completion, especially with an exponential search space. The proposed algorithm are build upon a new representation of the space of possible coalition structures which partitions the coalition structures into smaller and disjoint sub-spaces based on the sizes of the coalitions they contain. These subspaces can be explored independently in order to find an optimal solution. Based on this representation, the authors proposed an anytime Integer-Partition (IP)

algorithm to find an optimal coalition structure. The general idea of IP is that: it computes bounds and the best coalition structures with in every subspace and then incrementally searches the remaining subspaces and reduces the search spaces by applying the upper- and lower bounds as well as branch and bound search algorithm. Of course, by this incrementally searching scheme, the algorithm can be interrupted at anytime to provide a nearly optimal result. Experiments have shown that the algorithm provides very high quality results (more than 90% of an optimal solution) within only 10% of time required to find an optimal solution.

Simultaneous Localization and Mapping (SLAM) is a technique used by robots to estimate their own position and orientation and model the surrounded environment. Due to its useful applications, SLAM has been studied extensively in the literature. However, it is also regarded as one of the most challenging problem in autonomous navigation [203] due to its *chicken-and-egg* problem, uncertainty and high complexity. One of the most common techniques for SLAM is the Extended Kalman Filter (EKF) that recursively computes a concrete measure of the uncertainty in the covariance matrix. The covariance matrix contains the necessary information in order to reduce the risk of failure while making the data association and path planning. Unfortunately, storing and updating the covariance matrix are also a major bottle neck in SLAM. To deal with this problem, Nerurkar et al. [203] proposed an extended technique of ELK-SLAM called Power SLAM. Power SLAM is based on 3 main techniques: (1) Global-Map Postponement technique that delays the approximations over multiple time steps; (2) the Power Method for updating the covariance matrix; (3) the rank-2 update to increase the speed of convergence of the estimator. One of the key advantages of Power SLAM is the ability to adaptively trade the estimation accuracy in order to meet the availability of computational resources. This makes Power SLAM an anytime algorithm.

Sofman et al. [245] proposed an anytime algorithm to detect novel perception system input on an outdoor mobile robot. At each time step t, features are extracted from the inputs and then classified based on linear combination of the similarities w.r.t. previous classified features via an SVM-based classifier called NORMA. Multiple Discriminant Analysis (MDA) is used in order to reduce the dimensionality of input features thus enhancing the performance. If the result is larger than a predefined threshold then the input scene is classified as not novel and is discarded. Otherwise, the input scene is novel and is used to update the classifier. The anytime properties of algorithm are acquired via several addition schemes including: (1) early termination of the similarity calculation when it ex-

ceeds the threshold; (2) the new novel examples are considered first since they are more likely to impact the future queries; (3) the maximum number of storage examples is limited. These schemes help to reduce the calculation time thus allowing the anytime properties of the algorithm. In case the algorithm is interrupted too early, it returns a safe solution: a *false alarm*. Thus it does not cause missed detections.

The number partition problem is an NP-Complete problem that aims at dividing a set of numbers into 2 small subsets so that the sums of the numbers in each subset are as close as possible. In [146], the author proposed an algorithm called Complete Karmarkar-Karp (CKK) to efficiently solve this partition problem. One important advantage of CKK is that it is a *complete anytime* algorithm. CKK starts with the first polynomial-time approximation Karmarkar-Karp solution and continues to find better solutions until it reach the optimal solution.

Others. Saba et al. [188] proposed the construction, instrumentation, online measurement and runtime scheduling for an anytime algorithm that enable imprecise and approximate real-time computation on parallel architectures, in particular on the Graphic Processing Units (GPUs) architectures. In [15], Angelopoulos et al. focused on the design of an *interruptible* anytime algorithm for scheduling n equal problem instants on m identical processors using schedules of executions of anytime *contract* algorithms. In case of a single processors, the proposed schedule is proved to be optimal. When there are m processors available, the schedule is nearly optimal with only a small deviation. McMahan et al. [190] proposed an anytime algorithm that leverages fast best-response oracles to build a models of the convex games, that generalize zero-sum matrix by allowing arbitrary convex sets in place of probability simplices. Aine et al. [13] proposed an iterative anytime heuristic search algorithm called *Anytime Window A** (AWA*) which has many applications in planning and scheduling. They demonstrated the performance of their algorithm on 0/1 Knapshack problem [63] and Traveling Sale Man problem (TSP) [63]. The same efforts to propose an anytime algorithm for the heuristic search algorithm $A*$ can be found in [172, 107]. Kleinberg [143] proposed an anytime algorithm to deal with multi-armed bandit problems where learning and optimization should be balanced in order to achieve good cumulative performances. Kettle et al. [134] proposed an anytime algorithm to detect symmetry in the Boolean functions with reduced ordered binary decision diagrams (ROBDDs) which play an important role in CAD software to synthesize circuits (logic synthesis) and in formal verification. Haenni et al. [105] proposed an approximation algorithm for

computing arguments or explanations in the context of logic-based argumentative or abductive reasoning. Kumar et al. [161] proposed an anytime algorithm for Decentralized partially observable MDPs (DECPOMDPs), an emerging problem for modeling sequential decision making by a team of agents.

4.1.3 Anytime Clustering

Due to their attractive properties, anytime algorithms have been widely studied in the literature especially in the fields of Artificial Intelligent (AI) during the last decades. Recently, they are being employed to deal with the increasingly large and complex datasets in the field of Data Mining and Knowledge Discovery, e.g., anytime outlier detection for data streams [20], anytime nearest neighbor classification [279, 255], anytime Bayesian classification for data streams [232, 149, 150], anytime Bayesian network [176, 119], anytime classification for constant data streams [151, 241], anytime inductive logic programming [175], anytime support vector machine [72], anytime Naïve Bayes Text Classifier [223], anytime induction of decision tree [81], anytime data analysis [244], anytime boosting techniques [200], anytime top-k processing [17], anytime averaged probabilistic estimators for classification [285].

Anytime Clustering Algorithms

Though there are many anytime algorithms proposed in the literature, most of them focus on anytime classification, e.g., [279, 255, 232, 149, 150, 176, 119, 151, 241]. Anytime clustering, however, has not paid enough attentions. To the best of our knowledge, there are not many anytime clustering algorithms proposed in the literature so far, e.g., anytime clustering of data streams [147, 148], anytime time series clustering [174, 173, 300].

For the task of clustering data stream, Kranen et al. [147, 148] proposed a data structure called ClusTree in order to automatically adapt to the speeds of the data stream. ClusTree maintains a set of *micro cluster* features, denoted by a tuple of number of presented objects n in a cluster, their linear sum LS and their squared sum SS, into a hierarchical tree structure extended from R-Tree family [26] at different level of granularity. Each entry of ClusTree contains the cluster features of its respective subtrees. Each inner node contains a buffer b for temporary insertions of local aggregates. When a new object arrives, it is inserted into the subtree with the closest mean w.r.t. Euclidean distance following a top-

down scheme until the insertion process is interrupted by other coming objects or by users. The objects that do not reach the leaf node during the insertion process will be stored in the corresponding buffers waiting to be processed when the time is available. This scheme allows the algorithm to fully adapt to the changing speed of the data stream and makes it an anytime algorithm since the more time it has, the more accurate the object is located and thus the better the clustering. The final clustering result can be obtained from the set of micro clusters at leaf nodes using k-center clustering [207] or density-based clustering [53].

Lin et al. [173] proposed an algorithm called I-kMeans for clustering time series data. I-kMeans exploits the multi-resolution property of the Haar Wavelet transform to become an anytime algorithm. First, each time series is transformed into multiple levels of Wavelet coefficients using the Haar transform. At each level of resolution, the classical k-Means algorithm is performed on the coefficient vectors of time series at that level using the cluster centers returned at the previous level. Besides the anytime property, I-kMeans has several other attractive properties: (1) the clustering quality is usually better than the batch algorithm due to the avoidance of local minima; (2) it is faster than the batch algorithm even if it is run to the end. In [174], the authors extended the I-kMeans using the multi-resolution property of Piecewise Aggregate Approximation (PAA) [74] instead of the Haar Wavelet transform. A method for clustering streaming time series is also proposed by locally updating the clusters that contain any time series belonging to the neighborhood of a new coming time series.

Though Dynamic Time Warping (DTW) [132] has been proved an effective similarity measure for time series data [74], its quadratic time complexity is a bottle-neck in many data mining task such as time series clustering. Zhu et al. [300] proposed an approximation technique to estimate DTW distance between two time series based on its lower bounding and upper-bounding functions. The algorithm first initializes the distance matrix with these approximation distances of DTW. Then it incrementally replaces the approximate distances with the true DTW distances in a best first order, i.e., the pair of time series with a higher ratio between Euclidean distance and lower bounding distance will be updated first. The goal is to quickly approximate as close as possible to the true DTW distance matrix with each distance update. This scheme allows us to cast any clustering algorithms which are based on the distance matrix into anytime clustering algorithms, e.g., spectral clustering [263], hierarchical clustering [106], k-medoids clustering [106]. Obviously, the more time the algorithm has, the closer the distance matrix reaches the true DTW distance matrix thus the clustering results come closer to those of

the batch clustering algorithms.

Clustering Algorithms with Anytime Features

There exist in the literature many clustering algorithms with anytime features though not explicitly stated, e.g., [114, 67, 79, 121]. We briefly described some of them as examples below.

Active clustering. In order to deal with the similarity sparseness problem, Hofmann et al. [114] proposed a technique called *active clustering* that can actively select new data and use tentative knowledge to estimate the relevance of missing data. The algorithm is built upon a selection procedure, which is based on the so-called *Expected Value of Sampling Information* (EVSI), to select data, and a special clustering technique and objective functions, which is based on aggregate clustering membership of objects, to recluster data. Another example of this kind of active clustering algorithm is proposed in [79] where Ericksson et al. investigated how to perform hierarchical clustering of N objects using only a small subset of pairwise similarities instead of the complete set of $N(N-1)/2$ similarities. The algorithm actively and sequently selects meaningful pairwise similarities in an adaptive fashion to perform clustering. Under the *Tight Clustering* (TC) condition, the algorithm requires at most $3Nlog(N)$ pairwise similarities to reliably determine the unambiguous hierarchical clustering. In case the TC condition is not satisfied, the proposed algorithm requires only $O(Nlog^2N)$ pairwise similarities to produce high quality hierarchical clustering result.

Though these active clustering algorithms were not presented as anytime algorithms, they actually have some anytime features. For example, they could be trivially modified to support interruption and the clustering quality improves over time. However, one major problem of these algorithms is that their active schemes usually incur high computation cost, thus leads to expensive clustering processes, which oppose the efficiency purpose of anytime algorithms. In case of very expensive similarity measure, the benefit or similarity reduction of these algorithms may overcome the active selection cost. Thus, these algorithms could be regarded as anytime algorithms. The algorithm of Zhu et al. [300] described above is an example of this kind.

Metaheuristic clustering. Metaheuristic clustering algorithms [67] are an evolving research direction for data clustering. These algorithms formulate the clustering problems as optimization problems where the best partition of a dataset is

achieved by maximizing or minimizing one or multiple objective functions, e.g., minimizing the within-cluster sum of squares like classical k-means algorithm [121]. In order to acquire better result while solving these optimization problems, metaheuristic search algorithms such as Evolutionary algorithms [67] have been proved a useful approach and are widely used since they are able to avoid the local minima compared with traditional local search techniques [67, 94]. Some examples of metaheuristic clustering algorithms are: Genetic Algorithm (GA) [163], Bacterial Evolutionary Algorithm (BEA) [68], Evolutionary Algorithm (EA) [95], Ant Colony Algorithm [257], Particle Swarm Optimization (PSO) [210], etc. Due to a vast amount of these algorithms proposed in the literature, we only list a few in order to maintain readability and clarity, interested reader please refer to [67] for a comprehensive survey.

As discussed in 4.1.1, these clustering algorithms have anytime features. Thus, they could be regarded as anytime algorithms. However, one major drawback of these techniques is that they are only applicable for clustering problem with the goal of optimizing objective functions.

Iterative clustering. Classical k-Means algorithm and its extensions, e.g., k-Medoids, k-Medians [121] use an iterative refinement approach to assign objects to clusters until their objective functions converge to one of numerous local minima. Another examples for iterative clustering algorithms are Expectation Maximization (EM) clustering and its extension [121]. As discussed in 4.1.1, these algorithms also have anytime features and thus could be regarded as anytime algorithms.

Randomized clustering. The clustering algorithm CLARA [204] randomly draws a subset S of data ($40+2k$ objects where k is the number of clusters), computes k-medoids for S, labels all objects according to their most similar medoids. When the objects are drawn in a sufficiently random way, the medoids of S would approximate the medoids of the whole dataset. In order to acquire a better approximation, multiple sets of objects are drawn and the best results are reported in term of the average dissimilarity of the clustering results. The algorithms CLARANS [204] has the same randomized scheme with CLARA. However, it is based on the randomized search instead. As discussed above in 4.1.1, these algorithms would also be regarded as anytime algorithms.

4.1.4 Conclusions

Recent decades have witnessed constantly increasing numbers of large and complex datasets following the development of advanced data acquisition techniques which consequently require more efficient and effective algorithms to extract knowledge from them. Besides, the needs of interactive exploring of data has been arisen in order to let domain experts apply their knowledge by quickly testing hypotheses and performing exploratory data analysis [234, 9]. During the last decades, anytime algorithms [303] have been proved a flexible solution to cope with these challenges by providing a sequence of improved approximation results and allowing interruption during their executions. Consequently, they has been widely used in exploratory data analysis [279, 255, 232, 149, 150, 176, 119, 151, 241, 244, 17, 147, 148] recently. In this thesis, we focus on anytime algorithms for the task of data clustering, one of the main tasks of exploratory data mining.

In contrast to anytime classification [279, 255, 232, 149, 150, 176, 119, 151, 241], there exist only few algorithms in the literature that explicitly aim at anytime clustering, e.g., [173, 174, 300] to name a few. However, there exist in the literature many clustering algorithms with implicit anytime features, e.g., metaheuristic clustering [67], iterative clustering [121] or randomized clustering [204]. These algorithms would be easily adapted to anytime algorithms. Thus, they could be somehow regarded as anytime algorithms though they are not explicitly stated so.

Together with the evolvement of complex data such as time series [132] or trajectories [195], the more complex similarity measures such as Dynamic Time Warping (DTW) [74] or Longest Common Subsequence (LCS) [74] have been developed in order to efficiently cope with the new arisen challenges of complex data. Though they are efficient, these similarity measures have high time complexity which obviously becomes to a bottle-neck of the data clustering task. Thus, the need of more effective data clustering methods to deal with this problem has been arisen during the last decades, e.g., [45, 44]. As described above, anytime clustering algorithms could be a flexible solution for this problem. However, among those anytime algorithms described above, only some of them, e.g., [300], are designed to deal with expensive similarity measures.

In this thesis, we tend to fill in this gap by introducing an anytime clustering algorithm for complex similarity measures. Our algorithms is built upon the density-based clustering paradigm and thus called anytime density-based clustering (A-DBSCAN) [184, 185].

4.2 Active Clustering

Similar to anytime algorithms, active learning has attracted a lot research efforts recently [235]. In this Section, we first describe some backgrounds about active algorithms including their characteristics and their applications in reality. Then, we focus on a literature survey for active clustering algorithms, an emerging research field in data mining.

4.2.1 Active Learning

Active learning is a special case of *semi-supervised* machine learning and has become an emerging research field in the literature recently. The key idea behind active learning is that an algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns [235]. Active learning is a very useful solution for many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labeled data are difficult, time-consuming, or expensive to obtain.

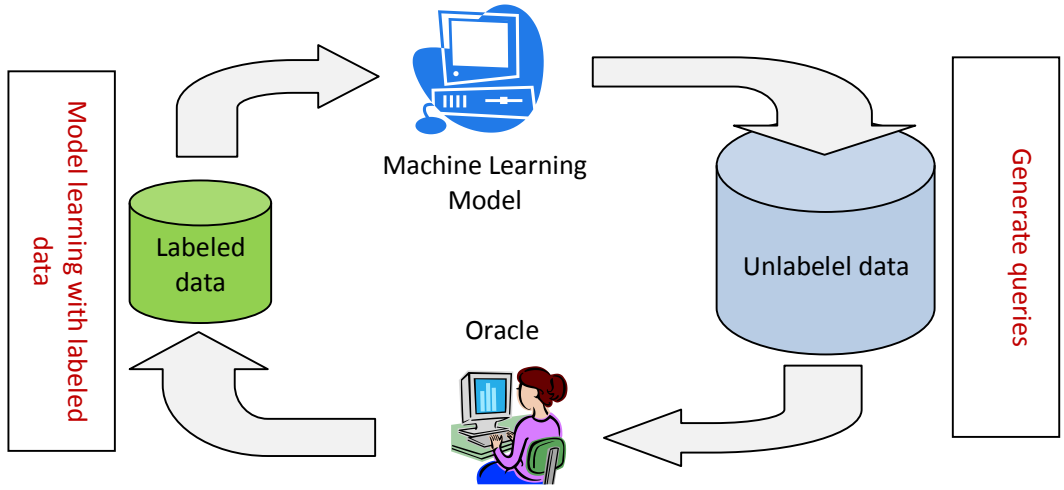

Figure 4.3: A common active learning scenario.

Figure 4.3 illustrates a common active learning scenario. The machine learning model actively generates queries, usually in the form of unlabeled data instances, and asks oracles, e.g., human annotators, for results. The acquired labeled data are used as training data to update the model. The whole process is repeated until it is terminated.

Performance of active algorithms. Generally, the performance of an active algorithm should increase w.r.t. the number of queries as illustrated in Figure 4.2. The more queries the algorithm generates, the better the results its produces.

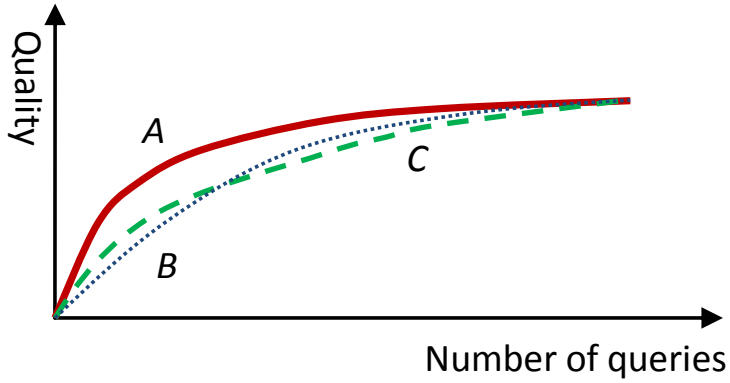

Figure 4.4: The performances of three different active algorithms. The algorithm A clearly outperforms others.

Figure 4.4 shows the performance comparison among three different anytime algorithms A, B and C. The algorithm A clearly outperforms others. However, the comparison between B and C is quite complicated since they dominate each other at different time periods. In this case, we prefer choosing B since it has better quality improvement at the beginning of execution.

4.2.2 Applications of Active Algorithms

During the past decades, active learning algorithms are widely used in many fields, e.g., [262, 33, 275, 278, 217, 249, 254, 43]. In this Section, we briefly describe some of them as examples.

Taking the fact that performing N separate pairwise similarity searches is much slower than a *one-versus-all* batch search scheme in biology, Voevodski et al. [262] proposed an active clustering algorithms which relies on *one-versus-all* query scheme to operate. Under some assumption about the structure of instances, their algorithm requires only $O(L)$ ($k \ll N$ where N is the number of instances) of one-versus-all queries to efficiently find an accurate clustering.

In [33], the authors proposed an active learning scheme to cluster images in which humans are involved in order to improve the accuracy. In the beginning, the

algorithm performs clustering using features extracted from images. The clustering result is then used to identify pairs of images that need to be justified by users and poses queries for results. The answers from users are then used as constraints to improve the clustering results. The process is repeated until the query budget is exhausted.

Wolf et al. [278] proposed a semi-automatic system for the task of clustering historical corpora. The proposed system is based on a graphical model that makes inferences based on catalog information provided for each leaf as well as on the pairwise similarities of handwriting. To help improve the accuracy in borderline cases, experts can involve to the clustering process by answering the queries actively posed by the system.

In [275], the authors introduced a problem of clustering photos acquired from a wearable camera, which may be useful in a variety of applications, for example, improving life quality for Alzheimer's patients or summarizing personal memories. Since the data is collected in an arbitrary manner, human must be involved to annotate the similarities among photos. Unfortunately it comes with a high cost. In order to reduce the payment cost, the authors proposed an algorithm which can accurately group the photos with only small number of annotated pairwise similarities. The algorithm starts with empty similarity matrix, iteratively selects the most informative pairs of photos and asks the annotators for the similarities between them until it is interrupted or a query budge is reached.

Others. In [217], the authors developed an active learning approach for classification galaxies and prediction Alzheimer from the structural MRI scan. In [249], an active learning support vector machine algorithm was developed for the text classification purpose. Tuia et al. [254] proposed an active learning system for the segmentation of remote sensing images. Bowring et al. [43] proposed an active learning system for automatic classification of software behavior.

4.2.3 Active Clustering

Though there are many active learning algorithm proposed in the literature, most of them aim to select data that will reduce the model's classification error or label uncertainty [235]. Compared with the active classification, active clustering, sometimes regarded as an unsupervised classification, has gained much less attention.

We roughly classify active clustering algorithms into two categories: active

learning for *sem-supervised* clustering, active learning for *unsupervised* clustering.

- Active learning for *semi-supervised* clustering: These algorithms focus on learning a set of constraints, mostly instance level constraints in the form of *must-link* and *cannot-link* constraints. These constraints are then used as training sets for semi-supervised clustering algorithms. Examples of these algorithms include [98, 265, 271, 187, 25, 267, 280, 12, 205, 266, 264], etc. Most of active learning for clustering algorithms fall into this category.

- Active learning for *unsupervised* clustering: These algorithms are mainly designed to deal with the sparseness of distance matrix instead of learning constraints from data. The general goal of these algorithms is to acquire a good clustering result with minimum number of used pairwise distances. In order to do so, a common approach is that these algorithms iteratively select the most informative pairwise distances with the goal of maximizing the clustering results to query until a predefined budget limitation is reached. Recently, these algorithms have attracted a lot of research efforts with many algorithms proposed recently, e.g., [275, 262, 79, 114, 236, 51].

In this thesis, we mainly focus on the latter case. In the rest of this thesis, we refer the latter to as *active clustering* in abbreviation.

Distance sparseness problem. Most data clustering techniques, e.g., k-Means [121], spectral clustering [263], require a matrix of pairwise distances to operate. Given N objects, there are $N(N-1)/2$ pairwise distances to calculate in total. However, in many real life applications, these pairwise distances may be difficult, time-consuming, expensive or event not available to obtain. This problem is commonly regarded as *distance sparseness* problem in the literature. As concrete examples, let consider these examples below.

A star light curve is the measurement of light intensity of a celestial object or region as a function of time. A light curve can be used to estimate the rotation period of a planet or comet nucleus in planetology or to discover supernovas in astronomy, etc. Clustering is commonly used to analyze the digital star light curves data using Dynamic Time Warping (DTW) as an effective similarity measure [300]. In [300], the authors pointed out that it cost around 127 days to cluster a mere 9236 curves under DTW due to the quadratic time complexity of DTW.

In [275], the authors introduced a real life application of clustering the photos acquired from a wearable camera, which may be useful in a variety of applications,

for example, improving life quality for Alzheimer's patients or summarizing personal memories. Due to the difficulty of automatically measuring the similarity among photos, human annotators must be involved to rate those similarities. Of course, it is a costly process since annotators must be paid for their works.

In model transportation monitoring and control system, GPS is usually used to collect the position of vehicles, people, airplanes, etc. Then, clustering algorithms are used to discover common or unusual movement patterns as in [201, 93]. However, in many cases, the GPS signals may be very noisy or may be temporarily lost due to bad weather, obstacles, etc. Thus, measuring the similarities among moving trajectories becomes hard or even infeasible. This problem is also common in many other application fields such as sensor networks. Moreover, efficient similarity measures for moving trajectories like Dynamic Time Warping (DTW) or Longest Common Subsequence (LCS) [195, 74] usually have high time complexity which makes the evaluation a very time-consuming process.

Many other examples can be found in [275, 262, 79, 114, 236, 51, 159], etc. Interested readers please refer to these works for more details.

Active Clustering. In other to tackle with this data sparseness problem described above, many active clustering algorithms have been proposed in the literature, e.g., [275, 262, 79, 114, 236, 51]. These algorithms often follow the same scenario: they iteratively select the most informative pairs of objects to query the distances and update the cluster results until a predefined budget limitation is reached. We briefly describe some of them below.

Hofmann et al. [114] proposed a technique called *active clustering* that can actively select new data and use tentative knowledge to estimate the relevance of missing data. The algorithm is built upon a selection procedure, which is based on the so-called *Expected Value of Sampling Information* (EVSI), to select data, and a special clustering technique and objective functions, which is based on aggregate clustering membership of objects, to recluster data. Since the proposed algorithm is based on its own definition and objective functions, its application may be limited. In [51], an active clustering algorithm for a hierarchical variant of [114] was proposed by Buhmann et al. In that work, *Bayesian Statistical Decision Theory* is applied as a tool for selecting new data which are most informative for the clustering task. In both works, the proposed techniques significantly outperform the naive random selection techniques.

The active k-Median clustering algorithm proposed by Voevodski et al. [262] uses an active selection strategy to choose a set of landmark points and constructs

clusters based on the distance between these landmarks and other points. The main difference between this algorithm and other algorithms in this category is that it is based on *one-versus-all* queries to form the clusters in contrast to the *one-to-one* query scheme of other techniques [79, 114]. This algorithm requires $O(k)$ *one-versus-all* queries in order to efficiently find an accurate clustering.

Ericksson et al. [79] investigated the problem of how to perform hierarchical clustering of N objects using only a small subset of pairwise similarities instead of the complete set of $N(N-1)/2$ similarities. The algorithm actively and iteratively selects meaningful pairwise similarities in an adaptive fashion to perform clustering. Under the *Tight Clustering* (TC) condition, the algorithm requires at most $3Nlog(N)$ pairwise similarities to reliably determine the unambiguous hierarchical clustering. In case the TC condition is not satisfied, the proposed algorithm requires only $O(Nlog^2N)$ pairwise similarities to produce high quality hierarchical clustering result.

Active spectral clustering algorithms were studied in [236]. Concretely, the authors focused on the problem of finding an approximation $\widetilde{v}_2$ of the 2nd eigenvector v_2 of the Laplacian matrix of spectral clustering. In case $||\widetilde{v}_2 - v_2|| \ll 1$, the clustering result acquired from $\widetilde{v}_2$ and v_2 are very similar as proven in the literature [128]. Besides a fast randomized algorithm, the author proposed an adaptive algorithm which is based on perturbation theory to choose the most informative pairwise distance. The general idea is to choose a pair of object that causes the most change on the vector $\widetilde{v}_2$ by following perturbation theory. In [275], the authors pointed out that if the clusters are well-separated then each pairwise distance should be choose so that the change on $\widetilde{v}_2$ is minimal. Their proposed algorithms are completely analogous to those of [236], except the key point described before. We note that, the algorithms from [236] and [275] are mainly designed for the binary spectral algorithm. However, they can be easily extended to the more general cases.

In [159], another active framework is developed for hierarchical clustering (it should be partition clustering instead). This framework repeatedly runs an off-the-shelf clustering algorithm on small subsets of the data and comes with guarantees on performance, measurement complexity and runtime complexity. The authors demonstrated their framework on spectral clustering and k-Means. For active spectral clustering, the authors showed that this algorithm recovers all clusters of size $\Omega(logN)$ using $O(Nlog^2N)$ pairwise similarities and runs in $O(nlog^3N)$ time complexity where N is the number of objects under some certain assumptions.

Other algorithms. Zhu et al. [300] proposed an approximation technique to estimate Dynamic Time Warping (DTW) distance between two time series based on its lower bounding and upper-bounding functions. The algorithm first initializes the distance matrix with this approximate distance function of DTW. Then it incrementally replaces the approximate distances with the true DTW distances in a best first order, i.e., the pair of time series with a higher ratio between Euclidean distance and lower bounding distance will be updated first. The goal is to quickly approximate as close as possible to the true DTW distance matrix with each distance update. This scheme can be use with any clustering algorithms which are based on the distance matrix, e.g., spectral clustering [263], hierarchical clustering [106], k-medoids clustering [106]. Generally, this algorithm could be regarded as an active clustering algorithm. However, it differs with the algorithms described above in several ways. First, this algorithm focuses on the change of distance matrix rather than cluster structure. Actually, maximizing a change in the distance matrix does not mean that the change in cluster structure is maximized as well and vice versa. Second, while each pairwise distance is iteratively evaluated and selected, this algorithm only ranks them one time in the beginning and selects them according to this ranking. In fact, this scheme makes the algorithm closer to an anytime algorithm than an active algorithm. Last, it requires the lower bounding and upper-bounding distances to be available while other active clustering algorithms do not since they start from an empty matrix.

In [22], the authors proposed a sampling technique to generates a hierarchy over a small set of random sample, which also implicitly represents a hierarchy over the entire data set. This technique is very useful when the amount of data is enormous such as in astrophysics and biology, under an assumption about the *good neighborhood properties* of data. Obviously, using a small part of data means that less distance calculations are performed. The proposed algorithm acquires the same goal however with different algorithmic scheme with the others. While other techniques like [236] try to maximize the change with each pairwise distance calculation iteratively, this technique does not operate in such the stepwise scheme. Thus, it is unable to cope with the budget problem. Moreover, it is not an incremental algorithm like active clustering algorithms described above.

4.2.4 Conclusions

Recently, advanced data acquisition techniques constantly produce data with increasing complexity. Together with the complexity of data, many difficulties have

arisen. Among those difficulties, one of the most challenging problems emerged recently is the distance sparseness problem.

Active clustering algorithms provide an efficient ways to cope with the distance sparseness problem. By allowing the algorithms to actively select the most informative pairwise distance to calculate, the total number of distance calculation is significantly reduced while ensuring good clustering results to be acquired. Due to its importance and interestingness, active clustering has become an emerging research in the literature and has attracted a lot of research efforts in recent years.

Though there exist in the literature active learning for k-Means, hierarchical clustering and spectral clustering. There is no active learning for density-based clustering proposed in the literature so far. Therefore, in this thesis, we introduce a first active density-based clustering algorithm called Act-DBSCAN [186] which is based on the DBSCAN paradigm.

Chapter 5

Anytime Density-based Clustering

Many clustering algorithms suffer from scalability problems on massive datasets and do not support any user interaction during runtime. To tackle these problems, anytime clustering algorithms are proposed. They produce a fast approximate result which is continuously refined during the further run. Also, they can be stopped or suspended anytime and provide an answer. In this Chapter, we present a novel anytime clustering algorithm based on the density-based clustering paradigm. Our algorithm called A-DBSCAN is applicable to many complex data such as trajectory and medical data. The general idea of our algorithm is to use a sequence of lower bounding functions (LBs) of the true distance function to produce multiple approximate results of the true density-based clusters. A-DBSCAN operates in multiple levels w.r.t. the LBs and is mainly based on two algorithmic schemes: (1) an efficient distance upgrade scheme which restricts distance calculations to core-objects at each level of the LBs; (2) a local reclustering scheme which restricts update operations to the relevant objects only. To further improve the performance, we propose a significant extension version of A-DBSCAN called A-DBSCAN-XS which is built upon the anytime scheme of A-DBSCAN and the μ-range query scheme of the extended Xseedlist. A-DBSCAN-XS requires less distance calculations at each level than A-DBSCAN and thus is more efficient. Extensive experiments demonstrate that A-DBSCAN and A-DBSCAN-XS acquire very good clustering results at very early stages of execution and thus save a large amount of computational time. Even if they run to the end, A-DBSCAN and A-DBSCAN-XS are still orders of magnitude faster than the original algorithm DBSCAN and its variants.

Publications. Parts of the material presented in this Chapter have been published in [184, 185]. The detailed information are described as follows:

- Son T. Mai, Xiao He, Jing Feng and Christian Böhm. Efficient Anytime Density-based Clustering. In SIAM International Conference on Data Mining (SDM), pages 112-120, 2013.

  In this work, S.T. M. contributed to the theory, implementation and experiment of the algorithm. C.B proposed an idea for the experiments. X.H. and J.F. helped with some experiments. The technique and results are discussed among all authors. All authors contributed to paper writing.

- Son T. Mai, Xiao He, Jing Feng, Claudia Plant and Christian Böhm. Anytime Density-based Clustering of Complex Data. Knowledge and Information System (KAIS), 2014. (accepted for publication).

  In this work, S.T.M. contributed to the theory, implementation and experiment of the algorithm. C.B. gave out an idea for the experiments. X.H. and J.F. participated in some experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

5.1 Introduction

Clustering is the task of assigning unlabeled objects into groups called clusters such that the similarity of objects within a group is maximized, and the similarity of objects between different groups is minimized. It plays a vital role for statistical data analysis in many fields including data mining, machine learning, pattern recognition, image analysis, information retrieval, etc. Although there are a vast amount of clustering algorithms proposed in the literature, most of them work in a *batch* scheme. They only produce a single result, and there is no interaction with end users during their executions.

For large databases, the idea of exploring the results during execution time has been proved to be a very useful approach [255, 300, 305]. The algorithms quickly produce an approximate result which is continuously improved over time and allow user interaction during their runtime. Users can terminate the algorithms anytime whenever they satisfied with existing results to save computation time. Moreover, the final results of these algorithms are often very similar to those of the

batch algorithms. Such algorithms are called *anytime* algorithms [255, 300, 305] and are an emerging research in many fields of data mining such as classification [232, 255, 149] and outlier detection [20, 19]. However, anytime clustering has not been paid enough attention. There are only a few works on anytime clustering algorithms, e.g., I-kMeans [173, 174].

Among various kinds of clustering algorithms such as partitioning methods and hierarchical methods, density-based clustering algorithms have attracted many attention in the data mining community due to their advantages compared with the others [228, 83, 111]. They can detect clusters of arbitrary shapes, do not require the number of clusters to be specified, and are robust to outliers. Besides many others, the density-based notion underlying the algorithm DBSCAN [83] is one of the most successful approaches to clustering with applications in many fields such as neuroscience and meteorology. Many clustering algorithms are successfully proposed based on this notion [16, 228, 83]. However, all of them only work in the batch scheme.

Contributions. Our contributions are summarized as follows:

1. In this Chapter, we propose for the first time a novel anytime clustering algorithm based on the cluster notion of DBSCAN [83]. Our algorithm called anytime density-based clustering (A-DBSCAN) is applicable to many complex data such as trajectory and medical data. The core idea of A-DBSCAN is to use a sequence of lower bounding functions (LBs) of the true distance function to produce multiple approximate results of DBSCAN. LBs are well-studied in the field of database indexing [216]. However, their applications in clustering have gained much less attention. By using LBs as distance measures, we are not only able to approximate the true clustering result but also speed up the algorithm significantly since the LBs often run very fast compared with the true distance function. A-DBSCAN operates in multiple levels w.r.t. the sequence of LBs. The result of each level is calculated by using the results of the previous levels. We propose efficient distance and cluster update schemes from level to level based on theoretical study of the way clusters change under the effect of LBs.

2. In order to further improve the efficiency, we propose a significant extension version of A-DBSCAN called A-DBSCAN-XS which is built upon the anytime scheme of A-DBSCAN and the μ-range query scheme of our extended Xseedlist. Compared with A-DBSCAN, A-DBSCAN-XS has better distance

calculation pruning power at each level. Hence it is more efficient than A-DBSCAN, especially when dealing with very expensive distance functions.

3. We theoretically prove that the final clustering results of our anytime algorithms A-DBSCAN and A-DBSCAN-XS are identical to that of DBSCAN.

4. Extensive experiments on real datasets such as time series and trajectories demonstrate that A-DBSCAN and A-DBSCAN-XS acquire very good clustering results at very early stages of execution, thus saving a large amount of runtime. Even if A-DBSCAN and A-DBSCAN-XS are run to the end, they are still orders of magnitudes faster than the algorithm DBSCAN and its variants. Such advantages are impressive since A-DBSCAN and A-DBSCAN-XS, as anytime algorithms, must perform clustering many times compared with only one time of the batch algorithm DBSCAN and its variants.

The rest of this Chapter is organized as follows. In Section 5.2, we briefly present some background. In Section 5.3, we delineate our anytime clustering algorithm A-DBSCAN. Section 5.4 propose an extension of A-DBSCAN called A-DBSCAN-XS. The distance measure and lower bounding functions are briefly described in Section 5.5. Section 5.6 reports experimental results. Section 5.7 is dedicated to related work and discussion. Section 5.8 concludes with a summary and future research.

5.2 Backgrounds

In order to enhance the readability, we briefly repeat some important backgrounds in this Section, though they are presented in previous Chapters.

5.2.1 Anytime Clustering

An anytime clustering algorithm works by trading execution time for quality of results [300, 305]. Anytime clustering produces a fast approximate result which is then refined during the further run. Users can examine the intermediate clustering results while the algorithm is continuing to produce the finer results at the next levels. According to [300, 305], an anytime clustering algorithm should satisfy some important properties such as:

1. It should produce good results which are close to the result of the batch algorithm at some early stages.

2. The final result should be similar to or better than the batch algorithm.

3. The total cumulative runtime of the algorithm should be only slightly larger than the batch algorithm.

5.2.2 The Algorithm DBSCAN

In density-based clustering, clusters are considered as areas of high object density separated by areas of low object density in the data space. The key idea of density-based clustering algorithm DBSCAN [83] is that the cardinality of the neighborhood of each object of a cluster has to exceed a predefined threshold.

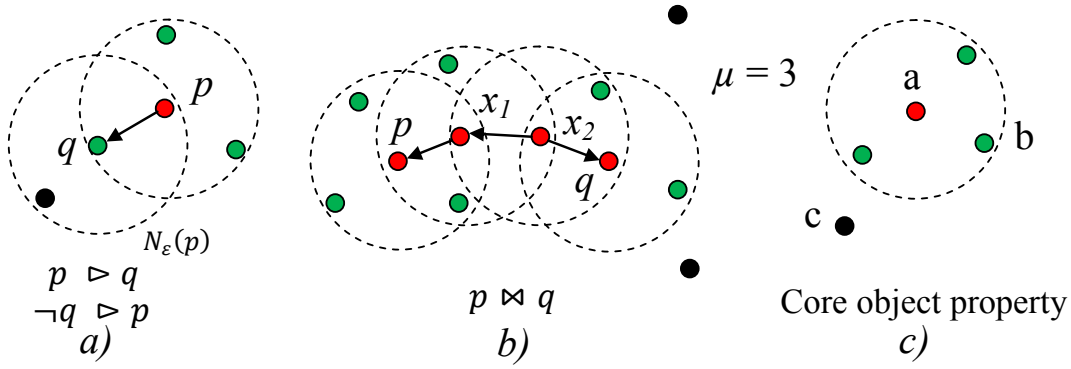

Figure 5.1: The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object;

Given a set of objects O, a distance function $d : O \times O \rightarrow R$ and two parameters $\epsilon \in R^+$ and $\mu \in N^+$.

Definition 6 *(ϵ-neighborhood) The ϵ-neighborhood of $p \in O$, denoted as $N_\epsilon(p)$, is defined by $N_\epsilon(p) = \{q \in O | d(p,q) \leq \epsilon\}$.*

Each object in O is classified either as core object, border object or noise object.

Definition 7 *(Core object property) An object $p \in O$ is a:*

1. *Core object, denoted as* $core(p)$*, iff* $|N_\epsilon(p)| \geq \mu$.
2. *Border object, denoted as* $border(p)$*, iff* $|N_\epsilon(p)| < \mu \wedge \exists q \in N_\epsilon(p) : |N_\epsilon(q)| \geq \mu$.
3. *Noise object, denoted as* $noise(p)$*, iff it is not a core object or a border object.*

Definition 8 *(Directly density-reachable) An object* $q \in O$ *is directly density-reachable from object* $p \in O$*, denoted as* $p \triangleright q$*, iff* $|N_\epsilon(p)| \geq \mu$ *and* $q \in N_\epsilon(p)$.

Definition 9 *(Density-connected) Two object* p *and* $q \in O$ *are density-connected, denoted as* $p \bowtie q$*, iff there exists a sequence* $(x_1, \ldots, x_m)$ *of objects such that* $\forall x_i : |N_\epsilon(x_i)| \geq \mu$ *and* $p \triangleleft x_1 \triangleleft \cdots \triangleright x_m \triangleright q$.

A cluster is defined as a maximal set of density-connected objects and is composed of core objects and border objects. A border object could belong to several clusters depending on the order of objects.

Definition 10 *(Cluster) A subset* $C \subseteq O$ *is called a cluster iff the two following conditions hold:*

1. *Maximality:* $\forall p \in C, \forall q \in O \setminus C : \neg p \bowtie q$
2. *Connectivity:* $\forall p, q \in C : p \bowtie q$

Figure 5.1 demonstrates some notions of DBSCAN. DBSCAN uses a data structure called the seed list S which contains a set of seed objects for cluster expansion. To construct a cluster, DBSCAN continuously extracts objects from S and performs the ϵ-range query to find neighbor objects and inserts them into S until S is empty.

5.3 Anytime Density-based Clustering

Given a set of objects O and a set D of n lower bounding functions $D = \{d_i | d_i : O \times O \rightarrow R \wedge \forall p, q \in O : d_i(p, q) \leq d(p, q) \wedge d_n(p, q) = d(p, q)\}$. Our algorithm A-DBSCAN works in a sequence of levels from L_1 to L_n. At each level L_i, the

clustering is performed by using the function d_i as the distance measure. Note that, d_{i+1} should be a tighter bound of d than d_i.

The naive approach. A naive algorithm should calculate the new distances between all objects and perform the clustering on the whole dataset at each level. Though it is simple, it is inefficient since the properties of LBs and DBSCAN are not exploited at all. Using the results at L_i to restrict the calculation in L_{i+1} is thus a reasonable approach to speed up the algorithm.

Our approach. A-DBSCAN maintains a neighborhood graph $G = (O, E)$ which connects each object $p \in S$ with objects in its ϵ-neighborhood. In the beginning, the graph G is fully connected. At each level, the graph G is updated to reflect the changes in the neighborhoods of objects w.r.t. the used distance function.

At level L_{i+1}, we define the ϵ-neighborhood of object p w.r.t. the graph G at L_i (G^i) as follows:

Definition 11 *(ϵ-neighborhood w.r.t. G^{i+1}) The ϵ-neighborhood of p at L_{i+1}, denoted as $N_\epsilon^{i+1}(p)$, is defined by $N_\epsilon^{i+1}(p) = \{q \in O | (p,q) \in E^i \wedge d_{i+1}(p,q) \leq \epsilon\}$.*

Following Definition 11, the graph G^{i+1} is created by removing every edge $(p,q) \in E^i$ which $d_{i+1}(q,p) > \epsilon$ from G^i. It is more efficient than the naive approach since only a part of the distances between objects must be updated at each level instead of all distances. However, while the neighborhoods of objects in naive algorithm at L_{i+1} depend only on the distance function d_{i+1}, the notion of the neighborhood of A-DBSCAN considers not only the current distance d_{i+1} but also the distances d_i at previous levels represented by the neighborhood graph G. As we shall see, this scheme is the heart of A-DBSCAN that allows it to be used with arbitrary sequences of LB distance functions, thus enhancing the applicability of our algorithm.

How the clusters change. A-DBSCAN works by exploiting the way the clusters change at each level. Assuming that we are currently at level L_{i+1}.

Lemma 2 *For every object $p \in O$ the neighborhood of p at L_{i+1} is a subset of the neighborhood of p at L_i.*

Proof 2 *Straightforward from Definition 11.*

According to Lemma 2, the neighborhood of each object p decreases at each level. Thus, from L_i to L_{i+1}, the core property of each object changes as the following:

Lemma 3 *From level L_i to L_{i+1}.*

1. *If p is a core object in L_i then it is a core, a border or a noise object in L_{i+1}.*
2. *If p is a border object in L_i then it is a border or a noise object in L_{i+1}.*
3. *If p is a noise object in L_i then it remains a noise object in L_{i+1}.*

Proof 3 *According to Lemma 2 and Definition 7, a core object p will become a border or a noise object if $|N_\epsilon^{i+1}(p)| < \mu$. A border object p will never be a core object because its neighborhood size never increases. But it will become a noise object if it does not have any core object in its neighbors. Since the neighborhood of a noise object p does not contain any core objects, it will remain a noise object in L_{i+1}.*

In DBSCAN, each cluster contains two kinds of objects: core and border objects and the core objects play a critical role to determine clusters. Because of their importance, we define a core cluster as follows:

Definition 12 *(Core Cluster) A subset $C \subseteq O$ is called a core cluster iff $\forall p \in C : core(p) \wedge C \subseteq \xi : \xi$ is a cluster.*

Lemma 4 *For all objects $p, q \in O$, if p and q are not density-connected at L_i (denoted as $\neg p \triangleright^i q$) then they are not density-connected at L_{i+1}.*

Proof 4 *According to Lemma 2 and 3, if $p \triangleright^{i+1} q \Rightarrow p \triangleright^i q$. Assuming that $p \bowtie^{i+1} q$, there exists a sequence of objects $(x_1, \ldots, x_m)$ so that $p \triangleleft x_1 \triangleleft \cdots \triangleright x_m \triangleright q$ at L_{i+1} according to Definition 9. Thus, $p \triangleleft x_1 \triangleleft \cdots \triangleright x_m \triangleright q$ at L_i. Therefore, we have $p \bowtie^i q$.*

Lemma 5 *For every core cluster C_u at L_{i+1} (denoted as C_u^{i+1}), there exists a core cluster C_v at L_i which $C_u \subseteq C_v$.*

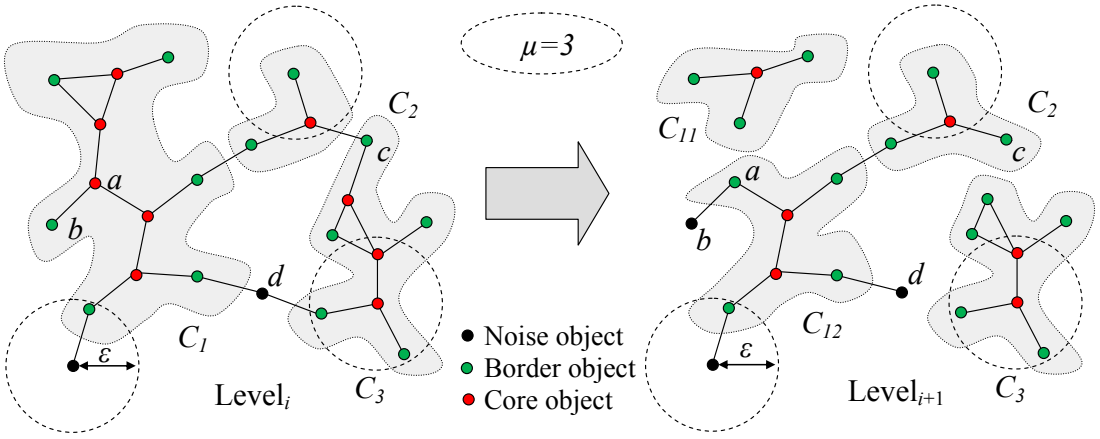

Figure 5.2: From level L_i to L_{i+1}, the core object a changes to border object. Object b changes from border to noise. The noise object d remains unchanged. Cluster C_1 at L_i is broken into two clusters C_{11} and C_{12} at L_{i+1}. Object c becomes border of cluster C_2 at the new level.

Proof 5 *Assuming that there exist* $p, q \in C_u^{i+1}$ *so that* $p \in C_k^i$ *and* $q \in C_l^i$. *From Definition 10 and Lemma 4, we have* $\neg p \bowtie q$ *at* $L_i \Rightarrow \neg p \bowtie q$ *at* L_{i+1}. *Thus,* p *and* q *belong to different clusters by Definition 10.*

Figure 5.2 illustrates the changes of clustering results and core properties of objects from level L_i to level L_{i+1}. A cluster at L_i may break into several smaller clusters at L_{i+1} following the division of its core cluster. The border objects of a cluster may be possessed by other clusters at the next level due to the changes of core properties of its neighbors, even if this cluster does not split. Such property of A-DBSCAN reminds us the monotonicity property of the subspace clustering algorithm SUBCLU [160]. Under the Euclidean distance (ED), the distances in subspace projections of the data could be consider as a sequence of increasing lower bounding distances ($\forall p, q \in O : d_i(p, q) \leq d_{i+1}(p, q)$), which is a special case of A-DBSCAN. By exploiting a graph structure, A-DBSCAN acquires the monotonicity property even for arbitrary sequences of LBs and is not restricted to ED like SUBCLU. Therefore, the applicability of the algorithm is increased. We also note that, SUBCLU is not an anytime algorithm.

To conclude, the changes of clustering results of A-DBSCAN are monotonic w.r.t. the changes of graph G at each level.

Anytime DBSCAN Algorithm. Our anytime algorithm is based on the notions

of DBSCAN. Let us start with a new enhanced notion for a cluster:

Definition 13 *(The border of a core cluster) A set B of border objects is called the border of a core cluster C iff $\forall p \in B$: (1) $\exists q \in C : q \triangleright p$ (2) $\neg\exists q \in O \setminus C$: $|N_\epsilon(q)| \geq \mu \wedge \forall r \in C : d(p,q) < d(p,r)$.*

Definition 14 *(A-Cluster) An anytime cluster C is an union of a core cluster C_c and its border C_b.*

Here, a border object is assigned to its nearest core object instead of being randomly assigned in DBSCAN. This brings up some benefits: (1) the compactness of clusters and thus the clustering quality are enhanced; (2) the cluster result does not depend on the the order of the input data. In our experiments in Section 5.6, this new notion helps to slightly improve the clustering results measured by NMI scores [258] on 13/32 datasets acquired from the UCR archives [1] while having the same results on the others [184].

Since A-DBSCAN operates in multiple levels, we have to efficiently solve 2 problems at each level:

1. How to upgrade the graph G?
2. How to perform the clustering?

According to Definition 11, we have to update the whole graph G from L_i to L_{i+1}. It is more efficient than the calculation of the distances between all objects in the naive approach. However, Lemma 2 to 5 suggest a more efficient way as follows. In graph G, there exist five kinds of edges: core-core, core-border, border-border, border-noise and noise-noise (the core-noise edges do not exist according to Definition 7). Since the edges between border and noise objects do not involve in clustering process, they can be safely ignored to save computation cost. Therefore, we just need to update the parts which involve the core objects: the core-core and core-border edges. This update scheme significantly reduces the cost of graph construction at each level, especially when the number of core objects is small. In other words, we only consider the subgraph with core-core and core-border edges. Note that, the graph G will no longer reflect the neighborhoods of all objects exactly. However, it will not affect the correctness of our algorithm as shown below.

[1] http://www.cs.ucr.edu/∼eamonn/time_series_data/

For clustering algorithm, some clusters may be split but there is no merging of clusters from L_i to L_{i+1} according to Lemma 5.

Corollary 1 *From level L_i to L_{i+1}, an A-cluster C_k may be split if:*

1. $\exists u, v \in C_k : (u, v) \in E^i \setminus E^{i+1} \wedge core(u) \wedge core(v)$.
2. $\exists u \in C_k : core(u)$ *at* $L_i \wedge \neg core(u)$ *at* L_{i+1}.

Corollary 1 is directly inferred from Definition 9 and 14. The deletion of an edge between two core objects or degradation of a core object in an A-cluster would break the density-connectivity of its core cluster, thus causing the splitting of the A-cluster. Assuming that this happened in cluster C_k at L_i, all we need is to recluster all the core objects in C_k instead of the whole dataset as in naive approach, thus saving significant amount of time (roughly $O(|C|^2)$ time with C is set of clusters). After that, all the border points are reassigned to the cluster labels of their nearest core objects following Definition 14. In case we use the original cluster notion of DBSCAN, only some border objects need to be reassigned, which is faster but may be less effective in terms of clustering quality.

Figure 5.3 shows the pseudocodes for the algorithm A-DBSCAN. For every level, first the graph G is updated. After that, we check all clusters to see if there are splitting possibilities, and recluster all of their core clusters to reflect the changes using the clustering scheme of DBSCAN again. All the border objects will then be added to clusters following their nearest core objects.

Correctness of algorithm. We show that, the final clustering result of A-DBSCAN is identical to that of DBSCAN except for those objects which can be border objects of two or more different clusters.

Definition 11 guarantees that the final neighborhoods of objects of A-DBSCAN are similar to DBSCAN. The graph update scheme ignores only the edges between noise and border objects which do not play any role to determine core properties of objects and density-connectedness of clusters. Therefore, the core properties of objects and the core clusters of A-DBSCAN at the last level are identical to those of DBSCAN. Since the border objects are assigned to their nearest core objects as stated in Definition 14, they are the only difference between A-DBSCAN and DBSCAN since the border objects of DBSCAN are assigned based on the order of objects. If we use original notion of cluster of DBSCAN then the final results of A-DBSCAN and DBSCAN are totally identical.

```
Function A-DBSCAN (O, μ, ε, F)
    G = fully connected graph
    C = {O};
    for all level L_i do
        % update the graph G
        for all core object p in O do
            for all adjacent core or border object q of p do
                e_pq = d_{i+1}(p,q)   // update edge e_pq
                if e_pq > ε then remove e_pq from G endif
            endfor
        endfor
        update the core property of objects
        % re-clustering the dataset O
        for all cluster C_i in C do
            if C_i may be split then
                recluster all core object in C_i and save the
                    changes into C again
            endif
        endfor
        for all border objects p do
            reassign cluster label for p according to the
            label of its nearest core object
        endfor
        return set of cluster C at level L_i
    endfor
    return final set of clusters C
EndFunction
```

Figure 5.3: Pseudocodes for A-DBSCAN.

A-DBSCAN and the naive algorithm. At a middle level L_i, the neighborhoods of objects of A-DBSCAN are not identical to those of the naive algorithm due to the new neighborhood notion in Definition 11. Therefore, the clustering result of A-DBSCAN at L_i is actually an approximation of the clustering result under the distance function d_i. They are identical if and only if D contains a sequence of increasing LBs ($\forall p, q : d_i(p, q) \leq d_{i+1}(p, q)$). Such condition, however, restricts the applicability of the algorithm since it is harder to satisfy than finding an arbitrary sequence of LBs.

Complexity analysis. In theory, time complexity of an anytime algorithm is usually higher than the batch one since it has to run many times. In our setting,

the complexity of the naive approach and A-DBSCAN is $\sum_{i=1}^{|D|} \theta_i |O|^2$ (where θ_i is complexity of d_i). However, since A-DBSCAN has efficient graph update and reclustering scheme as described above, A-DBSCAN is much faster than DBSCAN as we shall see in Section 5.6.

5.4 Extended A-DBSCAN

5.4.1 The Xseedlist

There exists several techniques to speed up DBSCAN in the literature by accelerating the ϵ-range query process [83]. The algorithm B-DBSCAN [45], in contrast, relies on the μ-range query to determine the core property of objects rather than the ϵ-range query. Thus it does not require calculating all the exact distances between p and its neighbors $N_\epsilon(p)$ like others. Therefore, the efficiency of the algorithm is much improved. B-DBSCAN uses a lower bounding function of the true distance measure and a data structure called the Xseedlist to perform the clustering.

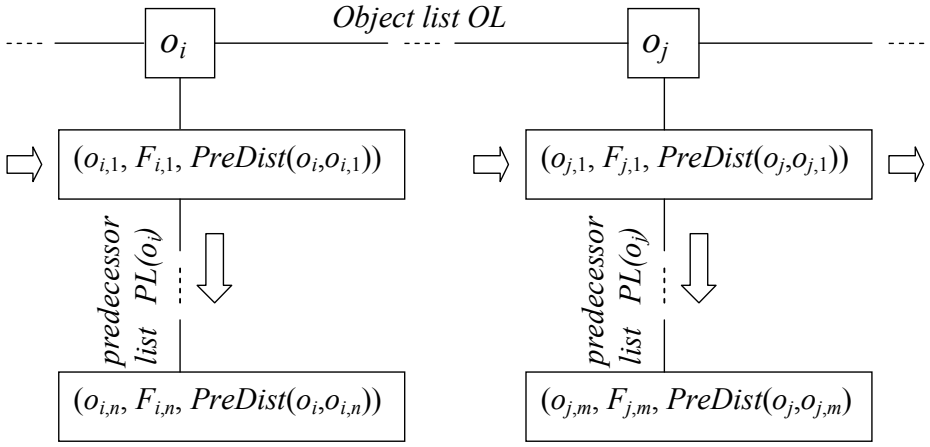

Figure 5.4: The data structure Xseedlist.

Figure 5.4 shows the data structure Xseedlist. It consists of an ordered object list OL. Each object o_i in OL is associated with a predecessor list $PL(o_i)$ and is sorted by the first element of $PL(o_i)$. Each entry of $PL(o_i)$ contains 3 items: (1) PreID: ID of a neighbor object $o_{i,k}$ $(1 \leq k \leq n)$; (2) PreFlag: indicates that the distance between o_i and $o_{i,k}$ is the LB or the true distance; (3) PreDist: the distance between o_i and $o_{i,k}$. The Xseedlist operates by using a LB distance as a

guide line to extend the clusters. For every object p, the Xseedlist determines all of its neighbors under the LB distance, sorts them and then updates the distance between p and its neighbors with the true distances until the core property of p is determined. The others will be updated only when they are necessary to determine the density connectivity of objects during the cluster expansion. Due to space limitation, interested readers please refer to [45] for more details.

Compared with the original Seedlist [83], Xseedlist can help to reduce the total number of used true distances up to orders of magnitudes [45]. However, it still suffers from several drawbacks: (1) It requires the full LB distances between all objects to be calculated; (2) It needs to sort the object list and predecessor list many times during the clustering process. Thus, the use of Xseedlist pays off only when the runtime difference between the LB and true distance is significant; (3) It is originally designed to work with a single LB distance. Though it can be extended as shown in Section 5.6 to work with multiple LBs, all the LB distances still have to be calculated.

5.4.2 The Algorithm A-DBSCAN-XS

In this part, we present our anytime clustering algorithm called A-DBSCAN-XS which is a significant extension of A-DBSCAN described above [184]. The general idea is to integrate the scheme of the μ-range query of the Xseedlist into the reclustering process of A-DBSCAN to update the cluster structure from L_i to L_{i+1}. By this way, we can reduce the number of calculated distances at each level thus significantly enhance the efficiency of the algorithm. A naive approach could use the Xseedlist with the distance function d_{i-1} as the LB distance for the distance function d_i to speed up the clustering process at level L_i. However, it is impossible since the LB function d_{i-1} is generally not a true lower bound of d_i in our setting ($\exists p, q \in O : d_{i-1}(p,q) > d_i(p,q)$), while the Xseedlist requires the LB property to operate. Even in a special case when d_{i-1} lower bounds d_i, directly applying the Xseedlist to recluster a cluster C in L_i is inefficient because the distance function d_{i-1} must be calculated for every pair of objects in O. Thus, we only save the distance calculations for the last distance function d_n (or d).

To the rest of this Section, we present how the Xseedlist can be extended and integrated with the anytime clustering scheme of A-DBSCAN to save the distance calculations for all d_i.

The extended graph G. Since our algorithm tries to reduce the number of

calculated distances at each level, the neighborhood graph G should be only partly updated from L_i to L_{i+1}. Therefore, for each edge (p, q), we additionally store the level $lev((p, q))$ at which it is already updated and the distance $dis((p, q)) = d_{lev((p,q))}(p, q)$. The main difference between A-DBSCAN-XS and A-DBSCAN is that all the edges related to the core objects must be updated at each level in A-DBSCAN while only a part of these edges is updated in A-DBSCAN-XS at each level. The extended neighborhood graph G plays a key role to integrate the Xseedlist into the anytime clustering scheme of A-DBSCAN.

The extended Xseedlist. The original Xseedlist [45] must be modified to work with our anytime clustering scheme. In our extended Xseedlist, the PreFlag and PreDist of $PL(o_i)$ will contain the value of $lev((o_i, o_{i,k}))$ and $dis((o_i, o_{i,k}))$. The order condition of the Xseedlist also needs to change to enhance the efficiency. For every pair of edges (p, q) and (r, s), we define a comparison function $\phi((p, q), (r, s))$ between them as follows:

$$\phi = \begin{cases} > \ if \ lev((p,q)) < lev((r,s)) \\ < \ if \ lev((p,q)) > lev((r,s)) \\ \begin{cases} > \ if \ dis((p,q)) > dis((r,s)) \\ = \ if \ dis((p,q)) = dis((r,s)) \\ < \ if \ dis((p,q)) < dis((r,s)) \end{cases} otherwise \end{cases}$$

The intuitive of this comparison function is that if an edge (p, q) is currently at higher level than (r, s), it should be considered to update first since we only need to calculate only $L_i - lev((p, q))$ distance functions to reach level L_i. Therefore, the number of calculated distances at each level is reduced. In our extended Xseedlist, we replace all the order conditions of the original Xseedlist which are only based on the distances of $((p, q))$ and $((r, s))$ by our new comparison function. As a result, the total number of calls for each function d_i is significantly reduced compared with the original Xseedlist as shown in Figure 5.9.

The algorithm A-DBSCAN-XS. We now can extend the clustering process of A-DBSCAN to work with the scheme of our extended Xseedlist. We called our algorithm A-DBSCAN with Xseedlist (A-DBSCAN-XS).

Figure 5.5 shows the pseudocode of our algorithm. Unlike A-DBSCAN, A-DBSCAN-XS does not separate the update process into two distinguish parts: distance update and cluster update. They are performed simultaneously during the clustering process instead. A-DBSCAN-XS starts with the extended neighbor-

```
Function A-DBSCAN-XS (O, μ, ε, D)
    Input: Dataset O, parameter μ and ε, a set of n functions D
BeginFunction
    G = ε-neighborhood graph with the LB function d1 at L1
    C = set of clusters provided by original DBSCAN on G
    for all level Li (2 ≤ i ≤ n) do
        for all cluster Ci in C do
            C = Update-Cluster (core(Ci), μ, ε, G, Li)
        endfor
        for all border objects or unclassified objects p do
            for all core object q in Nε(p) do
                Update-Edge ((p,q), ε, G, Li)
            endfor
            assign p to the cluster label of nearest core object
            or assign p as noise if there is no core object nearby
        endfor
        return set of cluster C at level Li
    endfor
EndFunction
```

```
Function Core-Check (q, Nε(q), μ, ε, G, Li)
    Input: object q and its neighbor Nε(q), parameter μ, ε,
           graph G and level Li
BeginFunction
    count = 0
    for all object p in Nε(q) do
        d = Update-Edge((p, q), ε, G, Li)
        if d < ε then count = count + 1 endif
        if count = μ then break endif
    endfor
    return count
EndFunction
```

```
Function Expand-Xseedlist (OL, C, q, Nε(q), μ, ε, G, Li)
    Input: Xseedlist OL, object list C, object q and its
           neighbor Nε(q), parameter μ, ε, graph G and level Li
BeginFunction
    for all object o in Nε(q) do
        if o not in C then continue endif
        if lev((q, o)) = Li and dis(q, o) ≤ ε then
            put (o, NIL) to beginning of OL and remove the old
                entry (if exists)
        else
            if entry o is not exist in OL then
                insert (o,(q, lev(o,q), dis(o,q)) into OL and
                    reorder OL according to the function φ
            else
                if PL(o) ≠ NIL then
                    insert (q,lev(o,q), dis(o,q)) into PL(o) and
                        reorder OL according to the function φ
                endif
            endif
        endif
    endfor
EndFunction
```

```
Function Update-Edge ((p,q), ε, G, Li)
    Input: Edge (p,q) of G, parameter ε, graph G and level Li
BeginFunction
    for j = lev(p,q) + 1 to Li do
        if dj(p,q) > ε then break endif
     if j = Li then lev(p,q) = Li and dis(p,q) = dl(p,q)
    else remove (p,q) from G
    endif
    return dis(p,q)
EndFunction
```

```
Function Update-Cluster (C, μ, ε, G, Li)
    Input: List of object C, parameter μ, ε, graph G and level Li
BeginFunction
    create an empty Xseedlist OL
    clear the cluster id of all object q in C
    cidnow = get a free cluster label
    for all object q in C do
        sort Nε(q) in ascending order according to function φ
        t = Core-Check (q, Nε(q), μ, ε, G, Li)
        if t < μ then set q as noise and continue the for loop endif
        set cluster id of q as cidnow and mark q as core object in Li
        Expand -Xseedlist (OL, C, q, Nε(q), μ, ε, G, Li, OL)
        while OL is not empty do
            o1 = get first object from OL
            if PL(o1) = NIL then
                set cluster id of o1 as cidnow and remove o1 from OL
                sort Nε(o1) in ascending order according to function φ
                t = Core-Check (o1, Nε(o1), μ, ε, G, Li)
                if t < μ then
                    set o1 as border object and continue while loop
                endif
                set o1 as a core object at Li
                Expand -Xseedlist (OL, C, o1, Nε(o1), μ, ε, G, Li, OL)
            else
                o1,1 = PreID of the first object in PL(o1)
                d = Update-Edge ((o1,o1,1), ε, G, Li)
                if d < ε then
                    set cluster id of o1,1 as cidnow and remove o1 from OL
                    sort Nε(o1) in ascending order according to function φ
                    t = Core-Check (o1, Nε(o1), μ, ε, G, Li)
                    if t < μ then
                        set o1 as border object and continue while loop
                    endif
                    set o1 as a core object at Li
                    Expand -Xseedlist (OL, C, o1, Nε(o1), μ, ε, G, Li, OL)
                else
                    if PL(o1).length > 1 then delete entry of o1,1 from PL(o1)
                    else delete o1 from OL
                    endif
                endif
            endif
        endwhile
        cidnow = get new cluster label
    endfor
    return new cluster membership for all object in C
Endfunction
```

Figure 5.5: Pseudocode for the algorithm A-DBSCAN-XS.

hood graph G and the cluster structure provided by performing DBSCAN with the first LB function d_1 as the main distance measure. At each level L_i, A-DBSCAN-XS performs the reclustering on the set of core objects of each cluster following the monotonicity property described in Section 5.3. Note that, A-DBSCAN only reclusters the clusters which may be split. Therefore, in terms of reclustering time, A-DBSCAN is more efficient than A-DBSCAN-XS. However, A-DBSCAN-

XS has much better distance pruning power, thus it is much more efficient than A-DBSCAN as we shall see in Section 5.6. After all the clusters are reclustered, A-DBSCAN-XS reassigns each border object to its nearest core object as in A-DBSCAN.

A-DBSCAN-XS relies on two main subroutines namely Update-Edge and Update-Cluster. To update an edge (p, q) to the current level L_i, we check all the functions d_j with $lev((p, q)) < j \leq L_i$ one after another. If $d_j(p, q) > \epsilon$, the edge (p, q) is removed from graph G. Otherwise, we set the new level for (p, q) as L_i ($lev((p, q)) = L_i$). The reclustering process still follows the basic clustering routine proposed in [45] but with some changes summarized as follows: (1) There is no range query with LB function. At level L_i, $|N_\epsilon(p)|$ is selected by all the neighbors of p in graph G. Therefore, for all object $q \in |N_\epsilon(p)|$, $lev(p, q)$ will be arbitrary in $[1, L_{i-1}]$, instead of L_{i-1}. This is a significant advantage since we can save a lot of distance calculation compared with the naive use of Xseedlist; (2) We used the extended Xseedlist described above instead of the original Xseedlist; (3) The procedure Update-Edge is called to update the distance of each edge to the current function d_i at L_i instead of the true distance function call; (4) Some other minor changes are also implemented to fit the extended Xseedlist with the anytime clustering scheme.

Correctness of the algorithm. We prove that, the result of A-DBSCAN-XS is identical to that of A-DBSCAN at every level.

First, due to the partly update scheme of graph G, for every object p at L_i, the neighborhood $|N_\epsilon(p)|$ acquired with A-DBSCAN-XS will be a superset of $|N_\epsilon(p)|$ acquired with A-DBSCAN. Since the Core-Check function examines all objects inside $|N_\epsilon(p)|$ until it finds μ objects q which has $d_j(p, q) \leq \epsilon$ $(1 \leq j \leq i)$, the core property of p is identical with both algorithm. Second, the Update-Cluster function guarantees that an object o will be added to the current cluster if $d_j(p, o) \leq \epsilon$ $(1 \leq j \leq i)$. Thus, the cluster structures of A-DBSCAN-XS and A-DBSCAN are identical at every level.

Complexity analysis. Since Xseedlist uses a sorting procedure to perform, the time complexity of A-DBSCAN-XS will be $\sum_{i=1}^{|D|} \theta_i |O|^2 + |O|^2 \log |O|$ (where θ_i is the time complexity of d_i). However, in case the distance functions d_i are expensive, the sorting cost will become negligible compared with the reduction of the distance calculation cost. In this case, A-DBSCAN-XS enjoys dramatic performance acceleration due to its efficient distance pruning scheme.

5.5 Similarity Measure and Lower bounding

Since A-DBSCAN is a general framework. It can be used with any kind of distance functions and their LBs. Given a distance function d, providing a set of lower bounding function $D = \{d_i \| d_i : O \times O \rightarrow R \wedge \forall p, q \in O : d_i(p,q) \leq d(p,q) \wedge d_n(p,q) = d(p,q)\}$ is an essential problem in our approach. In order for A-DBSCAN to work properly, these conditions should be fulfilled: (1) d_i should be faster than d; (2) d_{i+1} should be tighter than d_i ($d_i(p,q) \leq d_{i+1}(p,q)$ in general).

Euclidean distance. Recent research has introduced many distance measures for complex data such as Euclidean Distance (ED), Longest Common Subsequence (LCS) and Dynamic Time Warping (DTW) [74, 195]. Among them, ED is the most widely used one due to its simplicity and ubiquitousness.

Given two objects $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\} \in O$, we have: $d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$.

Though our algorithm can be used with all mentioned distance functions above, we simply choose ED as a representative to demonstrate our algorithm.

The Haar wavelet transform. Lower bounding functions for the ED are well-studied in the literature. There exist many proposed techniques such as Piecewise Aggregate Approximation (PAA) [174], Discrete Fourier Transform (DFT) [173] and Discrete Wavelet Transform (DWT) [216]. All of these techniques can be directly applied to A-DBSCAN. In this work, we simply choose DWT as a representative to build a sequence of LBs. Interested reader please refer to 3 for more details on DWT.

To build a sequence of LBs, we first transform all objects using DWT. Then, at each level L_i, we only use the first k_i coefficients of each object to calculate the distance function d_i with $k_i \leq k_{i+1}$. According to Lemma 1, the lower bounding condition 2 holds. Since we only need a few coefficients to have a good approximation of the original distance function, the runtime of the distance functions d_i is significantly faster than the original ED function d, which satisfies the runtime condition 1. Due to linear complexity of DWT, the time needed to transform the whole dataset S with dimensionality n is $O(n|O|)$ which is negligible compared with $O(n|O|^2)$ for DBSCAN. For a large time series dataset with 9236 objects with the length of each object $n = 8192$, it costs only 2.1 seconds to transform the whole dataset and an hour for clustering with DBSCAN.

5.6 Experiments

All experiments are conducted on a Workstation with 3.0 Ghz CPU, 8GB RAM under Windows Server 2008 using Java.

5.6.1 Evaluation Methodology

Datasets. We evaluate the performance of our algorithms on real datasets acquired from different sources, including:

- The eight real datasets from the UCR archives [131] (http://www.cs.ucr.edu/-~eamonn/time_series_data/) which contain small to large time series data from diverse fields. They are Mallat, CinC_ECG_torso, SonyAIBORobotSurface, Plane, ECGFiveDays, Symbols, DiatomSizeReduction and OliveOil.
- The two datasets Character Trajectory (CT) and Australian Sign Language (ASL) from the UCI archives [90] (http://archive.ics.uci.edu/ml). The CT dataset contains 2858 2D trajectories belonging to 20 different characters. The ASL contains 5 sign samples collected from five signers as 2D trajectories. Also, we add to ASL some 5% interpolated Gaussian noise and local time shifting samples as in [173] to form a dataset with 1050 2D trajectories.
- The COIL20 dataset acquired from the Columbia Object Image Library (http://www1. cs.columbia.edu/CAVE/software/softlib/coil-20.php). This dataset contains 1440 pictures of 20 different objects with 72 pictures per objects.

We note that the UCR and UCI datasets are re-interpolated to the length of $2^{\lfloor \log(n) \rfloor + 3}$ to use with DWT. However, it does not affect the evaluation of our algorithm since all the comparable algorithms described below produce almost identical results with A-DBSCAN and A-DBSCAN-XS (please refer to Section 5.3 and 5.4 for more details). Also, in our experiments on 32 UCR datasets, the change of the clustering qualities between the original and the re-interpolated datasets are negligible.

Algorithms. We compare our algorithms A-DBSCAN and A-DBSCAN-XS with 3 different variants of DBSCAN proposed in the literature [45, 83] including:

- The original DBSCAN algorithm proposed by Ester et al. [83].

- DBSCAN with multi-levels filter and refinement range query (M-DBSCAN) which uses LBs to speed up the range query process of DBSCAN as stated in [45]. For all objects p and q, we check all the LBs from d_1 to d_n (d_n is the true distance as defined in Section 5.3). If $d_i(p,q) > \epsilon$ then $d(p,q) > \epsilon$ since $d_i(p,q) \leq d(p,q)$. Thus q is not an ϵ-neighbor of q and vice versa. Since the runtime of LBs are smaller than the real one, this scheme can significantly speed up the range query process [227] and thus reduces the runtime of DBSCAN [45].
- DBSCAN with Xseedlist (B-DBSCAN) proposed by Brecheisen et al. [45]. Since the original algorithm only works with a single LB function, we slightly extend it to work with multiple LB functions by using multiple LBs from d_1 to d_{n-1} for each LB range query [45] as in M-DBSCAN and keeping d_{n-1} as the final LB distance.

The comparison between the naive anytime algorithms, A-DBSCAN and A-DBS-CAN-XS is omitted for clarity since the naive algorithms clearly perform much worse than DBSCAN, M-DBSCAN and B-DBSCAN.

Comparison Criteria. To compare the results of different clustering algorithms with the class labels provided for our experimental datasets, we use the DOM [76], Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) [258]. However, we only show the NMI for clarity because the results of DOM and AMI are the same as NMI. The result of NMI is in [0,1], with 0 means that the clustering result is independent of the ground truth and 1 means that the clustering result is the same as the ground truth.

For the runtime comparison, we report the *cumulative* runtime at each level for A-DBSCAN and A-DBSCAN-XS and the final runtime for B-DBSCAN, M-DBSCAN and DBSCAN. All the reported results are averaged over 20 runs. It is important to note that, A-DBSCAN and A-DBSCAN-XS do not necessary to be faster than B-DBSCAN, M-DBSCAN and DBSCAN in general since they have to perform the clustering at each level. However, they should quickly produce good clustering results at early levels [300].

Parameter Setting. For A-DBSCAN, A-DBSCAN-XS, B-DBSCAN and M-DBS-CAN, we use a sequence of LBs with 10 different functions $D = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 100\}$. Each function d_i at L_i uses first $D_i\%$ Wavelet coefficients to calculate lower bounding distance (100% means original ED distance). For the two parameters μ and ϵ, we first run DBSCAN with $2 \leq \mu \leq 30$ and 100 equally

distributed values of ϵ from min to max distance between objects to find the optimal parameters for DBSCAN for every dataset. Then all other algorithms are evaluated with these parameters. Therefore, we will have fair comparisons among all these algorithms and exclude any possible comparison bias. The selection of parameters will be further studied in Section 5.6.3.

5.6.2 Performance Evaluation

Performance on real datasets. Figure 5.6 and 5.7 show the clustering results for all real datasets used in our paper. For every figure, the upper part shows the NMI of the anytime algorithms A-DBSCAN and A-DBSCAN-XS at each level represented as curves and the NMI of the batch algorithms DBSCAN, M-DBSCAN and B-DBSCAN represented as horizontal lines (because these batch algorithms produce only 1 result). The parameters μ and ϵ of DBSCAN are shown beside the name of each dataset respectively. The lower part of each figure shows the cumulative runtimes of A-DBSCAN and A-DBSCAN-XS at each level represented as curves and the runtimes of B-DBSCAN and M-DBSCAN represented as horizontal lines. The runtimes of DBSCAN are shown beside the name of each dataset.

For the dataset MALLAT ($\mu = 7, \epsilon = 12.8$) as an example, the NMI score of DBSCAN (and also M-DBSCAN, B-DBSCAN) is 0.824. The runtimes are 15.7, 23.7, 164.4 seconds for B-DBSCAN, M-DBSCAN and DBSCAN respectively. At the first level L_1, A-DBSCAN and A-DBSCAN-XS require 9.7 and 9.4 seconds respectively to complete, which are about 3 times faster than M-DBSCAN with a clustering score of 0.747. The clustering scores of A-DBSCAN and A-DBSCAN-XS come to 0.826 at level 2-3 and 0.824 at level 4-10. When they come to the end, A-DBSCAN and A-DBSCAN-XS require only 21.3 and 11.3 seconds respectively and are about 8 and 15 times faster than DBSCAN.

For most datasets, the clustering scores become very close to the clustering scores of DBSCAN (more than 80% of NMI score of DBSCAN) from level 2. This means that A-DBSCAN and A-DBSCAN-XS acquire very good and stable clustering results at very early stages. For the dataset ECGFiveDays and SonyAIBORobotSurface, users can terminate the algorithm at level 3 or later to have satisfactory clustering results. For other datasets, the termination can be even earlier to acquire the speed up of 5 to 10 times compared with M-DBSCAN and B-DBSCAN and more than 10 times compared with DBSCAN. It is a remarkable advantage, especially when we are using the cheap ED as the distance measure. The difference would be much larger when we use expensive distance functions

such as DTW or LCS [74]. We demonstrate this fact in Section 9.3 by using DTW as the true distance measure.

For the dataset Cin_ECG_torso ($\mu = 22, \epsilon = 63.9$), the NMI scores at the early levels L_1 to L_3 (0.743, 0.739, 0.739 respectively) are even slightly better than the NMI scores of DBSCAN (0.738). For the dataset ECGFiveDays, the NMI score at level 3 is 0.724 which is slightly better than the NMI scores of DBSCAN (0.702). These facts are interesting since we can have better clustering results at some middle levels which could not be acquired with the true distance function. We will examine this problem more properly in Section 5.6.3.

For the datasets Mallat and DiatomSizeReduction, the final clustering results are slightly different with those of DBSCAN. It is due to the fact that we assign the border objects to their closest core objects (see Definition 14 in Section 5.3). In our experiment, this scheme helps to slightly improve the final clustering scores of A-DBSCAN and A-DBSCAN-XS on 13/32 real datasets from the UCR archives.

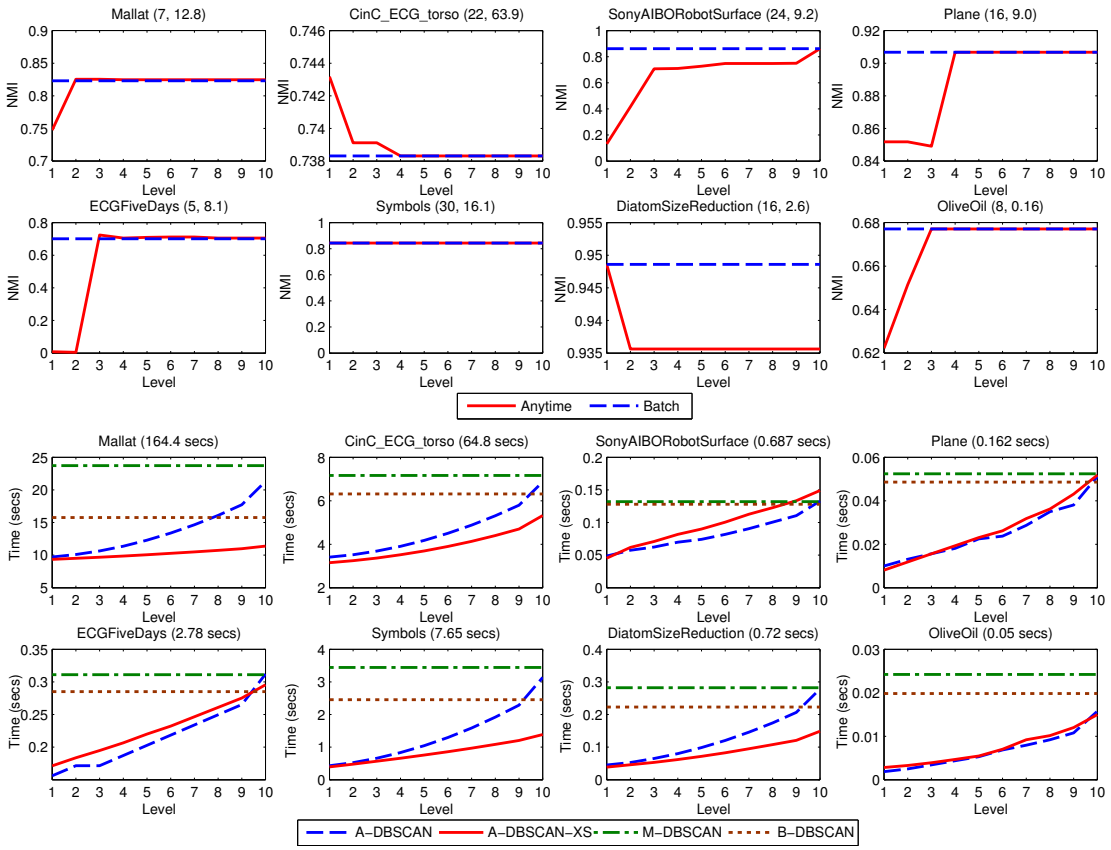

Figure 5.6: The performance of DBSCAN, M-DBSCAN, B-DBSCAN, A-DBSCAN and A-DBSCAN-XS on the 8 real datasets from the UCR archives.

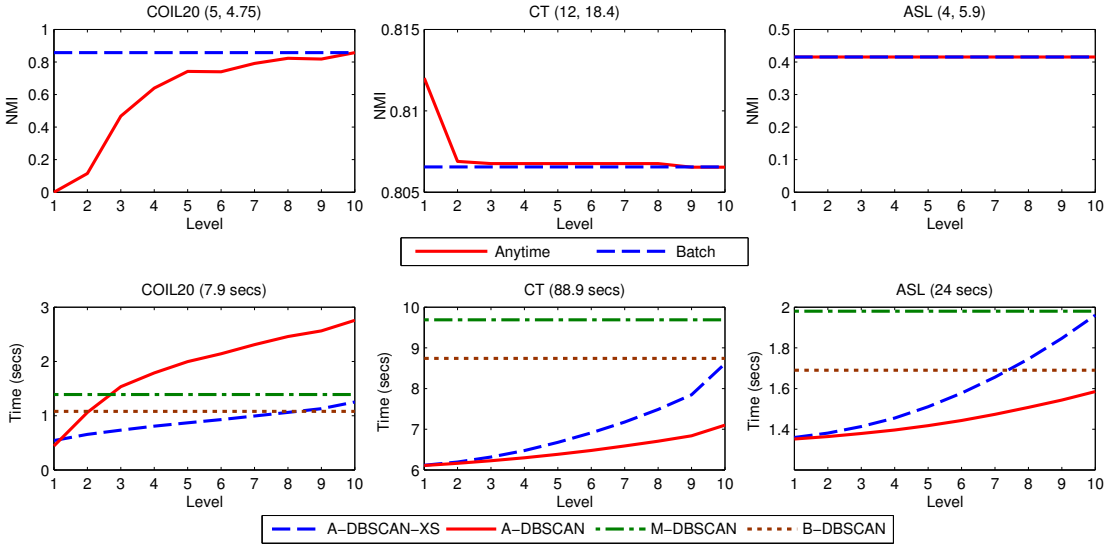

Figure 5.7: The performance of DBSCAN, M-DBSCAN, B-DBSCAN, A-DBSCAN and A-DBSCAN-XS on the 3 real datasets from other sources.

The final clustering scores of A-DBSCAN and A-DBSCAN-XS are slightly worst than DBSCAN on only 2/32 real datasets [184]. We note that if we use the original cluster definition of DBSCAN, then the results of DBSCAN, A-DBSCAN and A-DBSCAN-XS are identical.

For all datasets, the final cumulative runtimes of A-DBSCAN are slightly better than or identical to those of M-DBSCAN. The final cumulative runtimes of A-DBSCAN-XS are much better than those of B-DBSCAN on most datasets, except on the 3 datasets SonyAIBORobotSurface, Plane and COIL20. A-DBSCAN-XS is faster than A-DBSCAN on 6/11 datasets and is roughly identical to A-DBSCAN on 4/11 datasets. There is only one exception case on the dataset COIL20 where A-DBSCAN-XS is much slower than A-DBSCAN. The fact that A-DBSCAN is faster than M-DBSCAN and A-DBSCAN-XS is faster than B-DBSCAN is very interesting since A-DBSCAN and A-DBSCAN-XS have to perform the clustering process at each level. The main reason is due to the monotonicity property of A-DBSCAN and A-DBSCAN-XS. We will examine this problem below.

Why are the anytime algorithms faster than the batch algorithms? Figure 5.8 shows the percentages of the total distance function calls for every LB function d_i on 3 real datasets CinC_ECG_torso, SonyAIBORobotSurface and COIL20. We note that the percentages of the distance function d_1 is always 100% since we have to perform the full clustering at level 1.

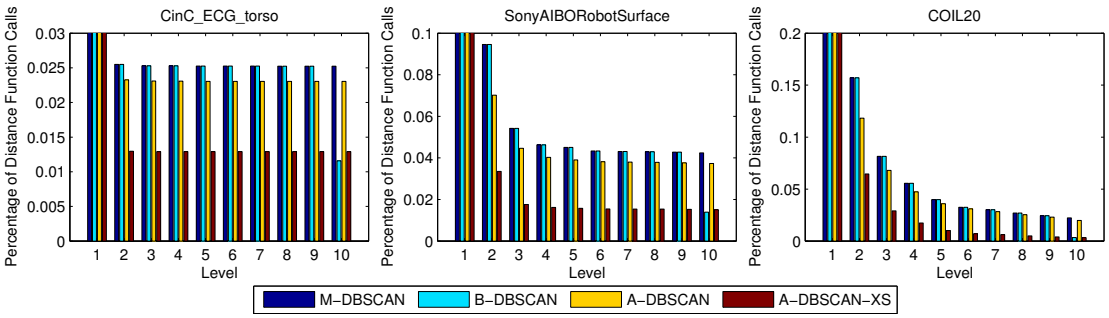

Figure 5.8: The percentages of the total distance function calls at each level on the 3 real datasets.

Due to the use of the monotonicity property to restrict the distance update, A-DBSCAN requires less distance calculations at each level than M-DBSCAN. By incorporating the monotonicity property and the extended Xseedlist, A-DBSCAN-XS clearly is the best among the 4 algorithms in terms of the total numbers of calls to distance functions. Unfortunately, A-DBSCAN and A-DBSCAN-XS must incur a remarkable cost for cluster update, though the monotonicity property helps to significantly reduce this cost following the local update scheme. Compared with A-DBSCAN, the cluster update cost of A-DBSCAN-XS is much larger due to the operation cost of the Xseedlist. A-DBSCAN-XS also has to update every cluster while A-DBSCAN only updates a cluster if it may split. Obviously, if the reduced cost of the distance calculations is larger than the cluster update cost, A-DBSCAN and A-DBSCAN-XS will be faster than their related algorithms M-DBSCAN and B-DBSCAN respectively and vice versa. In the case of the dataset COIL20 in Figure 5.7, the cluster update cost of A-DBSCAN-XS overwhelms the reduced cost for the distance calculations. Thus A-DBSCAN-XS performs worst.

The more expensive the used distance functions the better the performance of A-DBSCAN and especially A-DBSCAN-XS compared with M-DBSCAN and B-DBSCAN due to the trade-off between the cost for the cluster update and the cost for the distance calculation. In Chapter 9, we demonstrate this property by using DTW [74], which is much expensive than ED, as the distance function. The runtime difference among A-DBSCAN, A-DBSCAN-XS and others are much larger than with the use of ED in this Section.

Other experiments. The sorting scheme of the extended Xseedlist plays an important role to reduce the total numbers of call to distance functions. Figure 5.9 shows the total numbers of call to distance functions of our algorithm A-

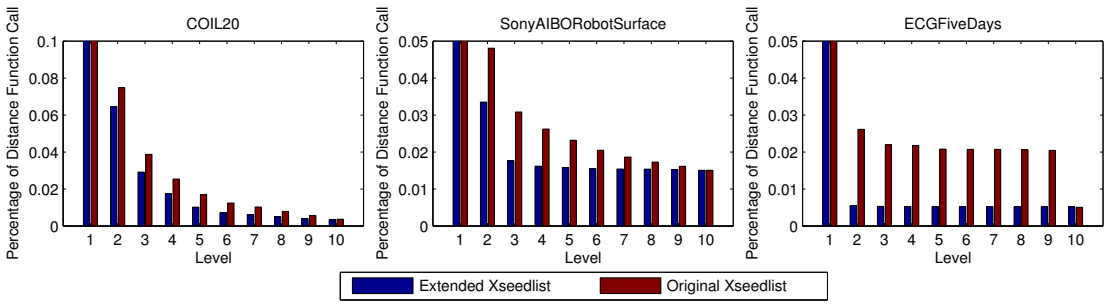

Figure 5.9: The percentages of the total distance function calls of A-DBSCAN-XS on the 3 real datasets with different sorting scheme for the Xseedlist.

DBSCAN-XS using the new sorting scheme (proposed in Section 5.4) and the original sorting scheme for the Xseedlist [45] on the 3 real datasets. As we can see, the new sorting scheme helps to significantly reduce the numbers of distance function calls, especially for the dataset ECGFiveDays.

Summary. A-DBSCAN and A-DBSCAN-XS acquire close clustering scores with DBSCAN at early stages of their execution. Thus, our algorithms accelerate the runtime up to orders of magnitude compared with B-DBSCAN, M-DBSCAN and DBSCAN. Though they both use LBs, A-DBSCAN and A-DBSCAN-XS are faster than M-DBSCAN and B-DBSCAN because they can exploit the monotonicity property to reduce redundant distance upgrades and reclustering cost as described in Section 5.3 and 5.4. Due to the high cost of the clustering update in contrast to the distance calculation reduction power of A-DBSCAN-XS, it works best when the used distance functions are expensive. In case, the distance functions are less expensive, A-DBSCAN should be chosen.

5.6.3 Parameter Analysis

The parameters μ and ϵ. Figure 5.10 shows the relationships between the two parameters μ (with $\epsilon = 4$), ϵ (with $\mu = 5$) and the performance of A-DBSCAN and A-DBSCAN-XS for the COIL20 dataset. The runtime of A-DBSCAN and A-DBSCAN-XS increases with ϵ since the increasing of the graph size leads to the increasing of the cost for the distance update and reclustering. The runtime of A-DBSCAN-XS is more sensitive to the choice of ϵ than that of A-DBSCAN due to the expensive cost of the data structure extended Xseedlist. The runtime of A-DBSCAN and A-DBSCAN-XS slightly decreases with μ since the reduction of

the number of core objects helps to reduce the total number of distance updates at each level.

The clustering quality is strongly affected by the choices of ϵ. For $\epsilon = 3.5$, A-DBSCAN and A-DBSCAN-XS quickly reach good NMI scores at level 2. The clustering quality slightly increases at each level until level 6 and then decreases. For $\epsilon = 5.0$, the clustering quality of A-DBSCAN and A-DBSCAN-XS increases and reaches the maximum value at the last level. This phenomenon can be explain via the relationship between the lower bounding distances and the parameter ϵ. Assuming that ϵ^* is the optimal parameter for ϵ ($\epsilon^* = 4.75$ for the dataset COIL20), if $\epsilon > \epsilon^*$ then the clustering qualities at all levels of A-DBSCAN and A-DBSCAN-XS will be smaller than the optimal one since the distance functions d_i lower bound the true distance d. However, if $\epsilon < \epsilon^*$ then the distance functions d_i at some middle levels may approximate the optimal cluster structure due to the lower bounding property. Thus, the clustering scores at some middle levels of A-DBSCAN and A-DBSCAN-XS are better than that of the final level. It is easy to see that A-DBSCAN and A-DBSCAN-XS work better when $\epsilon < \epsilon^*$ since they acquire the best result earlier at some middle levels and have smaller runtime as discussed above.

In contrast, the choices of the parameter μ seem less affect the clustering quality of A-DBSCAN and A-DBSCAN-XS. All the NMI score curves are generally the same with different values of μ. Therefore, A-DBSCAN and A-DBSCAN-XS work better when μ is big since they have smaller runtime as discussed above.

For real datasets, the relationship between the true distance and LB distance is somewhat arbitrary. For example, if $d(p, q) > d(r, s)$, it does not mean that $d_{lb}(p, q) > d_{lb}(r, s)$. Therefore, by using LBs, A-DBSCAN may reach results which are hard to acquire with DBSCAN at some middle levels since the LB distances may reflect the relationship of all the objects better than the true distance. For example, the best found score for COIL20 is 0.908 for A-DBSCAN and A-DBSCAN-XS ($\mu = 3$, $\epsilon = 4$) at level 7, while the best found score for DBSCAN is only 0.857 ($\mu = 5$, $\epsilon = 4.75$).

In this chapter, we do not focus on the problem of parameters finding for DBSCAN. The optimal choices for μ and ϵ could be selected by some existing heuristics proposed in the literature such as the k-dist graph [83] and entropy [166].

To summarize, A-DBSCAN and A-DBSCAN-XS seem more robust to the choices of parameters than DBSCAN due to its anytime scheme as discussed above.

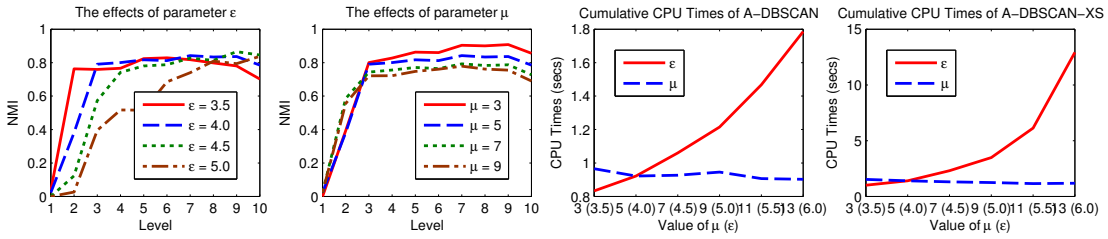

Figure 5.10: The relationships between the two parameters μ (with $\epsilon = 4$) and ϵ (with $\mu = 5$) and performance of A-DBSCAN for the COIL20 dataset.

Even the parameters are not optimally chosen, A-DBSCAN and A-DBSCAN-XS may still produce good results at some of its middle levels.

The lower bounding functions. Choosing a sequence of LBs is an important aspects of A-DBSCAN and A-DBSCAN-XS. Depending on their characteristics, different datasets require different numbers of coefficients to closely approximate the true ED distance. For example, for dataset ECGFiveDays, the use of the first 5% coefficients at level 1 is too small to have a good approximation of the ED distance. Thus, the clustering quality is too low. In contrast, the first 5% coefficients are too many for dataset Symbols. The lower bounding distance is close to the ED distance. Thus, the clustering qualities are the same at all levels.

Figure 5.11 shows the relationship between the performance of A-DBSCAN, A-DBSC-AN-XS and the tightness of LBs for the dataset Symbol ($\mu = 30, \epsilon = 16.1$). A-DBSCAN and A-DBSCAN-XS are run with 3 different sequences of LBs D_1 to D_3 (10 levels with different numbers of coefficients). As we see, the higher the numbers of the used coefficients at each level are, the tighter the lower bounding distances and the better the clustering qualities are.

The relationship between the runtime of A-DBSCAN and A-DBSCAN-XS and the numbers of the used coefficients at each level is somewhat theoretically complicated. Usually, smaller number of coefficients at each level means that the calculation time required for each LB function d_i is smaller. However, the pruning power of LB functions is decreased. Thus, the size of the graph G at each level is increased. It leads to the increasing of the total number of the distance updating and the time for reclustering at each level. The tradeoff between these two problems decides the performance of the algorithm. As we see from Figure 5.11 as an example, the runtime of A-DBSCAN is increased w.r.t. the numbers of used coefficients while the runtime of A-DBSCAN-XS is decreased. The tightness

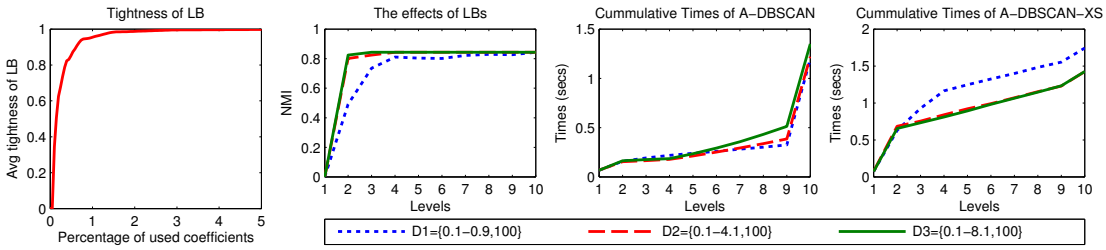

Figure 5.11: Performances of A-DBSCAN w.r.t. the tightness of LBs for the dataset Symbols.

of LB function could be a solution for this problem. By observing the tightness of LB function w.r.t. the number of used coefficients, we could balance this tradeoff. The higher the tightness of LB, the smaller the graph size at the next level but the larger the distance calculation cost.

Therefore, by randomly drawing a subset of data and calculating the averaged tightness of LBs w.r.t. the numbers of the used coefficients as demonstrated in Figure 5.11, users can choose the number of levels and number of used coefficients for each level based on their purposes.

5.7 Discussions

Anytime Clustering Algorithms. Anytime algorithms are algorithms that trade execution time for quality of result [305]. The quality of result of an anytime algorithm typically improves as the time increases to reach the result of the batch algorithm in the end. An anytime algorithm should satisfy some important properties described in Section 5.2 and [300, 305]. Anytime algorithms are currently an area of active research in many fields of data mining such as classification [255, 149] and outlier detection [20]. However, there is only little work on anytime clustering algorithms such as [147, 173, 174, 300].

Zhu et al. [300] proposed an approximation technique for Dynamic Time Warping which allows it to be used with anytime clustering algorithms. Kranen et al. [147] proposed an anytime clustering algorithm for streaming data.

Lin et al. [173, 174] exploited the multi-resolution property of DWT and PAA to casting k-Means into an anytime algorithm called I-kMeans. I-kMeans works by using the final cluster centers of level i as initial centers for level $i+1$. Though it is simple and efficient, I-kMeans is limited only for spherical shape clusters

while A-DBSCAN is able to detect clusters with arbitrary shapes and robust to outliers. The lower bounding property of DWT and PAA are also not exploited to construct clusters as in A-DBSCAN and A-DBSCAN-XS.

Density-based clustering. In density-based clustering, clusters are considered as high density areas, separated by low density areas. Among various kinds of density-based clustering algorithms, DBSCAN is one of the most successful algorithms with many extensions, e.g., [16, 45, 44, 160] and applications, e.g., [183, 166, 93, 56].

Ester et al. [82] proposed an incremental version of DBSCAN in a data warehousing environment. Based on the fact that insertion or deletion of an object affects the current clustering only in the neighborhood of this object, their algorithm called I-DBSCAN significantly speeds up DBSCAN even for large numbers of updates in a data warehouse environment. I-DBSCAN also exploits the nature of DBSCAN to do the clustering like our algorithm. However, I-DBSCAN is an incremental clustering algorithm, not an anytime algorithm. The changes of clusters in I-DBSCAN are caused by inserted or deleted objects, while the changes in A-DBSCAN are directed by the changes of the used distance functions.

The density-based subspace clustering algorithm SUBCLU [160] is based on the monotonicity property of DBSCAN w.r.t. distances in subspace projections of the data. The changes of clusters in A-DBSCAN are also monotonic. However, the monotonicity of A-DBSCAN and A-DBSCAN-XS is caused by the reduction of a special neighborhood graph related to a sequence of LBs, which is more general than SUBCLU. Thus, A-DBSCAN can be used with many different kinds of distance measures such as DTW and LCS [74] and arbitrary sequences of LBs while SUBCLU is limited to ED. Moreover, SUBCLU is designed for the discovery of subspace clustering of vector data and is also not an anytime algorithm.

In [44], a client-server parallel version of DBSCAN is proposed. The monotonicity property of DBSCAN [83] and OPTICs [16] is used to split objects to different clients and to merge the results returned from clients at the server. This monotonicity property is also similar to SUBCLU and is a special case of the monotonicity property used with A-DBSCAN.

LBs could be used in density-based clustering to accelerate the range query process thus improving the performance [45]. In [45] the authors integrate a lower bounding function into DBSCAN [83] to speed up these algorithms based on a data structure called the Xseedlist to reduce the number of true distance calculations. One major drawback of this technique (B-DBSCAN) is that Xseedlist requires a

sorting procedure which leads to higher time complexity than the normal Seedlist of DBSCAN. Thus it may not be suitable for cheap distance functions and very high number of objects. B-DBSCAN produces only one result and thus is not an anytime algorithm. In our algorithm, we use many LBs to produce multiple approximate clustering results during the runtime. A-DBSCAN and A-DBSCAN-XS rely on the monotonicity property and the restricted neighborhood graph to perform the clustering, which is fundamentally different from [45].

The idea of A-DBSCAN-XS is built upon the ideas of anytime clustering of A-DBSCAN and the ideas of the μ-range query from B-DBSCAN [45]. The main advantage of A-DBSCAN-XS over B-DBSCAN and A-DBSCAN is the way the Xseedlist is modified and combined with the neighborhood graph G and the monotonicity property of the cluster structure to significantly reduce the distance calculation at each level. We note that the original Xseedlist of B-DBSCAN is only designed to work with a single LB function, though we can easily modify it to work with multiple LB functions in a naive way as proposed in Section 5.6. Besides the integration of the Xseedlist, the main difference between A-DBSCAN and A-DBSCAN-XS is that A-DBSCAN contains two separated phases: the distance update and the reclustering phase at each level, while A-DBSCAN-XS combines the distance update and reclustering into a single phase at each level. This combination allows A-DBSCAN-XS substantially reducing the total number of distance calculation at each level. However, it increases the cost for updating cluster compared with A-DBSCAN since A-DBSCAN-XS is unable to detect which clusters will be split to recluster like A-DBSCAN. The high operation cost of Xseedlist of A-DBSCAN-XS is another difference with A-DBSCAN. Thus, A-DBSCAN-XS is more suitable to be used with very expensive distance function than A-DBSCAN.

Anytime DBSCAN. In general, A-DBSCAN and A-DBSCAN-XS are unique in the ways that they: (1) exploit multi-levels lower bounding functions to produce multiple approximate results of the final clustering result; (2) maintain the graph structure to acquire the monotonicity property even for arbitrary sequences of LBs. By this way, A-DBSCAN and A-DBSCAN-XS can be used with any kind of distance measures and arbitrary sequences of lower bounding functions. Thus they would have great applicability in reality.

Due to their update scheme, A-DBSCAN and A-DBSCAN-XS work very well on noisy datasets which contain large amount of noise and border objects. Also, A-DBSCAN and A-DBSCAN-XS are extremely useful when using with very expensive distance functions such as DTW or LCS [74]. Moreover, A-DBSCAN and

A-DBSCAN-XS can easily be parallelized.

5.8 Conclusions

We propose two anytime density-based clustering algorithms called A-DBSCAN and A-DBSCAN-XS which are applicable for many complex data such as time series and trajectory. Our algorithms work by exploiting a sequence of LBs to produce multiple approximate results of the true density-based clusters. To enhance performance, we propose an efficient distance update scheme which partially updates the distances among objects and a local reclustering scheme to save computational time at each level. Some changes in the notions of DBSCAN are made to improve the clustering results. An efficient heuristic for parameter setting is also proposed. Experiments on real datasets have shown that A-DBSCAN and A-DBSCAN-XS produce very good clustering results at very early stages of execution thus saving a large amount of computational time. Even if they run to the end, A-DBSCAN and A-DBSCAN-XS are still much faster than DBSCAN and its variants, despite the fact that they have to produce clustering results at every level.

Chapter 6

Active Density-based Clustering

The density-based clustering algorithm DBSCAN is a fundamental technique for data clustering with many attractive properties and applications. However, DBSCAN requires specifying all pairwise (dis)similarities among objects that can be non-trivial to obtain in many applications. To tackle this problem, in this Chapter, we propose a novel active density-based clustering algorithm, named Act-DBSCAN, which works under a restricted number of used pairwise similarities. Act-DBSCAN exploits the pairwise lower bounding (LB) similarities to initialize the cluster structure. Then, it adaptively selects the most informative pairwise LB similarities to update with the real ones in order to reconstruct the result until the budget limitation is reached. The goal is to approximate as much as possible the true clustering result with each update. Our Act-DBSCAN framework is built upon a proposed probabilistic model to score the impact of the update of each pairwise LB similarity on the change of the intermediate clustering structure. Deriving from this scoring system and the monotonicity and reduction property of our active clustering process, we propose the two efficient algorithms to iteratively select and update pairwise similarities and cluster structure. Experiments on real datasets show that Act-DBSCAN acquires good clustering results with only a few pairwise similarities, and requires only a small fraction of all pairwise similarities to reach the DBSCAN results. Act-DBSCAN also outperforms other related techniques such as active spectral clustering.

Publications. Parts of the material presented in this Chapter have been published in [186, 182]. The detailed information are described as follows:

- Son T. Mai, Xiao He, Nina Hubig, Claudia Plant and Christian Böhm. Active Density-based Clustering. In International Conference on Data Mining

(ICDM), pages 508-517, 2013.

In this work, S.T.M. proposed the theory, did most of the implementation and experiments. X.H. implemented an active spectral clustering algorithm and participated in the experimental comparison with Active Spectral Clustering. All authors discussed the technique and the results and contributed to paper writing.

- Son T. Mai, Xiao He. Active Density-based Clustering for Complex Data. Technical Report, University of Munich, 2014.

  In this work, S.T.M. developed the theory, implemented the algorithm and did experiments. X.H. implemented an active spectral clustering algorithm and participated in the experimental comparison with Active Spectral Clustering.

6.1 Introduction

Density-based clustering is a fundamental technique for data clustering with applications in many fields. In density-based clustering, clusters are regarded as areas of high object density in the data space separated by areas of lower object density. The algorithm DBSCAN [83] formalizes a density notion for clustering by measuring the cardinality of the neighborhood of each object. Compared with other clustering algorithms, DBSCAN has many attractive benefits, e.g., robustness against noise, support of general metric data and the ability to detect arbitrarily-shaped clusters. Therefore, DBSCAN has attracted a lot of research efforts and many extensions have been proposed in the literature over the past decades [45, 44, 82, 160]. However, DBSCAN requires specifying all (dis)similarities among all pairs of objects.

In many cases, gathering all pairwise similarities among objects is a non-trivial problem due to high computational cost, financial requirement, need for human annotation or even unavailableness for all pairs of objects [79, 236, 262, 275]. For example, clustering of proteins often involves a computational expensive alignment process before two proteins can be compared [28, 275]. In [275], the authors consider an application of clustering snapshots taken by a wearable camera in which human annotators have to be involved to rate the similarities between two pictures. This process is of course time consuming and expensive. For many

tasks such as detecting community structure in social networks, obtaining full data on the networks and the relations between its members is often hard or even impossible due to a large number of interactions [236]. Another example is the clustering of yeast proteins from the protein-protein interaction network where measuring all possible interactions reliably is often infeasible [233]. The potential difficulties of obtaining all pairwise similarities motivate the tradeoff between the desired clustering quality and the number of required similarities [275]. To tackle these problems, active clustering algorithms which tradeoff between the desired clustering quality and required amount of data have become an emerging topic, e.g., [236, 275, 79, 262, 114, 51]. However, to the best of our knowledge, there is no density-based active clustering algorithm proposed in the literature so far.

In this chapter, we focus on the novel problem of performing DBSCAN under a *budget* constraint, i.e., we can only use a limited number of pairwise similarities to acquire well comparable clustering result as if we had the entire similarity matrix at hand. In general, our algorithm does not have access to all pairwise similarities in the beginning. However, it iteratively and actively selects pairs of objects and updates the similarity matrix with their real similarities until the budget limitation is reached. The goal is to approximate as close as possible to the true clustering result with each similarity update thus reducing the total cost of accessing similarities. We call our algorithm the active density-based clustering (Act-DBSCAN).

Act-DBSCAN is based on the assumption that although the true pairwise similarities of data are time consuming, expensive or hard to obtain, there exists a fast, cheap and easy to obtain lower bounding (LB) function for them which can be used to support the clustering process. Such assumption is satisfied in many situations. For example, many effective similarity measures, e.g., Dynamic Time Warping, Longest Common Subsequence, have quadratic time complexity which makes them infeasible to deal with large datasets. However, there exist many constant or linear time complexity LB functions for them which clearly require much less computational effort [74]. Another example comes from *crowdsourcing* mechanisms where the ratings of multiple annotators for the similarities between objects are averaged to form the final results as used for the wearable camera problem in [275]. Instead of waiting all 10 annotators to rate a pair of pictures and averaging the results, the available ratings of several annotators could be summed and divided by 10 to form a LB similarity. Thus, the payment for the annotators is clearly much cheaper. For the clustering of sensor networks time-series, obtaining the true pairwise similarities may be hard or even impossible due to the missing

data during the signal transmission. However, the LB similarities can be easily estimated from the available data, e.g., using the corresponding distances at all available time-points. Many other examples could be found in other applications as well.

In general, Act-DBSCAN initializes with the cluster structure provided by a LB matrix. Then it iteratively selects the most informative pairwise LB similarities to update with the real ones. The selection is based on a proposed probabilistic model, which is built upon the *Kernel Density Estimation* (KDE) and *Poisson Binomial Distribution* (PBD), to score the impact of the update of each pairwise LB similarity to the change of the existing cluster structure. We call this scoring scheme the *Shared Core Object* (SCO) score. In our Act-DBSCAN framework, we propose two algorithms namely Splitting with SCO score (SP-SCO) and Merging with SCO score (MG-SCO) following the splitting (SP) and merging (MG) schemes respectively. Inspired by the monotonicity and reduction property of our clustering update process, these algorithms rely on the SCO scores to actively select the most informative pairwise LB similarities at each step in order to reconstruct the cluster structure. The general idea of them is to maximize the change in the existing cluster structure towards the desired result with each update. We theoretically prove the correctness of Act-DBSCAN by showing that the clustering result of Act-DBSCAN is identical with that of DBSCAN if the budget is large enough.

Contributions. Our contributions are summarized as follows:

1. We propose for the first time a novel active density-based clustering algorithm called Act-DBSCAN following the cluster notion of DBSCAN to deal with the sparseness (incompleteness) of the similarity matrix.

2. The Shared Core Object (SCO) score is proposed to measure the impact of each pairwise LB similarity based on a theoretical study of the intermediate clustering structure change and a probabilistic model to predict the core property of objects.

3. We propose two algorithms, named Splitting with SCO score (SP-SCO) and Merging with SCO score (MG-SCO) to actively select and update the similarity matrix based on the SCO score, monotonicity and reduction properties of our algorithm.

4. Extensive experiments have been conducted to demonstrate the performance of Act-DBSCAN. It acquires good clustering results with only a few pairwise

similarities, and requires only a small fraction of all pairwise similarities to reach the DBSCAN results. Act-DBSCAN also significantly outperforms other related techniques, e.g., active spectral clustering algorithms [236, 275].

This chapter is organized as follows. In Section 6.2, we briefly describe some backgrounds. Section 6.3 presents our algorithm Act-DBSCAN. Section 6.4 briefly describes some used similarity measures. Experiments are conducted in Section 6.5. We discuss relevant related approaches in Section 6.6. Finally, Section 6.7 concludes the paper.

6.2 Backgrounds

We briefly repeat some important backgrounds in this Section in order to enhance the readability, though they are presented in previous Chapters.

6.2.1 Density-based Clustering

The key idea of clustering algorithm DBSCAN [83] is that the cardinality of the ϵ-neighborhood of each object in a cluster has to exceed a predefined threshold μ. Compared with traditional clustering algorithms such as k-Means [121], spectral clustering [263], etc., DBSCAN has several attractive benefits as described above. Therefore, it is the main focus of our work.

Given a set of objects O with N objects, a similarity function $d : O \times O \rightarrow R$, parameters $\epsilon \in R^+$ and $\mu \in N^+$.

Definition 15 *(ϵ-neighborhood) The ϵ-neighborhood of $p \in O$, denoted as $N_\epsilon(p)$, is defined by $N_\epsilon(p) = \{q \in O | d(p, q) \leq \epsilon\}$.*

Definition 16 *(Directly density-reachable) An object $q \in O$ is directly density-reachable from object $p \in O$, denoted as $p \triangleright q$, iff $|N_\epsilon(p)| \geq \mu$ and $q \in N_\epsilon(p)$.*

Definition 17 *(Density-connected) Two objects p and $q \in O$ are density-connected, denoted as $p \bowtie q$, iff there exists a sequence $(x_1, \ldots, x_m)$ of objects such that $\forall x_i : |N_\epsilon(x_i)| \geq \mu$ and $p \triangleleft x_1 \triangleleft \cdots \triangleright x_m \triangleright q$.*

Definition 18 *(Cluster) A subset $C \subseteq O$ is called a cluster iff the two following conditions hold:*

1. *Maximality:* $\forall p \in C, \forall q \in O \setminus C : \neg p \bowtie q$
2. *Connectivity:* $\forall p, q \in C : p \bowtie q$

Definition 19 *(Core object property) An object $p \in O$ is:*

1. *A core object, denoted as core(p), iff $|N_\epsilon(p)| \geq \mu$.*
2. *A border object, denoted as border(p), iff $|N_\epsilon(p)| < \mu$ and $\exists q \in N_\epsilon(p)$: $|N_\epsilon(q)| \geq \mu$.*
3. *A noise object, denoted as noise(p), iff it is neither a core object nor a border object.*

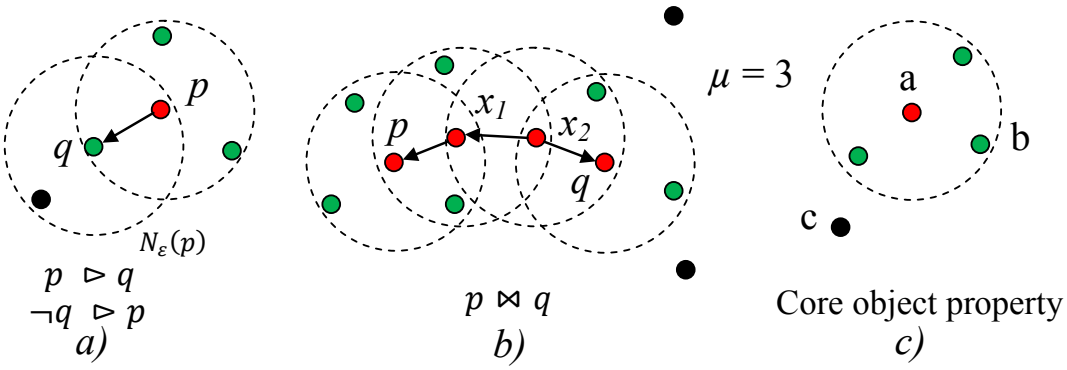

Figure 6.1: The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object;

Figure 6.1 (a), (b) and (c) show some notions used in DBSCAN. To construct a cluster, DBSCAN continuously extracts objects from a seedlist S and performs ϵ-range queries to find neighbor objects and inserts them into S until S is empty. A cluster of DBSCAN contains core and border objects. The noise objects do not belong to any cluster.

6.2.2 Active Clustering

Due to the potential difficulties of acquiring all pairwise similarities among objects in many applications, active clustering algorithms have recently become an emerging research topic, e.g., active spectral clustering [236, 275], active hierarchical clustering [79] and active k-Median clustering [262]. These algorithms focus on the tradeoff between the clustering quality and the number of used similarities. However, to the best of our knowledge, there is no active density-based clustering algorithm proposed in the literature so far.

Function Active_clustering (O, N, b, B)
  Input: Dataset O with N objects
    Number of similarity updates per step b and budget B
BeginFunction
  Initialize a similarity matrix $M_{N\times N}$
  $c = 1$
  while $c \leq B$ **and** $c \leq N(N\text{-}1)/2$ ***do***
    Choose b entries of M and update them with their true value
    Perform the clustering update with the updated matrix M
    $c = c + b$
  endwhile
EndFunction

Figure 6.2: A general framework for active clustering.

Figure 6.2 shows a general framework for active clustering algorithms. The algorithm starts with an arbitrary similarity matrix M. Then, b pairwise similarities are actively selected and updated with their real values before a clustering update is performed. The algorithm continues its execution until the matrix M is fully updated or the number of used pairwise similarities exceeds a predefined budget B.

6.3 Active Density-based Clustering

6.3.1 The Algorithm Act-DBSCAN

Given a set of objects O and a similarity function $d : O \times O \rightarrow R$, a lower bounding (LB) function of d is a function $d_{lb} : O \times O \rightarrow R$ where $\forall p, q \in O : d_{lb}(p, q) \leq$

$d(p, q)$. In our approach, we use LB similarities as a guideline for Act-DBSCAN to select meaningful pairwise similarities.

In general, Act-DBSCAN follows the general framework in Figure 6.2 for active clustering algorithms. In the initialization step, Act-DBSCAN starts with a similarity matrix M_{lb} acquired by the LB function d_{lb}. Due to the lower bounding property, if a pair of objects (p, q) has $d_{lb}(p, q) > \epsilon$ then $d(p, q) > \epsilon$ (since $d_{lb}(p, q) \leq d(p, q)$). Thus the similarity update of (p, q) is meaningless and should be considered last since it does not cause any change in the intermediate cluster structure following the cluster notions of DBSCAN in Section 6.2.1. In other words, let $G_{lb} = (O, E_{lb})$ be the ϵ-neighborhood graph acquired with the LB function d_{lb} (for all edge $(p, q) \in E_{lb}$, we have $d_{lb}(p, q) \leq \epsilon$). As described above, we only need to update pairs of objects (p, q) which $(p, q) \in E_{lb}$. It is clearly more efficient than updating the whole similarity matrix M_{lb}. For simplicity and readability, we use the ϵ-neighborhood graph G to represent the similarity matrix M w.r.t. the cluster notion of DBSCAN in the rest of this chapter.

Following the Framework 6.2, we still need a clustering update scheme and a pairwise similarity selection scheme.

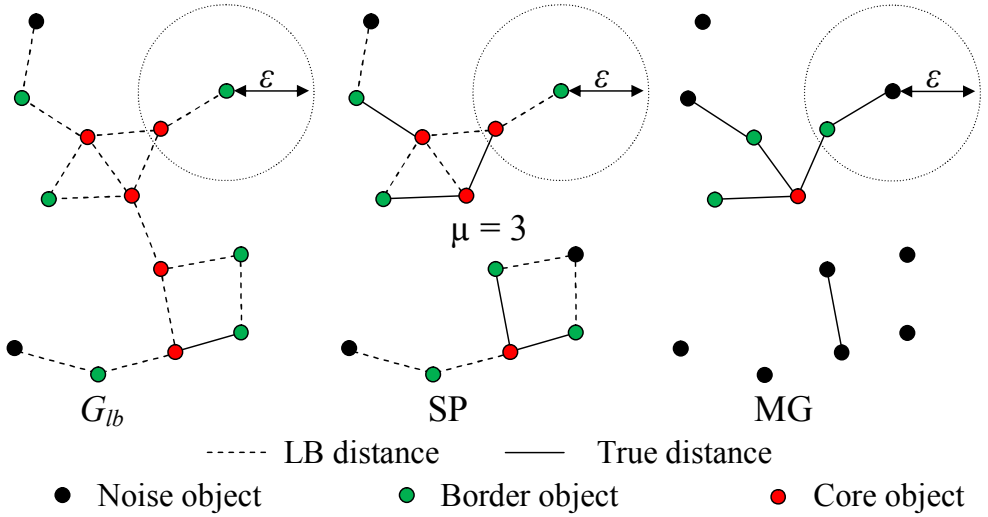

Figure 6.3: Two clustering update schemes used in our algorithm based on G_{lb}. The splitting scheme (SP) starts with $G = G_{lb}$ and removes the edge (p, q) from graph G if $d(p, q) > \epsilon$. The merging scheme (MG) starts with empty graph G and adds an edge (p, q) to graph if $d(p, q) \leq \epsilon$.

The clustering update schemes. Figure 6.3 describes two clustering update

schemes used in our paper based on G_{lb}.

Method 1 - *Splitting* (SP) starts with $G = G_{lb}$. At each iteration, an edge (p, q) is selected and updated with the true similarity. If $d(p, q) > \epsilon$, (p, q) will be removed from G and the corresponding cluster may be split into smaller clusters due to the monotonicity property described in Section 6.3.2.

Method 2 - *Merging* (MG) starts with an empty graph G and continuously updates edge (p, q) in G_{lb}. If $d(p, q) \leq \epsilon$, (p, q) will be added to the graph G. This clearly may lead to the merge of the two corresponding clusters.

For both methods, original DBSCAN clustering is performed on the result graph G to update the cluster structure. While SP can exploit the existing clustering result produced by the LB function and thus has a good starting point, it may suffer from a problem that many edges might need to be removed before an object could be separated from a cluster. Thus the improvement of clustering quality may be stepwise and slow. MG, in contrast, is not affected by this problem, however, it does not have a good starting point like SP.

The similarity selection scheme. Randomly selecting an edge $(p, q) \in E_{lb}$ to update for both methods (denoted as SP-Rand and MG-Rand) is simple and straightforward. However the results are unsatisfactory in our experiments. Therefore, we propose two methods to actively select edges to update based on their impacts on the change of intermediate cluster structure which are measured by the Shared Core Object (SCO) scores proposed in Section 6.3.5. These methods are named Splitting with SCO score (SP-SCO) and Merging with SCO score (MG-SCO) and described in the Act-DBSCAN framework below.

The Act-DBSCAN framework. Figure 6.4 shows a general framework for SP-SCO and MG-SCO based on the SCO scores. For both methods, we first calculate the probability $P_{del}((p, q))$ (Section 6.3.3) that an edge (p, q) of E_{lb} will have the true similarity $d(p, q) > \epsilon$. Then all edges of E_{lb} are divided into two separate lists L_1 and L_2 so that every pair of objects in L_1 has a higher likelihood to have $d(p, q) > \epsilon$ and vice versa. A list L of edges to update is then created by putting L_1 and L_2 together. Based on the monotonicity property (Section 6.3.2), we remove meaningless edges from L to improve the performance of the algorithm. After that we calculate the SCO scores (Section 6.3.5) of all edges of E_{lb} and initialize the graph G as described above.

For each iteration of Act-DBSCAN, we select an unprocessed set S of edges based on their SCO scores and update them with their true similarities. We

Function Active_clustering (O, N, b, B, μ, ε)
Input: Dataset O with N objects
Number of similarity updates per step b and budget B
BeginFunction
% Initialize the matrix and clustering results
Initialize ε-neighborhood graph G_{lb}= (O, E_{lb}) with d_{lb}
Perform DBSCAN on G_{lb}
% Create an update list L of edges
Calculate the edge probabilities for all edges (p,q) in E_{lb}
$L_1 = \{(p,q)|(p,q) \in E_{lb} \wedge P_{del}(p,q) \geq \tau\}$
$L_2 = \{(p,q)|(p,q) \in E_{lb} \wedge P_{del}(p,q) < \tau\}$
Create update list L of all edges from E_{lb} by arranging them according to their appearance in L_1 or L_2 first
Remove redundant edges from L by using monotonicity property
% Calculate the impact of each edge by SCO score
Calculate SCO score for all edges (p,q) in E_{lb}
% Actively select edges to query and update clusters
Initialize graph G
c = 1
while $c \leq B$ ***and*** $c \leq N(N\text{-}1)/2$ ***do***
if L is not empty ***then***
Remove set S of b edges of G from L_1 or L_2
Update the true values of all edges $(p,q) \in S$
Update the graph G
Perform DBSCAN on G to update the cluster structure
Remove meaningless edges from L by using monotonicity property and reduction property
Update the SCO scores for every edge in L
else
Randomly update b unprocessed edges or break the alg.
endif
$c = c + b$
endwhile
EndFunction

Figure 6.4: General framework for Act-DBSCAN.

then update the graph G following these changes. After that, DBSCAN is performed based on the graph G to acquire a new clustering result. Then, we remove meaningless edges from L following the monotonicity property (Section 6.3.2) and

reduction property (Section 6.3.6) of our algorithm. Lastly, we update the SCO scores of all edges in L.

When L is empty, all the remaining pairwise LB similarities have no effect on the current cluster structure. Thus, they can be randomly updated, or we can stop the algorithm.

The algorithm SP-SCO. The algorithm SP-SCO starts with $G = G_{lb}$. Due to its splitting scheme, the cluster structure will change most if the updated edge (p, q) is removed from G. Therefore, the edges in L_1 should be considered first. Thus, the update list is $L = L_1 \oplus L_2$. Also, to maximize the change in the cluster structure, (p, q) should cause a cluster to split. Thus, it should be the one with highest impact following the comparison function ϕ (Section 6.3.5).

The algorithm MG-SCO. The algorithm MG-SCO starts with an empty graph G. Due to its merging scheme, the cluster structure will change most if the updated edge (p, q) is added to G. Thus, the edges in L_2 should be considered first. The update list is therefore $L = L_2 \oplus L_1$. To maximize the change in the cluster structure, an edge (p, q) should be selected so that it will connect existing clusters together. Thus, it should be the one with the lowest impact following the comparison function ϕ (Section 6.3.5).

6.3.2 Monotonicity Property

The nature of DBSCAN permits efficient clustering algorithms with the monotonicity of cluster structures under some certain conditions, e.g., subspace projection of data [160], the use of a sequence of LB functions [184]. Here, we prove that the monotonicity property of the cluster structure still holds under the partial upgrade of the similarity matrix from the LB to the true similarity in our algorithm.

Let $G_i = (O, E_i)$ and $G_j = (O, E_j)$ be the partial upgraded ϵ-neighborhood graphs of dataset O at iteration i and j $(i \leq j)$ of Act-DBSCAN as stated in Section 6.3.1. Let $N_\epsilon^G(p)$ be the ϵ-neighborhood of object p under the graph G. We have:

Lemma 6 *For every object $p \in O$, $N_\epsilon^{G_i}(p)$ is a superset of $N_\epsilon^{G_j}(p)$.*

Proof 6 *Straightforward from LB property and update scheme stated in Section 6.3.1.*

Lemma 7 *For every object $p \in O$, we have:*

1. *if p is a core object under graph G_i, then it is a core, a border or a noise object under G_j.*
2. *if p is a border object under graph G_i then it is a border or a noise object under G_j.*
3. *if p is a noise object under graph G_i, then it remains a noise object under G_j.*

Proof 7 *According to Lemma 6, a core object p will become a border or a noise object if $N_\epsilon^{G_j}(p) < \mu$. Due to the reduction of its neighborhood size, a border object p will never become a core object, but it will become a noise object if all of its neighbor core objects lose their core property. A noise object p will not change because it does not have any core object inside its neighbors.*

For every cluster $C \subseteq O$, let $core(C)$ be a set of all core objects in C. We have the following lemma:

Lemma 8 *For all clusters C_v under graph G_i ($C_v^{G_i}$), there exists a cluster C_u under graph G_j ($C_u^{G_j}$) such that $core(C_u) \subseteq core(C_v)$.*

Proof 8 *Assume that there exist $p, q \in C_u^{G_j}$ so that $p \in C_k^{G_i}$ and $q \in C_l^{G_i}$. From Definition 18, we have $\neg p \bowtie^{G_i} q \Rightarrow \neg p \bowtie^{G_j} q$ according to Lemma 6 and 7. Thus, p and q belong to different clusters under G_j by Definition 18.*

Figure 6.5 illustrates the monotonicity property of Act-DBSCAN from G_i to G_j. Cluster C_1 at G_i is broken into two clusters C_{11} and C_{12} at G_j due to the deletion of edge (a, b). The core object a at G_i changes to a border object at G_j. Similarly, object b changes from a core to a noise object. The border object c becomes a noise object. The noise object d remains unchanged.

Why is the monotonicity property interesting? In the graph G, there exist 5 kinds of edges: core-core, core-border, border-border, border-noise and noise-noise (the core-noise edges do not exist according to Definition 19). Due to the Lemma 7, we do not need to update the border-border, border-noise and noise-noise edges since they do not participate in the clustering process. Thus, it helps

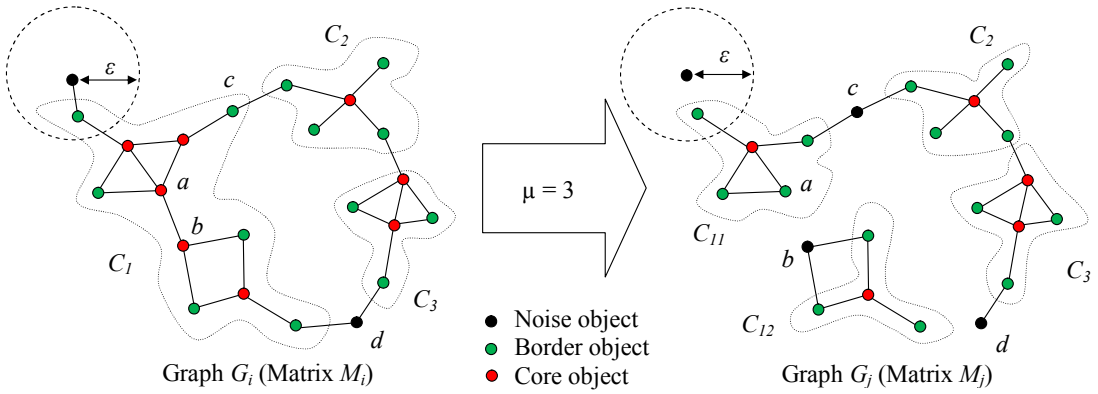

Figure 6.5: The change of clustering result of Act-DBSCAN from G_i to G_j. The core object a changes to a border object. Object b changes from a core to a noise object. The border object c becomes a noise object. The noise object d remains unchanged. Cluster C_1 is broken into two clusters C_{11} and C_{12}.

the algorithm to reach the desired clustering result faster by ignoring or postponing these meaningless updates.

From Lemma 8, a cluster may be split if one of its core-core edges is deleted or one of its core objects loses its core property. Thus it plays an important role for the selection of meaningful edges. The two algorithms SP and MG are inspired from this property. It also motivates the proposed SCO score (Section 6.3.5) to measure the impact of each edge. By choosing edges which may cause a cluster to be split to update first, we can quickly acquire the final cluster structure from the clustering result acquired by the LB similarities.

The monotonicity property allows an efficient reclustering scheme at each iteration of the active clustering. Assume that edge (p, q) is removed from cluster C, all we need is to perform DBSCAN on C to update the cluster structure instead of the whole dataset. Thus, it helps to speed up the clustering roughly about $O(N^2/|C|^2)$ time.

6.3.3 Edge Probability

Finding an edge $(p, q) \in E_{lb}$ of G_{lb} such that the true similarity $d(p, q) > \epsilon$ plays an important role in Act-DBSCAN to avoid meaningless updates which do not bring any change to the clustering result. For example, if we choose (p, q) with $d(p, q) \leq \epsilon$ to update with the algorithm SP (Section 6.3.1) then the cluster structure does not change. This problem however is non-trivial. For most kinds of

similarity measures, the relationship between LB and true similarity is somewhat arbitrary in our experiments. Therefore, one possible way is using a small training set T to learn the probability distribution of the difference between them and predict the true similarities based on the LB similarities. In our work, we use Kernel Density Estimation (KDE) to estimate this probability distribution rather than the traditional Gaussian distribution since KDE can robustly estimate the probability for any unknown distribution [242]. Figure 6.6 (left) shows an example for the dataset Symbols, the result of KDE is more close to the true distribution, represented by a histogram, than the Gaussian distribution obviously.

Kernel Density Estimation (KDE). Let $(x_1, \ldots, x_n)$ be a sample drawn from the difference between LB and true similarity distribution of training set T with an unknown density f, the kernel density estimation is defined as follows:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} k(\frac{x - x_i}{h}) = \frac{1}{n} \sum_{i=1}^{n} k_h(x - x_i)$$

where $k(\cdot)$ is a symmetric but not necessarily positive kernel function that integrates to one, $h > 0$ is a smoothing parameter called the bandwidth and $k_h(x) = h^{-1}k(x/h)$.

The cumulative distribution of KDE is then defined as:

$$\hat{F}_h(x) = \int_{-\infty}^{x} \hat{f}_h(u)du = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i)$$

where

$$K(x) = \int_{-\infty}^{x} k(u)du$$

is integrated kernel and $K_h(u) = K(u/h)$. As common choice, we choose $k(x) = (2\pi)^{-1/2}exp(-x^2/2)$ (standard Gaussian distribution). And the bandwidth h is estimated by Silverman's rule of thumb [242] as follows:

$$h = (\frac{4\hat{\sigma}^5}{3n})^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-\frac{1}{5}}$$

where $\hat{\sigma}$ is the standard deviation of the samples. In contrast to parametric statistics, KDE allows to robustly estimate the probability for any unknow distribution [242].

Calculate the deletion probability. For an edge $(p, q) \in E_{lb}$, the probability that it is removed from E_{lb} under the true similarity function (the probability that $d(p, q) > \epsilon$) is:

$$\begin{aligned} P_{del}((p,q)) &= P(d(p,q) > \epsilon) \\ &= P(d(p,q) - d_{lb}(p,q) > \epsilon - d_{lb}(p,q)) \\ &= 1 - \hat{f}_h(x > \epsilon - d_{lb}(p,q)) \\ &= 1 - \hat{F}_h(\epsilon - d_{lb}(p,q)) \end{aligned}$$

We note that $P_{del}((p,q))$ can be calculated in constant time since the size of the training set T is small and fixed.

Estimate the probability threshold. We need to find a probability threshold $\tilde{\theta}$ so that if $P_{del}((p,q)) \geq \tilde{\theta}$ then (p,q) has high likelihood to have $d(p,q) > \epsilon$.

Given n pairwise similarities, for each pair (p,q) (called a trial), we have only 2 states: (1) $d(p,q) > \epsilon$ (success) with probability θ and (2) $d(p,q) \leq \epsilon$ (failure) with probability $1-\theta$. Then the probability f that we have k successes follows the Binomial distribution [11] as follows:

$$f(k,n,\theta) = \begin{pmatrix} k \\ n \end{pmatrix} \theta^k (1-\theta)^{n-k}$$

From the training set T, we have δ the percentage of success trials over all trials. Therefore, for n pairwise similarities, we expect to have δn successes. By assuming that all trials have same probability, we can estimate the probability $\tilde{\theta}$ that $k = \delta n$ with $(1-\alpha)\%$ confidence interval using Agresti-Coull rule [11] as follows:

$$\tilde{\theta} = \frac{\delta n + z^2_{1-\alpha/2}/2}{n + z^2_{1-\alpha/2}}$$

with the confidence interval:

$$\tilde{\theta} \pm z_{1-\alpha/2} \sqrt{\frac{\tilde{\theta}(1-\tilde{\theta})}{n + z^2_{1-\alpha/2}}}$$

where $z_{1-\alpha/2}$ is the $1-\alpha/2$ percentile of a standard normal distribution. For example, for a 95% confidence interval, we have $\alpha = 0.05$, so $z_{1-\alpha/2} = 1.96$. Clearly, we have $n = N(N-1)/2$ is the total number of pairwise similarities.

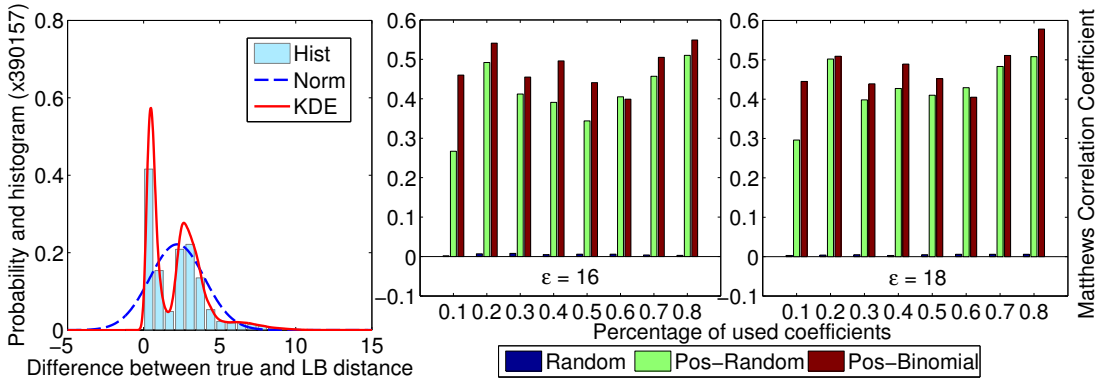

Figure 6.6: The dataset Symbols: (Left) Kernel Density Estimation (KDE) can estimate the true probability distribution better than traditional Gaussian distribution. (Right) The proposed technique Pos-Binomial outperforms other techniques in term of the prediction accuracy.

How good is the prediction? To the best of our knowledge, we are the first to solve this particular problem. Thus, we compare our technique with two simple methods. Method 1 (Random) randomly classifies E_{lb} into 2 classes. Method 2 (Pos-Random) randomly selects the threshold $\tilde{\theta}$ to perform classification.

We use the Matthews Correlation Coefficient (MCC) [23] to assess the quality of this binary classification.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP, FP, TN, FN are true positive, false positive, true negative, false negative respectively. MCC has value in [-1,1] with 1 being perfect prediction, 0 being similar to random prediction, -1 being totally different with the ground truth. Compared with other techniques such as Accuracy and Precision, MCC can avoid inflated performance estimates on imbalanced datasets. Thus, MCC is a reasonable choice in our case, since the numbers of success and failure are usually very different. We report the best results for Random and average results for Pos-Random over 100 runs.

Figure 6.6 shows the prediction results for the real dataset Symbols under Euclidean Distance and Haar Transform as LB function (please refer to Section 6.4 and 6.5 for descriptions). We set $\epsilon = 16$ and 18 respectively and 5% number of objects is randomly sampled as the training set T. Our algorithm, denoted as Pos-Binomial (with 95% confident interval), outperforms Pos-Random and Random in

most cases, especially when the quality of LB function, denoted by the tightness of LB (averaged ratio between LB and true similarity), is low. The same results are also observed with other datasets.

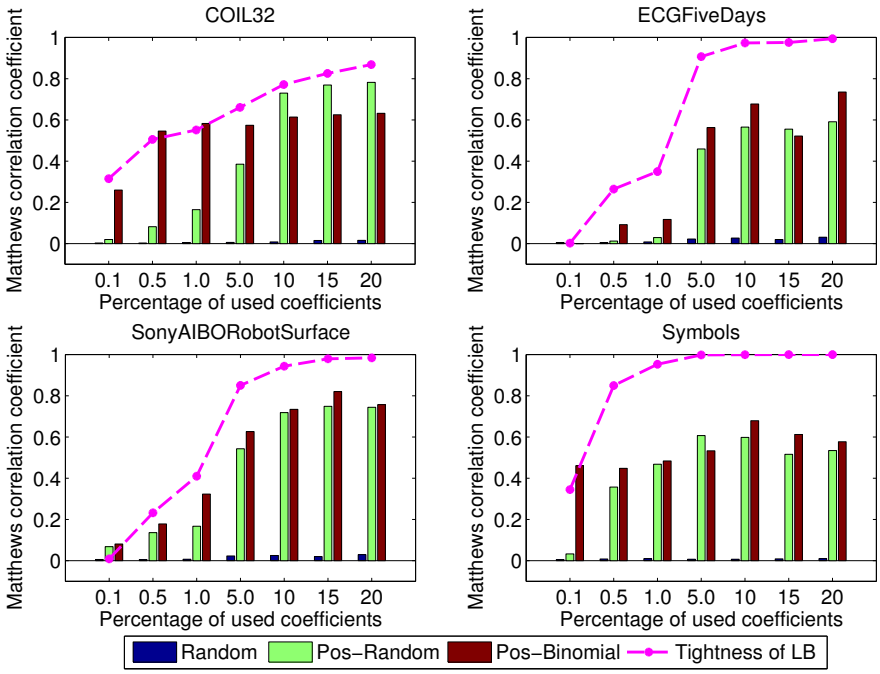

Figure 6.7: The comparison of different prediction techniques for 4 real datasets. Pos-Binomial outperforms other techniques in term of prediction accuracy.

Figure 6.7 additionaly shows the comparison of different prediction techniques on 4 more real datasets. As we can see, the Pos-Binomial outperforms other techniques, especially when the quality of LB function is low.

The role of the training set T. The training set T is only used to study the difference between the true similarity and the LB similarity. In case the training set T is not available, we can assume a probabilistic model such as Gaussian in the beginning. For each pairwise similarity update, the difference between the true similarity and the LB similarity is then used with Kernel Density Estimation (KDE) to recalculate the probability of each edge. The more updated similarities, the better the prediction. Though continuously updating the edge probability leads to the improvement of prediction and thus performance of the algorithm, we use in our paper a fixed training set T to calculate the deleted probability of each edge for clarity.

6.3.4 Core Object Probability

If object $o \in O$ is a core object under the LB similarity, is o still a core object under the true similarity?

Let $X = (x_1, \ldots, x_n)$ be the set of neighbors of o under d_{lb}. For each object x_i, let $P(x_i) = P(d(o, x_i) \leq \epsilon) = 1 - P_{del}(o, x_i)$ be the probability that x_i is a neighbor of o under the true similarity d ($P(x_i) \neq P(x_j)$). Then the probability that o has k neighbors follows the Poisson Binomial distribution [58] as follows:

$$f(k, n) = \sum_{A \in X_k} \prod_{i \in A} P(x_i) \prod_{j \in X \setminus A} (1 - P(x_j))$$

where X_k is the set of all subsets of k objects in X. According to [58], $f(n, k)$ can be computed in $O(kn)$ time by using the recursive formula:

$$f(k, n) = \begin{cases} \prod_{i=1}^{n} P(x_i) & if\ k = 0 \\ \frac{1}{k} \sum_{i=1}^{k} (-1)^{i-1} f(k - i, n) T(i) & if\ k > 0 \end{cases}$$

where

$$T(i) = \sum_{j=1}^{n} (\frac{P(x_j)}{1 - P(x_j)})^i.$$

Let $\tilde{X} = (x'_1, \ldots, x'_m)$ be the subset of X whose all pairs (o, x'_i) have already been updated with the true similarity d ($P(x'_i) = 1$). In case $m < \mu$, o will need at least $\mu - m$ objects in its neighbors under the function d to be a core object. Thus, the possibility that o is a core object ($P_{core}(o)$) is:

$$P_{core}(o) = \begin{cases} 1 & if\ m \geq \mu \\ 0 & if\ n < \mu \\ 1 - \sum_{i=1}^{\mu - m - 1} f(i, n - m) & otherwise \end{cases}$$

Following the recursive formula, $P_{core}(o)$ can be calculated in $O(N)$ time ($O(N^2)$ overall) since μ is constant and $|N_\epsilon(o)| \leq N$.

6.3.5 Edge Score and Comparison

Given a pair of objects (p, q) under d_{lb}, assessing its impact on the cluster structure under the similarity d is one of the most important problems in Act-DBSCAN. Let $S = \{o | o \in N_\epsilon(p) \cap N_\epsilon(q) \wedge core(o)\}$ be a set of shared core objects inside

the ϵ-neighborhoods of p and q. The Shared Core Objects (SCO) score of (p, q) is defined as:

$$SCO((p,q)) = \sum_{o \in S} P_{core}(o) + \beta|S|$$

where β in $[0, 1]$ is used to control the importance of current state versus predicted state. In our paper, we set $\beta = 1$ in all experiments. Intuitively, if (p, q) has a high SCO score, then (p, q) belongs to a high density area and has higher chance to be in the same cluster.

SCO score for an edge (p, q) can be calculated in $O(N)$ time. Thus, the time needed to calculate the SCO scores of the whole edges is $O(N^3)$. In our algorithm, SCO scores are re-computed at each iteration. However, due to its local scheme, for each iteration of Act-DBSCAN, we only need to update the related SCO scores. Thus, the time complexity to update SCO scores for each iteration is $O(bN)$ or $O(N)$, since b is a constant and the neighborhood size is smaller than N.

Given 2 edges (p, q) and (r, s), the comparison function ϕ between them is determined by SCO scores and LB functions as follows:

$$\phi((p,q),(r,s)) = \begin{cases} > \ if \ SCO((p,q)) < SCO((r,s)) - \tau \\ < \ if \ SCO((p,q)) > SCO((r,s)) + \tau \\ \begin{cases} > \ if \ d_{lb}(p,q) > d_{lb}(r,s) \\ = \ if \ d_{lb}(p,q) = d_{lb}(r,s) \quad otherwise \\ < \ if \ d_{lb}(p,q) < d_{lb}(r,s) \end{cases} \end{cases}$$

where τ is a user defined threshold (which is always set to 1 in our paper). If $\phi((p, q), (r, s)) =' >'$ then (p, q) has higher impact than (r, s) to cluster structure (since its has lower SCO score and higher LB similarity, it is more likely to be removed under the true similarity d and thus causes a cluster to be split if the algorithm is SP (Section 6.3.1)).

6.3.6 Reduction Property

Assuming that p and q are core objects of a cluster C under the true similarity d. Then, the edge (p, q) will never need to be updated since p and q are already density-connected. This scheme helps to reduce the meaningless updates significantly, especially for very dense clusters, since each object only needs to have μ neighbors to be a core object under the true similarity. Figure 6.8 shows an

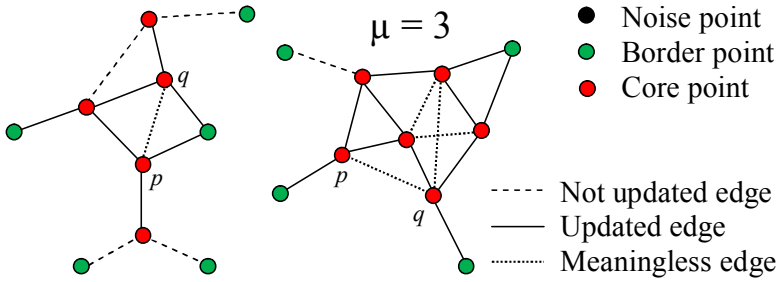

Figure 6.8: An example for reduction property. The edge (p, q) is meaningless and does not need to be updated since p and q are already density-connected. In a high density area (right), there are more meaningless edges than in a low density area (left).

example for reduction property of Act-DBSCAN. In the high density areas, there are more meaningless edges than low density areas.

6.3.7 Algorithm Analysis

Correctness. We prove that Act-DBSCAN requires only $B_0 \ll N(N-1)/2$ number of updates to reach the same cluster structure as DBSCAN.

The filter of the LB similarity, the monotonicity property and the reduction property significantly reduce the number of edges which needs to be updated from the list L at each iteration. Therefore, the total number of similarity updates is B_0 with $B_0 \ll N(N-1)/2$. Since these techniques only ignore edges which do not play any role in the clustering process, the core properties of objects are preserved. Thus, Act-DBSCAN produces the same results as DBSCAN, if budget $B \geq B_0$. If $B < B_0$, Act-DBSCAN may only approximate the result of DBSCAN. In our experiments, B_0 is usually less than 20% of the total number of all pairwise similarities. Also, Act-DBSCAN produces identical results as DBSCAN with only $B_1 \ll B_0$ similarity updates.

Complexity analysis. Since active clustering algorithms repeatedly perform clustering with each pairwise similarity update, the worst case time complexity for the naive algorithms such as SP-Rand and MG-Rand will be $O(\vartheta N^2 + N^4)$ where $O(\vartheta)$ is the complexity of the similarity function d in general. For the two algorithms SP-SCO and MG-SCO, the time needed to calculate the edge probability and the SCO scores of all edges is $O(N^2)$ and $O(N^3)$ respectively (see

Section 6.3.3, 6.3.4 and 6.3.5). At each iteration, the time needed to recalculating the SCO scores and updating the clusters is $O(N)$ and $O(N^2)$ in general respectively. Thus the worst case time complexity is also $O(\vartheta N^2 + N^4)$. Note that, we assume $B = N(N-1)/2$ and $b = 1$ for clarity and the time complexity of the LB function d_{lb} is much lower than d. However, unlike the active spectral algorithms (with $O(\vartheta N^2 + N^5)$ time complexity), the actual time complexity of Act-DBSCAN is much smaller than the worst case due to many useful properties, e.g., the monotonicity and reduction property.

In many complex databases such as time-series, multimedia, gene expression databases, each object contains thousands to millions points which makes the similarity measure between them extremely time consuming, especially with expensive distance measures, e.g., such as DTW, LCS (both with quadratic time complexity) [74]. In this case, the overhead of active scheme will be overwhelmed. Thus, Act-DBSCAN, due to its monotonicity property and reduction scheme, will enjoy dramatically performance improvement compared with DBSCAN as shown in Section 6.5.

6.4 Similarity Measures

Since Act-DBSCAN is a general framework, it can be used with any kind of (dis)similarity measures and their LB functions.

Euclidean Distance. Recent researches have introduced many kinds of (dis)similarity measures such as Euclidean Distance (ED), Dynamic Time Warping (DTW), Longest Common Subsequence (LCS), etc. [74]. In this chapter, we apply the ED to demonstrate the performance of our algorithm due to its simplicity and ubiquitousness.

Given two objects $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\} \in O$, we have: $d(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$.

Lower bounding function. In the literature, there exist many different kinds of LB functions for ED, e.g., Discrete Wavelet Transform (DWT) [216], Piecewise Aggregate Approximation (PAA) [74]. Though all of them can be used, we simply choose DWT as a representative. Interested reader please refer to Chapter 3 for more details on DWT.

For the LB function, we use DWT to transform each object into a sequence of Wavelet coefficients. For every pair of objects, we use only first $\kappa\%$ coefficients

of each object ($\kappa \ll 1$) and calculate the ED between these feature vectors. The result is a lower bounding of the ED between the two objects as shown in [216].

6.5 Experiments

All experiments are conducted on a 3 Ghz Workstation with 8 GB RAM under Window Server 2008 using Java.

6.5.1 Algorithms and Comparison Criteria

Datasets. We use six datasets from UCR achieves [131] (http://www.cs.ucr.edu/-~eamonn/time_series_data/), namely Trace, OliveOil, CinC_ECG_torso, Mallat and Symbols. These datasets contain many time-series objects acquired from diverse fields and are scale from small to large datasets. In addition, the dataset Coil20 acquired from Columbia Object Image Library (http://www1.cs.columbia.edu/-CAVE/software/softlib/coil-20.php) is also examined. This dataset contains 1440 pictures of 20 different objects (72 pictures per object). As common methods, we use ED distance between all pixel intensities to calculate the similarity. Note that the UCR datasets are re-interpolated to the length of $2^{\lfloor \log(n) \rfloor + 3}$ to use with DWT. This however does not affect the comparisons since all SP-Rand, SP-SCO, MG-Rand, MG-SCO produce the same results as DBSCAN.

Cluster Evaluation. To compare the clustering results with the ground truths, we use three different cluster evaluation methods, namely Dom [76], NMI [258] and AMI [258]. However, we only show the NMI for clarity, since the results of AMI and Dom are similar to NMI. The result of NMI is in [0,1], with 0 means that the clustering result is independent of the ground truth and 1 means that the clustering result is the same as the ground truth.

Algorithms. Since there is no active density-based clustering algorithm proposed in the literature, we compare the 4 heuristics SP-Rand, SP-SCO, MG-Rand, MG-SCO of Act-DBSCAN proposed in Section 6.3.1. Note that we do not compare with the pure random techniques SP and MG which randomly select pairwise similarities from all pairwise similarities because they clearly perform worse than SP-Rand and MG-Rand which randomly select pairwise similarities only from neighborhood graph G_{lb} (see Section 6.3.1). We also compare Act-DBSCAN with

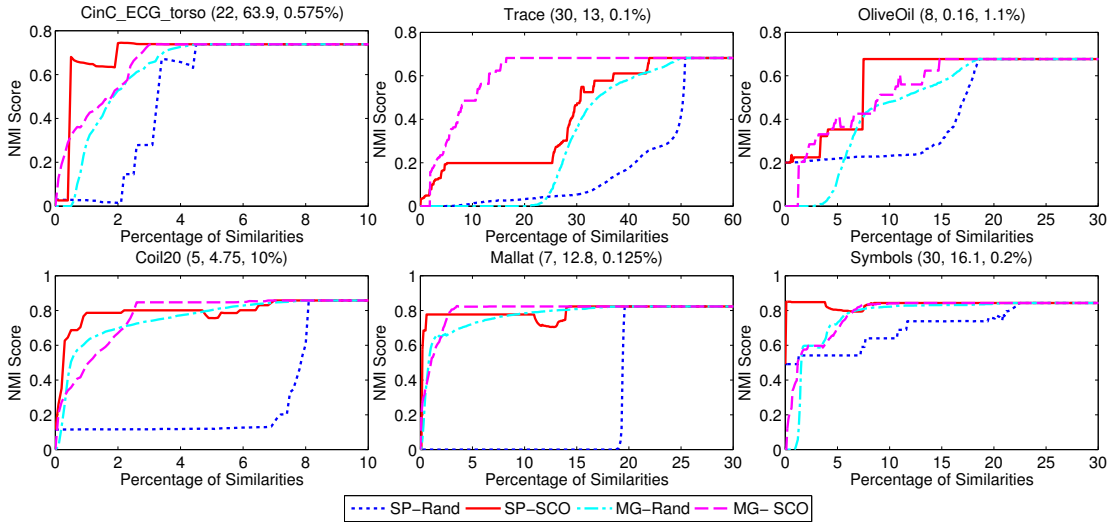

Figure 6.9: The comparison of the four algorithms SP-Rand, SP-SCO, MG-Rand and MG-SCO for six real datasets. SP-SCO and MG-SCO clearly outperforms other random techniques.

the active spectral clustering algorithms [236, 275]. The faster the algorithms reach the true clustering result w.r.t. the number of used true similarity measures, the better the algorithms are [236, 275].

Parameter Setting. To fully understand the performance of Act-DBSCAN, we set the budget limitation as the full similarity matrix ($B = N(N-1)/2$). Act-DBSCAN is run with 1000 steps. At each step, $b = N(N-1)/2000$ pairwise similarities are chosen to update with the true ones. For the two parameters μ and ϵ, we run DBSCAN with many different values of μ and ϵ to find the best parameters. Then these settings are used for our algorithm Act-DBSCAN. Thus this excludes any possible comparison bias.

6.5.2 Performance Analysis

Figure 6.9 shows the performance of the four algorithms SP-Rand, SP-SCO, MG-Rand, MG-SCO for the six real datasets. The parameters μ, ϵ of DBSCAN, and κ of the Haar transform are shown beside the names of the datasets respectively.

For most datasets, the performance of SP-SCO and MG-SCO clearly outperform random techniques SP-Rand and MG-Rand, except on the dataset Coil20 where the performance of MG-SCO and MG-Rand are somewhat hard to distinguish. Due to the update problem of the SP heuristic described in Section 6.3.1,

SP-Rand performs worst and SP-SCO has staircase shapes on dataset OliveOil. In contrast, MG-Rand and MG-SCO acquire smooth and stable performance since they are not affected by this problem. In general, SP-SCO performs the best on most datasets, except the dataset Trace where MG-SCO is clearly the best. MG-SCO does not start with existing cluster structure provided by the LB function thus they need more time to reach some certain results. This makes them less efficient than SP-SCO in many cases.

For all datasets, only small fractions of pairwise similarity are required to reach the desired results. For most datasets, SP-SCO and MG-SCO require less than 10% the total number of pairwise similarities to acquire satisfactory results, except for the dataset Trace.

Summary. Act-DBSCAN acquires very good performance on real datasets. It requires only few true pairwise similarities to reach the true clustering result, especially with SP-SCO.

6.5.3 Parameter Analysis

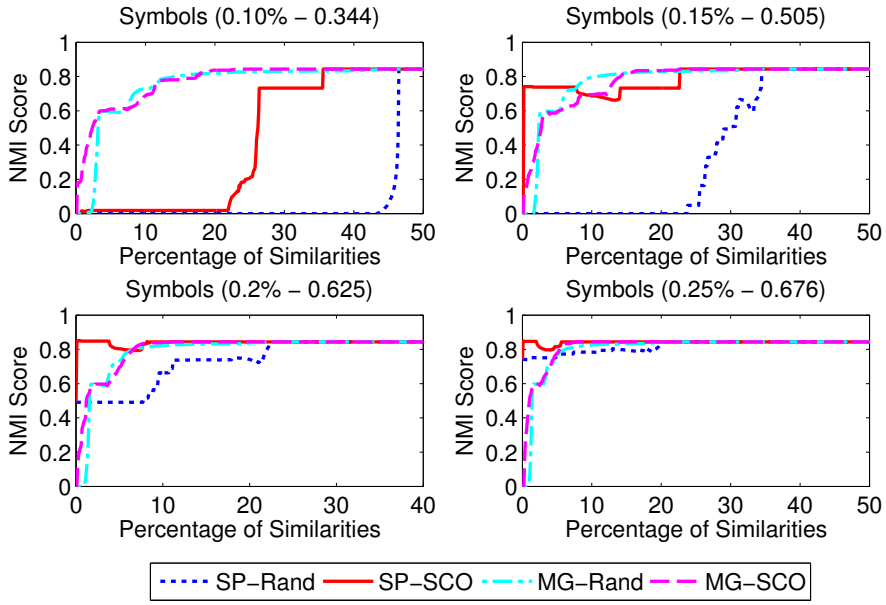

Figure 6.10: The effect of the LB function on dataset Symbols.

The LB function. Figure 6.10 shows the effect of LB functions on dataset Symbols. The percentages of Haar coefficients κ are shown beside the name of

dataset together with the Tightness of LB function (TLB). As we can see, the performance of SP-SCO varies with the quality of the LB function (indicated by the TLB). When the quality of LB is too low (0.344), SP-SCO performs worse than MG-SCO and MG-Rand. However, when the quality of LB is good (0.505, 0.625 and 0.676), SP-SCO clearly outperforms the others. MG-SCO, in contrast with SP-SCO, produces more stable results due to its merging scheme. For both algorithms, the higher the TLB, the better the performance that they acquire. The same results are also observed in other datasets as well.

By measuring the TLB on the training set T, we can determine which heuristic we should use. If the TLB is too low (e.g., $TLB < 0.5$) then MG-SCO should be chosen. If the TLB is good enough (e.g., $TLB \geq 0.5$) then SP-SCO should be chosen. On the contrary, if we want to choose the LB function to use with SP-SCO, then it should have high TLB. In case of ED with Haar wavelet as LB distance, higher TLB means that κ is bigger.

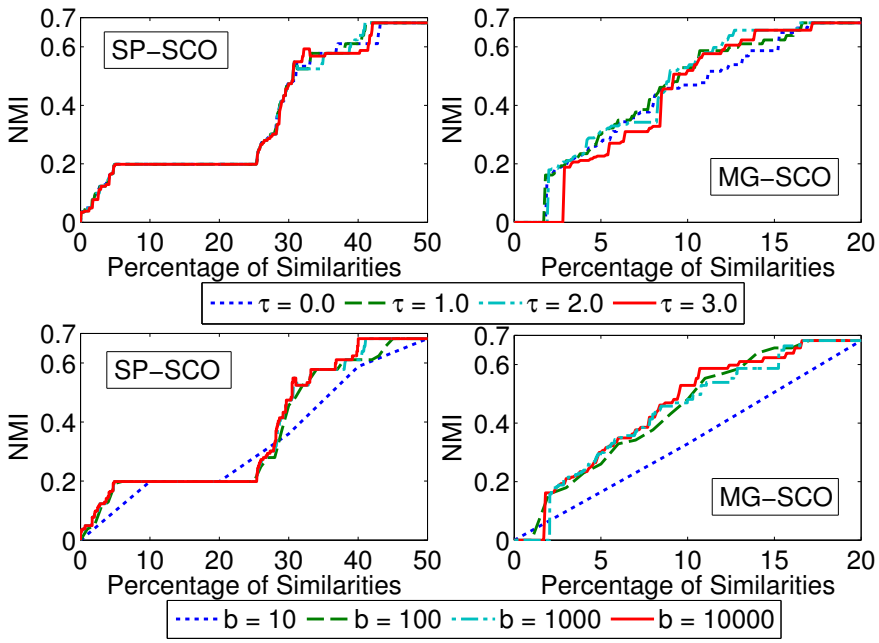

Figure 6.11: The effect of the parameter b and τ on SP-SCO and MG-SCO for the dataset Trace.

The parameter *b*. Figure 6.11 (bottom) shows the relationship between the clustering quality of SP-SCO, MG-SCO and the parameter b of Act-DBSCAN. We run Act-DBSCAN with 10, 100, 1000 iterations respectively (that means $b = N(N-1)/20, N(N-1)/200$ and $N(N-1)/2000$ respectively). As we see,

the smaller the value of b, the better the performance of Act-DBSCAN, since the cluster structure is updated more frequently thus the role of each pairwise similarity is evaluated more properly. However, too small values would not bring much benefit.

Parameter τ. Figure 6.11 (top) shows the relationship between the clustering quality of SP-SCO, MG-SCO and the threshold τ of the comparison function ϕ described in Section 6.3.5 for the dataset Trace. As we can see, our algorithm is robust to the choices of τ. The quality curves are almost the same with different values of τ.

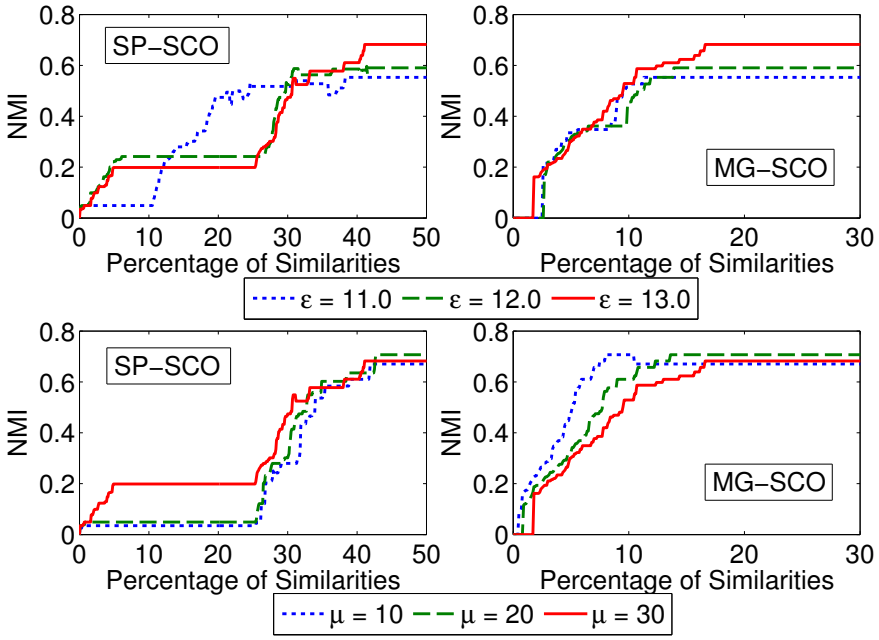

Figure 6.12: The effect of the parameter μ and ϵ on SP-SCO and MG-SCO for the dataset Trace.

Parameter μ and ϵ. Figure 6.12 shows the effects of μ and ϵ on SP-SCO and MG-SCO. The bigger the parameter μ and the smaller the parameter ϵ, the faster SP-SCO reaches the final result and the slower MG-SCO reaches the final result. The reason is that SP-SCO needs to remove less edges to change the core property of an object thus causing a cluster to be split while MG-SCO needs to add more edges to cause the merging of two clusters.

6.5.4 Comparison with Spectral Clustering

Although there exist several active clustering algorithms for spectral clustering [236, 275], hierarchical clustering [79], etc., the comparison between these algorithms and our algorithm seems unintuitive since they have different natures which have been extensively studied in the literature. However, it is still of interest for us to evaluate the performance of our algorithm. Here, we compare Act-DBSCAN with the four active spectral clustering algorithms denoted as AspecW, AspecW-Inter [275] and AspecS, AspecS-Inter [236] ("Inter" means *interleave* version [236, 275]) since they are closest to the nature of DBSCAN and can be extended to used with the LB functions. For each algorithm, we also report the result of an extended version using LB similarity matrix as an initialization, denoted by the suffix "-LB", to ensure the fair comparison with Act-DBSCAN.

We note that the comparison with active spectral clustering is conducted on two cluster datasets as in [275] and [236].

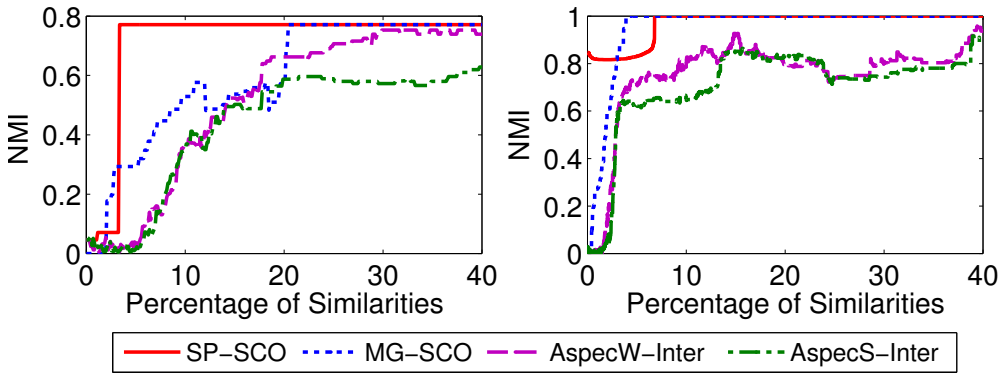

Figure 6.13: Comparison with Active Spectral clustering on 2 real datasets OliveOil (6, 0.18, 2%, 0.08) and Coil20 (6, 2.5, 5%, 2). SP-SCO and MG-SCO acquire better performance than active spectral clustering algorithms.

Figure 6.13 shows the comparison results for two real datasets OliveOil and Coil20. However, we only show the 2 best results out of 8 different algorithms for clarity. The parameters μ, ϵ of DBSCAN, κ of Haar and σ of Gaussian Kernel of spectral clustering are shown beside the names of datasets. As we can see, SP-SCO and MG-SCO have better performance since the use of LB function provides an efficient way to select meaningful pairwise similarities. Unlike Act-DBSCAN, the use of LB similarity matrix does not benefit the overall performance of active spectral algorithms due to the non-correlation between the LB and true similarity.

With the use of LB similarity matrix, they acquire better performance in the beginning. However, their performance becomes worse in the long term. The reason why the LB similarity matrix does not help to improve the performance of active spectral clustering is that the relationship between LB and true similarities is somewhat arbitrary, i.e., two objects with very big similarity may have very small LB similarity and vice versa. Thus, the second eigenvector cannot reflex the cluster structure correctly with the LB similarities. Act-DBSCAN, however, is less affected by this phenomenon.

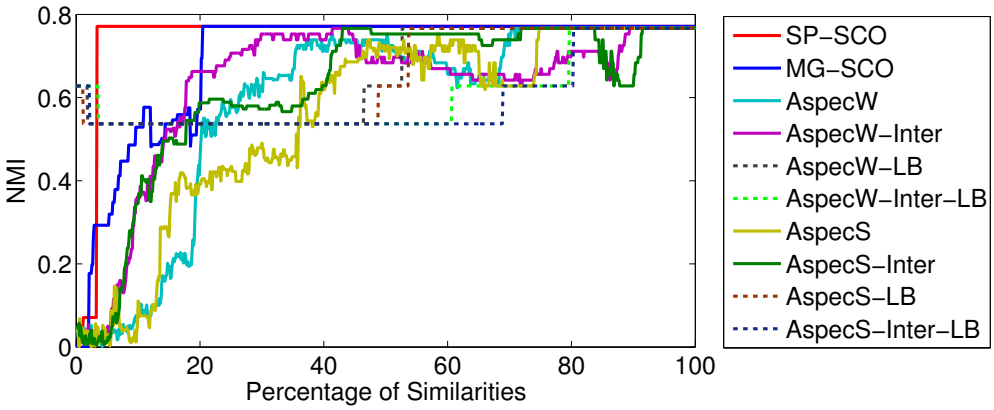

Figure 6.14: Comparison with Active Spectral clustering on dataset OliveOil. SP-SCO and MG-SCO outperforms active spectral clustering algorithms.

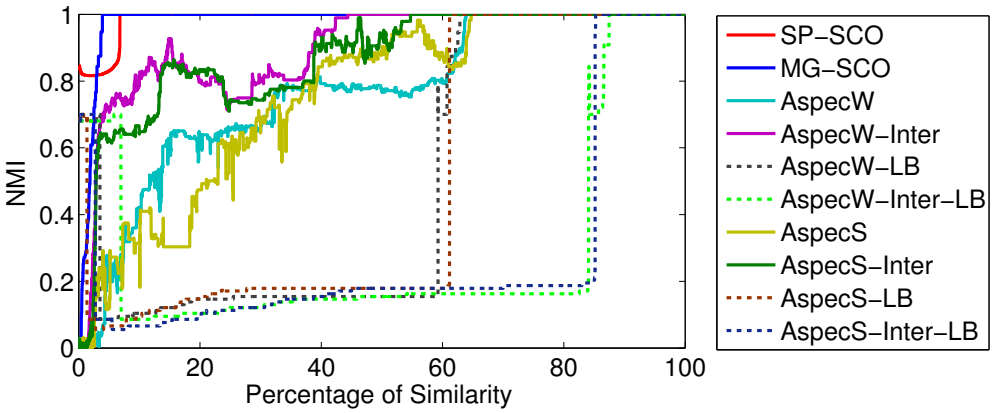

Figure 6.15: Comparison with Active Spectral clustering on dataset COIL20. SP-SCO and MG-SCO outperforms active spectral clustering algorithms.

Figure 6.14 and 6.15 show the comparison between SP-SCO, MG-SCO and all the eight variants of active spectral clustering for the real dataset OliveOil and

COIL20.

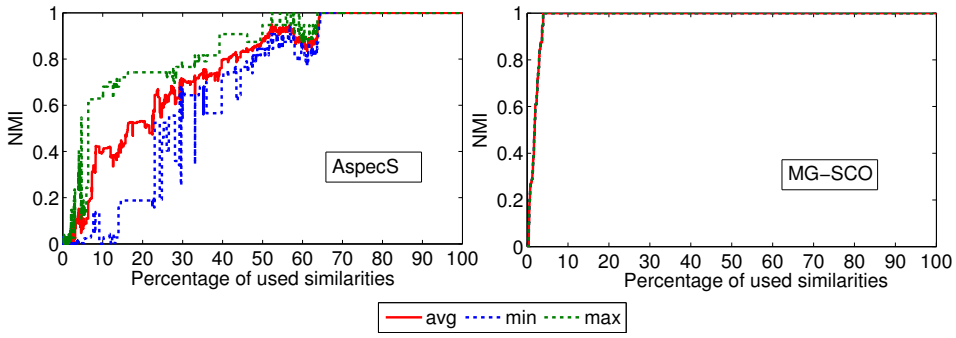

Figure 6.16: Stability of active spectral algorithm AspecS and MG-SCO on dataset Coil20. The performance of AspecS varies significantly.

Figure 6.16 shows the min, max and average performance of active spectral clustering algorithms AspecS and MG-SCO for dataset Coil20. The performance of AspecS varies significantly at each run. The same results are observed with other active spectral clustering algorithms. In contrast to active spectral clustering, the performance of Act-DBSCAN (MG-SCO and SP-SCO) is much more stable. The result varies negligible at each run. The instability of active spectral clustering algorithms are caused by the eigen decomposition of the similarity matrix while the instability of Act-DBSCAN is caused by the choice of parameter τ. The larger the parameter τ, the more unstable the result of Act-DBSCAN.

Summary. Act-DBSCAN requires less pairwise similarities to produce the same results as active spectral clustering.

6.5.5 How many similarities do other algorithms use?

In Figure 6.17, the vertical lines indicate the total numbers of true pairwise similarities that other algorithms used to produce the final results of the dataset Symbols. These algorithms, namely anytime DBSCAN (A-DBSCAN) [184], DBSCAN with LB as a filter (M-DBSCAN) [45], DBSCAN with XSeedList (B-DBSCAN) [45], try to speed up DBSCAN by reducing the total number of used pairwise similarities. Note that, none of these algorithms is active clustering algorithm and can not deal with the budget problem of Act-DBSCAN. Thus, we only use the vertical lines to represent the total numbers of pairwise similarities that they used.

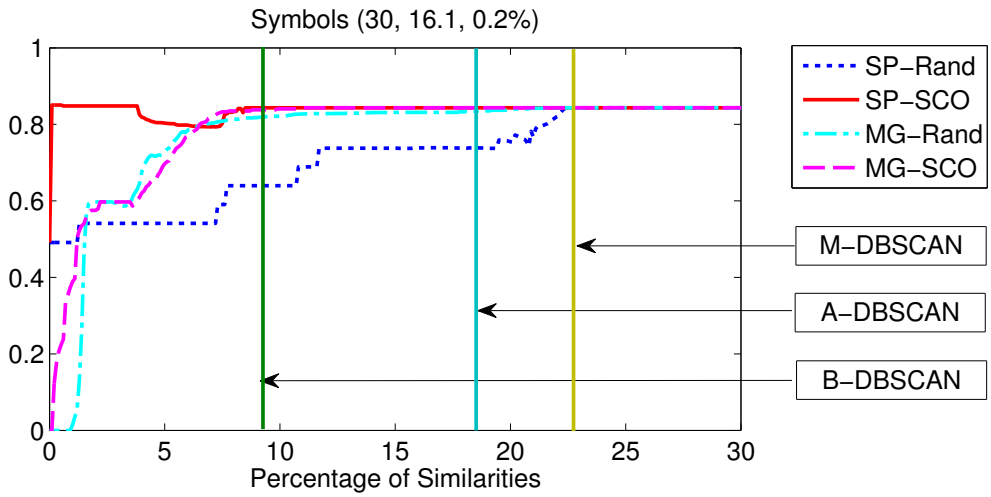

Figure 6.17: The total numbers of used pairwise similarities acquired from different algorithms.

It is interesting to see that Act-DBSCAN, especially with SP-SCO, requires very small number of pairwise similarities to reach a satisfactory result. The same results are also observed with other datasets as well. The effectiveness of Act-DBSCAN comes from the combination of the monotonicity and reduction property of our algorithm which leads to significant reduction of unnecessary pairwise similarity updates. Moreover, the efficiency of the SCO scoring system helps the algorithm to efficiently identify the most informative pairwise similarities to update first. Hence, Act-DBSCAN reaches the desired results much faster than random scheme.

6.5.6 Runtime Analysis

In many complex databases such as time-series, multimedia, gene expression databases, each object contains thousands to millions points which makes the similarity measure between them extremely time consuming, especially with expensive distance measures, e.g., such as DTW, LCS (with $O(M^2)$ time complexity) [74]. In this case, the overhead of active scheme will be overwhelmed. Thus, Act-DBSCAN, due to its monotonicity property and reduction scheme, will enjoy dramatically performance improvement compared with DBSCAN.

To demonstrate this property of Act-DBSCAN, we use DTW as the similarity measure. In contrast to EU, DTW [132] allows flexible matching between object

points and thus is one of the most effective similarity measures for many kinds of complex objects such as time-series, etc. [74]. However, its $O(M^2)$ time complexity remains a bottle neck in many applications. DTW requires a parameter $\xi \in [0, 1]$ which determines the size of matching window to be set in percentage of the length of objects. Base on this window scheme, Keogh et al. [132] proposed an $O(M)$ time complexity LB distance called LB_Keogh which is still one of the best LB techniques for DTW nowadays.

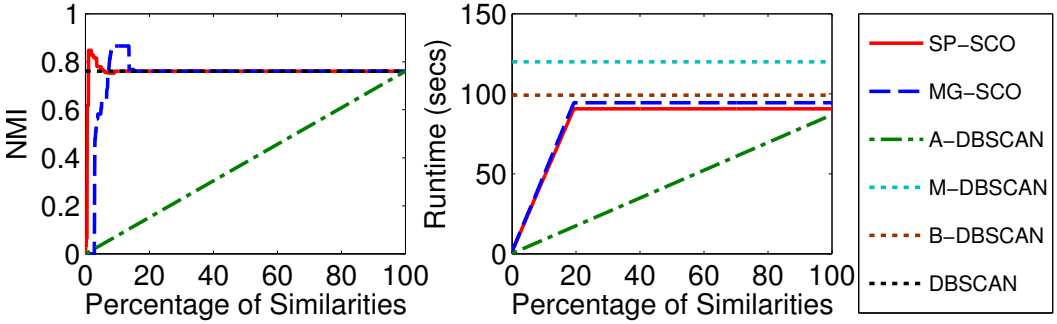

Figure 6.18: Runtime comparison for the dataset Trace. The parameters μ, ϵ and warping window size for LB_Keogh are 30, 75.0 and 30% respectively. Since they produce only 1 result during their runtime, the results of DBSCAN, M-DBSCAN and B-DBSCAN are presented by horizontal lines.

Figure 6.18 shows the comparison among Act-DBSCAN, DBSCAN [83], M-DBSCAN and B-DBSCAN [45] and A-DBSCAN [184] for the dataset Trace using DTW and LB_Keogh [74] as the similarity and LB measures. As we see, SP-SCO and MG-SCO acquire the same result with DBSCAN after 5% percents of all similarity measures. Thus if we stop the algorithm at this point, SP-SCO and MG-SCO require around 25 seconds, which is up to 20 times faster than DBSCAN (494 seconds) and its variants. Even if they run to the end, SP-SCO and MG-SCO is still faster than or at least equivalent to others. That means the pruning power of Act-DBSCAN overwhelms the overhead for the active scheme in this case.

6.6 Discussions

Active Clustering. Most active learning algorithms for clustering focus on the learning of pairwise constraints (usually in the form of *must-link* and *cannot-link* constraint) for semi-supervised clustering such as [98, 265, 271, 187, 25, 267, 280,

98]. Recently, active clustering algorithms which focus on the sparseness of the similarity measure matrix have become an emerging research in the literature, e.g., [51, 79, 114, 236, 262, 275].

Eriksson et al. [79] propose an active clustering method for hierarchical clustering which requires $O(N \log N)$ pairwise similarities under the Tight Clustering (TC) condition. The active k-median clustering proposed by Voevodski et al. [262] uses an active selection strategy to choose a set of landmark points and constructs clusters based on the distance between the landmarks and other points. This algorithm requires $O(k)$ *one-versus-all* queries to form the clusters. In [236, 275], the authors propose active spectral clustering algorithms based on matrix perturbation theory under some certain assumptions about the cluster structures. Other approaches include the use of expected value of information and mean field annealing optimization to produce efficient active clustering methods [51, 114]. To the best of our knowledge, there is no active density-based clustering algorithm proposed in the literature.

Density-based clustering. Ester et al. [82] propose an incremental version of DBSCAN (I-DBSCAN) for data warehouse environment. The splitting and merging of clusters of I-DBSCAN are caused by the update, insertion and deletion of objects while those of our algorithm are caused by adding or removing an edge to the neighborhood graph. SUBCLU [160] is axis-parallel subspace clustering and is based on the monotonicity of DBSCAN under subspace projection of vector data. In [44], the authors propose a client-server parallel version of DBSCAN by exploiting the monotonicity property of DBSCAN under a LB function. Mai et al. [184] introduce an anytime version of DBSCAN by exploiting the monotonicity of DBSCAN under a sequence of LB functions to reduce number of similarity calculations and to have faster cluster updates. In contrast to these algorithms, the monotonicity of Act-DBSCAN is caused by the partial update of the similarity matrix from the LB to the true similarities. Brecheisen et al. [45] integrate a LB function into the clustering process to speed up DBSCAN based on a data structure called Xseedlist. This algorithm tries to reduce computation cost by using μ-query instead of ϵ-query to determine the core objects. This idea somewhat closes to the reduction property used in Act-DBSCAN. In [157], the authors extend DBSCAN to deal with uncertain databases using a probabilistic model to calculate the probability of a core object. However, it is also fundamentally different from our work which is based on the exact similarities between objects. There also exist many other extensions of DBSCAN proposed in the literature (see Chapter 2 for

more details). However, none of them is designed to deal with the *budget* problem.

Act-DBSCAN. Act-DBSCAN is unique in the way that it exploits a LB function to become an active clustering algorithm. The use of LB function: (1) provides useful information to select meaningful pairwise similarities to be updated; (2) allows the detection of meaningless pairwise similarities and exclude them to help Act-DBSCAN to reach the desired clustering result faster. The Act-DBSCAN framework can be easily extended to work with the upper-bounding functions or the combination of both the lower bounding and the upper-bounding functions to further enhance the efficiency. However, we only focus on lower bounding functions in this work for clarity. Note that, existing techniques for reducing the number of calculated pairwise similarities to enhance the efficiency of DBSCAN like [83, 45, 184] fundamentally differs from solving an active clustering problem which the algorithm has to produce as close as possible to the desired clustering structure within an arbitrary allowed number of used pairwise similarities.

The anytime clustering algorithm proposed in [300] exploits the lower bounding and upper-bounding functions of Dynamic Time Warping (DTW) to approximate the similarity matrix in the beginning. Then, each entry in the similarity matrix is sequentially selected and updated with its true DTW similarity following a predetermined ranking scheme. The general goal is to approximate the true similarity matrix at much as possible with each entry update. By this way, the proposed algorithm can be used to transform all clustering algorithms that use the similarity matrix into anytime clustering algorithms including the density-based clustering algorithm, though the authors only mentioned hierarchical clustering, k-Medoids and spectral clustering in their work. Due to its ranking and selection scheme, the proposed algorithm could somehow be regarded as an active clustering algorithm. However, it differs with our algorithms in some major ways. First, this algorithm focuses on the change of distance matrix rather than cluster structure. Actually, maximizing a change in the distance matrix does not mean that the change in cluster structure is maximized as well and vice versa. Thus, it limits the performance of the algorithm. Second, while each pairwise distance is iteratively evaluated and selected, this algorithm only ranks them one time in the beginning and selects them according to this ranking. In fact, this scheme makes the algorithm closer to an anytime algorithm than an active algorithm. In contrast, our algorithm iteratively re-evaluates each remaining entry of similarity matrix after each entry update. Last, the proposed algorithm is limited with Dynamic Time Warping where both the lower and upper bounding functions are available while

our algorithm can be used with any kind of similarity measure assuming that only lower bounding function is provided. Due to these differences, the comparison between this algorithm and our algorithm is thus excluded.

6.7 Conclusions

In this chapter, we propose a novel active density-based clustering algorithm to deal with the sparseness (incompleteness) of the pairwise similarity matrix in many applications. Based on the availability of a LB similarity matrix, Act-DBSCAN iteratively selects the most informative pairwise LB similarities to update with their true similarities and refines the cluster structure. The general idea is to reach as close to the desired clustering result of DBSCAN as possible with each update. Act-DBSCAN contains an efficient probabilistic model and a scoring system called the Shared Core Object (SCO) score to evaluate the impact of the update of each pairwise LB similarity on the change of the intermediate cluster structure. Deriving from the monotonicity and reduction property of our clustering scheme and the SCO score, the two algorithms Splitting with SCO (SP-SCO) and Merging with SCO (MG-SCO) provide two different and efficient ways to actively select and update pairwise similarities and cluster results. Experiments on various real datasets have shown that Act-DBSCAN requires only a tiny fraction of all pairwise similarities to reach the clustering results of DBSCAN. Act-DBSCAN also outperforms other related techniques such as active spectral clustering.

Part III

Application for Fiber Clustering

Chapter 7

Background on Fiber Segmentation

Recently, fiber segmentation has become an emerging technique in neuroscience for grouping the white matter fiber tracks acquired from Diffusion Tensor Imaging (DTI) into anatomical meaningful bundles for the study of various brain structures and diseases. In this Chapter, we briefly present some backgrounds and related works for fiber segmentation problem including Diffusion Tensor Imaging (DTI), fiber similarity measure techniques and fiber clustering techniques.

7.1 Diffuse Tensor Imaging

Understanding anatomical connectivity of human brain is one of the major challenges in neuroscience. However, how to study the brain structures in a non-invasive way is a critical issue. In the past decades, the emergence of the Diffusion Tensor Imaging (DTI) technology [194] provides a promising way to study the white matter structure in human brain in *vivo* by exploiting characteristics of water diffusion in biological tissues [194]. In such fibrous tissues, water tends to diffuse parallelly along fiber pathways. By following the major direction indicated by principal eigenvector of the underlying diffusion tensor field, the neural fiber tracts can be reconstructed as a set of 3D streamlines. Such kind of techniques is called fiber tractography [24, 194, 31]. Fiber tractography has been widely used in visualization and brain connectivity analysis and is an important tool surgical planning and in studies of various diseases such as Alzheimer and Parkinson. Figure 7.1 shows a full brain fiber dataset acquired from tractography technique as

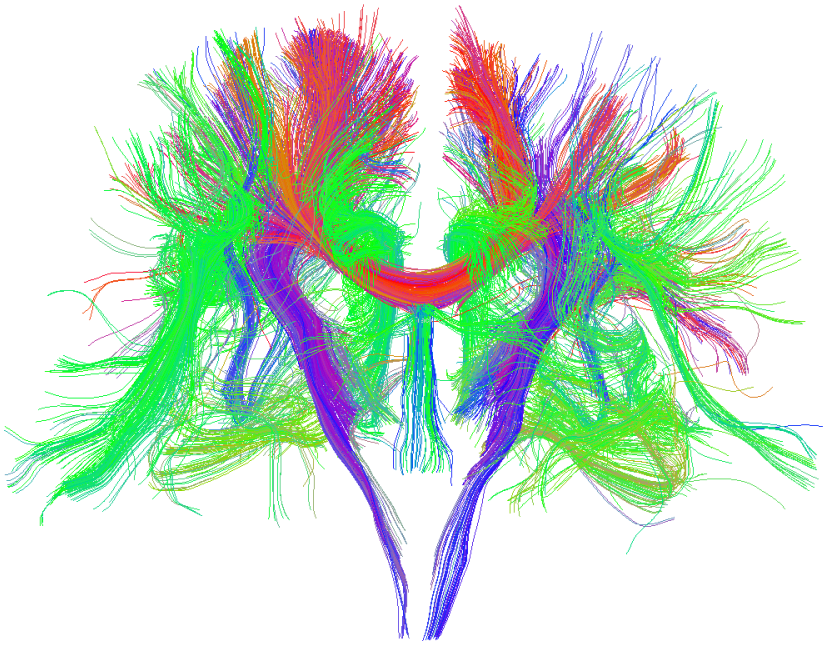

Figure 7.1: An examples of a full brain fiber dataset. Each fiber is represented by a streamline in three-dimensional space.

an example.

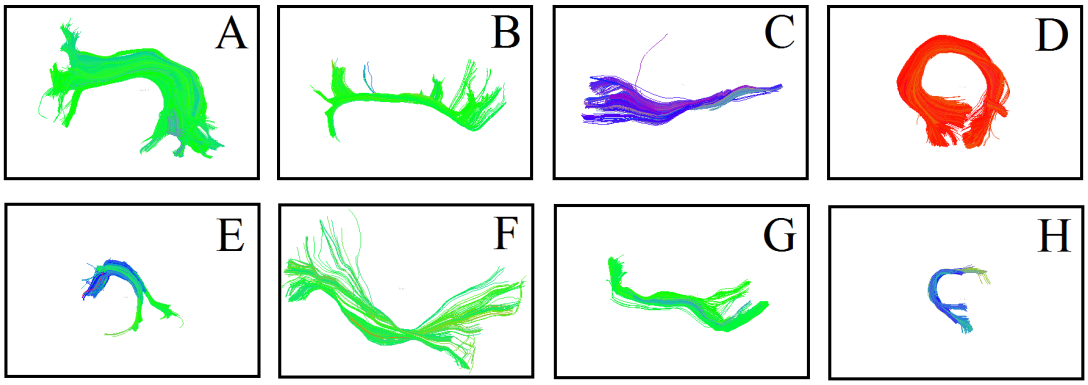

Figure 7.2: The eight major fiber bundles in human brain including Arcuate (A), Cingulum (B), Corticospinal (C), Forceps Major (D), Fornix (E), Inferior Occipitofrontal Fasciculus (F), Superior Longitudinal Fasciculus (G) and Uncinate (H). These bundles are parts of datasets used in our study.

In brain connectivity analysis, fibers are classified according to their anatomical function and shape. Fibers with same anatomical structure are grouped into a bundle. Each bundle connects different parts of brain and has different functions.

Figure 7.2 shows eight major fiber bundles in human brain including Arcuate, Cingulum, Corticospinal, Forceps Major, Fornix, Inferior Occipitofrontal Fasciculus, Superior Longitudinal Fasciculus and Uncinate. Each bundle connects different parts of brain. For example, the superior longitudinal fasciculus is a long associative bundle composed of long and short fibers that connect the frontal lobe with the parietal, occipital, and temporal lobes. The long fibers of the inferior longitudinal fasciculus bundle runs through the temporal lobe, connecting the temporal pole with the occipital pole. The short fibers emerge from nonpolar temporal and occipital areas and connect neighboring gyri. The uncinate fasciculus connects the temporal pole to the orbitofrontal cortex. The long fibers of the cingulum bundle connect the frontal lobe to the temporal lobe, whereas the short fibers connect neighboring areas of the cingulate and medial gyri of the frontal, parietal, occipital, and temporal lobes [55].

Fiber segmentation. The tractography process produces a large number of fibers (usually from 10^3 to 10^6 fibers), which embarrasses the analysis of the white matter structures. They need to be grouped into anatomical meaningful bundles before experts can use them for their studies. Various techniques are proposed to segment fibers into meaningful anatomical structures for quantification and comparison between individuals. Catani et al. [55] used a technique called *virtual dissection* to interactively select fibers passing through some manually defined regions of interests (ROIs). Though this technique is highly flexible, it is expensive and also very time consuming due to a large amount of complex fiber structures. Moreover, the results may be biased by subjective opinions of experts. Therefore, automatic fiber clustering algorithms, e.g., [64, 75, 50, 209, 208, 291, 192, 65, 168, 291, 140], which do not require user experts and thus exclude undesirable bias have become interesting and supplementary approach.

Contents. In this Chapter, we briefly review some techniques for fiber similarity measures (Section 7.2) and fiber segmentation algorithms (Section 7.3) proposed in the literature.

7.2 Fiber Similarity Measures

From DTI images, we acquire a set of fiber tracks after tractography. Each fiber trajectory is represented by an ordered set of points in 3D space with different numbers of points and arc-lengths. Providing an efficient and effective similarity

measure for fibers is one of the most important task for fiber segmentation problem.

Brun et al. [50] consider two fibers similar if their start and end points are close together. However, this assumption is not always reasonable, since fibers in the same bundle may start and end in different regions [55]. Also, the shape similarity is totally ignored.

Most successful techniques use point-to-point correspondences to measure similarity. Zhang et al. [292] used the average of distances from points in the shorter fiber to their closest points in the longer one if they are larger than a predefined threshold. The choice of the threshold may be a drawback of this approach. Ding et al. [75] defined similarity by using the mean Euclidean distance and ratio between corresponding segments of fibers. One of the major drawbacks of this technique is that there is no clue to find those segments.

Corouge et al. [64] introduced the three widely used similarity functions: closest point distance, mean of closest point distance (MCP) and Hausdorff distance (HDD) which measure fiber similarity by using distances between pairs of points of two fibers. Shao et al. [238] used Dynamic Time Warping (DTW) to measure shape similarity between fibers. All these techniques are sensitive to noise which may occur in fibers. Their distance mechanism is not strong enough to tell us whether two fibers have similar shape or they are separated by a small distance. The contribution of start and end points of fibers is also ignored although it plays an important role in the segmentation [55]. Moreover, they have $O(n^2)$ time complexity which is obviously undesirable especially for very large fiber datasets.

There exists some techniques which use some techniques to approximate fibers such as [142]. However, these techniques are out of scope of our study.

7.3 Fiber Segmentation Algorithms

Fiber segmentation algorithms are used to automatically segment fibers into anatomical meaningful bundles.

In Corouge et al. [64], two fibers are considered in a same cluster if the distance between them is lower than a predefined threshold. Clusters with a low cardinality (less than 10% of total number of fibers) are regarded as outliers and thus are removed from the final results. To construct clusters, the algorithm calculates the distance matrix among all fibers and then propagates label from neighboring fiber to neighboring fiber until all fibers are labeled. Though the algorithm is simple, it may connect different bundles together due to its transitivity property.

The k-most-similar-fibers algorithm from Ding et al. [75] has similar label propagation mechanism with [64]. However, a label of a fiber is only propagated to its k-nearest neighbor instead of all neighbors that lie within a predefined threshold like [64]. This algorithm suffers from the same problem with [64].

Brun et al. [50] use spectral embedding techniques called Laplacian eigenmaps to map the fibers to a 3D feature space forming by the first, second and third largest eigenvectors. Each point in feature space is then mapped to a color using a color map. A fiber will be colored according to the color of its corresponding point. We note that, this algorithm is not a clustering algorithm. It is only used to color the fibers. In [141, 208], spectral clustering algorithms are used to group fibers into bundles. The main different between these algorithms and the algorithm of Brun et al. is that k-Means is used on the feature space to group the fibers instead of mapping the colors to fibers. One significant drawback of these spectral clustering algorithms is that they require quadratic space and have cubic time complexity. These make them infeasible when clustering large fiber datasets.

In [292], an agglomerative hierarchical clustering method was used to group fibers. It starts by putting each data point into an individual cluster, then at each stage of the algorithm the two most similar clusters are joined until all fibers are in a same cluster. The main drawback of this techniques is to find a suitable partition of the dendrogram.

The algorithm k-Means was used in [252] to group fibers in low dimensional projected space acquired from Locally Linear Embedding (LLE) technique on the distance matrix produced by the minimum spanning trees between pairwise fiber tracts. One major drawback of this technique is that it may not be able to deal with large fiber datasets.

Recently, the density-based clustering algorithm was used in [237] in order to group fibers and to reject spurious fibers as outliers. This algorithm requires parameters that may be hard to set.

In contrast to previous algorithms which produce a hard segmentation of fiber structure, the EM clustering algorithm is used to produce a soft segmentation of fibers into bundle in [178, 179].

7.4 Conclusions

Fiber clustering provides a useful tool in neuroscience for visualization and brain connectivity analysis. Though, there are many fiber clustering algorithms pro-

posed in the literature, they are still suffer from various problems, especially performance on large fiber datasets.

In order to enhance the efficiency, we propose a novel similarity model for fibers based on the combination of the shape and the connectivity similarity which helps to improve the classification accuracy [35, 183]. To enhance the performance when segmenting large fiber datasets, we introduce the novel concept of anytime fiber clustering algorithm which trades off between quality of results and runtime [185]. To the best of our knowledge, there is no anytime fiber clustering algorithm proposed in the literature so far.

Chapter 8

Advantage Fiber Similarity Measure Techniques

Though there are many similarity measures for fiber tracts proposed in the literature, they mostly focus on the shape similarity or have high time complexity. Thus, this limits their efficiency. In this Chapter, we propose a novel similarity model for fiber tracts based on shape similarity and connection similarity. For shape similarity, we propose some new techniques adapted from existing similarity measures for trajectory data. Besides, a new technique called Warped Longest Common Subsequence (WLCS) for which we additionally developed a lower bounding distance function to speed-up the segmentation process is also proposed. Our segmentation algorithm is based on an outlier-robust density-based clustering algorithm. Extensive experiments from diffusion tensor images of the white matter of the brain demonstrate the efficiency and effectiveness of our technique.

Publications. Parts of the material presented in this Chapter have been published in [183, 35]. The detailed information are described as follows:

- Christian Böhm, Jing Feng, Xiao He, Son T. Mai, Claudia Plant and Junming Shao. A Novel Similarity Measure for Fiber Clustering using Longest Common Subsequence. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Workshop on Data Mining for Medical and Healthcare (DMMH), pages 1-9, 2011.

  In this work, S.T.M. developed the theory and the algorithms and implemented them. J.F and X.H helped with some experiments. J.S. provided

some datasets for experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

- Son T. Mai, Sebastian Goebl and Claudia Plant. A Similarity Model and Segmentation Algorithm for White Matter Fiber Tracts. In International Conference on Data Mining (ICDM), pages 1014-1019, 2012.

  In this work, S.T.M. developed the theory, implemented the algorithm and did experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

8.1 Introduction

Over the past decades, Diffusion-Tensor Magnetic Resonance Imaging (DTI) has become an important tool for quantification and comparison of white matter structures of human brains in *vivo* [194]. From DTI images, thousands of fiber tracts are extracted via *tractography* technique. They need to be grouped into meaningful anatomical structures for quantification and comparison between individuals by some fiber segmentation techniques.

Most automatic techniques for segmenting fibers are based on geometric properties of fibers. Two fibers are usually grouped into a bundle if they are separated by a small distance, have comparable length and have similar shape [75]. However, these criteria might be insufficient, since two fibers with different shapes can be grouped into a bundle if they start and end at the same region [50]. Moreover, the white matter tracts contain many spurious and noisy fibers which make the similarity measure and segmentation non-trivial problems. So, fiber segmentation remains an area of active research [50, 64, 75, 292], etc.

The most successful techniques use point-to-point distance as a basis for measuring similarity [64, 75, 292, 238]. However, these techniques are sensitive to noise due to their point-to-point distance mechanism. Their effectiveness is also limited when detecting fibers with similar shapes. They also ignore the contribution of the start and end points of fibers which actually plays an important role in the segmentation [55, 50]. Moreover, their quadratic time complexity makes them hard to deal with very large fiber datasets.

Contributions. In this Chapter, we investigate the problem of the efficient and effective similarity measure for the segmentation of the white matter fiber tracts

in human brain. Our major contributions are:

1. We propose some new techniques for fiber similarity measure by employing and adapting various existing similarity techniques for trajectory databases [195, 74] and study their performances in comparison with existing techniques for fibers [64]. Such comparison is essential and interesting since the performances of these new techniques have never been studied properly in neuroscience.

2. For shape similarity, we introduce a new view about the similarity of fibers using fiber envelope. Based on this scheme, a technique called Warped Longest Common Subsequence technique (WLCS) is also proposed. Compared with other techniques, WLCS is more accurate and more robust to noise and local time shifting within fibers. A lower bounding distance is also proposed for WLCS to speed up the comparison thus enhancing the efficiency of segmentation process.

3. We introduce a novel and robust similarity model called SIM for fiber segmentation by combining both shape similarity and connection similarity of fibers. Such approach provides a robust and flexible way to deal with the complexity of white matter structures.

4. Extensive experiments on real datasets are conducted to demonstrate the efficacy of our algorithms and to provide a close view about their characteristics.

The rest of this Chapter is organized as follows. In Section 8.2, we introduce various new techniques for fiber similarity measure including WLCS technique, its lower bounding and our fiber similarity model. Section 8.3 presents fiber segmentation algorithm. Experiments are displayed in Section 8.4. Further discussions about our approaches, related works and other important issues are located in Section 8.5. Finally, Section 8.6 summarizes our work.

8.2 Fiber Similarity Measure

Providing a similarity measure for fibers is an essential problem in any automatic fiber segmentation algorithm. During the past decades, many techniques are proposed such as [50, 75, 292, 64, 238, 252, 142]. However, most of them suffer from

high time complexity such as [64, 238]. Their performance are somewhat limited due to the complexity of fiber structure. Also, most algorithms focus only on the shape similarity and ignore the contribution of start and end points of fibers.

In this section, we propose a novel fiber similarity model which encapsulates structure and connectivity similarity of fibers. We propose some new techniques to measure shape similarity of fibers by adapting various existing techniques for trajectory databases. We also introduce a new technique called Warped Longest Common Subsequence and its lower bounding distance for fibers.

8.2.1 Shape Similarity of Two Fibers

After tractography, fibers are extracted from DTI images and represented as a set of streamlines in 3D space.

To measure the shape similarity between two fibers A and B, we build an ϵ-envelope around A, and then compare this envelope with B. If B is inside the envelope of A, they have similar shape. Such comparison provides a new view about the shape similarity, which differs from the previous approaches like distance-based techniques [64]. Consider Figure 8.1, by distance-based mechanism, we cannot know whether the shape of fiber B or C is more similar to A, because $Dist(A, B) \approx Dist(A, C)$. However, the envelope scheme successfully discovers that the shape of C is more similar to A than to B, because a large part of C lies inside the envelope of A.

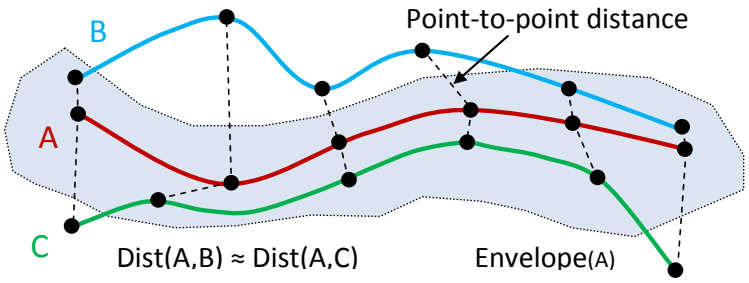

Figure 8.1: By distance-based techniques, both B and C are similar to A. By envelope-based techniques, C is more similar to A than to B.

This envelope scheme can be simulated by considering fibers as trajectories and adapting existing techniques such as LCS [261] or EDR model [57] to measure

their shape similarity. Table 8.1 shows the definitions of LCS and EDR adapted for the shape similarity of fibers together with the existing techniques HDD, MCP [64] and DTW [238]. The parameter *subcost* of EDR equals to 1 if $a_i \sim b_j$ and 0 otherwise. All algorithms have $O(n^2)$ complexity.

Since LCS and EDR only allow one-to-one matching mechanism, the measure is coarse. As a result, they do not distinguish fibers with similar common subsequences, although they belong to different bundles. They also perform poorly when fibers have very few points or have very different lengths. To enhance the accuracy, one-to-many matching mechanism like DTW [261] should be employed, which means that one point in a fiber can be matched to many points in the others. The similar idea can be found in music retrieval community. Guo et al. [102] proposed Time-Warped Longest Common Subsequence (T-WLCS) technique to efficiently deal with problems like variations in speed and inaccuracies in the rhythm. We extend this scheme to deal with 3D fiber trajectories.

Assuming that we have two fibers $A = (a_1, \cdots, a_n)$ and $B = (b_1, \cdots, b_m)$. And let A_i be the first i points of A.

Definition 20 *Two fibers A and B are close to each other at position i and j respectively if the coordinates of points at i and j are not different more than a predefined similarity threshold ϵ.*

$$a_i \sim b_j \Leftrightarrow |a_i(x) - b_j(x)| \leq \epsilon \wedge |a_i(y) - b_j(y)| \leq \epsilon \\ \wedge |a_i(z) - b_j(z)| \leq \epsilon$$

Definition 21 *Given a time constraint δ and a similarity threshold ϵ, the longest common subsequence of two fibers A and B, denoted as $wlcs_{\delta,\epsilon}(A_n, B_m)$ (or $wlcs_{\delta,\epsilon}(A, B)$ in short), is defined as follows:*

$$wlcs_{\delta,\epsilon}(A_n, B_m) = \begin{cases} 0 \quad \text{if } A \text{ or } B \text{ is empty} \\ 1 + max(wlcs_{\delta,\epsilon}(A_{n-1}, B_{m-1}), \\ \qquad wlcs_{\delta,\epsilon}(A_n, B_{m-1}), \\ \qquad wlcs_{\delta,\epsilon}(A_{n-1}, B_m)) \\ \qquad \text{if } a_n \sim b_m \wedge |n - m| \leq \delta \\ max(wlcs_{\delta,\epsilon}(A_{n-1}, B_m), \\ \qquad wlcs_{\delta,\epsilon}(A_n, B_{m-1})) \text{ otherwise} \end{cases}$$

where δ constrains the matching regions in time to avoid two sequences to be compared at too far away positions, which may be nonsense and unnecessary.

WLCS can also be calculated by using dynamic programming approach [261] to construct a cumulative cost matrix $M_{n\times m}$, where the value of $M_{i,j}$ can be calculated by the values of its adjacent cells. The time complexity is thus $O(\delta(n+m))$.

Definition 22 *Given two fiber A, B and a cost matrix $M_{n\times m}$, the warping path $P = p_1, ..., p_K$ is defined as a sequence of matrix cells $M_{i,j}$, in which the points at position i of A and j of B match. For any two parts $p_k = M_{i,j}$ and $p_{k+1} = M_{i',j'}$, the following properties hold:*

- *Non-Boundary: The warping path needs not to start at $p_1 = M_{1,1}$ and end at $p_K = M_{n,m}$.*
- *Monotony: $i' - i \geq 0$ and $j' - j \geq 0$, which means that the warping path is monotonically spaced in time.*
- *Non-continuity: Some parts of A and B may remain unmatched, which means that the property $i' - i \leq 1$ and $j' - j \leq 1$ does not hold.*
- *The length K of the warping path (or the value of $wlcs(A, B)$) cannot be larger than $n + m - 1$.*

Figure 8.2 illustrates the calculation of WLCS for two fibers A and B in 1D. One point in fiber A can be matched with many points in fiber B and vice versa.

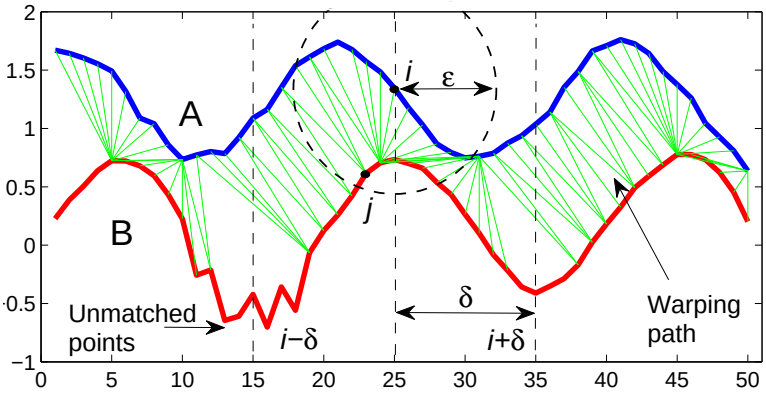

Figure 8.2: The comparison of two fibers A and B by WLCS model in 1D with time constraint δ and similarity threshold ϵ. For each point a_i in fiber A, every point b_j in fiber B (with j in $[i-\delta, i+\delta]$) which lies inside the ϵ-circle of a_i can be matched with a_i.

Definition	**Similarity Function**
$d_{hdd}(A,B) = max_{a_i \in A}\{min_{b_j \in B}\|a_i - b_j\|\}$	$HDD(A,B) = max(d_{hdd}(A,B), d_{hdd}(A,B))$
$d_{mcp}(A,B) = avg_{a_i \in A}\{min_{b_j \in B}\|a_i - b_j\|\}$	$MCP(A,B) = avg(d_{mcp}(A,B), d_{mcp}(A,B))$
$d_{dtw}(A,B) = \|a_n - b_m\| + min(d_{dtw}(A_{n-1}, B_m), d_{dtw}(A_n, B_{m-1}), d_{dtw}(A_{n-1}, B_{m-1}))$	$DTW(A,B) = \frac{d_{dtw}}{K}$
$lcs_{\delta,\epsilon}(A_n, B_m) = \begin{cases} 0 \quad \text{if } A \text{ or } B \text{ is empty} \\ 1 + lcs_{\delta,\epsilon}(A_{n-1}, B_{m-1}) \\ \quad \text{if } a_n \sim b_m \wedge \lvert n-m\rvert \leq \delta \\ max(lcs_{\delta,\epsilon}(A_{n-1}, B_m), \\ \quad lcs_{\delta,\epsilon}(A_n, B_{m-1})) \text{ otherwise} \end{cases}$	$LCS_{\delta,\epsilon}(A,B) = 1 - \frac{lcs_{\delta,\epsilon}(A,B)}{min(n,m)}$
$edr_{\delta,\epsilon}(A_n, B_m) = \begin{cases} n \quad \text{if } B \text{ is empty} \\ m \quad \text{if } A \text{ is empty} \\ min(edr_{\delta,\epsilon}(A_{n-1}, B_{m_1}) + subcost, \\ \quad edr_{\delta,\epsilon}(A_n, B_{m-1}) + 1, \\ \quad edr_{\delta,\epsilon}(A_{n-1}, B_m) + 1) \text{ otherwise} \end{cases}$	$EDR_\epsilon(A,B) = \frac{edr_\epsilon(A,B)}{max(a,b)}$

Table 8.1: Some similarity measure techniques for fibers used in our papers. MCP, HDD and DTW are existing techniques, and LCS and EDR are new adapted techniques.

In order to be able to compare sequences with different lengths, we need to normalize the cost of WLCS.

Definition 23 *Given a time constrain δ and a similarity threshold ϵ, we have:*

$$WLCS_{\delta,\epsilon}(A,B) = 1 - \frac{wlcs_{\delta,\epsilon}(A,B)}{n+m-1}$$

We use $WLCS_{\delta,\epsilon}(A,B)$ as the shape similarity of two fibers A and B. The smaller the value of $WLCS_{\delta,\epsilon}(A,B)$ is, the more similar the shapes of two fibers A and B are.

8.2.2 Lower Bounding Distance for WLCS

In this part, we propose a lower bounding distance to speed up the comparison of WLCS. We assume w.l.o.g. that fiber A is longer than fiber B. The lower bounding distance of $WLCS_{\delta,\epsilon}(A,B)$ can be calculated by using the Minimum Bounding Envelope of A ($MBE_{\delta,\epsilon}(A)$) [261]. For clarity, we also assume that A and B are now 1D fibers. However, the notion of the $MBE_{\delta,\epsilon}$ can be trivially extended to 3D.

$$Envlow \leq MBE_{\delta,\epsilon}(A) \leq Envhigh$$

where

$$\begin{cases} Envhigh_i = max(a_j) + \epsilon & \forall j, |i-j| \leq \delta \\ Envlow_i = min(a_j) + \epsilon & \forall j, |i-j| \leq \delta \end{cases}$$

Figure 8.3 illustrates the construction of $MBE_{\delta,\epsilon}(A)$. The envelope defines areas of possible matching for all points of A. Every point which lies outside MBE of A can never be matched.

Definition 24 *The longest common subsequence between B and $MBE_{\delta,\epsilon}(A)$ is the number of points of B which lie inside the $MBE_{\delta,\epsilon}$ of A.*

$$lcs(MBE_{\delta,\epsilon}(A),B) = \sum_{i=1}^{n} \begin{cases} 1 \text{ if } b_i \text{ is in the envelope of } A \\ 0 \text{ otherwise} \end{cases}$$

Lemma 9 *For two fibers A and B, $lcs(MBE_{\delta,\epsilon}(A),B)+n-1$ is the upper bound of $wlcs_{\delta,\epsilon}(A,B)$ with assumption that A is longer than B.*

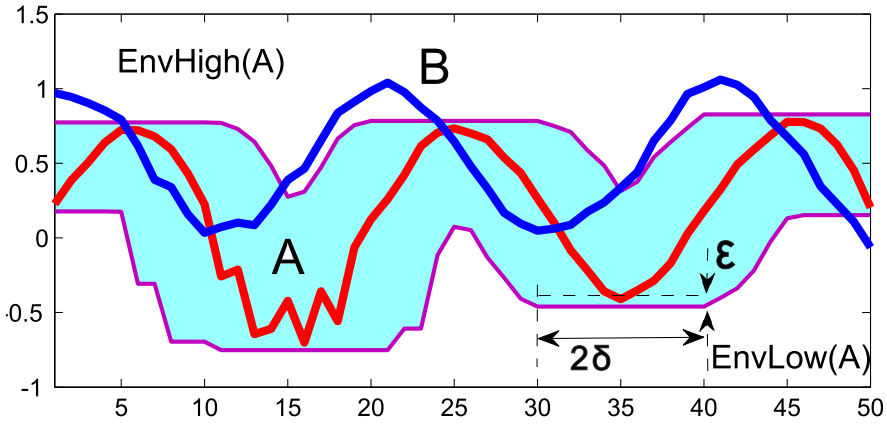

Figure 8.3: The Minimum Bounding Envelope (MBE) of fiber A with respect to the time constraint δ and similarity threshold ϵ. Only points of B that lie inside $MBE_{\delta,\epsilon}$ of A can be matched with A.

Proof 9 *Let x be the number of points of B which lie outside the MBE of A. For any point $b_k \in B$ which lies outside MBE of A, assuming that b_i and b_j are the closest front and rear points of b_k which lie inside MBE of A. Also, $a_{i'}$ and $a_{j'}$ are matched points of b_i and b_j under WLCS respectively. There always exists a way to add a pair $(b_k, a_{k'})$ where $i' \leq k' \leq j'$ to the warping path P which does not violate the monotony property in Definition 22 (see Figure 8.4 as an example). Since the full-length warping path is bounded by $n+m-1$, we have:*

$$
\begin{aligned}
& wlcs_{\delta,\epsilon}(A,B) + x \leq n + m - 1 \\
\Rightarrow\ & wlcs_{\delta,\epsilon}(A,B) \leq m - x + n - 1 \\
\Rightarrow\ & wlcs_{\delta,\epsilon}(A,B) \leq lcs(MBE_{\delta,\epsilon}(A),B) + n - 1.
\end{aligned}
$$

Lemma 10 *For two fibers A and B, $1 - \frac{lcs(MBE_{\delta,\epsilon}(A),B)+n-1}{m+n-1}$ is the lower bound of $wlcs_{\delta,\epsilon}(A,B)$ (assuming that A is longer than B).*

Proof 10 *Straightforward derived from Definition 23 and Lemma 9.*

8.2.3 Connection Similarity of Two Fibers

Definition 25 *Given two fibers A and B with the start points p_A, p_B and the end points q_A, q_B. The connection similarity between them is defined as follows:*

$$Conn(A,B) = \|p_A - p_B\| + \|q_A - q_B\|$$

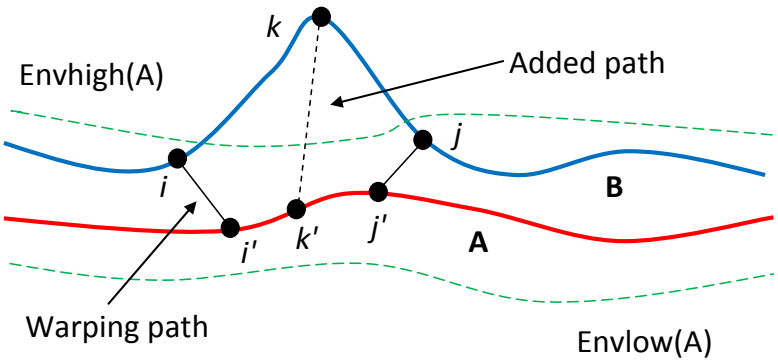

Figure 8.4: A matched pair $(b_k, a_{k'})$ can be added to the existing warping path without violating the monotony property.

The connection similarity measures how close the two fibers start and end. Thus it plays an important role to distinguish fiber bundles. Many techniques rely only on this scheme to segment fibers either manually [55] or automatically [50].

8.2.4 Unified Fiber Similarity Measure

Definition 26 *The similarity between two fibers A and B is defined as a weighted sum of their shape and connection similarities.*

$$Sim_{\delta,\epsilon,\alpha}(A,B) = \alpha \cdot Shape_{\delta,\epsilon}(A,B) + (1-\alpha) \cdot Conn(A,B)$$

where $\alpha \in [0,1]$ is used to control the balance between the shape and connection similarities.

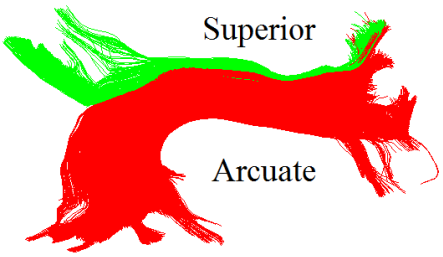

Figure 8.5: Two bundles Arcuate and Superior Longitudinal Fasciculus are close to each other and hard to distinguish if only based on the shape similarity.

$Sim_{\delta,\epsilon,\alpha}(A,B)$ unifies both important aspects: anatomical structure similarity and connectivity similarity of fiber bundles. As a result, it provides a flexible way

to enhance the effectiveness of fiber similarity measure. For example, with two close bundles Arcuate and Superior (Figure 8.5), it is hard to distinguish them only by shape similarity. Therefore, the use of connection similarity could help to enhance the result.

8.2.5 Other Important Characteristics of Fiber Similarity

Due to the process of DTI tractography, the fibers may contain noise, which affects the similarity between them [55]. Assuming that A and B in the upper part of Figure 8.6 are two real fibers and in the lower parts are noisy fibers, we calculate $EDR_{0.2}$, $LCS_{10,0.2}$, DTW, HDD, MCP and $WLCS_{10,0.2}$ (or $SIM_{10,0.2,1}$)) to see the effect of noise in each measure. As we see, LCS, EDR and WLCS are more robust to noise than other methods. Their values do not change because the noisy parts are just ignored.

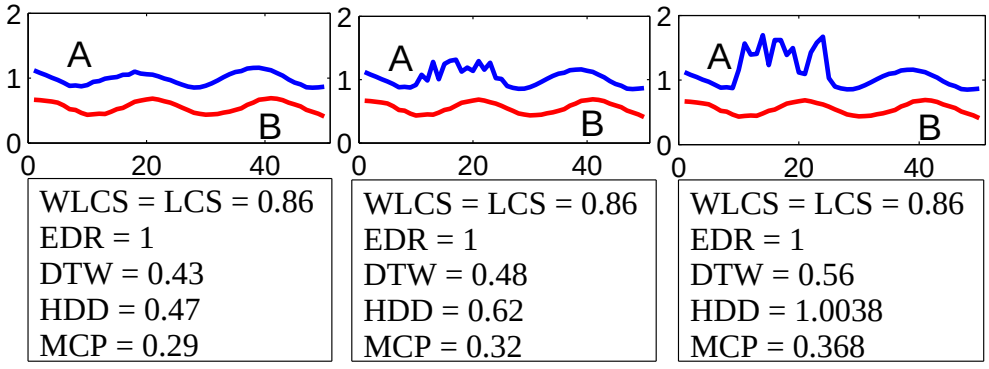

Figure 8.6: The effect of noise on different similarity measure functions between two fibers A and B. LCS, EDR and WLCS (SIM) are more robust to noise than other techniques.

When using techniques like DTW, LCS, EDR and WLCS, the similarity between A and B changes significantly with respect to the orders of points in A and B. This phenomenon often happens in tractography when one fiber is recognized in a direction which is contrary to the rests in a group. To overcome this problem, we use 2-phases approach. We calculate the similarity between (A, B) and ($Reverse(A)$,B) and choose the smallest result as follows:

$$Sim2_{\delta,\epsilon,\alpha}(A,B) = min(Sim_{\delta,\epsilon,\alpha}(A,B) + Sim_{\delta,\epsilon,\alpha}(Reverse(A),B)$$

Figure 8.7 (a) shows the result of the clustering using 1-phase approach. There are many false direction fibers which could not be grouped correctly. Figure 8.7 (b) shows perfect clustering result (exactly the same as gold standard) if we use 2-phases approach. In the rest of this chapter, we always use the 2-phases approach for DTW, LCS, ERD and WLCS.

Figure 8.7: The results of clustering using (a) 1-phase similarity measure (10 clusters) and (b) 2-phases similarity measure (5 clusters). The 2-phases approach produces the gold standard exactly.

8.3 Fiber Segmentation

To segment fibers into bundles, many clustering algorithms are used such as such as EM clustering [178], spectral clustering [50], hierarchical clustering [292] and density-based clustering [238, 35, 237].

In density-based clustering algorithms, clusters are considered as areas of high object density and separated by areas of low object density. This notion has several attractive benefits. It helps to detect clusters of arbitrary shapes, and is robust to outliers. Moreover, users do not need to specify the number of clusters. Among many proposed approaches namely DENCLUE, OPTICS and DBCLASD, we employ the well-known algorithm DBSCAN [83] to segment fiber bundles and to reject spurious fibers. DBSCAN is based on the idea that each core-point of a cluster has to contain at least *minpts* points within its *eps*-neighborhood. Figure 8.8 shows the pseudocode for DBSCAN clustering algorithm.

```
Function DBSCAN(D, minpts, eps)
  currentID = Fist_ID
  set all object p in D to unprocessed
  for all object p in D do
    if p is processed then continue endif
    S = rangeQuery(p, ε, D)
    if S.size < minpts then
      assign cluster id of p as noise
    else
      assign cluster id of p as currentID
      assign cluster id for all objects in S as CurrentID
      while S not empty do
        q = S.first( )
        T = rangeQuery(q, ε, D)
        if T.size >= minpts then
          for all objects r in T do
            if r is unprocessed or noise
              if r is unprocessed then
                insert r into S
              endif
              set cluster id of r as currentID
            endif
          endfor
        endif
        remove q from S
      endwhile
      currentID = Next_ID
    endif
  endfor
EndFunction
```

Figure 8.8: Pseudocode for DBSCAN algorithm used in our work.

8.4 Empirical Evaluation

In this section, we present our experiments on real datasets to prove the efficiency and effectiveness of our algorithm. First, we study the performances of different shape similarity measure techniques including HDD, MCP, DTW, LCS, EDR and WLCS. Then, we demonstrate the effectiveness of the unified similarity measure (SIM) and explore some of its characteristics. Last, we discuss the efficiency of the algorithms.

All experiments are conducted on a Workstation with 2.0 Ghz CPU, 4GB RAM under Window XP SP2 using Java language. The labeled datasets are acquired from Pittsburgh Brain Connectivity Competition (http://pbc.lrdc.pitt.edu/?q=20-09b-home).

To compare the clustering results with the gold standards, we use three different cluster evaluation methods, namely Dom [76], NMI [258] and AMI [258]. These methods, in contrast to others, namely Rand Index or Cluster Purity, can compare results with different numbers of clusters. However, we only show the NMI for clarity, since the results of AMI and Dom are the same with NMI. The result of NMI is in [0,1], with 0 means that the clustering result is independent of the gold standard and 1 means that the clustering result is the same as the gold standard.

For DBSCAN, we need to adjust two parameters *minpts* and *eps*. After trying some values, we fix the parameter $minpts = 5$ as suggested in [83]. For the parameter *eps*, we explore the search space from the minimum value 0.01 to 1.0 with search step 0.01 to ensure that we do not miss any good result. For DTW, LCS, EDR and WLCS, we try various combinations of δ (up to 100) and ϵ (between 0.01 to 2.0) and report the best found results.

8.4.1 Effectiveness of The Shape Similarity Measures

To evaluate the effectiveness of different similarity measures, we use one nearest neighbor classification technique as suggested by Keogh et al. [130]. The class label of each fiber is predicted based on the class label of its nearest neighbor. The error rate is then defined as the percentage of wrongly predicted fibers.

We randomly extract 5 labeled datasets from the PBC datasets. Each data set contains from 500 to 1500 fibers that belong to 5 to 8 different bundles. To evaluate the robustness of the similarity measure, we also create 10 noisy datasets

⋆	HDD	MCP	DTW	LCS	EDR	WLCS
Norm	2.815	0.367	0.319	0.228	0.135	0
Noisy	13.03	3.110	3.208	1.287	1.724	0.64

Table 8.2: The error rates of one nearest neighbor classification for the six similarity measure techniques.

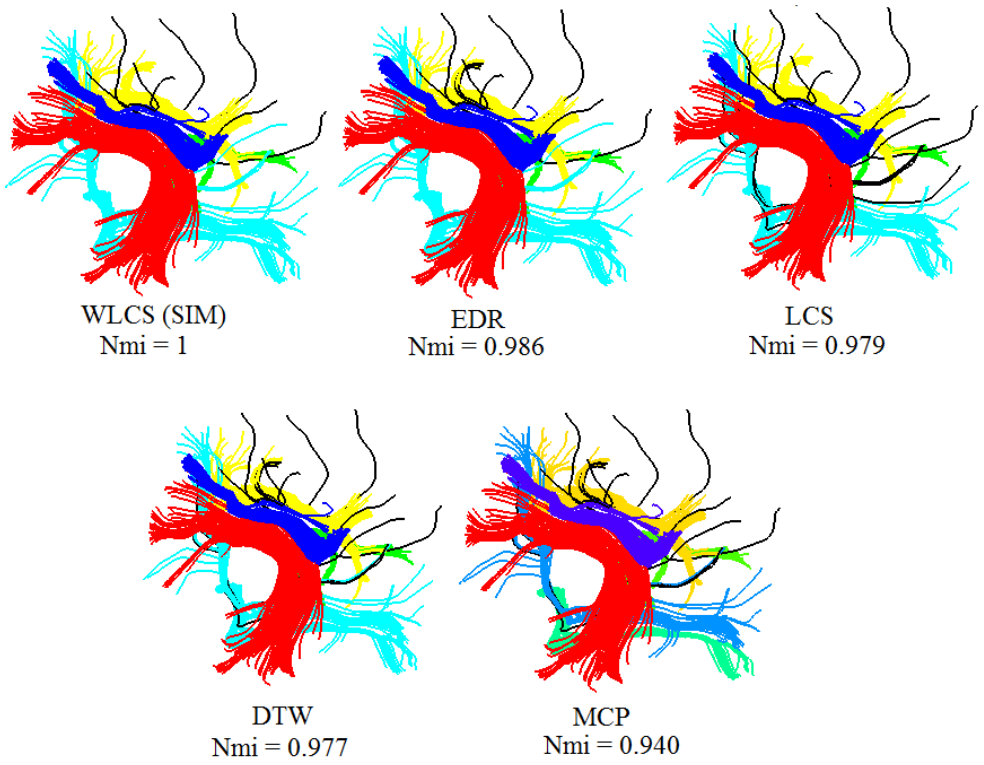

Figure 8.9: Effectiveness of different shape similarity measure techniques for real data set DS0. Only WLCS produces the gold standard exactly.

by adding some Gaussian noise and local time shifting to each fiber of the five datasets above as in [57]. Then, for each data set, 25 fibers are randomly selected as the training set, the rest is used as the test set. Table 8.2 shows the average results of the error rate for each technique over the 5 normal and 10 noisy datasets. WLCS performs better than other techniques on normal and especially on noisy datasets. Since HDD uses max distance of closest point-pairs as a measure, it is extremely affected by noise and performs worst. All threshold-based techniques LCS, EDR and WLCS show better performances than distance-based techniques like HDD, MCP and DTW.

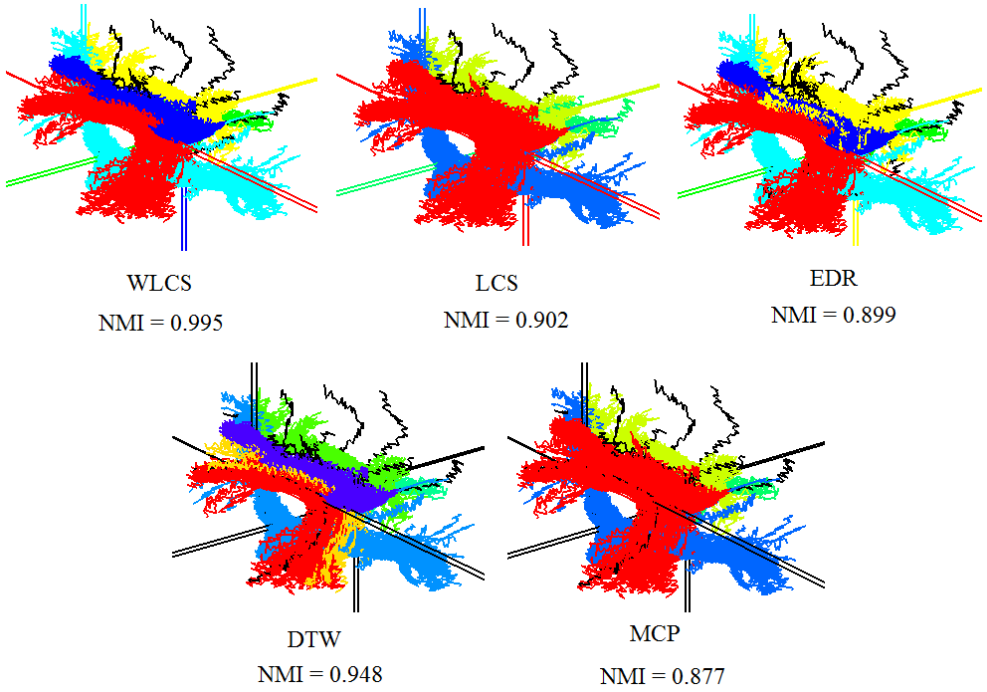

Figure 8.10: Effectiveness of different shape similarity measure techniques for noisy data set. The clustering score of WLCS reduces slightly compared with the clustering scores of other techniques. WLCS is more robust to noise than the others.

Figure 8.9 shows clustering results on data set (DS0) which contains 500 fibers belonging to 5 bundles namely Arcuate, Cingulum, Fornix, Inferior and Superior. Also, 5 fibers from other bundles are added as noise. Only WLCS ($\delta = 75$, $\epsilon = 0.1$, $eps = 0.24$) produces the gold standard exactly, while HDD ($eps = 0.26$) and MCP ($eps = 0.06$) result in 6 clusters. DTW ($\sigma = \infty$, $eps = 0.11$), LCS ($\sigma = 75$, $\epsilon = 0.07$, $eps = 0.12$) and EDR ($\epsilon = 0.17$, $eps = 0.18$) detect 5 clusters with some minor errors. We do not show HDD in Figure 8.9 for clarity.

In order to to see how all techniques perform on noisy datasets, we add 5% random Gaussian noise to dataset DS0 and do the clustering again. Figure 8.10 shows the results for noisy dataset. While the clustering scores of WLCS is slightly reduced, the clustering scores of other techniques are significantly reduced. WLCS is more robust to noise than other techniques.

Let us consider another important aspect of the effectiveness of similarity measure. As we know, the parameter *eps* of DBSCAN specifies the range of the core objects. Thus, it plays an important role to distinguish fiber bundles. Therefore,

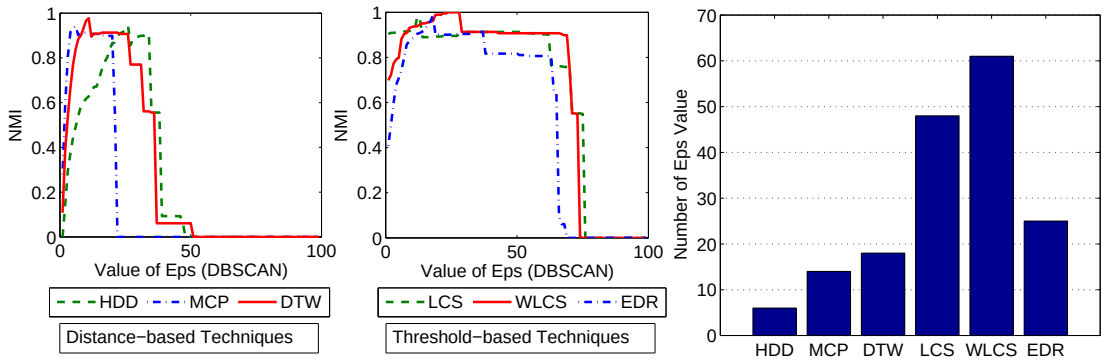

Figure 8.11: Effectiveness of different similarity measure techniques based on the range of *eps*. WLCS is more robust and can distinguish bundles better than other techniques.

a better similarity measure should support the wider range of *eps* (assuming that we fixed *minpts*). In this experiment, we let *eps* run in the range from 0.01 to 1.0 with step size 0.01 and count the numbers of *eps* values which result in NMI score larger than 0.9 on the data set DS0 in Figure 8.9. As we can see from Figure 8.11, the performances of WLCS and LCS are very stable, while the performances of the others decreases quickly when *eps* becomes bigger. Compared with the predefined threshold, WLCS succeeds 61 over 100 times, while the values for LCS are 48 times. The performance of EDR varies significantly with different values of ϵ. However, it is still more stable than DTW, MCP and HDD. Compared with distance-based techniques, threshold-based techniques are more robust. The results are the same for all other datasets in our experiments.

Further comparisons can be found in Table 8.3. WLCS outperforms other techniques on all real datasets. LCS, EDR and DTW are ranked next while HDD performs worst. All threshold-based techniques provide better results than distance-based techniques since they are based on the efficient envelope-based scheme for fiber similarity.

To conclude, WLCS acquires better and more stable performance than other techniques on both normal and noisy fiber datasets.

8.4.2 Effectiveness of the Unified Similarity Measure SIM

In this section, we examine the effectiveness of the unified similarity measure (SIM) and its characteristics. For the parameters of SIM, we simply choose $\delta = 50$,

$\epsilon = 0.05$ and $\alpha = 0.5$ unless otherwise stated.

Figure 8.12 demonstrates the clustering ability of SIM on 4 real datasets DS1, DS2, DS3 and DS4. All of them are clustered exactly as the gold standard.

Table 8.3 shows the clustering results for various techniques on the 10 real datasets DS0 to DS9, which are randomly extracted from labeled PBC datasets. These datasets contain 500 to 1500 labeled fibers belonging to 8 famous bundles namely Arcuate, Cingulum, Fornix, Inferior Occipitofrontal Fasciculus, Superior Longitudinal Fasciculus, Forceps Major and Corticospinal. Some fibers from other groups are also added as noise. SIM significantly improves the clustering results. DS0-5 and DS7-8 are clustered exactly as gold standards, the other three are grouped with nearly perfect results. To further evaluate the performances of our algorithms on the task of whole brain clustering, we use 10 more datasets. Each dataset contains 5000 fibers which are randomly extracted from the PBC whole brain data set. The clustering scores are measured based on the labeled fibers only. The last column (Full Brain) in Table 8.3 shows the averaged NMI scores for all techniques. Threshold-based techniques are better than distance-based techniques. WLCS outperforms the other shape similarity measure techniques. And SIM significantly improves the clustering results.

Figure 8.13 shows clustering results for some real datasets acquired from [238] to assess our algorithm. Although we do not have gold standards to compare, all results are well confirmed by our experts. DF1 was clustered with $\delta = 100$, $\epsilon = 0.1$ and $eps = 0.31$. DF2 was clustered with $\delta = 100$ and $eps = 0.46$.

8.4.3 Characteristics of SIM

Although SIM is superior to other techniques, it requires 3 parameters: the time constraint δ, the similarity threshold ϵ and the weight α, which may confuse us at the first glance. However, these parameters are relatively easy to set up.

Figure 8.15 shows the relationships between δ, ϵ and NMI score for the data set DS0. SIM is more robust to the choices of the parameters δ and ϵ than WLCS. The combination with connection similarity not only enhances the effectiveness but also the robustness of the fiber similarity measure. The same results are observed with the other datasets as well.

Figure 8.12: Clustering results with SIM for 4 datasets DS1, DS2, DS3 and DS4 with $eps = 0.62, 0.57, 0.41$ and 0.48 respectively.

$\star$	DS0	DS1	DS2	DS3	DS4	DS5	DS6	DS7	DS8	DS9	Full Brain
HDD	0.943	0.925	0.952	0.953	0.945	0.938	0.974	0.954	0.955	0.945	0.871
MCP	0.940	**1**	**1**	0.953	0.976	0.935	0.907	0.904	0.922	0.971	0.897
DTW	0.977	**1**	**1**	0.991	0.976	0.970	0.907	0.975	0.972	0.975	0.907
LCS	0.979	**1**	**1**	0.993	0.986	0.972	0.976	0.989	0.962	0.981	0.916
EDR	0.986	**1**	**1**	0.974	0.976	0.986	0.932	0.957	0.958	0.973	0.913
WLCS	**1**	**1**	**1**	0.993	0.990	**0.995**	0.984	0.991	0.979	0.993	0.932
SIM $\alpha = 0.5$	**1**	**1**	**1**	**1**	**1**	**0.995**	**0.995**	**1**	**1**	**0.998**	0.942
SIM Best	**1**	**1**	**1**	**1**	**1**	**0.995**	**0.995**	**1**	**1**	**0.998**	**0.956**

Table 8.3: NMI scores of some similarity measure techniques for some real datasets.

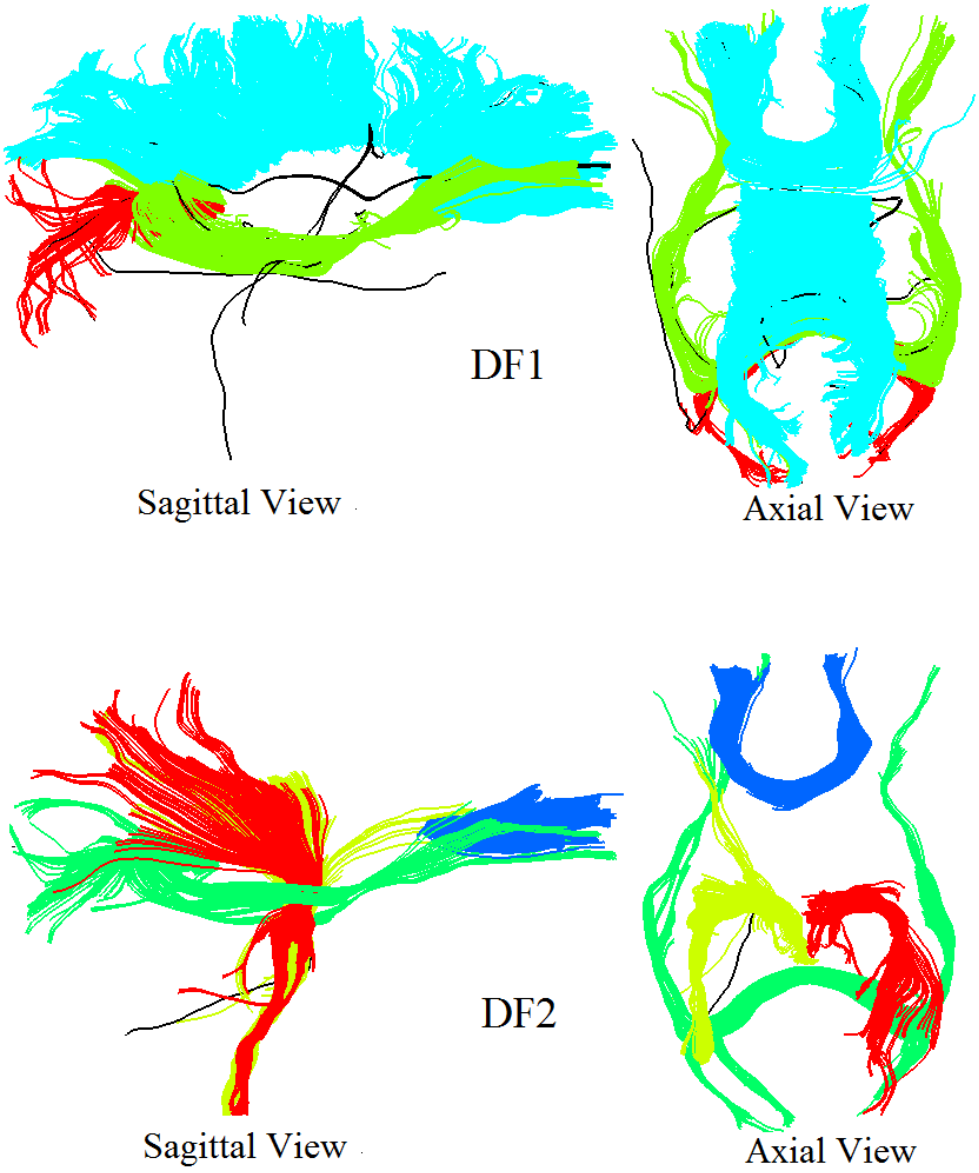

Figure 8.13: All datasets are well grouped according to our experts.

Figure 8.14 shows the relationship between α and NMI scores as well as their *eps* ranges (with the threshold of 0.9) for the five datasets from DS0 to DS4. As we see, each data set depends on α in slightly different ways. But all acquire good and stable performances when $\alpha \geq 0.3$. Besides, the *eps* ranges in all datasets increase with α (usually best at $\alpha = 0.8$ or 0.9), which means that the larger the value of α is, the more robust our algorithm is.

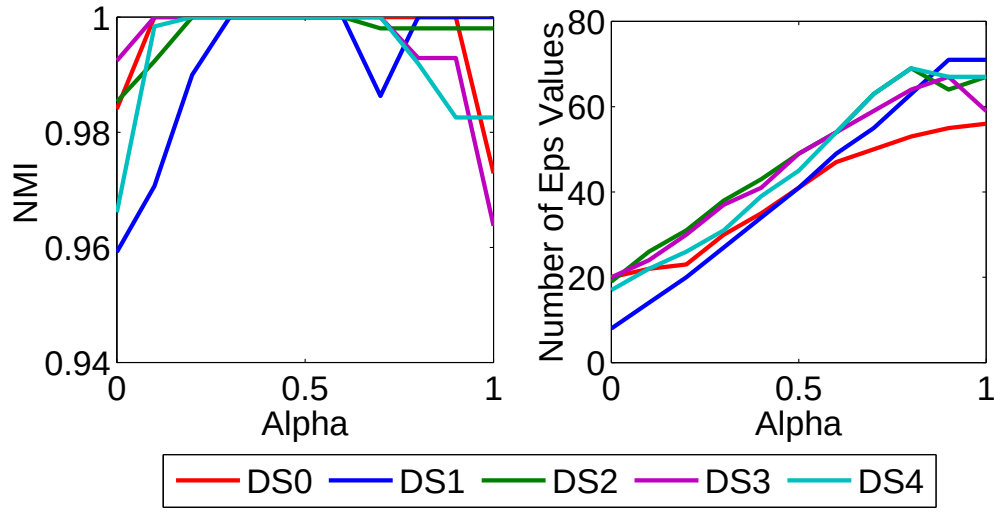

Figure 8.14: The relationship between α and the clustering scores NIM as well as its *eps* ranges on five real datasets.

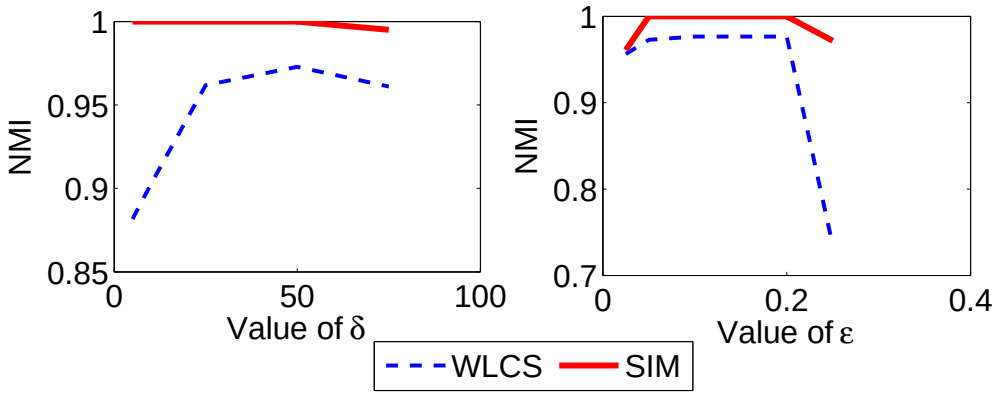

Figure 8.15: The relationships between parameter δ, ϵ and NMI score on data set DS0. SIM is more robust to the choices of parameters than WLCS.

8.4.4 Efficiency of the Unified Similarity Measure

The combination of the shape and the connection similarity enhances not only the effectiveness but also the efficiency of algorithm. When comparing two fibers, the connection similarity measure serves as the first level lower bounding with only $O(1)$ time complexity. After the first level has failed, the second level, the lower bounding distance of WLCS, is then examined with $O(n)$ time complexity. And the final step is WLCS itself. By the use of this multi-levels lower bounding distance, the running time of algorithm speeds up significantly.

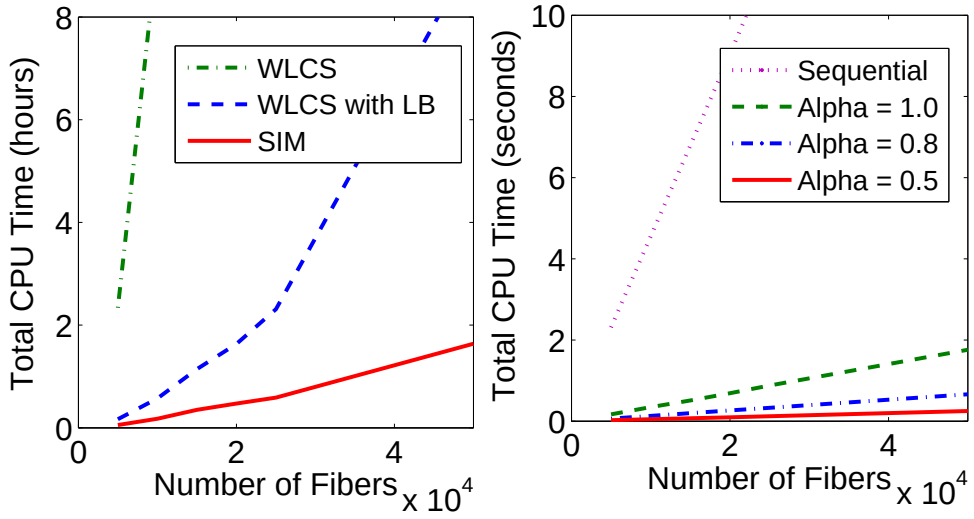

Figure 8.16: The total CPU times for (a) fiber segmentation and (b) *eps*-range query on real datasets with 5000 to 50000 fibers.

Figure 8.16 (b) shows the running time of *eps*-range queries of SIM with different values of α. As we can see, the smaller the value of α is, the faster the query time is. And so is the segmentation process. Figure 8.16 (a) shows the running times of WLCS on some real datasets. The parameters are $\alpha = 0.5$, $\delta = 50$ and $\epsilon = 0.05$. Since other techniques have $O(n^2)$ complexity like WLCS, we do not show them for clarity. As we can see, the use of the lower bounding distance significantly improves the performance of WLCS, especially for large datasets. With the multi-level lower bounding distance, the running times are further decreased. For example, with 5000 fibers, WLCS requires 8416 seconds. WLCS with lower bounding distance requires about 607 seconds. And SIM requires only 196 seconds. Compared with the other techniques, for the data set with 25000 fibers,

SIM requires 30 minutes to finish, while DTW [238] requires 10 hours and MCP requires 3 days. SIM is faster than the others up to about 150 times. For a massive data set with 250000 fibers, SIM finishes within a day.

8.5 Discussions

Brun et al. [50] consider two fibers similar if their start and end points are close together. However, this assumption is not always reasonable, since fibers in the same bundle may start and end in different regions [55]. Also, the shape similarity is totally ignored.

Most successful techniques use point-to-point correspondences to measure similarity. Zhang et al. [292] used the average of distances from points in the shorter fiber to their closest points in the longer one if they are larger than a predefined threshold. The choice of the threshold may be a drawback of this approach. Ding et al. [75] defined similarity by using the mean Euclidean distance and ratio between corresponding segments of fibers. One of the major drawbacks of this technique is that there is no clue to find those segments. Corouge et al. [64] introduced the three widely used similarity functions: closest point distance, mean of closest point distance (MCP) and Hausdorff distance (HDD) which measure fiber similarity by using distances between pairs of points of two fibers. Shao et al. [238] used Dynamic Time Warping (DTW) to measure shape similarity between fibers. All these techniques are sensitive to noise which may occur in fibers. Their distance mechanism is not strong enough to tell us whether two fibers have similar shape or they are separated by a small distance. The contribution of start and end points of fibers is also ignored although it plays an important role in the segmentation [55]. Moreover, they have $O(n^2)$ time complexity which is obviously undesirable especially for very large fiber datasets.

In data mining community, there exist many similarity measure techniques, for example, HMM-based distance [195], Longest Common Subsequence (LCS) [261], Edit Distance on Real sequence (EDR) [57], Sequence Weighted Alignment model (Swale) [74], etc. Although the most promising techniques like LCS and EDR are well-studied in the data mining fields, they are still unknown in the field of neuroscience. Therefore, understanding their performances on fiber context is essential for further researches on more efficient and effective methods for segmenting fibers. And our paper provides such a comprehensive study for this problem.

The envelope scheme proposed in this chapter provides an effective view about

the shape similarity measure of fibers. Our new techniques based on this scheme like LCS, EDR and WLCS also have a point-to-point mechanism like MCP, HDD and DTW. However, while these distance-based techniques can only detect whether two fibers are separated by a small distance or not, the threshold-based techniques can provide us more information, for example, the shape similarity between fibers, etc. As a result, our new envelope-based techniques acquire better performances and robustness than the existing distance-based techniques on fiber datasets. By allowing one-to-many matching mechanism, WLCS allows more efficient matching of coarse and different length fibers than other techniques. WLCS can be seen as a combination of DTW and LCS. But it is more suitable for the nature of fiber tracts than DTW and LCS.

The combination of shape and connection similarity provides a novel and robust similarity model for the white matter fiber tracts. The connection similarity serves as a lower bounding distance, which decreases the running time of fiber segmentation, as well as a supplement for the shape similarity measure. As a result, SIM is more robust to the choices of parameters and acquires better performance than the sole use of the shape similarity. In case the shape similarity is solely required in cross-subject comparison, WLCS still performs better than the others. Moreover, the combination of the shape and the connection similarity provides a flexible way for experts to customize their notions of fiber similarity. Depending on their opinions and their purposes, the experts can decide which is more important: the shape or the connection similarity by setting a suitable value for α. Thus, they may have diversified views about the white matter structures.

In Demiralp et al. [73], the authors assigned for each pair of points a weight based on their positions. This scheme is totally different with our SIM model which measure the shape similarity by weighting two important factors: structure and connectivity similarity of fibers. WLCS, LCS and EDR can also be improved by using weighted model of Demiralp et al. [73]. Also, in [170] the authors used a weighted scheme which looks similar to our scheme. However, this scheme has a totally different meaning. It focus on the shape similar only.

There exists some techniques which use some techniques to approximate fibers such as [142]. However, these techniques are out of scope of our study.

Moreover, our proposed lower bounding distance allows us to use the Minimum Bounding Envelope technique proposed in [132, 261] to index the time series. When it is applied to fiber segmentation, it could significantly improving the performance of algorithm by reducing the time needed for range query [83]. This

issue however is also out of scope of this chapter.

8.6 Conclusions

In this chapter, we propose a novel and robust similarity model (SIM) for fiber segmentation based on the combination of the shape and the connection similarity measure. By using the connection similarity as a supplement, this combination not only significantly improves quality but also speeds up the segmentation process.

Based on the new view about shape similarity of white matter tracts, we propose a new technique called WLCS which can efficiently and effectively capture the shape similarity between fibers. Also, various existing similarity models for trajectory like LCS, EDR are also adapted to measure the shape similarity of fibers. Our experiments have shown that WLCS outperforms other techniques on real fiber datasets. Also, threshold-based techniques like LCS, EDR acquire better and stable performances than distance-based techniques like HDD, MCP and DTW. A lower bounding distance is also proposed for WLCS to speed up the comparison, thus significantly decreasing the computation time.

Our future works aim at the development of an efficient approximation techniques for fiber similarity measure to further speed up the segmentation as well as the applications of WLCS on different fields such as biomedicine and spatio-temporal data mining.

Chapter 9

Advantage Fiber Clustering Techniques

Most existing fiber segmentation techniques suffer from high runtime and do not allow interaction with experts during their executions. To tackle with these limitations, we present a novel approach for fiber segmentation following an anytime clustering scheme. The general idea is to allow the algorithm quickly produces an approximation result and the continuously refines it until terminated. In this Chapter, the anytime density-based clustering algorithm A-DBSCAN and A-DBSCAN-XS are employed for segmenting fibers using Dynamic Time Warping as a represented technique for fiber similarity measure. Experiments on real fiber datasets are conducted to demonstrate the performance and characteristic of the proposed approaches.

Publications. Parts of the material presented in this Chapter have been published in [181]. The detailed information are described as follows:

- Son T. Mai, Xiao He, Jing Feng, Claudia Plant and Christian Böhm. Anytime Density-based Clustering of Complex Data. Knowledge and Information System (KAIS), 2014. (accepted for publication).

  In this work, S.T.M. contributed to the theory, implementation and experiment of the algorithm. C.B. gave out an idea for the experiments. X.H. and J.F. participated in some experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

9.1 Introduction

Exploring and analyzing fiber tracts are non-trivial problems due to the complexity of the white matter structure and huge amounts of fiber tracts (usually from 10^3 to 10^6 fibers) [55]. Therefore, many fiber clustering techniques [50, 75, 64, 252, 59, 178, 238] have been used to automatically group similar fibers into anatomical bundles. They help to reduce the complexity of data, improve the visualization and allow robust quantification and comparison between subjects to find out abnormalities or unusual features in the brains.

Although they are useful, most fiber clustering algorithms still suffer from several problems. First, they have very high running times which are caused by the high time complexity of the fiber similarity measures and clustering algorithms (usually quadratic time complexity). For example, the density-based fiber clustering algorithm [35] with the well-known Mean of Closest Point (MCP) distance [64] as the fiber similarity measure requires about 10 hours to group 10000 fibers of average length 75 points per fiber on 2Ghz Workstation with 4GB RAM. Since fiber datasets are usually large, this limitation is truly undesirable. Second, interactive exploration of DTI fibers has been proved to be a useful approach for brain connectivity analysis [59, 240, 91]. However, many existing fiber clustering algorithms such as [50, 64, 178] only work in a *batch* scheme. They do not allow interaction with experts during their runtime. Third, providing multiple results for experts to examine would be more useful due to complex structures of fiber tracts [269]. However, most existing techniques only produce one clustering result.

Since experts usually expect fast response times and interaction with program during the clustering process of the white matter fiber tracts, the anytime clustering scheme would be very useful. The algorithm quickly produces an approximate result which is continuously refined during the further runs. During its runtime, experts can examine the results, suspend, resume or stop the algorithm anytime whenever they satisfy with the existing results to save computation cost, or continue the algorithm to look for better results. On the other aspect, experts can have different views about the fiber structure and they can choose the best results according to their opinions from many results produced by the algorithm. To the best of our knowledge, this anytime scheme has never been applied for fiber clustering before.

Contributions. Our contributions are summarized as follows:

1. We proposed advantage techniques for fiber clustering using anytime density-

based clustering algorithm A-DBSCAN [184] and A-DBSCAN-XS [185].

2. Extensive experiments on real datasets are conducted to demonstrate the performance of our algorithms.

The rest of this Chapter is organized as follows. In Section 9.2, we propose a similarity measure for fiber using DTW and construct a sequence of LB distances for it. Experiments are displayed in Section 9.3. Section 9.4 summarizes our work.

9.2 Fiber Similarity Measure

After tractography, fibers are represented as a set S of streamlines in 3D with different lengths. Recent researches have introduced many similarity models for fibers such as Hausdoff distance [64], Mean of Closest Point distance [64] and Dynamic Time Warping [237]. Although these similarity models are efficient and widely used, they have quadratic time complexity which is a bottle neck of the fiber clustering algorithms. Therefore, anytime clustering model becomes a useful approach. In this work, we adopt Dynamic Time Warping (DTW) as the main similarity model for fibers due to its simplicity and ubiquitousness in many fields [132]. However, we note that our algorithm can be used with all mentioned techniques above.

Given two fibers $A = (a_1, \cdots, a_N)$ and $B = (b_1, \cdots, b_M)$ which contain N and M points in 3D. Let A_i be the first i points of A. We need to find a warping path $W = (w_1, \cdots, w_K)$ $(min(M, N) \leq K \leq M + N - 1)$ which contains pairs of indices of A and B $(w_k = (i, j))$ so that $\sum_{k=1}^{K} f(a_i, b_j)$ is minimum. We define distance function f between two 3D points as $f(a_i, b_j) = \sum_{d=1}^{3} |a_{id} - b_{jd}|$.

DTW can be calculated by using dynamic programming approach [132] to fill the cost matrix $D_{N \times M}$ with $D(i, j) = f(a_i, b_j) + min(D(i, j-1), D(i-1, j), D(i-1, j-1)))$. We have $DTW(A, B) = D(N, M)$.

To reduce the effect of different lengths of fibers, we define the similarity of two fibers A and B as follows:

$$Sim(A, B) = \frac{DTW(A, B)}{N + M - 1}$$

Note that this formula is slightly different with [237] which uses the length of warping path between two fibers as the denominator.

Lower bounding Fiber Similarity. In order to construct a sequence of lower bounding functions, there exist many lower bounding techniques for DTW [220] which can be employed such as LB_Keogh [132], LB_Zhu [301], LB_Yi [287, 132] and LB_Kim [132, 138]. Sakurai et al. [227] propose a technique called LB_Sakurai which divides 1D time-series into smaller segments and constructs the lower bounding distance for DTW based on the upper and lower values of these segments. By decreasing the length of each segment to 1, the lower bounding distance increases towards the true DTW distance. This property is well suited with our philosophy. Therefore, we extend the LB_Sakurai to use with 3D fiber trajectories.

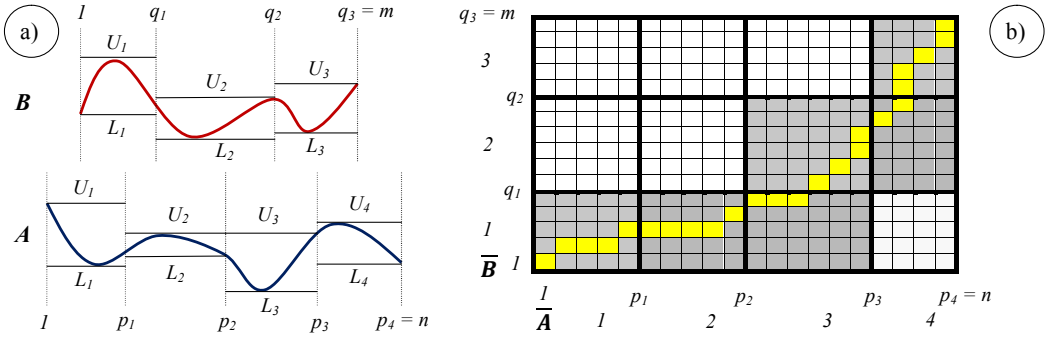

Figure 9.1: Example of LB_Sakurai: (a) A and B is divided into 4 and 3 segments respectively; (b) the cost matrix and the warping path for $DTW(A, B)$ (yellow cells) and $DTW(\overline{A}, \overline{B})$ (bold gray cells).

Assuming that, A and B are divided into n and m non-overlapping segments at positions $P = (p_1, \cdots, p_n = n)$ and $Q = (q_1, \cdots, q_m = m)$ respectively. For each segment i, we calculate its upper value (U_i) and lower value (L_i) at each dimension. The lower bounding distance of A and B is calculated by DTW of segmented representations of A (denoted as $\overline{A}$) and B (denoted as $\overline{B}$) respectively. Figure 9.1 shows an example of LB_Sakurai in 1D. Since the size of the cost matrix of $\overline{A}$ and $\overline{B}$ is much smaller than the cost matrix of A and B, the running time of LB_Sakurai is much faster than the original DTW.

To calculate LB_Sakurai, we define the distance between two segments p and q with length l_p and l_q as follows:

$$f_s(p,q) = min(l_p, l_q) \cdot \sum_{d=1}^{3} \begin{cases} |U_{pd} - L_{qd}| & (U_{pd} \geq L_{qd}) \\ |L_{qd} - U_{pd}| & (L_{qd} \geq U_{pd}) \\ 0 & (otherwise) \end{cases}$$

Figure 9.2 illustrates the distance between two segments p and q in 1D (a) and in 2D (b).

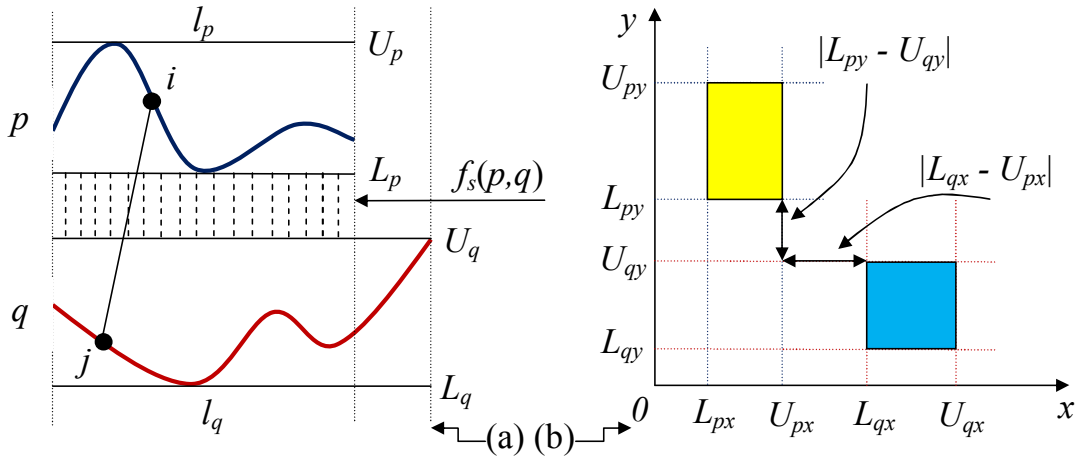

Figure 9.2: The distance between two segments p and q in 1D (a) and in 2D (b).

Lemma 11 *Given two points p_i in p and q_j in q, we have:*

$$\sum_{d=1}^{3} |p_{id} - q_{jd}| \geq \sum_{d=1}^{3} \begin{cases} |U_{pd} - L_{qd}| & (U_{pd} \geq L_{qd}) \\ |L_{qd} - U_{pd}| & (L_{qd} \geq U_{pd}) \\ 0 & (otherwise) \end{cases}$$

Proof 11 *The expression is true at each dimension (see Figure 9.2 (a) for an example). So is the sum.*

Lemma 12 *Given two fibers A and B and their segmented representation $\overline{A}$ and $\overline{B}$ respectively. We have $DTW(\overline{A}, \overline{B}) \leq DTW(A, B)$.*

Proof 12 *Following Lemma 11, the proof of Lemma 12 is similar to the lower bounding proof in [227].*

Following Lemma 12, the LB_Sakurai of $Sim(A, B)$ is defined as $DTW(\overline{A}, \overline{B})/(N+M-1)$.

In this work, we divide each fiber into equal segments of length l (except the last segment). Therefore, the running time of LB_Sakurai is around $O(n^2/l^2)$ (assuming that the length of fibers are n), which is l^2 times faster than original DTW.

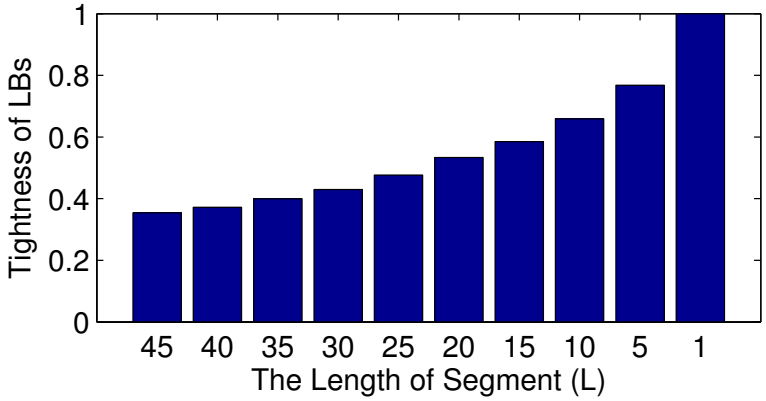

Figure 9.3: The average of tightness of LB_Sakurai on real fiber datasets.

To construct a sequence of lower bounding functions to use with our algorithms A-DBSCAN and A-DBSCAN-XS, all we need to do is decreasing the value of l to 1 (in case $l = 1$ it is clear that $DTW(\overline{A}, \overline{B}) = DTW(A, B)$). Figure 9.3 shows the averaged tightness of lower bounding [132] which is measured by the ratio between $DTW(\overline{A}, \overline{B})$ and $DTW(A, B)$ w.r.t. different values of l on all our real fiber datasets. The smaller value of l, the higher the tightness of LBs.

9.3 Empirical Evaluation

Datasets. We evaluate the performance of our algorithm on 6 real labeled datasets DS1 to DS6 which are randomly extracted from Pittsburgh Brain Competition (PBC) dataset (http://pbc.lrdc.pitt.edu/?q=home). These datasets contain 500 to 1500 fibers belonging to 5 to 8 famous fiber bundles namely Arcuate, Cingulum, Fornix, Inferior Occipitofrontal Fasciculus, Superior Longitudinal Fasciculus, Forceps Major and Corticospinal. For each dataset, 5 fibers from other bundles are also added as noise.

Parameters Setting. A-DBSCAN and A-DBSCAN-XS always run on 8 different levels with the length of segment $l = \{35, 30, 25, 20, 15, 10, 5, 1\}$ respectively ($l = 1$ means the original DTW distance). Note that, experts can construct different sequences of lower bounding functions based on their need: large l means worse results and shorter running times and vice versa. We fix the parameter $\mu = 5$ as suggested by various researchers [83].

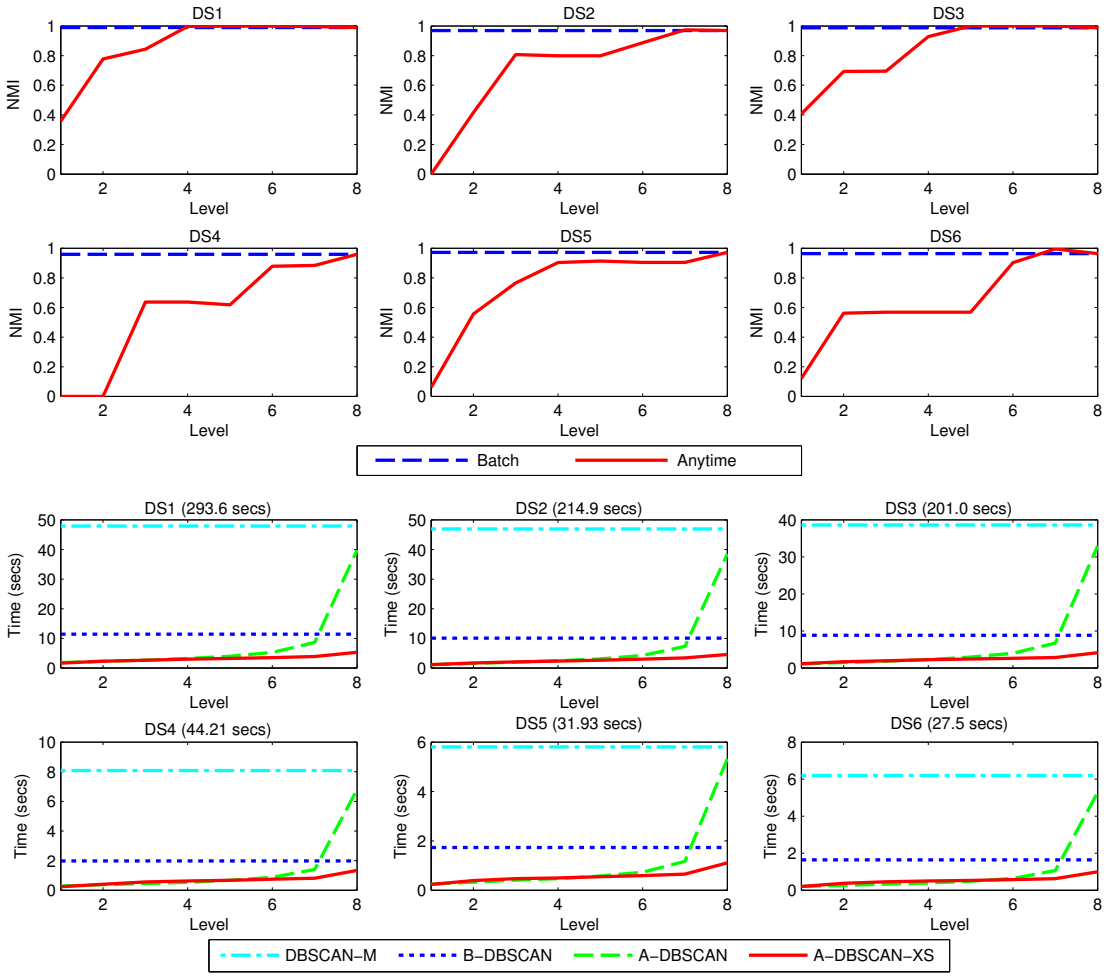

Figure 9.4: The comparison of A-DBSCAN-XS and the others on 6 real datasets DS1 to DS6. Good results are acquired at very early stages which require very small running times.

Performance comparison. Figure 9.4 shows the comparison of A-DBSCAN-XS, A-DBSCAN, M-DBSCAN, B-DBSCAN and DBSCAN on 6 real datasets DS1 ($\epsilon = 0.07$), DS2 ($\epsilon = 0.1$), DS3 ($\epsilon = 0.08$), DS4 ($\epsilon = 0.12$), DS5 ($\epsilon = 0.055$) and DS6 ($\epsilon = 0.1$) (the runtime of DBSCAN is written beside the name of each dataset). The NMI scores of the anytime algorithm (A-DBSCAN and A-DBSCAN-XS) come closer to NMI scores of the batch algorithm (DBSCAN, M-DBSCAN and B-DBSCAN) at each level. On all datasets, the NMI scores of A-DBSCAN-XS become very close to the results of DBSCAN after level 3 or 4. Therefore, experts can stop the algorithm as soon as it reaches level 3 or 4 to save computation time. For DS1, A-DBSCAN-XS acquires the NMI score of 0.842 at

level 3 which is a good score compared with the final score 0.988. The cumulative running time of A-DBSCAN-XS at level 3 is only 2.62 seconds which is 112 times faster than DBSCAN (293.6 seconds), 18 times faster than M-DBSCAN (47.9 seconds) and 5 times faster than B-DBSCAN (11.4 seconds). When it comes to the end, A-DBSCAN-XS requires only 5.3 seconds which is 55 times faster than DBSCAN, 10 times faster than M-DBSCAN, 2 times faster than B-DBSCAN, and 8 times faster than A-DBSCAN. For the dataset DS4, DS5 and DS6, the cumulative runtime of A-DBSCAN-XS at some mid-levels is slightly slower than A-DBSCAN since the overhead of the Xseedlist is much higher than its benefits. For DS1, DS2, DS3 and DS6, the best results are found at some mid-levels instead of the last one. It is interesting since we can have results which might never be acquired with the batch clustering algorithm.

Figure 9.5 (a) shows the numbers of calls to different LBs for all techniques on dataset DS6 as an example. The result clearly proves the performance acceleration of A-DBSCAN-XS compared with the others. Figure 9.5 (b) shows the numbers of calls to different LBs between using the extended Xseedlist with the sorting function ϕ and the original one. The extended Xseedlist helps to reduce the numbers of calls to LBs significantly.

To summarize, good results are found at very early levels by A-DBSCAN-XS. Thus it helps to speed up A-DBSCAN-XS by orders of magnitudes compared with other techniques. Since DTW is much expensive than ED, the performance acceleration of A-DBSCAN and A-DBSCAN-XS on DTW is much higher than on ED as described in Section 5.6. Even when it runs to the end, A-DBSCAN-XS is about 50 times faster than DBSCAN, 2 times faster than B-DBSCAN, and 8 times faster than A-DBSCAN.

More experiments. Figure 9.6 shows the clustering results of A-DBSCAN-XS at each level for DS5 which contains 5 bundles. A-DBSCAN-XS detects only 1 bundle at level 1, 2 bundles at level 2, 3 bundles at level 3. From level 4 to 7, it discovers 4 bundles with only some minor changes. The result at level 8 is closest to the ground truth.

One of the advantages of our algorithm is that experts can be able to choose whatever they think reasonable from the bunch of results produced by our algorithm. For example, since many experts consider the 2 bundles Arcuate and Superior as similar (the yellow and red bundles at level 8), they prefer the results at level 3 to 7 while other experts prefer the results at level 8. This is extremely useful in neuroscience since the goodness of the classification of fibers sometime

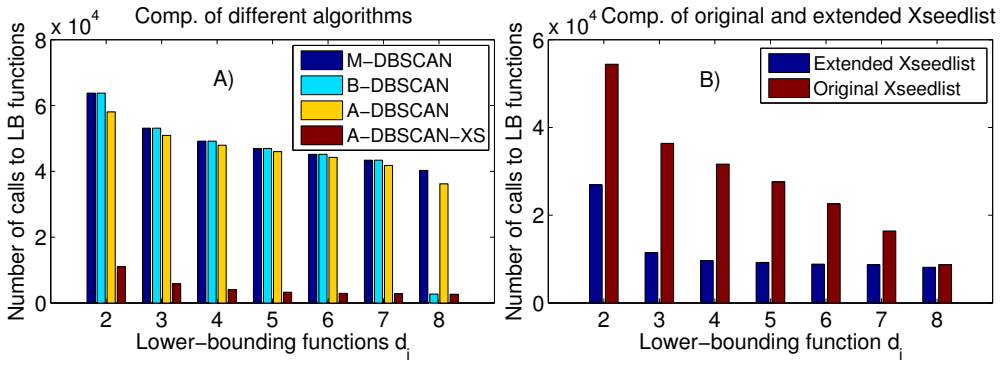

Figure 9.5: The numbers of call to LBs (a) for all algorithms and (b) for the original and extended Xseedlist on the dataset DS6

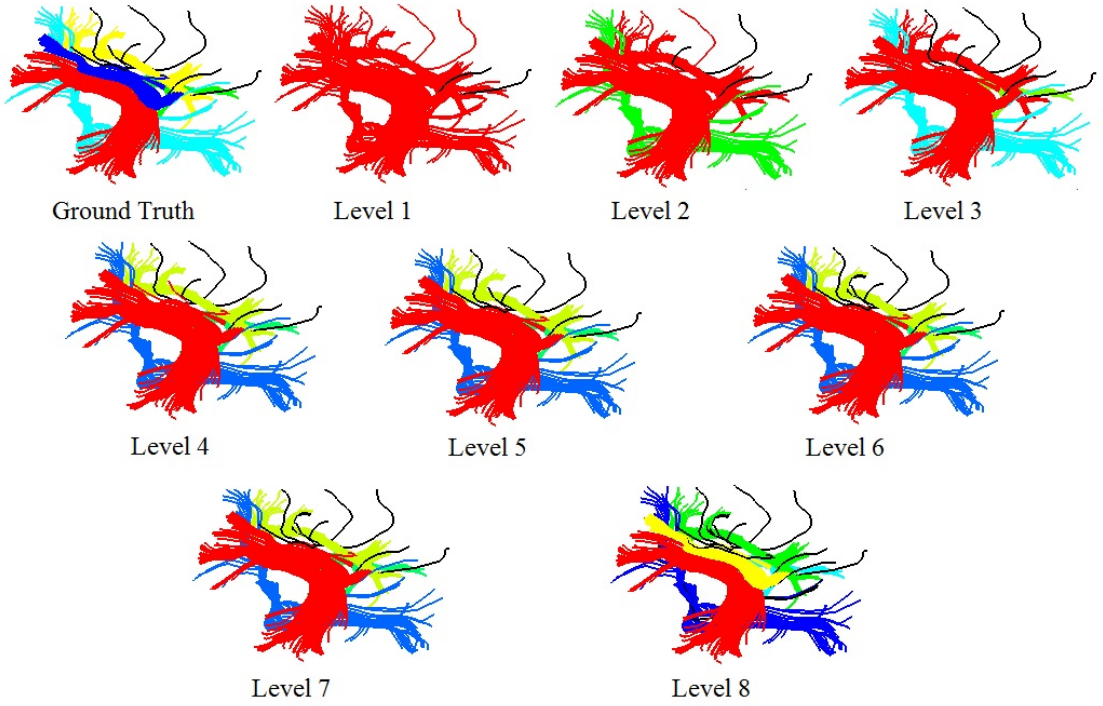

Figure 9.6: The clustering results for dataset DS5 at each level (outliers are always draw in black). All the results from level 4 to 8 are well-grouped.

depends on different opinions of experts. Experts can stop the algorithm at level 4, thus acquiring up to 70 times speed up compared with DBSCAN.

Corpus Callosum. Figure 9.7 shows the clustering results of A-DBSCAN-XS for the Corpus Callosum ($\mu = 5$, $\epsilon = 0.03$), the biggest and the most important bundles which connects the left and right cerebral hemispheres in human brain. Although we do not have the ground truth to compare, the results are well confirmed by our experts. Moreover, the results show the Corpus Callosum with different resolution from the coarser to finer one.

Other results. Figure 9.8 shows the clustering results for the dataset DS1 at each level. The clustering results at level 4 to 8 are well-grouped with 8 bundles are detected with only some minor errors. The best clustering results are at level 6 and 7.

Figure 9.9 shows the clustering results for the dataset DR1 at each level. Although we do not have a gold standard to compare. The results are well-grouped according to our experts.

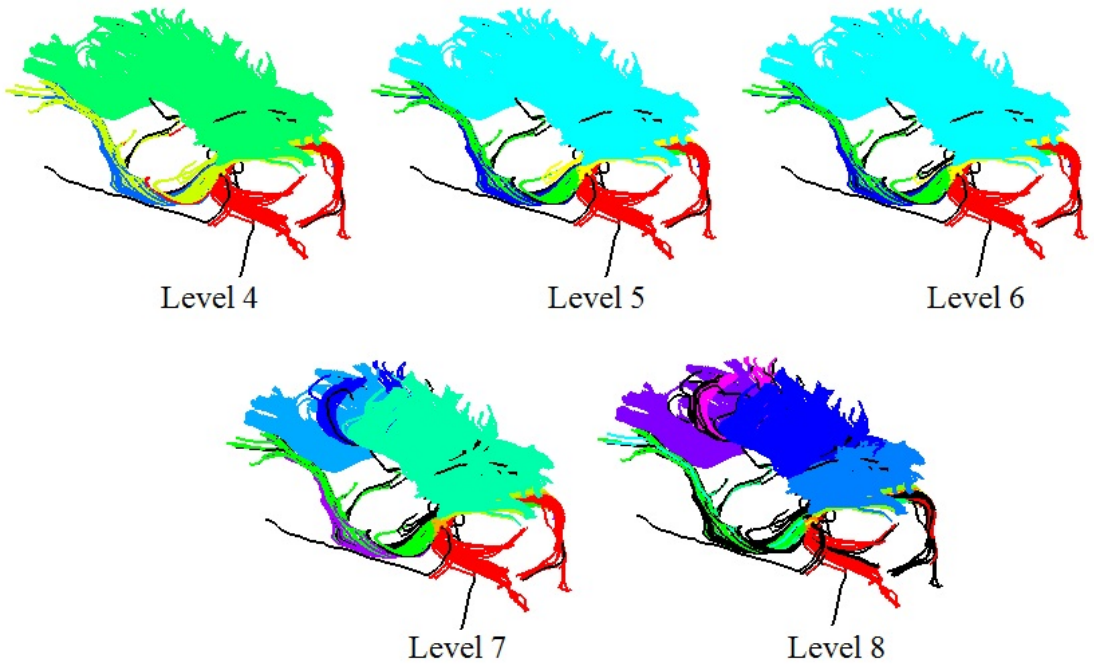

Figure 9.7: The clustering results for the Corpus Callosum dataset at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well confirmed according to our experts.

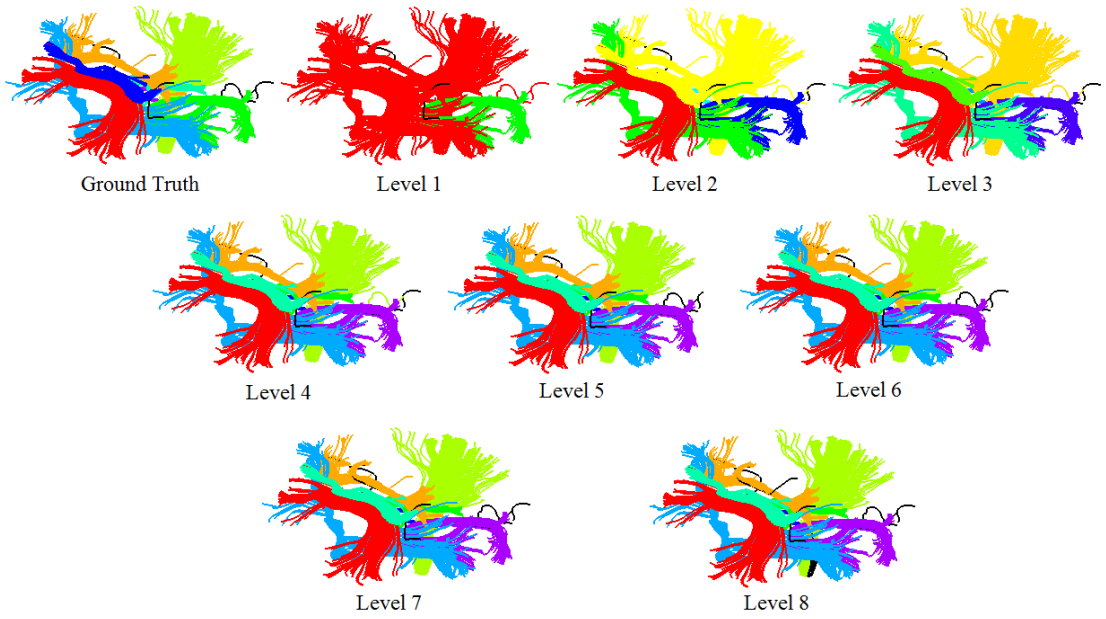

Figure 9.8: The clustering results for the dataset DS1 at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well-grouped with 8 bundles are detected with only some minor errors.

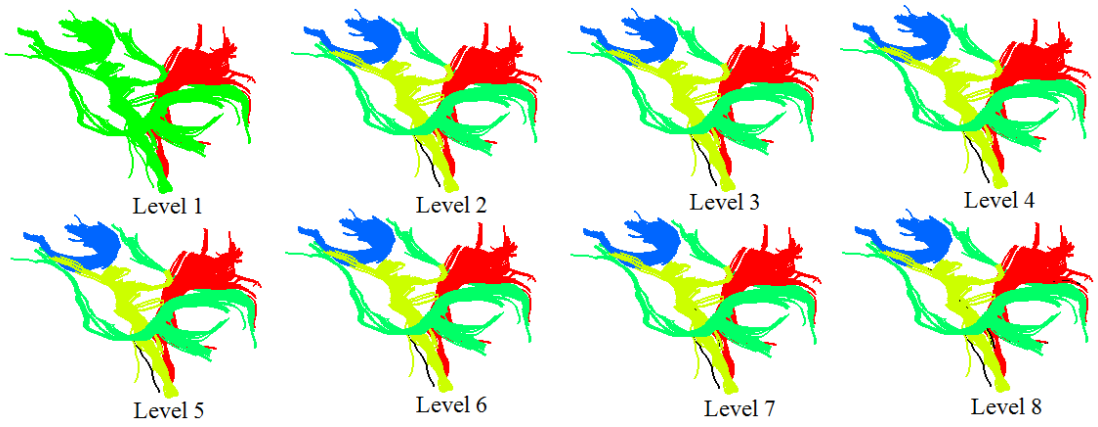

Figure 9.9: The clustering results for the dataset DS1 at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well-grouped with 8 bundles are detected with only some minor errors.

Scalability. Figure 9.10 shows the scalability of different algorithms for very large fiber datasets which contain 2000 to 8000 fibers with $D = \{8, 6, 4, 2, 1\}$, $\mu = 5$ and $\epsilon = 0.04$. A-DBSCAN acquires better performance than M-DBSCAN. Among all the algorithms, A-DBSCAN-XS is the best one. For the large dataset with 8000 fibers, it requires only 1467.3 seconds to finish and 754.4 seconds to acquire a good

result, while DBSCAN requires more than 5 hours and still does not finish. Note that, we report the final cumulative runtimes of A-DBSCAN and A-DBSCAN-XS in this experiment. If A-DBSCAN and A-DBSCAN-XS are terminated at some middle levels, the acceleration factor will be much larger.

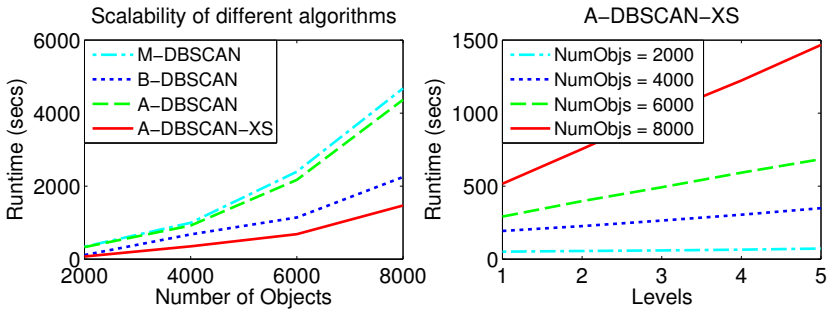

Figure 9.10: Scalability of different algorithms for very large fiber datasets.

9.4 Conclusions

In this Chapter, we propose a novel approach for fiber clustering using anytime clustering scheme. In contrast to the fiber clustering techniques, our algorithm quickly produces a segmentation result and then continuously refines it during the runtime. Moreover, it allows the interaction with experts during its execution. Experiments confirm that our proposed fiber clustering techniques can help to speed up the segmentation process significantly. Since it produces multiple segmentation results during the runtime, experts thus could have different views about fiber structures.

To summarize, our approach provides a unique scheme to tackle with the fiber segmentation problem.

Part IV

Summary

Chapter 10

Summary

In this thesis, we focused on developing anytime and active techniques for the density-based clustering algorithm DBSCAN and their applications in neuroscience. In this Chapter, we sum up all our contributions in this thesis and discuss some future researches.

10.1 Conclusions

Nowadays, the developments of model technologies have opened the possibility of generating, storing and collecting a large amount of complex data acquired from many different fields, e.g. medicine, environment. As a consequence, the need of new data mining technologies to deal with the new challenges for complex data has been arisen during the last decades. In the first part of this thesis, we aimed at the task of data clustering, one of the most common techniques in advanced data analysis, for complex data. In particular, we focused on the density-based clustering algorithm DBSCAN, a fundamental data clustering technique proposed in the literature.

The core idea of the density-based clustering algorithm DBSCAN is that each object within a cluster must have a certain amount of other objects inside its neighborhood. Compared with other clustering algorithms, DBSCAN has many attractive benefits, e.g. it can detect clusters with arbitrary shape and is robust to outlier. Thus, DBSCAN has attracted a lot of research interest in many fields during the last decades.

However, like many other clustering algorithms, DBSCAN suffers from poor performance problem when facing with expensive distance measures for com-

plex data. To tackle this problem, we proposed a new approach for DBSCAN, called Anytime Density-based Clustering (A-DBSCAN), that works in an anytime scheme. In contrast to the original batch scheme of DBSCAN, the algorithm Any-DBSCAN first produces a quick approximation of the final clustering result and then continuously refines the result during the further run. Experts can interrupt the algorithm, examine the results and stop the algorithm at any time whenever they satisfied with the result to save runtime or continuous the algorithm to acquire better results. Such kind of anytime scheme has been proved in the literature as a very useful technique when dealing with time consuming problems. Concretely, A-DBSCAN works in multiple levels w.r.t. a sequence of LB distances of the true expensive distance function. At each level, the density-based clustering algorithm is performed on different LB distance functions. And the clustering results at previous levels are used to refine the clustering result at current level. From the current level to the next level, the monotonicity property of density-based clusters is exploited to exclude redundance distance upgrades and to reduce cost of cluster update time. Hence the total final cumulative runtime of A-DBSCAN is usually better than that of DBSCAN. The speed up factor is much better if A-DBSCAN is interrupted at some middle levels.

We also introduced an extended version of A-DBSCAN called A-DBSCAN-XS which is more efficient and effective than DBSCAN when dealing with expensive distance measures. The algorithm A-DBSCAN-XS is built upon the data structure called the extended XSeedlist to further reduce the total number of distance calculations at each level. Here, the monotonicity of clustering results also plays an important role to improve the runtime in the similar way with A-DBSCAN. Beside the power of the extended XSeedlist in reducing the number of distance calculations, maintaining the XSeedlist is an expensive process due to its sorting scheme. Thus, A-DBSCAN-XS is more useful than A-DBSCAN when handling expensive distance measures.

Since DBSCAN relies on the cardinality of the neighborhood of objects, it requires all the distance among objects to perform. For complex data, these distances are usually expensive, time consuming or even unavailable to acquire due to financial problem, high time complexity, noisy and missing data, etc. Motivated by these potential difficulties of acquiring the distances among objects, we proposed another approach for DBSCAN, called Active Density-based Clustering (Act-DBSCAN). Given a budget limitation B, Act-DBSCAN is only allowed to use up to B pairwise distances to produce the same result as if it has the entire distance matrix at hand. The general idea of Act-DBSCAN is that it actively

selects the most meaningful pairwise objects to calculate the distances between them and tries to approximate as much as possible the desired clustering result with each distance calculation. This scheme provides an efficient way to reduce the total cost needed to perform the clustering. Thus it limits the potential weakness of DBSCAN when dealing with the distance sparseness problem of complex data.

In particular, Act-DBSCAN is initialized with a lower-bounding distance matrix of the true distance measure to guide the clustering process. At each iteration, some pairwise distances are actively selected and updated with the true distances until the budget limitation is reached. Act-DBSCAN contains an efficient probabilistic model and a scoring system called the Shared Core Object (SCO) score to evaluate the impact of the update of each pairwise LB similarity on the change of the intermediate cluster structure. Deriving from the monotonicity and reduction property of our clustering scheme and the SCO score, the two algorithms Splitting with SCO (SP-SCO) and Merging with SCO (MG-SCO) are proposed to provide two different and efficient ways to actively select and update pairwise similarities and cluster results. Extensive experiments on real datasets have demonstrated the performance of Act-DBSCCAN. It requires only a small fraction of all pairwise similarities to reach the clustering results of DBSCAN. Act-DBSCAN also outperforms other techniques such as active spectral clustering.

In the last part of this thesis, we focused on applications of density-based clustering algorithms in the field of neuroscience, in particular the problem of automatically segmenting the white matter structure in human brain via Diffusion Tensor Imaging technique. In order to segment the fibers into anatomical meaningful bundle, a similarity model between fibers is required beforehand. Therefore, we proposed a model to evaluate the similarity between two fibers as a combination of the structure and connectivity similarity of fiber tracts. Various distance measure techniques from other fields are adapted to calculate the structure similarity of fibers. Density-based clustering algorithm is used as the segmentation algorithm to evaluate the efficiency of our proposed model. Our similarity model not only is more efficient than other existing similarity measure techniques for fibers but also is more robust to noise. Moreover, it can help to speed up the clustering process due to its multi-level lower-bouding property. It also provides a flexible way to cope with the complex notion of fiber similarity.

For segmenting the fibers into bundle, we showed how anytime density-based clustering algorithms like A-DBSCAN and A-DBSCAN-XS can be used as novel solutions for the segmentation of massive fiber datasets and for providing unique features to assist experts during the fiber clustering process.

10.2 Future Works

Based on the results presented in this thesis, several promising and challenging directions of future works arise.

Though the use of LB distance to initialize the cluster structure in DBSCAN provides an efficient way to estimate the true cluster structure and to select meaningful pairwise distances to update, it limits the applicability of Act-DBSCAN in case the LB distance is not available. Thus, how to extend Act-DBSCAN to work with an approximate distance matrix or an empty distance matrix in the beginning is thus an interesting work. However, it is a non-trivial problem since there are less obvious available information to choose meaningful pairwise distance to update due to the lack of some important properties such as the monotonicity property of the cluster structure.

Building an interactive system for fiber clustering under the anytime and active density-based clustering scheme is another extension of our work. Obviously, such kind of system would be a very useful tool for analyzing fiber structure. One of the main challenges is how to capture the changes or constraints posed by experts during the clustering process instead of allowing them to interactively explore the results only. A promising direction is to allow expert to continuously pose instance level constraints during the clustering process.

There exists in the literature many other extensions of DBSCAN. Since these algorithms rely on the pairwise similarities among objects, they also suffer from the similarity sparseness problem described in Chapter 1. Therefore, developing anytime and active schemes for these algorithms to cope with the similarity sparseness problem remains open and interesting works.

Another promising direction is to parallelize the clustering process of A-DBSCAN, A-DBSCAN-XS and Act-DBSCAN to enhance the performance on large-scale complex datasets.

Bibliography

[1] E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[2] A. A. Abbasi and M. F. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, 2007.

[3] E. Achtert, C. Böhm, H.-P. Kriegel, and P. Kröger. Online Hierarchical Clustering in a Data Warehouse Environment. In *ICDM*, pages 10–17, 2005.

[4] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Finding Hierarchies of Subspace Clusters. In *PKDD*, pages 446–453, 2006.

[5] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. On exploring complex relationships of correlation clusters. In *SSDBM*, page 7, 2007.

[6] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Robust, complete, and efficient correlation clustering. In *SDM*, pages 413–418, 2007.

[7] E. Achtert, C. Böhm, and P. Kröger. DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking. In *PAKDD*, pages 119–128, 2006.

[8] E. Achtert, C. Böhm, P. Kröger, and A. Zimek. Mining Hierarchies of Correlation Clusters. In *SSDBM*, pages 119–128, 2006.

[9] C. C. Aggarwal and C. K. Reddy, editors. *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[10] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *SIGMOD Conference*, pages 94–105, 1998.

[11] A. Agresti and B. A. Coull. Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, 52(2):119–126, 1998.

[12] E. B. Ahmed, A. Nabli, and F. Gargouri. {SHACUN} Semi-supervised Hierarchical Active Clustering Based on Ranking Constraints. In *ICDM*, pages 194–208, 2012.

[13] S. Aine, P. P. Chakrabarti, and R. Kumar. AWA* - A Window Constrained Anytime Heuristic Search Algorithm. In *IJCAI*, pages 2250–2255, 2007.

[14] G. Andrade, G. S. Ramos, D. Madeira, R. S. Oliveira, R. Ferreira, and L. C. da Rocha. G-DBSCAN: A GPU Accelerated Algorithm for Density-based Clustering. In *ICCS*, pages 369–378, 2013.

[15] S. Angelopoulos and A. López-Ortiz. Interruptible Algorithms for Multi-Problem Solving. In *IJCAI*, pages 380–386, 2009.

[16] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *SIGMOD*, pages 49–60, 1999.

[17] B. Arai, G. Das, D. Gunopulos, and N. Koudas. Anytime measures for top-k algorithms. In *VLDB*, pages 914–925, 2007.

[18] D. Arlia and M. Coppola. Experiments in Parallel Clustering with DBSCAN. In *Euro-Par*, pages 326–331, 2001.

[19] I. Assent, P. Kranen, C. Baldauf, and T. Seidl. Detecting Outliers on Arbitrary Data Streams Using Anytime Approaches. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, StreamKDD '10, pages 10–15, New York, NY, USA, 2010. ACM.

[20] I. Assent, P. Kranen, C. Baldauf, and T. Seidl. AnyOut: Anytime Outlier Detection on Streaming Data. In *DASFAA (1)*, pages 228–242, 2012.

[21] F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[22] M.-F. Balcan and P. Gupta. Robust hierarchical clustering. In *COLT*, pages 282–294, 2010.

[23] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424, 2000.

[24] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. In Vivo Fiber Tractography using DT-MRI data. *Magn. Reson. Med*, 44:625–632, 2000.

[25] S. Basu, A. Banjeree, E. Mooney, A. Banerjee, and R. J. Mooney. Active Semi-Supervision for Pairwise Constrained Clustering. In *SDM*, pages 333–344, 2004.

[26] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.

[27] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The x-tree : An index structure for high-dimensional data. In *VLDB*, pages 28–39, 1996.

[28] S. Bereg, M. Kubica, T. Walen, and B. Zhu. RNA multiple structural alignment with longest common subsequences. *J. Comb. Optim.*, 13(2):179–188, 2007.

[29] P. Berkhin. A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data*, pages 25–71. 2006.

[30] E. Biçici and D. Yuret. Locally Scaled Density Based Clustering. In *ICANNGA (1)*, pages 739–748, 2007.

[31] D. L. Bihan, J.-F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabrait. Diffusion Tensor Imaging: Concepts and Applications. *Journal of Magnetic Resonance Imaging*, 13:534–546, 2001.

[32] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.*, 60(1):208–221, 2007.

[33] A. Biswas and D. W. Jacobs. Active image clustering: Seeking constraints from humans to complement algorithms. In *CVPR*, pages 2152–2159, 2012.

[34] C. Böhm, B. Braunmüller, M. M. Breunig, and H.-P. Kriegel. High Performance Clustering Based on the Similarity Join. In *CIKM*, pages 298–305, 2000.

[35] C. Böhm, J. Feng, X. He, S. T. Mai, C. Plant, and J. Shao. A Novel Similarity Measure for Fiber Clustering using Longest Common Subsequence. In *KDD Workshops*, pages 1–9, 2011.

[36] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density Connected Clustering with Local Subspace Preferences. In *ICDM*, pages 27–34, 2004.

[37] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing Clusters of Correlation Connected Objects. In *SIGMOD Conference*, pages 455–466, 2004.

[38] C. Böhm, R. Noll, C. Plant, and B. Wackersreuther. Density-based Clustering using Graphics Processors. In *CIKM*, pages 661–670, 2009.

[39] C. Böhm, R. Noll, C. Plant, B. Wackersreuther, and A. Zherdin. Data Mining Using Graphics Processing Units. *T. Large-Scale Data- and Knowledge-Centered Systems*, 1:63–90, 2009.

[40] C. Böhm and C. Plant. HISSCLU: a hierarchical density-based method for semi-supervised clustering. In *EDBT*, pages 440–451, 2008.

[41] B. Borah and D. K. Bhattacharyya. An Improved Sampling-Based DBSCAN for Large Spatial Databases. *ICISIP*, 2004.

[42] B. Borah and D. K. Bhattacharyya. DDSC : A Density Differentiated Spatial Clustering Technique. *JCP*, 3(2):72–79, 2008.

[43] J. F. Bowring, J. M. Rehg, and M. J. Harrold. Active learning for automatic classification of software behavior. In *ISSTA*, pages 195–205, 2004.

[44] S. Brecheisen, H. Kriegel, and M. PfeifIe. Parallel Density-Based Clustering of Complex Objects. In *PAKDD*, pages 179–188, 2006.

[45] S. Brecheisen, H. Kriegel, and M. Pfeifle. Efficient density-based clustering of complex objects. In *ICDM*, pages 43–50, 2004.

[46] S. Brecheisen, H.-P. Kriegel, and M. Pfeifle. Multi-step density-based clustering. *Knowl. Inf. Syst.*, 9(3):284–308, 2006.

[47] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering. In *SIGMOD Conference*, pages 79–90, 2001.

[48] M. M. Breunig, H.-P. Kriegel, and J. Sander. Fast Hierarchical Clustering Based on Compressed Data and OPTICS. In *PKDD*, pages 232–242, 2000.

[49] R. Brooks. Fast Image Alignment Using Anytime Algorithms. In *IJCAI*, pages 2078–2083, 2007.

[50] A. Brun, H.-J. Park, H. Knutsson, and C.-F. Westin. Coloring of DT-MRI Fiber Traces using Laplacian Eigenmaps. In *EUROCAST*, pages 564–572, 2003.

[51] J. M. Buhmann and T. Zöller. Active Learning for Hierarchical Pairwise Data Clustering. In *ICPR*, pages 2186–2189, 2000.

[52] R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *PAKDD (2)*, pages 160–172, 2013.

[53] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, pages 328–339, 2006.

[54] C. Cassisi, A. Ferro, R. Giugno, G. Pigola, and A. Pulvirenti. Enhancing density-based clustering: Parameter reduction and outlier detection. *Inf. Syst.*, 38(3):317–330, 2013.

[55] M. Catani, R. J. Howard, S. Pajevic, and D. K. Jones. Virtual in Vivo Interactive Dissection of White Matter Fasciculi in The Human Brain. *Neuroimage*, 17(1):77–94, 2002.

[56] M. E. Celebi, Y. A. Aslandogan, and P. R. Bergstresser. Mining biomedical images with density-based clustering. In *ITCC (1)*, pages 163–168, 2005.

[57] L. Chen and M. T. Özsu. Robust and Fast Similarity Search for Moving Object Trajectories. In *SIGMOD*, pages 491–502, 2005.

[58] S. X. Chen and J. S. Liu. Statistical applications of the Poisson-Binomial and conditional Bernoulli distributions. *Statistica Sinica*, 7(4), 1997.

[59] W. Chen, Z. Ding, S. Zhang, A. MacKay-Brandt, S. Correia, H. Qu, J. A. Crow, D. F. Tate, Z. Yan, and Q. Peng. A Novel Interface for Interactive Exploration of DTI Fibers. *IEEE Trans. Vis. Comput. Graphics*, 15(6):1433–1440, Nov. 2009.

[60] X. Chen, W. Liu, H. Qiu, and J.-H. Lai. Apscan: A parameter free algorithm for clustering. *Pattern Recognition Letters*, 32(7):973–986, 2011.

[61] Y. Chen and L. Tu. Density-based clustering for real-time stream data. In *KDD*, pages 133–142, 2007.

[62] C. K. Chui. *An Introduction to Wavelets*. Academic Press Professional, Inc., San Diego, CA, USA, 1992.

[63] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[64] I. Corouge, G. Gerig, and S. Gouttard. Towards a Shape Model of White Matter Fiber Bundles Using Diffusion Tensor MRI. In *ISBI*, pages 344–347, 2004.

[65] S. Correia, S. Y. Lee, T. Voorn, D. F. Tate, R. H. Paul, S. Zhang, S. P. Salloway, P. F. Malloy, and D. H. Laidlaw. Quantitative tractography metrics of white matter integrity in diffusion-tensor {MRI}. *NeuroImage*, 42(2):568 – 581, 2008.

[66] B.-R. Dai and I.-C. Lin. Efficient map/reduce-based dbscan algorithm with optimized data partition. In *IEEE CLOUD*, pages 59–66, 2012.

[67] S. Das, A. Abraham, and A. Konar. *Metaheuristic Clustering*, volume 178 of *Studies in Computational Intelligence*. Springer, 2009.

[68] S. Das, A. Chowdhury, and A. Abraham. A bacterial evolutionary algorithm for automatic data clustering. In *IEEE Congress on Evolutionary Computation*, pages 2403–2410, 2009.

[69] M. Dash, H. Liu, and X. Xu. '1 + 1 > 2': Merging distance and density based clustering. In *DASFAA*, pages 32–39, 2001.

[70] I. Davidson and S. Basu. A Survey of Clustering with Instance Level Constraints. In *KDD*, 2006.

[71] T. L. Dean and M. S. Boddy. An Analysis of Time-Dependent Planning. In *AAAI*, pages 49–54, 1988.

[72] D. DeCoste. Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry. In *ICML*, pages 99–106, 2002.

[73] C. Demiralp and D. H. Laidlaw. Similarity Coloring of DTI Fiber Tracts. In *DMFC Workshop*, 2009.

[74] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *PVLDB*, 1(2):1542–1552, 2008.

[75] Z. Ding, J. C. Gore, and A. W. Anderson. Classification and Quantification of Neuronal Fiber Pathways Using Diffusion Tensor MRI. *Mag. Res. in Med.*, 49:716–721, 2003.

[76] B. E. Dom. An Information-Theoretic External Cluster- Validity Measure. Technical Report RJ 10219, IBM, 2001.

[77] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan. A local-density based spatial clustering algorithm with noise, journal = Inf. Syst. 32(7):978–986, 2007.

[78] Y. El-Sonbaty, M. A. Ismail, and M. Farouk. An efficient density based clustering algorithm for large databases. In *ICTAI*, pages 673–677, 2004.

[79] B. Eriksson, G. Dasarathy, A. Singh, and R. D. Nowak. Active Clustering: Robust and Efficient Hierarchical Clustering using Adaptively Selected Similarities. *Journal of Machine Learning Research*, 15:260–268, 2011.

[80] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SDM*, pages 47–58, 2003.

[81] S. Esmeir and S. Markovitch. Anytime induction of decision trees: An iterative improvement approach. In *AAAI*, pages 348–355, 2006.

[82] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental Clustering for Mining in a Data Warehousing Environment. In *VLDB*, pages 323–333, 1998.

[83] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density- Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pages 226–231, 1996.

[84] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. Density-Connected Sets and their Application for Trend Detection in Spatial Databases. In *KDD*, pages 10–15, 1997.

[85] A. M. Fahim, G. Saake, A.-B. M. Salem, F. A. Torkey, and M. A. Ramadan. An Enhanced Density Based Spatial clustering of Applications with Noise. In *DMIN*, pages 517–523, 2009.

[86] T. Falkowski, A. Barth, and M. Spiliopoulou. DENGRAPH: A Density-based Community Detection Algorithm. In *Web Intelligence*, pages 112–115, 2007.

[87] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, 1996.

[88] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[89] Z. Francis, C. Villagrasa, and I. Clairand. Simulation of DNA damage clustering after proton irradiation using an adapted DBSCAN algorithm. *Computer Methods and Programs in Biomedicine*, 101(3):265–270, 2011.

[90] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml.

[91] P. A. Freeborough, N. C. Fox, and R. I. Kitney. Interactive algorithms for the segmentation and quantitation of 3-d {MRI} brain scans. *Computer Methods and Programs in Biomedicine*, 53(1):15 – 25, 1997.

[92] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[93] M. Gariel, A. N. Srivastava, and E. Feron. Trajectory Clustering and an Application to Airspace Monitoring. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1511–1524, 2011.

[94] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[95] M. Gong, L. Jiao, L. Wang, and L. Bo. Density-sensitive evolutionary clustering. In *PAKDD*, pages 507–514, 2007.

[96] M. Gorawski and R. Malczok. Calculation of Density-Based Clustering Parameters Supported with Distributed Processing. In *DaWaK*, pages 417–426, 2006.

[97] M. Gorawski and R. Malczok. Towards automatic eps calculation in density-based clustering. In *ADBIS*, pages 313–328, 2006.

[98] N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5):1834–1844, 2008.

[99] S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. *Inf. Syst.*, 26(1):35–58, 2001.

[100] S. Günnemann, B. Boden, and T. Seidl. Db-csc: A density-based approach for subspace clustering in graphs with feature vectors. In *ECML/PKDD (1)*, pages 565–580, 2011.

[101] S. Günnemann, B. Boden, and T. Seidl. Finding density-based subspace clusters in graphs with feature vectors. *Data Min. Knowl. Discov.*, 25(2):243–269, 2012.

[102] A. Guo and H. T. Siegelmann. Time-Warped Longest Common Subsequence Algorithm for Music Retrieval. In *ISMIR*, 2004.

[103] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[104] D. Habich, P. B. Volk, W. Lehner, R. Dittmann, and C. Utzny. Error-aware density-based clustering of imprecise measurement values. In *ICDM Workshops*, pages 471–476, 2007.

[105] R. Haenni. A Query-Driven Anytime Algorithm for Argumentative and Abductive Reasoning. In *Soft-Ware*, pages 114–127, 2002.

[106] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[107] E. A. Hansen and R. Zhou. Anytime Heuristic Search. *J. Artif. Intell. Res. (JAIR)*, 28:267–297, 2007.

[108] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl. Density-based projected clustering of data streams. In *SUM*, pages 311–324, 2012.

[109] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce. In *ICPADS*, pages 473–480, 2011.

[110] A. Hinneburg and H.-H. Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *IDA*, pages 70–80, 2007.

[111] A. Hinneburg and D. A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *KDD*, pages 58–65, 1998.

[112] A. Hinneburg and D. A. Keim. A general approach to clustering in large databases with noise. *Knowl. Inf. Syst.*, 5(4):387–415, 2003.

[113] A. Hinneburg, D. A. Keim, and M. Wawryniuk. Hd-eye: Visual mining of high-dimensional data. *IEEE Computer Graphics and Applications*, 19(5):22–31, 1999.

[114] T. Hofmann and J. M. Buhmann. Active Data Clustering. In *NIPS*, 1997.

[115] M. C. Horsch and D. L. Poole. An Anytime Algorithm for Decision Making under Uncertainty. In *UAI*, pages 246–255, 1998.

[116] E. Horvitz. Reasoning about Beliefs and Actions Under Computational Resource Constraints. In *UAI*, pages 301–324, 1987.

[117] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39(2):133–155, 2009.

[118] P.-H. Huang, W.-C. Chou, and W.-T. Lin. Using som and dbscan-based models for landslide hazard and spatial correlations analysis: A case study in central taiwan. In *Geoinformatics*, pages 1–5, 2012.

[119] G. Hulten and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *KDD*, pages 525–531, 2002.

[120] Y. Hwang and H.-K. Ahn. Convergent bounds on the euclidean distance. In *NIPS*, pages 388–396, 2011.

[121] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[122] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[123] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. DBDC: Density Based Distributed Clustering. In *EDBT*, pages 88–105, 2004.

[124] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable Density-Based Distributed Clustering. In *PKDD*, pages 231–244, 2004.

[125] D. Jiang, J. Pei, and A. Zhang. DHC: A Density-Based Hierarchical Clustering Method for Time Series Gene Expression Data. In *BIBE*, pages 393–400, 2003.

[126] K. Kailing, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Clustering Multi-represented Objects with Noise. In *PAKDD*, pages 394–403, 2004.

[127] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD*, pages 364–381, 2005.

[128] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[129] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *IEEE Computer*, 32(8):68–75, 1999.

[130] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *KDD*, pages 102–111, 2002.

[131] E. Keogh, Q. Zhu, B. Hu, H. Y., X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering. http://www.cs.ucr.edu/~eamonn/time_series_data/.

[132] E. J. Keogh. Exact Indexing of Dynamic Time Warping. In *VLDB*, pages 406–417, 2002.

[133] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.

[134] N. Kettle and A. King. An Anytime Algorithm for Generalized Symmetry Detection in ROBDDs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(4):764–777, 2008.

[135] F. Khani, M. J. Hosseini, A. A. Abin, and H. Beigy. An algorithm for discovering clusters of different densities or shapes in noisy data sets. In *SAC*, pages 144–149, 2013.

[136] H. Kil and W. Nam. Efficient anytime algorithm for large-scale QoS-aware web service composition. *IJWGS*, 9(1):82–106, 2013.

[137] M.-S. Kim and J. Han. A particle-and-density based evolutionary clustering method for dynamic networks. *PVLDB*, 2(1):622–633, 2009.

[138] S.-W. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pages 607–614, 2001.

[139] S. Kisilevich, F. Mansmann, and D. A. Keim. P-DBSCAN: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In *COM.Geo*, 2010.

[140] J. Klein, P. Bittihn, P. Ledochowitsch, H. K. Hahn, O. Konrad, J. Rexilius, and H. O. Peitgen. Grid-based spectral fiber clustering. In *SPIE*, volume 6509, 2007.

[141] J. Klein, F. Ritter, H. K. Hahn, J. Rexilius, and H.-O. Peitgen. Brain structure visualization using spectral fiber clustering. In *SIGGRAPH*, New York, NY, USA, 2006. ACM.

[142] J. Klein, H. Stuke, B. Stieltjes, O. Konrad, H. K. Hahn, and H.-O. Peitgen. Efficient fiber clustering using parameterized polynomials. In *Proc. SPIE 6918, 69182X (2008).*

[143] R. D. Kleinberg. Anytime algorithms for multi-armed bandit problems. In *SODA*, pages 928–936, 2006.

[144] M. Klusch, S. Lodi, and G. Moro. Distributed clustering based on sampling local density estimates. In *IJCAI*, pages 485–490, 2003.

[145] T. Kobayashi, M. Iwamura, T. Matsuda, and K. Kise. An Anytime Algorithm for Camera-Based Character Recognition. In *ICDAR*, pages 1140–1144, 2013.

[146] R. E. Korf. A Complete Anytime Algorithm for Number Partitioning. *Artif. Intell.*, 106(2):181–203, 1998.

[147] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. Self-Adaptive Anytime Stream Clustering. In *ICDM*, pages 249–258, 2009.

[148] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The ClusTree: indexing micro-clusters for anytime stream mining. *Knowl. Inf. Syst.*, 29(2):249–272, 2011.

[149] P. Kranen, S. Günnemann, S. Fries, and T. Seidl. MC-Tree: Improving Bayesian Anytime Classification. In *SSDBM*, pages 252–269, 2010.

[150] P. Kranen, M. Hassani, and T. Seidl. BT* - An Advanced Algorithm for Anytime Classification. In *SSDBM*, pages 298–315, 2012.

[151] P. Kranen and T. Seidl. Harnessing the strengths of anytime algorithms for constant data streams. *Data Min. Knowl. Discov.*, 19(2):245–260, 2009.

[152] H.-P. Kriegel, P. Kröger, and I. Gotlibovich. Incremental optics: Efficient computation of updates in a hierarchical cluster ordering. In *DaWaK*, pages 224–233, 2003.

[153] H.-P. Kriegel, P. Kröger, I. Ntoutsi, and A. Zimek. Density based subspace clustering over dynamic data. In *SSDBM*, pages 387–404, 2011.

[154] H.-P. Kriegel, P. Kröger, M. Renz, and S. H. R. Wurst. A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data. In *ICDM*, pages 250–257, 2005.

[155] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. Density-based clustering. *Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[156] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[157] H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *KDD*, pages 672–677, 2005.

[158] H.-P. Kriegel and M. Pfeifle. Hierarchical density-based clustering of uncertain data. In *ICDM*, pages 689–692, 2005.

[159] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. In *ICML*, 2012.

[160] P. Kröger, H.-P. Kriegel, and K. Kailing. Density-Connected Subspace Clustering for High-Dimensional Data. In *SDM*, pages 246–256, 2004.

[161] A. Kumar and S. Zilberstein. Anytime Planning for Decentralized POMDPs using Expectation Maximization. In *UAI*, pages 294–301, 2010.

[162] K. A. Kumar and C. P. Rangan. Privacy Preserving DBSCAN Algorithm for Clustering. In *ADMA*, pages 57–68, 2007.

[163] K. Kummamuru and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(3):433–439, 1999.

[164] W. W. Kywe, D. Fujiwara, and K. Murakami. Scheduling of Image Processing Using Anytime Algorithm for Real-time System. In *ICPR (3)*, pages 1095–1098, 2006.

[165] C. Lai and N. T. Nguyen. Predicting density-based spatial clusters over time. In *ICDM*, pages 443–446, 2004.

[166] J. Lee and J. Han. Trajectory Clustering: A Partition-and-Group Framework. In *SIGMOD*, pages 593–604, 2007.

[167] L. Lelis and J. Sander. Semi-supervised Density-Based Clustering. In *ICDM*, pages 842–847, 2009.

[168] H. Li, Z. Xue, L. Guo, T. Liu, J. Hunter, and S. T. Wong. A hybrid approach to automatic clustering of white matter fibers. *NeuroImage*, 49(2):1249 – 1258, 2010.

[169] J. Li, J. Sander, R. J. G. B. Campello, and A. Zimek. Active Learning Strategies for Semi-Supervised DBSCAN. In *Canadian Conference on AI*, pages 179–190, 2014.

[170] X. Liang, Q. Zhuang, N. Cao, and J. Zhang. Shape modeling and clustering of white matter fiber tracts using fourier descriptors. In *CIBCB*, pages 292–297, 2009.

[171] T. W. Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[172] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *ICAPS*, pages 262–271, 2005.

[173] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative Incremental Clustering of Time Series. In *EDBT*, pages 106–122, 2004.

[174] J. Lin, M. Vlachos, E. Keogh, D. Gunopulos, J. Liu, S. Yu, and J. Le. A MPAA-Based iterative clustering algorithm augmented by nearest neighbors search for time-series data streams. In *PAKDD*, pages 333–342, 2005.

[175] T. Lindgren. Anytime inductive logic programming. In *Computers and Their Applications*, pages 439–442, 2000.

[176] C.-L. Liu and M. P. Wellman. On state-space abstraction for anytime evaluation of bayesian networks. *SIGART Bulletin*, 7(2):50–57, 1996.

[177] D. Ma and A. Zhang. An Adaptive Density-Based Clustering Algorithm for Spatial Database with Noise. In *ICDM*, pages 467–470, 2004.

[178] M. Maddah, W. E. L. Grimson, and S. K. Warfield. Statistical modeling and EM clustering of white matter fiber tracts. In *ISBI*, pages 53–56, 2006.

[179] M. Maddah, W. M. W. III, S. K. Warfield, C.-F. Westin, and W. E. L. Grimson. Probabilistic Clustering and Quantitative Analysis of White Matter Fiber Tracts. In *IPMI*, pages 372–383, 2007.

[180] S. T. Mai. A Survey on Anytime and Active Clustering Algorithms. Technical report, University of Munich, 2013.

[181] S. T. Mai. Density-based Clustering: A Comprehensive Survey. Technical report, University of Munich, 2013.

[182] S. T. Mai. Active Density-based Clustering for Complex Data. Technical report, University of Munich, 2014.

[183] S. T. Mai, S. Goebl, and C. Plant. A Similarity Model and Segmentation Algorithm for White Matter Fiber Tracts. In *ICDM*, pages 1014–1019, 2012.

[184] S. T. Mai, X. He, J. Feng, and C. Böhm. Efficient Anytime Density-based Clustering. In *SDM*, pages 112–120, 2013.

[185] S. T. Mai, X. He, J. Feng, C. Plant, and C. Böhm. Anytime Density-based Clustering of Complex Data. *Knowledge and Information System (KAIS)*, 2014 (to appear).

[186] S. T. Mai, X. He, N. Hubig, C. Plant, and C. Böhm. Active Density-Based Clustering. In *ICDM*, pages 508–517, 2013.

[187] P. K. Mallapragada, R. Jin, and A. K. Jain. Active query selection for semi-supervised clustering. In *ICPR*, pages 1–4, 2008.

[188] R. Mangharam and A. A. Saba. Anytime Algorithms for GPU Architectures. In *RTSS*, pages 47–56, 2011.

[189] J. L. Matthews and J. Trostel. An Improved Storm Cell Identification and Tracking (SCIT) Algorithm based on DBSCAN Clustering and JPDA Tracking Methods. In *IIPS*, pages 8.3+, 2010.

[190] H. B. McMahan and G. J. Gordon. A Fast Bundle-based Anytime Algorithm for Poker and other Convex Games. In *AISTATS*, pages 323–330, 2007.

[191] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[192] B. Moberts, A. Vilanova, and J. J. Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *IEEE Trans. Vis. Comput. Graphics*, page 9, 2005.

[193] G. Moise, A. Zimek, P. Kröger, H.-P. Kriegel, and J. Sander. Subspace and projected clustering: experimental evaluation and analysis. *Knowl. Inf. Syst.*, 21(3):299–326, 2009.

[194] S. Mori. *Introduction to Diffusion Tensor Imaging*. Elsevier Science, May 2007.

[195] B. Morris and M. M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *CVPR*, pages 312–319, 2009.

[196] A.-I. Mouaddib and S. Zilberstein. Knowledge-Based Anytime Computation. In *IJCAI*, pages 775–783, 1995.

[197] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. Density-based clustering validation. In *SDM*, 2014.

[198] E. Müller, I. Assent, R. Krieger, T. Jansen, and T. Seidl. Morpheus: interactive exploration of subspace clustering. In *KDD*, pages 1089–1092, 2008.

[199] E. Müller, S. Günnemann, I. Färber, and T. Seidl. Discovering Multiple Clustering Solutions: Grouping Objects in Different Views of the Data. In *ICDE*, pages 1207–1210, 2012.

[200] K. Myers, M. J. Kearns, S. P. Singh, and M. A. Walker. A boosting approach to topic spotting on subdialogues. In *ICML*, pages 655–662, 2000.

[201] M. Nanni and D. Pedreschi. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.*, 27(3):267–289, 2006.

[202] E. N. Nasibov and G. Ulutagay. Robustness of density-based clustering methods with various neighborhood relations. *Fuzzy Sets and Systems*, 160(24):3601–3615, 2009.

[203] E. D. Nerurkar and S. I. Roumeliotis. Power-SLAM: a linear-complexity, anytime algorithm for SLAM. *I. J. Robotic Res.*, 30(6):772–788, 2011.

[204] R. T. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.*, 14(5):1003–1016, 2002.

[205] B. M. Nogueira, A. M. Jorge, and S. O. Rezende. Hierarchical confidence-based active clustering. In *SAC*, pages 216–219, 2012.

[206] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel. Density-based projected clustering over high dimensional data streams. In *SDM*, pages 987–998, 2012.

[207] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha. Streaming-data algorithms for high-quality clustering. In *ICDE*, pages 685–694, 2002.

[208] L. O'Donnell, K. M, M. E. Shenton, M. Dreusicke, W. E. L. Grimson, and C.-F. Westin. A method for clustering white matter fiber tracts. *AJNR*, 27(5):1032–1036, 2006.

[209] L. J. Odonnell and C. fredrik Westin. Automatic tractography segmentation using a highdimensional white matter atlas. *IEEE Trans. Med. Imag*, pages 1562–1575, 2007.

[210] M. G. Omran, A. A. Salman, and A. P. Engelbrecht. Dynamic clustering using particle swarm optimization with application in image segmentation. *Pattern Anal. Appl.*, 8(4):332–344, 2006.

[211] S. R. Pamudurthy, S. Chandrakala, and C. C. Sekhar. Local density estimation based clustering. In *IJCNN*, pages 1249–1254, 2007.

[212] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[213] M. M. A. Patwary, D. Palsetia, A. Agrawal, W. keng Liao, F. Manne, and A. N. Choudhary. A new scalable parallel DBSCAN algorithm using the disjoint-set data structure. In *SC*, page 62, 2012.

[214] M. M. A. Patwary, D. Palsetia, A. Agrawal, W. keng Liao, F. Manne, and A. N. Choudhary. Scalable parallel OPTICS data clustering using graph algorithmic techniques. In *SC*, page 49, 2013.

[215] M. Petrik and S. Zilberstein. Anytime Coordination Using Separable Bilinear Programs. In *AAAI*, pages 750–755, 2007.

[216] K. pong Chan and A. W.-C. Fu. Efficient Time Series Matching by Wavelets. In *ICDE*, pages 126–133, 1999.

[217] B. Qian, X. Wang, F. Wang, H. Li, J. Ye, and I. Davidson. Active learning from relative queries. In *IJCAI*, 2013.

[218] T. Rahwan, S. D. Ramchurn, V. D. Dang, A. Giovannucci, and N. R. Jennings. Anytime Optimal Coalition Structure Generation. In *AAAI*, pages 1184–1190, 2007.

[219] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci. An Anytime Algorithm for Optimal Coalition Structure Generation. *J. Artif. Intell. Res. (JAIR)*, 34:521–567, 2009.

[220] C. A. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. In *SDM*, 2005.

[221] J. Rosswog and K. Ghose. Efficiently detecting clusters of mobile objects in the presence of dense noise. In *SAC*, pages 1095–1102, 2010.

[222] J. Rosswog and K. Ghose. Detecting and tracking coordinated groups in dense, systematically moving, crowds. In *SDM*, pages 1–11, 2012.

[223] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 441–448, 2001.

[224] C. Ruiz, M. Spiliopoulou, and E. M. Ruiz. Density-based semi-supervised clustering. *Data Min. Knowl. Discov.*, 21(3):345–370, 2010.

[225] S. J. Russell and S. Zilberstein. Composing real-time systems. In *IJCAI*, pages 212–217, 1991.

[226] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[227] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: Fast Similarity Search under The Time Warping Distance. In *PODS*, pages 326–337, New York, NY, USA, 2005. ACM.

[228] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Min. Knowl. Discov.*, 2(2):169–194, June 1998.

[229] S. Sarmah, R. D. Sarmah, and D. K. Bhattacharyya. An Effective Density-Based Hierarchical Clustering Technique to Identify Coherent Patterns from Gene Expression Data. In *PAKDD (1)*, pages 225–236, 2011.

[230] S. Schlitt, T. E. Gorelik, A. A. Stewart, E. Schömer, T. Raasch, and U. Kolb. Application of clustering techniques to electron-diffraction data: determination of unit-cell parameters. *Acta Crystallographica Section A*, 68(5):536–546, Sep 2012.

[231] N. Schlitter, T. Falkowski, and J. Lässig. Dengraph-ho: Density-based hierarchical community detection for explorative visual network analysis. In *SGAI Conf.*, pages 283–296, 2011.

[232] T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann. Indexing density models for incremental learning and anytime classification on data streams. In *EDBT*, pages 311–322, 2009.

[233] T. Z. Sen, A. Kloczkowski, and R. L. Jernigan. Functional clustering of yeast proteins from the protein-protein interaction network. *BMC Bioinformatics*, 7:355, 2006.

[234] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results. *IEEE Computer*, 35(7):80–86, 2002.

[235] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[236] O. Shamir and N. Tishby. Spectral Clustering on a Budget. *Journal of Machine Learning Research*, 15:661–669, 2011.

[237] J. Shao, K. Hahn, Q. Yang, C. Böhm, A. M. Wohlschläger, N. Myers, and C. Plant. Combining Time Series Similarity with Density-Based Clustering to Identify Fiber Bundles in the Human Brain. In *ICDM Workshops*, pages 747–754, 2010.

[238] J. Shao, K. Hahn, Q. Yang, A. M. Wohlschläger, C. Böhm, N. Myers, and C. Plant. Hierarchical Density-Based Clustering of White Matter Tracts in the Human Brain. *IJKDB*, 1(4):1–25, 2010.

[239] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *VLDB*, pages 428–439, 1998.

[240] A. Sherbondy, D. Akers, R. Mackenzie, R. Dougherty, and B. Wandell. Exploring Connectivity of the Brain's White Matter with Dynamic Queries. *IEEE Trans. Vis. Comput. Graphics*, 11(4):419–430, July 2005.

[241] J. Shieh and E. J. Keogh. Polishing the Right Apple: Anytime Classification Also Benefits Data Streams with Constant Arrival Times. In *ICDM*, pages 461–470, 2010.

[242] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[243] S. Singh and A. Awekar. Incremental shared nearest neighbor density-based clustering. In *CIKM*, pages 1533–1536, 2013.

[244] P. Smyth and D. Wolpert. Anytime exploratory data analysis for massive data sets. In *KDD*, pages 54–60, 1997.

[245] B. Sofman, J. Bagnell, and A. Stentz. Anytime online novelty detection for vehicle safeguarding. In *ICRA*, pages 1247–1254, May 2010.

[246] E. Stefanakis. NET-DBSCAN: Clustering The Nodes of A Dynamic Linear Network. *Int. J. Geogr. Inf. Sci.*, 21(4):427–442, Jan. 2007.

[247] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[248] A. Tepwankul and S. Maneewongvatana. U-DBSCAN : A density-based clustering algorithm for uncertain objects. In *ICDE Workshops*, pages 136–143, 2010.

[249] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[250] A. Tramacere and C. Vecchio. gamma-ray dbscan: a clustering algorithm applied to fermi-lat gamma-ray data. i. detection performances with real and simulated data. 2012.

[251] T. N. Tran, K. Drab, and M. Daszykowski. Revised {DBSCAN} algorithm to cluster data with dense adjacent clusters. *Chemometrics and Intelligent Laboratory Systems*, 120(0):92 – 96, 2013.

[252] A. Tsai, C.-F. Westin, A. O. Hero, and A. S. Willsky. Fiber Tract Clustering on Manifolds With Dual Rooted-Graphs. In *CVPR*, 2007.

[253] C.-F. Tsai and H.-F. Yeh. Npust: An efficient clustering algorithm using partition space technique for large databases. In *IEA/AIE*, pages 787–796, 2009.

[254] D. Tuia, J. Muñoz-Marí, and G. Camps-Valls. Remote sensing image segmentation by active queries. *Pattern Recognition*, 45(6):2180–2192, 2012.

[255] K. Ueno, X. Xi, E. J. Keogh, and D.-J. Lee. Anytime Classification Using the Nearest Neighbor Algorithm with Applications to Stream Mining. In *ICDM*, pages 623–632, 2006.

[256] Ö. Uncu, W. A. Gruver, D. B. Kotak, D. Sabaz, Z. Alibhai, and C. Ng. GRIDBSCAN: GRId Density-Based Spatial Clustering of Applications with Noise. In *SMC*, pages 2976–2981, 2006.

[257] H. Van Dyke Parunak, R. Rohwer, T. C. Belding, and S. Brueckner. Dynamic Decentralized Any-time Hierarchical Clustering. In *ESOA*, pages 66–81, 2007.

[258] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML*, pages 1073–1080, 2009.

[259] P. Viswanath and V. S. Babu. Rough-DBSCAN: A fast hybrid density based clustering method for large data sets. *Pattern Recognition Letters*, 30(16):1477–1488, 2009.

[260] P. Viswanath and R. Pinkesh. l-dbscan : A fast hybrid density based clustering method. In *ICPR (1)*, pages 912–915, 2006.

[261] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. J. Keogh. Indexing Multi-dimensional Time-series with Support for Multiple Distance Measures. In *KDD*, pages 216–225, 2003.

[262] K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Active Clustering of Biological Sequences. *Journal of Machine Learning Research*, 13:203–225, 2012.

[263] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[264] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier. Active learning for semi-supervised k-means clustering. In *ICTAI (1)*, pages 12–15, 2010.

[265] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier. Boosting clustering by active constraint selection. In *ECAI*, pages 297–302, 2010.

[266] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier. An efficient active constraint selection algorithm for clustering. In *ICPR*, pages 2969–2972, 2010.

[267] V.-V. Vu, N. Labroche, and B. Bouchon-Meunier. Improving constrained clustering with active query selection. *Pattern Recognition*, 45(4):1749–1758, 2012.

[268] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang. Density-based clustering of data streams at multiple resolutions. *TKDD*, 3(3), 2009.

[269] Q. Wang, P.-T. Yap, H. Jia, G. Wu, and D. Shen. Hierarchical Fiber Clustering based on Multi-scale Neuroanatomical Features. In *MIAR*, pages 448–456, 2010.

[270] W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, pages 186–195, 1997.

[271] X. Wang and I. Davidson. Active spectral clustering. In *ICDM*, pages 561–568, 2010.

[272] X. Wang and H. J. Hamilton. DBRS: A Density-Based Spatial Clustering Method with Random Sampling. In *PAKDD*, pages 563–575, 2003.

[273] X. Wang, C. Rostoker, and H. J. Hamilton. Density-based spatial clustering in the presence of obstacles and facilitators. In *PKDD*, pages 446–458, 2004.

[274] X. Wang, C. Rostoker, and H. J. Hamilton. A density-based spatial clustering for physical constraints. *J. Intell. Inf. Syst.*, 38(1):269–297, 2012.

[275] F. L. Wauthier, N. Jojic, and M. I. Jordan. Active spectral clustering via iterative uncertainty reduction. In *KDD*, pages 1339–1347, 2012.

[276] B. Welton, E. Samanas, and B. P. Miller. Mr. scan: extreme scale density-based clustering using a tree-based network of gpgpu nodes. In *SC*, page 84, 2013.

[277] D. H. Widyantoro, T. R. Ioerger, and J. Yen. An Incremental Approach to Building a Cluster Hierarchy. In *ICDM*, pages 705–708, 2002.

[278] L. Wolf, L. Litwak, N. Dershowitz, R. Shweka, and Y. Choueka. Active clustering of document fragments using information derived from both images and catalogs. In *ICCV*, pages 1661–1667, 2011.

[279] X. Xi, K. Ueno, E. J. Keogh, and D.-J. Lee. Converting non-parametric distance-based classification to anytime algorithms. *Pattern Anal. Appl.*, 11(3-4):321–336, 2008.

[280] Q. Xu, M. desJardins, and K. L. Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science*, pages 294–307, 2005.

[281] R. Xu and D. C. W. II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[282] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A Distribution-based Clustering Algorithm for Mining in Large Spatial Databases. In *ICDE*, pages 324–331, 1998.

[283] X. Xu, J. Jäger, and H.-P. Kriegel. A Fast Parallel Clustering Algorithm for Large Spatial Databases. *Data Min. Knowl. Discov.*, 3(3):263–290, 1999.

[284] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: a structural clustering algorithm for networks. In *KDD*, pages 824–833, 2007.

[285] Y. Yang, G. I. Webb, K. B. Korb, and K. M. Ting. Classifying under computational resource constraints: anytime classification using probabilistic estimators. *Machine Learning*, 69(1):35–53, 2007.

[286] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB*, pages 385–394, 2000.

[287] B.-K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. In *VLDB*, pages 385–394, 2000.

[288] N. A. Yousri, M. S. Kamel, and M. A. Ismail. A novel validity measure for clusters of arbitrary shapes and densities. In *ICPR*, pages 1–4, 2008.

[289] J. Y. Yu and P. H. J. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(1-4):32–48, 2005.

[290] O. R. Zaïane and C.-H. Lee. Clustering Spatial Data when Facing Physical Constraints. In *ICDM*, pages 737–740, 2002.

[291] S. Zhang, S. Correia, and D. H. Laidlaw. Identifying White Matter Fiber Bundles in DTI Data Using an Automated Proximity-Based Fiber Clustering Method. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1044–1053, 2008.

[292] S. Zhang, C. Demiralp, and D. H. Laidlaw. Visualizing Diffusion Tensor MR Images Using Streamtubes and Streamsurfaces. *Vis'03*, 9:454–462, 2003.

[293] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD Conference*, pages 103–114, 1996.

[294] W. Zhao, Q. He, H. Ma, and Z. Shi. Effective semi-supervised document clustering via active learning with instance-level constraints. *Knowl. Inf. Syst.*, 30(3):569–587, 2012.

[295] S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowl. Inf. Syst.*, 8(3):374–384, 2005.

[296] B. Zhou, D. W.-L. Cheung, and B. Kao. A fast algorithm for density-based clustering in large database. In *PAKDD*, pages 338–349, 1999.

[297] J. Zhou and J. Sander. Data Bubbles for Non-Vector Data: Speeding-up Hierarchical Clustering in Arbitrary Metric Spaces. In *VLDB*, pages 452–463, 2003.

[298] S. Zhou, A. Zhou, J. Cao, W. Jin, Y. Fan, and Y. Hu. Combining sampling technique with dbscan algorithm for clustering large spatial databases. In *PAKDD*, pages 169–172, 2000.

[299] Y. Zhou, Y. Liang, K. H. Lynch, J. J. Dennis, and D. S. Wishart. PHAST: A Fast Phage Search Tool. *Nucleic Acids Research*, 39(Web-Server-Issue):347–352, 2011.

[300] Q. Zhu, G. E. A. P. A. Batista, T. Rakthanmanon, and E. J. Keogh. A Novel Approximation to Dynamic Time Warping allows Anytime Clustering of Massive Time Series Datasets. In *SDM*, pages 999–1010, 2012.

[301] Y. Zhu and D. Shasha. Warping Indexes with Envelope Transforms for Query by Humming. In *SIGMOD*, pages 181–192, New York, NY, USA, 2003. ACM.

[302] S. Zilberstein. Operational Rationality through Compilation of Anytime Algorithms. *AI Magazine*, 16(2):79–80, 1995.

[303] S. Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73–83, 1996.

[304] S. Zilberstein and S. J. Russell. Anytime Sensing Planning and Action: A Practical Model for Robot Control. In *IJCAI*, pages 1402–1407, 1993.

[305] S. Zilberstein and S. J. Russell. Approximate Reasoning Using Anytime Algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. 1995.

List of Figures

1.1 Knowledge Discovery in Databases (KDD) process. 3

2.1 The clustering results on a toy example. Due to its ability of detecting clusters with arbitrary shapes, DBSCAN can group data exactly as the ground truth. The traditional algorithm k-Means however cannot group data correctly since it can detect spherical clusters only. 11

2.2 The clustering results on a toy example. Outliers are drawn in black. While density-based clustering algorithm DBSCAN can detect those outliers, traditional clustering algorithm k-Means is unable to classify those outliers. 12

2.3 The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object; (d) the seed list S for cluster expansion. DBSCAN is currently constructing the cluster C_2. Object p is extracted from S and examined. Object a and b which lie inside the ϵ-neighborhood of p are not processed and thus are inserted into S. 15

2.4 The reachability plots (left) and clustering results (right) of OPTICS w.r.t. different extract thresholds (the dotted horizon lines in the reachability plots). Outliers are drawn in black. The parameter ϵ^* is set to 6. From the top to the bottom, the threshold value ϵ is set to 3, 2.5 and 1.5 respectively. 17

2.5 The clustering results on a dataset with clusters of varying densities. Outliers are drawn in black. DBSCAN cannot detect all the clusters exactly. 25

2.6 The data structure Xseedlist. It consists of an ordered object list OL. Each object o_i in OL is associated with a predecessor list $PL(o_i)$ and is sorted by the first element of $PL(o_i)$. Each entry of $PL(o_i)$ contains 3 items: (1) PreID: ID of a neighbor object $o_{i,k}$ ($1 \leq k \leq n$); (2) PreFlag: indicates that the distance between o_i and $o_{i,k}$ is the LB or the true distance; (3) PreDist: the distance between o_i and $o_{i,k}$. The Xseedlist operates by using a LB distance as a guideline to extend the clusters. For every object p, the Xseedlist determines all of its neighbors under the LB distance, sorts them and then updates the distance between p and its neighbors with the true distances until the core property of p is determined. The others will be updated only when they are necessary to determine the density connectivity of objects during the cluster expansion. . . 30

3.1 An example of Haar wavelet transformation. 56

4.1 The progress of an anytime algorithm. 63

4.2 The performance comparison among three different anytime algorithms. The algorithm A clearly outperforms others. 64

4.3 A common active learning scenario. 76

4.4 The performances of three different active algorithms. The algorithm A clearly outperforms others. 77

5.1 The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object; 89

5.2 From level L_i to L_{i+1}, the core object a changes to border object. Object b changes from border to noise. The noise object d remains unchanged. Cluster C_1 at L_i is broken into two clusters C_{11} and C_{12} at L_{i+1}. Object c becomes border of cluster C_2 at the new level. 93

5.3 Pseudocodes for A-DBSCAN. 96

5.4 The data structure Xseedlist. 97

5.5 Pseudocode for the algorithm A-DBSCAN-XS. 100

5.6 The performance of DBSCAN, M-DBSCAN, B-DBSCAN, A-DBSCAN and A-DBSCAN-XS on the 8 real datasets from the UCR archives. 106

5.7 The performance of DBSCAN, M-DBSCAN, B-DBSCAN, A-DBSCAN and A-DBSCAN-XS on the 3 real datasets from other sources. . . . 107

5.8 The percentages of the total distance function calls at each level on the 3 real datasets. 108

5.9 The percentages of the total distance function calls of A-DBSCAN-XS on the 3 real datasets with different sorting scheme for the Xseedlist. 109

5.10 The relationships between the two parameters μ (with $\epsilon = 4$) and ϵ (with $\mu = 5$) and performance of A-DBSCAN for the COIL20 dataset. 111

5.11 Performances of A-DBSCAN w.r.t. the tightness of LBs for the dataset Symbols. 112

6.1 The notions of DBSCAN : (a) q is directly density-reachable from p; (b) p and q is density-connected; (c) object a (red) is a core object, b (green) is border object, c (black) is noise object; 122

6.2 A general framework for active clustering. 123

6.3 Two clustering update schemes used in our algorithm based on G_{lb}. The splitting scheme (SP) starts with $G = G_{lb}$ and removes the edge (p, q) from graph G if $d(p, q) > \epsilon$. The merging scheme (MG) starts with empty graph G and adds an edge (p, q) to graph if $d(p, q) \leq \epsilon$. 124

6.4 General framework for Act-DBSCAN. 126

6.5 The change of clustering result of Act-DBSCAN from G_i to G_j. The core object a changes to a border object. Object b changes from a core to a noise object. The border object c becomes a noise object. The noise object d remains unchanged. Cluster C_1 is broken into two clusters C_{11} and C_{12}. 129

6.6 The dataset Symbols: (Left) Kernel Density Estimation (KDE) can estimate the true probability distribution better than traditional Gaussian distribution. (Right) The proposed technique Pos-Binomial outperforms other techniques in term of the prediction accuracy. 132

6.7 The comparison of different prediction techniques for 4 real datasets. Pos-Binomial outperforms other techniques in term of prediction accuracy. 133

6.8 An example for reduction property. The edge (p, q) is meaningless and does not need to be updated since p and q are already density-connected. In a high density area (right), there are more meaningless edges than in a low density area (left). 136

6.9 The comparison of the four algorithms SP-Rand, SP-SCO, MG-Rand and MG-SCO for six real datasets. SP-SCO and MG-SCO clearly outperforms other random techniques. 139

6.10 The effect of the LB function on dataset Symbols. 140

6.11 The effect of the parameter b and τ on SP-SCO and MG-SCO for the dataset Trace. 141

6.12 The effect of the parameter μ and ϵ on SP-SCO and MG-SCO for the dataset Trace. 142

6.13 Comparison with Active Spectral clustering on 2 real datasets OliveOil (6, 0.18, 2%, 0.08) and Coil20 (6, 2.5, 5%, 2). SP-SCO and MG-SCO acquire better performance than active spectral clustering algorithms. 143

6.14 Comparison with Active Spectral clustering on dataset OliveOil. SP-SCO and MG-SCO outperforms active spectral clustering algorithms. 144

6.15 Comparison with Active Spectral clustering on dataset COIL20. SP-SCO and MG-SCO outperforms active spectral clustering algorithms. 144

6.16 Stability of active spectral algorithm AspecS and MG-SCO on dataset Coil20. The performance of AspecS varies significantly. 145

6.17 The total numbers of used pairwise similarities acquired from different algorithms. 146

6.18 Runtime comparison for the dataset Trace. The parameters μ, ϵ and warping window size for LB_Keogh are 30, 75.0 and 30% respectively. Since they produce only 1 result during their runtime, the results of DBSCAN, M-DBSCAN and B-DBSCAN are presented by horizontal lines. 147

7.1 An examples of a full brain fiber dataset. Each fiber is represented by a streamline in three-dimensional space. 154

7.2 The eight major fiber bundles in human brain including Arcuate (A), Cingulum (B), Corticospinal (C), Forceps Major (D), Fornix (E), Inferior Occipitofrontal Fasciculus (F), Superior Longitudinal Fasciculus (G) and Uncinate (H). These bundles are parts of datasets used in our study. 154

8.1 By distance-based techniques, both B and C are similar to A. By envelope-based techniques, C is more similar to A than to B. 162

8.2 The comparison of two fibers A and B by WLCS model in 1D with time constraint δ and similarity threshold ϵ. For each point a_i in fiber A, every point b_j in fiber B (with j in $[i-\delta, i+\delta]$) which lies inside the ϵ-circle of a_i can be matched with a_i. 164

8.3 The Minimum Bounding Envelope (MBE) of fiber A with respect to the time constraint δ and similarity threshold ϵ. Only points of B that lie inside $MBE_{\delta,\epsilon}$ of A can be matched with A. 167

8.4 A matched pair $(b_k, a_{k'})$ can be added to the existing warping path without violating the monotony property. 168

8.5 Two bundles Arcuate and Superior Longitudinal Fasciculus are close to each other and hard to distinguish if only based on the shape similarity. 168

8.6 The effect of noise on different similarity measure functions between two fibers A and B. LCS, EDR and WLCS (SIM) are more robust to noise than other techniques. 169

8.7 The results of clustering using (a) 1-phase similarity measure (10 clusters) and (b) 2-phases similarity measure (5 clusters). The 2-phases approach produces the gold standard exactly. 170

8.8 Pseudocode for DBSCAN algorithm used in our work. 171

8.9 Effectiveness of different shape similarity measure techniques for real data set DS0. Only WLCS produces the gold standard exactly. 173

8.10 Effectiveness of different shape similarity measure techniques for noisy data set. The clustering score of WLCS reduces slightly compared with the clustering scores of other techniques. WLCS is more robust to noise than the others. 174

8.11 Effectiveness of different similarity measure techniques based on the range of *eps*. WLCS is more robust and can distinguish bundles better than other techniques. 175

8.12 Clustering results with SIM for 4 datasets DS1, DS2, DS3 and DS4 with $eps = 0.62, 0.57, 0.41$ and 0.48 respectively. 177

8.13 All datasets are well grouped according to our experts. 178

8.14 The relationship between α and the clustering scores NIM as well as its *eps* ranges on five real datasets. 179

8.15 The relationships between parameter δ, ϵ and NMI score on data set DS0. SIM is more robust to the choices of parameters than WLCS. 179

8.16 The total CPU times for (a) fiber segmentation and (b) *eps*-range query on real datasets with 5000 to 50000 fibers. 180

9.1 Example of LB_Sakurai: (a) A and B is divided into 4 and 3 segments respectively; (b) the cost matrix and the warping path for $DTW(A, B)$ (yellow cells) and $DTW(\overline{A}, \overline{B})$ (bold gray cells). 188

9.2 The distance between two segments p and q in 1D (a) and in 2D (b). 189

9.3 The average of tightness of LB_Sakurai on real fiber datasets. 190

9.4 The comparison of A-DBSCAN-XS and the others on 6 real datasets DS1 to DS6. Good results are acquired at very early stages which require very small running times. 191

9.5 The numbers of call to LBs (a) for all algorithms and (b) for the original and extended Xseedlist on the dataset DS6 193

9.6 The clustering results for dataset DS5 at each level (outliers are always draw in black). All the results from level 4 to 8 are well-grouped. 193

9.7 The clustering results for the Corpus Callosum dataset at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well confirmed according to our experts. 194

9.8 The clustering results for the dataset DS1 at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well-grouped with 8 bundles are detected with only some minor errors. 195

9.9 The clustering results for the dataset DS1 at each level (outliers are always draw in black). The clustering results at level 4 to 8 are well-grouped with 8 bundles are detected with only some minor errors. 195

9.10 Scalability of different algorithms for very large fiber datasets. . . . 196

List of Tables

8.1 Some similarity measure techniques for fibers used in our papers. MCP, HDD and DTW are existing techniques, and LCS and EDR are new adapted techniques. 165

8.2 The error rates of one nearest neighbor classification for the six similarity measure techniques. 173

8.3 NMI scores of some similarity measure techniques for some real datasets. 177

Publications

The details of relevant accepted publications accepted are listed as the following.

- Christian Böhm, Jing Feng, Xiao He, Son T. Mai, Claudia Plant and Junming Shao. A Novel Similarity Measure for Fiber Clustering using Longest Common Subsequence. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Workshop on Data Mining for Medical and Healthcare (DMMH), pages 1-9, 2011.

  In this work, S.T.M. developed the theory and the algorithms and implemented them. J.F and X.H helped with some experiments. J.S. provided some datasets for experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

- Son T. Mai, Sebastian Goebl and Claudia Plant. A Similarity Model and Segmentation Algorithm for White Matter Fiber Tracts. In International Conference on Data Mining (ICDM), pages 1014-1019, 2012.

  In this work, S.T.M. developed the theory, implemented the algorithm and did experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

- Son T. Mai, Xiao He, Jing Feng and Christian Böhm. Efficient Anytime Density-based Clustering. In SIAM International Conference on Data Mining (SDM), pages 112-120, 2013.

  In this work, S.T. M. contributed to the theory, implementation and experiment of the algorithm. C.B proposed an idea for the experiments. X.H. and J.F. helped with some experiments. The technique and results are discussed among all authors. All authors contributed to paper writing.

- Son T. Mai, Xiao He, Nina Hubig, Claudia Plant and Christian Böhm. Active Density-based Clustering. In International Conference on Data Mining (ICDM), pages 508-517, 2013.

  In this work, S.T.M. proposed the theory, did most of the implementation and experiments. X.H. implemented an active spectral clustering algorithm and participated in the experimental comparison with Active Spectral Clustering. All authors discussed the technique and the results and contributed to paper writing.

- Son T. Mai, Xiao He, Jing Feng, Claudia Plant and Christian Böhm. Anytime Density-based Clustering of Complex Data. Knowledge and Information System (KAIS), 2014. (accepted for publication). The final publication is available at Springer via http://dx.doi.org/10.1007/s10115-014-0797-0.

  In this work, S.T.M. contributed to the theory, implementation and experiment of the algorithm. C.B. gave out an idea for the experiments. X.H. and J.F. participated in some experiments. All authors discussed the principles of the technique and the results and contributed to paper writing.

The detail of relevant publications that are currently under review or are going to submit are listed as the following.

- Son T. Mai. Density-based Clustering: A Comprehensive Survey. Technical Report, University of Munich, 2013.

  In this work, S.T.M. did the major part including the literature review, experiments and paper writing.

- Son T. Mai. A Survey on Anytime and Active Clustering Algorithms. Technical Report, University of Munich, 2013.

  In this work, S.T.M. did the major part including the literature review, experiments and paper writing.

- Son T. Mai, Xiao He. Active Density-based Clustering for Complex Data. Technical Report, University of Munich, 2014.

  In this work, S.T.M. developed the theory, implemented the algorithm and did experiments. X.H. implemented an active spectral clustering algorithm and participated in the experimental comparison with Active Spectral Clustering.

Acknowledgements

I would like to express my warmest gratitude to my family, my friends, my students and all the people who have supported me during the past years while I have been working on this thesis.

In particular, I would like to express my gratitude to my supervisor Prof. Dr. Christian Böhm for his supervision, advice and supports. Furthermore, I would like to thank Prof. Dr. Xiaowei Xu for his interest in my work and his willingness to act as the second referee on this thesis.

I am very grateful to my parents and my relatives for their love, cultivation and encouragement for advanced education. I appreciate my younger sister, younger brother for taking care of our parents while I were away. My gratefulness is also to my girl friend Phan Thi Minh Diep for her endless love and supports during the last years.

I would like to thank my students especially Phan Thanh Huy, Nguyen Nam Giang, Huynh Huu Nghia, Luong Thi Mai Nhan, Nguyen Tuan Phong, Tran Duy Dong, Trinh Duy Sam, Do Hoang Dinh Tien, Nguyen Huyen Thao Vy, Nguyen An Ninh, Thai Duc Uy, Truong Thi Xuan, Nguyen Ngoc Tam and many others for their loves, cares and supports during the last years. Watching them growing up day to day has been the delight of my life.

My warmest thanks also go to my friends especially Nghiem Xuan Anh, Nguyen Hoang Vu, Pham Dang Ninh, Le Quang Loc, Vo Thi Minh Hanh, Tran Anh Quan, Andreas Grueber, Angelika Schindler, Anna Diet and her family, Do Duy Thanh, Pham Thanh Van, Nguyen Thi Truc Linh, Tran Anh Duong, Nguyen Van Te Ron, Pham Hung Thinh, Vu Quoc Huy, Nguyen Thi Anh Phuong, Duong Tuan Anh, Bui Kim Anh, Le Pham Hai Yen, Tran Hoang Phuong, Tuan-Van family and their beloved children Sandra and Julia, Tuan-Thoa family, Nguyen Minh Ngoc, Nguyen Van Hau, Nguyen Trung Thanh, Pham Quang Ninh, Giap Binh Nga, Phung Quang Di, Nguyen Duy Linh, Nguyen Thi Yen, Nguyen Tri Hieu, Nguyen Thi Bach Hue, Le Van Quoc Anh, To Ba Lam, Nguyen Luong Anh Tuan, Nguyen

Tran Cao Tuan Khoa, Phan Anh Tuan, Le Van Thanh, Nguyen Quoc Viet Hung, Truong Quoc Hung, Dang Hoang Vu, Hoang Thai Son, Nguyen Mai Anh, Doan Hong Nhung, Nguyen Hong Nhung, Phan Ngoc Thuy Tien, Bui Thi Thu Ha, Do Phan Nam Tien and many other friends for their cares, supports and joyful times during the last years.

My special thanks to Frank Krojer for providing me computers to do experiments and his endless technical supports.

I also thank to all my colleagues at the data mining group including Jing Feng, Xiao He, Wei Ye, Linfei Zhou, Annika Tonch, Bianca Wackersreuther, Peter Wackersreuther, Andrew Zherdin, Nina Hubig, Samuel Maurus, Nikola Müller, Katrin Haegler, Bettina Konte, Frank Fiedler, Michael Plavinski, Can Altinigneli, Annahita Oswald, Qinli Yang, Junming Shao, Sebastian Goebl, and Claudia Plant for their helps and cooperations.

Last but not least, I would like to thank Ministry of Education and Training (Vietnam) (MOET) and German Academic Exchange Service (DAAD) for their financial supports during the last years.

Munich, 9 Sep 2013

Mai Thai Son

Curriculum Vitae

Mai Thai Son received the B.Sc. and M.Sc. degrees in Computer Science from University of Technology, HoChiMinh City, Vietnam. He is currently a Ph.D. student at the Institute for Mathematics, Informatics and Statistics, University of Munich, Munich, Germany. His researches focus on computer graphic, computational geometry, metaheuristic search and local search techniques, time series data mining, time series prediction, dimensionality reduction techniques, machine learning, data mining algorithms for high dimension and complex data with applications in various fields such as Neuroscience, Biomedicine, Bioinformatics, Transportation and Environmental Science.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Mai Thai Son

Name, Vorname

Munich, 26.06.2014

Ort, Datum

Mai Thai Son

Unterschrift Doktorand/in

Formular 3.2