Accurate and efficient clustering algorithms for very large data sets

Syed Abdul Quddus

This thesis is submitted in total fulfilment of the requirement
for the degree of Doctor of Philosophy

Faculty of Science and Technology
Federation University Australia
PO Box 663
University Drive, Mount Helen
Ballarat, Victoria, Australia 3353

Submitted in April 2017

Abstract

The ability to mine and extract useful information from large data sets is a common concern for organizations. Data over the internet is rapidly increasing and the importance of development of new approaches to collect, store and mine large amounts of data is significantly increasing.

Clustering is one of the main tasks in data mining. Many clustering algorithms have been proposed but there are still clustering problems that have not been addressed in depth especially the clustering problems in large data sets. Clustering in large data sets is important in many applications and such applications include network intrusion detection systems, fraud detection in banking systems, air traffic control, web logs, sensor networks, social networks and bioinformatics. Data sets in these applications contain from hundreds of thousands to hundreds of millions of data points and they may contain hundreds or thousands of attributes.

Recent developments in computer hardware allows to store in random access memory and repeatedly read data sets with hundreds of thousands and even millions of data points. This makes possible the use of existing clustering algorithms in such data sets. However, these algorithms require a prohibitively large CPU time and fail to produce an accurate solution. Therefore, it is important to develop clustering algorithms which are accurate and can provide real time clustering in such data sets. This is especially important in a big data era.

The aim of this PhD study is to develop accurate and real time algorithms for clustering in very large data sets containing hundreds of thousands and millions of data points. Such algorithms are developed based on the combination of heuristic algorithms with the incremental approach. These algorithms also involve a special procedure to identify dense areas in a data set and compute a subset most informative representative data points in order to decrease the size of a data set.

It is the aim of this PhD study to develop the center-based clustering algorithms. The success of these algorithms strongly depends on the choice of starting cluster

centers. Different procedures are proposed to generate such centers. Special procedures are designed to identify the most promising starting cluster centers and to restrict their number.

New clustering algorithms are evaluated using large data sets available in public domains. Their results will be compared with those obtained using several existing center-based clustering algorithms.

DECLARATION

I, Syed Abdul Quddus, declare that the PhD thesis entitled "Efficient algorithms for solving clustering problems in very large data sets" contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the Federation University Australia Library, being made available for loan and photocopying, subject to the provision of the copyright Act 1968. I also give permission for the digital version of my thesis to be made available on the web, via the University Library catalogue, the Australian Digital Theses Program (ADTP) and also through web search engines, unless permission has been granted by the Federation University Australia to restrict access for a period of time.

Signature.....

April 2017

Syed Abdul Quddus

Acknowledgement

It is my pleasure to express my deep gratitude to my principal supervisor, Associate Professor Adil Bagirov, whose great expertise, consideration, and extreme patience helped me to construct my PhD thesis with meaning and structure. He is a perfect mentor and I feel truly fortunate to have worked under his supervision.

I would like to thank my associate supervisors Associate Professor Madhu Chetty and Dr. Dean Webb for their help and guidance.

This project would not have been completed without the morale support of the Faculty of Science and Technology, and to them I extend my gratitude. The atmosphere and environment in our faculty, both academic and administrative, have been extremely supportive and enjoyable. I would like also to thank PhD coordinator of the Faculty of Science and Technology Dr. Stephen Carey for his help and support throughout my PhD study. I would like also to thank Research Services of Federation University Australia for their support.

Last, but not least, I owe my loving thanks to my whole family for their unfailing support, constant encouragement and care they have given me throughout the entire period of my study.

Dedication

I dedicate my dissertation work to my family. A special feeling of gratitude to my loving mother and father, whose words of encouragement ring in my ears.

Table of Contents

Abstract **2**

Declaration **4**

Acknowledgement **5**

Dedication **6**

1 Introduction **18**

2 Literature review **22**

2.1 Data collection, cleaning and pre-processing 22

2.2 Definition of clustering problems 24

2.3 Clustering algorithms for large data sets 25

2.3.1 *k*-means algorithm 26

2.3.2 Modifications of *k*-means algorithm 26

2.3.3 Partial/merge *k*-means algorithm 27

2.3.4 Parallel *k*-means algorithm 28

2.3.5 BIRCH algorithm 28

2.3.6 CADD algorithm 28

2.3.7 Patch clustering algorithm 29

2.3.8 Grid based clustering 29

2.3.9 Incremental clustering 30

2.3.10 Subspace clustering 30

2.4 Evolutionary algorithms for clustering 31

2.4.1 Genetic Algorithm 32

2.4.2 Artificial Bee Colony Algorithm 32

2.4.3 Particle Swarm Optimization Algorithm 34

2.4.4 Ant Colony Optimization 35
2.5 Data mining software 35
2.5.1 WEKA 35
2.5.2 R 36
2.6 Summary 37

3 Optimization models and algorithms of clustering 38
3.1 Combinatorial model of the clustering problem 38
3.2 Integer programming model of clustering problems 39
3.2.1 Nonsmooth optimization models of clustering problems . . 40
3.2.2 Comparison of two optimization models 42
3.2.3 The auxiliary clustering problem 42
3.2.4 Optimization algorithms for clustering problems 44
3.3 Summary 47

4 Fast incremental clustering algorithms 48
4.1 Computation of starting points 48
4.2 Algorithm for reduction of data points 52
4.3 Reduction of computational effort 54
4.4 An incremental clustering algorithm 58
4.5 Hyperbolic smoothing of cluster functions 59
4.5.1 Computational complexity of the fast modied global k-means algorithm 62
4.5.2 Incremental Algorithms for fast modified global k-means . . 63
4.6 A smooth incremental clustering algorithm 64
4.7 A modified smooth incremental clustering algorithm 66

5 Computational results: small data sets 68
5.1 Data sets 69
5.2 Results 70
5.3 Summary 75

6 Computational results: medium size data sets 89
6.1 Data sets 89
6.2 Results 93
6.3 Summary 98

7 Computational results: large data sets **117**
7.1 Data sets 117
7.2 Results 119
7.3 Summary 123

8 Conclusions and future work **139**

Bibliography **142**

List of Tables

5.1 Small size data sets 69
5.2 Results with Wilt data set: Cluster function values, (The best results are highlighted) 71
5.3 Results for Wilt data set: CPU time in seconds, (this Table corresponds to the Figure 5.1), (The best results are highlighted) 71
5.4 Results for Wine Quality data set: Cluster function values, (The best results are highlighted). 72
5.5 Results for Wine Quality data set: CPU time in seconds,(this Table corresponds to the Figure 5.3), (The best results are highlighted) . . 72
5.6 Results for Waveform Generator data set: Cluster function values, (The best results are highlighted). 73
5.7 Results for Waveform Generator data set: CPU time in seconds,(this Table corresponds to the Figure 5.5), (The best results are highlighted) 73
5.8 Results for Turkiye Student Evaluation data set: Cluster function values, (The best results are highlighted) 74
5.9 Results for Turkiye Student Evaluation data set: CPU time in seconds,(this Table corresponds to the Figure 5.7), (The best results are highlighted) 74
5.10 Results for Drug yprop 41 data set: Cluster function values, (The best results are highlighted) 75
5.11 Results for Drug yprop 41 data set:CPU time in seconds, (The best results are highlighted) 75
5.12 Results for Combined Cycle Power Plant data set: Cluster function values, (The best results are highlighted) 76

5.13 Results for Combined Cycle Power Plant data set: CPU time in seconds,(this Table corresponds to the Figure 5.9), (The best results are highlighted) 76
5.14 Results for Gesture Phase Segmentation data set: Cluster function values, (The best results are highlighted). 77
5.15 Results for Gesture Phase Segmentation data set: CPU time in seconds, (The best results are highlighted) 77
5.16 Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms. 87
5.17 Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in terms of its efficiency,when tested over five small data sets,in comparison with other algorithms, with a breakdown of the numbers for different k values 88

6.1 Description of medium data size data sets 90
6.2 Results for Gas Sensor Array Drift data set,(this Table corresponds to the Figure 6.1), (The best results are highlighted) 93
6.3 Results for D15112 data set, (this Table corresponds to the Figure 6.11), (The best results are highlighted) 94
6.4 Results for Letter Recognition data set, (this Table corresponds to the Figure 6.9), (The best results are highlighted) 95
6.5 Results for Chess (King-Rook vs. King) data set, (this Table corresponds to the Figure 6.10), (The best results are highlighted) 96
6.6 Results for Online News popularity data set, (this Table corresponds to the Figure 6.12), (The best results are highlighted) 97
6.7 Results for Bank Marketing data set, (this Table corresponds to the Figure 6.3), (The best results are highlighted) 98
6.8 Results for TamilNadu Electricity Board Hourly Reading data set, (The best results are highlighted) 99
6.9 Results for KEGG Metabolic Relation Network data set, (this Table corresponds to the Figure 6.5), (The best results are highlighted) . . 100
6.10 Results for Shuttle Control data set, (The best results are highlighted) 100

6.11 Results for Jester Collaborative Filtering data set,(The best results are highlighted) 101
6.12 Results for Programmed Logic Array (Pla85900) data set and (this Table corresponds to the Figure 6.6), (The best results are highlighted) 101
6.13 Results for Sensit-vehicle-acoustic data set, (this Table corresponds to the Figure 6.7), (The best results are highlighted) 102
6.14 Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms. 102
6.15 Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in comparison with other algorithms, with a breakdown of the numbers for different k values 116

7.1 The brief description of large data sets. 118
7.2 Results for shuttle2 data set, (this Table corresponds to the Figure 7.1), (The best results are highlighted) 120
7.3 Results for Localization Data for Person Activity data set, (this Table corresponds to the Figure 7.6),(The best results are highlighted) 121
7.4 Results for Online Video Characteristics and Transcoding Time data set, (The best results are highlighted) 122
7.5 Results for artificial-2state-sequence-data set,(The best results are highlighted) 123
7.6 Results for Skin-non skin Segmentation data set,(this Table corresponds to the Figure 7.7),(The best results are highlighted) 124
7.7 Results for Cod Coma data set,(The best results are highlighted) . . 125
7.8 Results for Online Retail data set, (The best results are highlighted) 125
7.9 Results for Algebra training data set, (this Table corresponds to the Figure 7.10), (The best results are highlighted) 126
7.10 Results for Phones Accelrometer data set, (this Table corresponds to the Figure 7.8), (The best results are highlighted) 126
7.11 Results for Ijcnn1 data set, (The best results are highlighted) 127

7.12 Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms. 127
7.13 Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in comparison with other algorithms, with a breakdown of the numbers for different k values 138

List of Figures

5.1 The CPU time vs the number of clusters: Wilt data set,(this Figure corresponds to Table. 5.3) 78
5.2 The number of distance function evaluations vs the number of clusters: Wilt data set 79
5.3 The number CPU time vs the number of clusters: Wine Quality data set,(this Figure corresponds to Table. 5.5) 80
5.4 The number of distance function evaluations vs the number of clusters: Wine Quality data set 81
5.5 The CPU time vs the number of clusters: Waveform data set,(this Figure corresponds to Table. 5.7) 82
5.6 The number of distance function evaluations vs the number of clusters: Waveform data set 83
5.7 The CPU time vs the number of clusters: Turkiye Students evaluations data set,(this Figure corresponds to Table. 5.9) 84
5.8 The number of distance function evaluations vs the number of clusters: Phase Gesture data set,(this Figure corresponds to Table. 5.15). 85
5.9 The number of clusters vs the CPU time: Combined Cycle Power Plant data set,(this Figure corresponds to Table. 5.13). 86

6.1 The CPU time vs the number of clusters: Drift data set,(this Figure corresponds to Table. 6.2) 103
6.2 The number of distance function evaluations vs the number of clusters: Drift data set 104
6.3 The CPU time vs the number of clusters: Bank Marketing data set,(this Figure corresponds to Table. 6.7) 105

6.4 The number of distance function evaluations vs the number of clusters: Bank Marketing data set 106
6.5 The CPU time vs the number of clusters: Relation Network data set, (this Figure corresponds to Table. 6.9) 107
6.6 The CPU time vs the number of clusters: Programmed Logic Array (Pla85900) data set,(this Figure corresponds to Table. 6.12) 108
6.7 The CPU time vs the number of clusters: Sensit-vehicle-acoustic data set, (this Figure corresponds to Table. 6.13) 109
6.8 The number of distance function evaluations vs the number of clusters: Sensit-vehicle-acoustic data set 110
6.9 The CPU time vs the number of clusters: Letters data set, (this Figure corresponds to Table. 6.4) 111
6.10 The CPU time vs the number of clusters: Chess (King-Rook vs. King) data set,(this Figure corresponds to Table. 6.5) 112
6.11 The CPU time vs the number of clusters: D15112 data set,(this Figure corresponds to Table. 6.3) 113
6.12 The CPU time vs the number of clusters: Online popularity data set ,(this Figure corresponds to Table. 6.6) 114
6.13 The number of distance function evaluations vs the number of clusters: Tamilnadu data set 115

7.1 The CPU time vs the number of clusters: Shuttle2 Mldata data set, (this Figure corresponds to Table. 7.2) 128
7.2 The number of distance function evaluations vs the number of clusters: Shuttle2 Mldata data set 129
7.3 The number of distance function evaluations vs the number of clusters: Artificial-2state-sequence-data data set 130
7.4 The number of distance function evaluations vs the number of clusters: Skin-non-skin segmentation data set 131
7.5 The number of distance function evaluations vs the number of clusters: Phones Accelerometer data set 132
7.6 The CPU time vs the number of clusters: Localization for person Activity data set, (this Figure corresponds to Table. 7.3) 133
7.7 The CPU time vs the number of clusters:Skin-non-skin segmentation data set, (this Figure corresponds to Table. 7.6) 134

7.8 The CPU time vs the number of clusters:Phones Accelerometer data set (this Figure corresponds to Table. 7.10) 135
7.9 The CPU time vs the number of clusters:Online Retail Dataset,(this Figure corresponds to Table. 7.8) 136
7.10 The CPU time vs the number of clusters:Algebra 2005 2006 train Dataset, (this Figure corresponds to Table. 7.9) 137

Publications:

The following papers have been published and submitted for publication:

1. S. Quddus, Fast algorithms for unsupervised learning in large data sets, In Proceedings of the conference: Computer Science and Information Technology, Dubai, 2017, January 27-28, pages 15-17.

2. A.M. Bagirov and S. Quddus, Efficient and accurate clustering algorithms for very large data sets, Submitted for publication.

Chapter 1

Introduction

The expression Knowledge Discovery in Databases (KDD) refers to the extensive process of discovering useful knowledge in a collection of data. Main steps of KDD are data collection and selection, data cleansing and preparation, data transformation, data mining, incorporating prior knowledge on data sets and data evaluation and interpretation [23, 98, 102, 103] .

The data mining, an analysis step of KDD, is the computational process of discovering and extraction of useful information from data sets. This useful information may be a model, summary or just derived values relating to a problem definition [19]. Data mining is a field of computer science which deals with collecting, analysing, predicting, learning and discovering patterns from data sets. In many cases, the data mining is used as synonym of KDD [23].

Data mining has been applied in many areas of human activity. Such areas include spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, internet security and many other applications [17]. New developments in information technology and the demands for more information shows that the data mining now embraces more areas than ever before.

There are different type of data mining problems including clustering (or unsupervised classification), supervised data classification, feature selection and extraction. Clustering is among most important problems in data mining. A process of partitioning or grouping a set of data into meaningful similar subsets based on some criteria, typically a distance function between objects, called clustering. In other words, clustering is a certain kind of unsupervised machine learning which means that the clustering algorithms learns from unlabeled data [77, 98, 101]. Clustering

has many applications in business, finance and science [77, 98, 101].

Clustering now is a quite advanced area. Many algorithms proposed over the last fifty years to solve clustering problems. However, there are still clustering problems that have not been addressed in depth. Clustering problems in very large data sets are among those problems. Such problems have many applications including network intrusion detection systems, fraud detection in banking systems, air traffic control, web logs, sensor networks, social networks and bio-informatics. Data sets in these applications contain from hundreds of thousands to hundreds of millions of data points and they may contain hundreds or thousands of attributes.

Recent developments in computer hardware allows one to store in random access memory (RAM) and repeatedly read data sets with hundreds of thousands and even millions of data points. These developments make possible the use of existing clustering algorithms in such data sets. However, existing clustering algorithms are either not accurate or require prohibitively large computational effort in such data sets. The development of accurate real-time clustering algorithms for such data sets is highly important, especially in a big data era.

The aim of this PhD study is to develop accurate and real time algorithms for clustering in very large data sets. The term "very large data sets" in this thesis means that a data set contains from hundreds of thousands to millions of data points. Accurate and real-time algorithms are developed based on the combination of heuristic algorithms with the incremental approach. These algorithms also involve a special procedure to identify dense areas in a data set and compute a subset of most informative data points which can represent the whole data set. This allows one to significantly decrease the size of a data set and keep only those data points which determine the cluster distribution of a data set.

This PhD study aims to develop the center-based clustering algorithms. The success of these algorithms strongly depends on the choice of starting cluster centers. In this thesis various procedures are proposed to generate such centers. Thus, main objectives of this research are as follows.

Objective 1: Development of a special procedure to identify dense areas in a data set and introduce an algorithm for data reduction based on this procedure. This will be done by introducing a tolerance which depends on the size of data set in the whole search space.

Objective 2: Development of algorithms for finding starting cluster centers. The

success of most clustering algorithms strongly depends on the choice of starting cluster centers. This is especially important in large data sets. Using the nonconvex nonsmooth optimization model of the clustering problem we will develop an algorithm which will allow to identify most promising starting cluster centers among data points.

Objective 3: Develop clustering algorithms using data points with weights and without weights. This algorithms are based on the incremental approach, that is they build clusters gradually starting from one cluster.

Objective 4: Implement and evaluate proposed clustering algorithms and compare them with existing clustering algorithms based on optimization techniques. New clustering algorithms will be evaluated using large data sets available in public domains. Their results will be compared with those obtained using several existing center-based clustering algorithms based on optimization techniques.

Structure of the thesis

The thesis is structured as follows. In the current chapter, Chapter 1 is an introductory chapter where we formulate main objectives of this thesis and outline the structure of the thesis. Chapter 2 presents a literature review of existing algorithms, including evolutionary algorithms, for solving clustering problems in large data sets. This chapter also describes widely used data mining software packages containing various clustering algorithms.

In Chapter 3, we define nonsmooth nonconvex optimization models of clustering problems and discuss incremental approaches for solving these problems. In this chapter we also consider procedures for finding starting cluster centers.

Chapter 4 introduces new versions of the incremental algorithms considered in Chapter 3 where different schemes are applied to accelerate the incremental clustering algorithms.

Chapters 5, 6 and 7 present computational results. Each of these chapters contains information about data sets used in numerical experiments. Chapter 5 presents results with small, Chapter 6 with medium size and Chapter 7 with large and very large data sets. All these three chapters also include comparison with other mainly optimization based incremental clustering algorithms

Finally, in Chapter 8 we give a summary of results obtained in this thesis and also discuss possible directions of future research.

Chapter 2

Literature review

In this chapter first we briefly describe main steps in data analysis, then give overview of clustering algorithms for large data and clustering algorithms based on evolutionary algorithms. We also give a brief description of existing publicly available data mining software.

2.1 Data collection, cleaning and pre-processing

In this section we discuss main steps in data analysis before applying any clustering algorithms. These steps are: data collection, data cleaning and data pre-processing.

Data collection can be done in many different ways. There is a large amount of pre-existing data which can be obtained from repositories all over the world. Also many companies or academic institutions collect data relevant to solving their problems. Most important questions here are about the integrity of data and the means used to source the data. The data collecting process may include a number of repeat collections, before useful patterns can be found [25, 28, 30].

The methods used to collect data depend on the application, aims for collecting the data and problem definition. For example, social scientists generally collect data via observational techniques. This is a systematic process of watching and recording behavior and characteristics. Another method is interview based, where questioning of respondents is done either individually or as a group.

Another example of the data collection is their collection using sensors measuring certain activities or data generators designed to create certain distributions for testing or simulating problems. Generally this is an automated process, where large

amounts of data can be generated.

The collected data must be stored in a such way that relevant data remains related to each other and can be called up to be used quickly and easily. With large amounts of data sourced, it is possible that the data may be stored on multiple machines and over multiple sites [25, 28, 30].

Since in most cases humans are involved in the collection process, possible errors can be made. In the example considered above about collecting data in social sciences respondents or observers could be tired or distracted leading to incorrect values being recorded. Survey questions can be ambiguous or misleading which can lead to incorrect, redundant or no answers at all.

Moreover, it is obvious that during manual data collection, due to human weaknesses, boredom or inadequate surveys, there may occur errors and these erroneous values can make their way into data. Automated data collection may also experience problems (missing, noise or outlier data) due to possible damaged sensors or incorrectly calibrated measuring equipment. These outlier data values are non-consistent values with other values in other data observations and may cause difficulties during the data mining. These occur rarely, generally only representing about two percent of all measurements pertaining to the observations [19].

If data pre-processing has not been applied, then it is likely that useful hidden patterns will be disguised leading to the decreased performance of the data mining techniques. The overall benefits of pre-processing means a smaller, cleaner and higher quality data set which yields more concentrative patterns [36,37]. Therefore, in any data mining applications data pre-processing, including data cleaning, is an important step.

Sometimes, noisy data may also be considered as outliers, though only in the case of irregular occurrences. For example, the noisy data automatically collected by sensors is more problematic, as there may occur random noise both as bufirsts or just intermittently. Noisy data can disagree with values describing classes, breaking down the natural pattern correlations and reducing the ability to differentiate between the certain patterns. However, unrelated attributes do not add any noise because their values are not indicative of any inherent patterns [17].

There has been a lot of work undertaken in data analysis for pre-processing and preparation of data. As a result, a lots of data pre-processing tools are available. A number of data pre-processing techniques can be found in [36]. The paper [31] proposed a new method for dealing with missing values by using an imputation

method based on the k-nearest neighbor algorithm. The papers [31, 32] present overview of a number of existing methods. These methods include ignoring missing values, disregarding instances containing missing values and maximum likelihood procedures.

The paper [34] presents a comprehensive survey of contemporary outlier detection techniques. These include a large array of statistical, neural and machine learning techniques. This paper highlights the advantages and disadvantages of each technique and point out that an algorithm's suitability is based on the data's distribution model, correct attribute types, scalability, speed and modelling accuracy. The paper [34] demonstrates the benefits of removing observations in terms of noise and data size complexity.

2.2 Definition of clustering problems

In cluster analysis we assume that we have been given a finite set of points A in the n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1, \ldots, a^m\}, \text{ where } a^i \in \mathbb{R}^n,\ i = 1, \ldots, m.$$

The hard unconstrained clustering problem is the distribution of the points of the set A into a given number k of disjoint subsets $A^j,\ j = 1, \ldots, k$ with respect to predefined criteria such that:

1) $A^j \neq \emptyset,\ j = 1, \ldots, k$;

2) $A^j \bigcap A^l = \emptyset,\ j, l = 1, \ldots, k,\ j \neq l$;

3) $A = \bigcup\limits_{j=1}^{k} A^j$;

4) no constraints are imposed on the clusters $A^j,\ j = 1, \ldots, k$.

Remark 1. Constraints in the clustering problems can be on size of clusters (their radius or the number of data points in each of them) or that given set of points should belong to the same cluster (this may rely on a priori information on the cluster structure of a data set). However, in this thesis we consider clustering problems without any constraints on clusters.

The sets $A^j,\ j = 1,\ldots,k$ are called clusters. We assume that each cluster A^j can be identified by its center $x^j \in \mathbb{R}^n,\ j = 1,\ldots,k$. The problem of finding these centers is called *the k-clustering (or k-partition) problem*.

It is assumed that points from the same cluster are similar to each other and points from different clusters are dissimlar to each other. Therefore, the similarity measure is fundamental in cluster analysis. This measure, in particular, can be defined using different distance functions or norms. For example, the Euclidean norm, otherwise known as the L_2-norm, Manhetten norm, known also as the L_1-norm and L_∞-norm can be used to define the similarity measure. Note that the distance functions and the similarity measures are not the same. In some cases the similarity may not satisfy the so-called triangle inequality for distance functions (or norms).

In this thesis, we define the similarity measure using the squared Euclidean distance. Consider the center $x^j,\ j = 1,\ldots,k$ and a point $a \in A$. Then the similarity of these two point is defined as:

$$\|x^j - a\|^2 = \sum_{i=1}^{n} (x_i^j - a_i)^2. \tag{2.2.1}$$

In this case the clustering problem is also known as the minimum sum-of-squares clustering problem and center $x^j\ j = 1,\ldots,k$ is also called centroid. In general, any clustering problem is an NP-hard combinatorial optimization problem.

2.3 Clustering algorithms for large data sets

In this section we briefly describe mainstream clustering algorithms for large data sets. Clustering algorithms need to efficiently scale up with both the dimensionality and the size of the data set. Due to the amount of data, many standard data mining approaches can not be applied at all, batch processing of the full data set is infeasible due to memory restrictions, while online processing requires several passes over the data set, i.e. it is infeasible due to time constraints. Due to these circumstances, researchers have worked on modifications of various clustering algorithms such that they run in a single or few passes over the data and such that they require only a priori fixed amount of allocated memory.

2.3.1 k-means algorithm

The k-means is the most popular clustering algorithm. Here k stands for the number of clusters. It is typically a user input to the algorithm. Some criteria can be used to automatically estimate k. The k-means algorithm is an iterative algorithm in nature. It works only for data sets with numerical attributes [44].

The k-means clustering algorithm for finding the k-partition of the set A proceeds as follows:

Algorithm 1. A k-means algorithm for clustering.

Step 1. Choose seed solution consisting of k centers.

Step 2. Allocate all data points $a \in A$ to its closest centers and obtain the k-partition of A.

Step 3. Recompute centers for this new partition and go to Step 2. Continue until no more data points change their clusters.

In Step 1 of the k-means algorithm initial cluster centers can be chosen randomly but not necessarily among data points. This algorithm is a local search algorithm and it converges only to a local minimum of the clustering problem. The k-means algorithm is fast algorithm even in very large data sets. However, a solution obtained by this algorithm differs significantly from the global solution of the clustering problem as the size of a data set increases. Therefore, different versions of this algorithm have been developed to improve the quality of a solution. We will consider some of these algorithms in next subsections.

2.3.2 Modifications of k-means algorithm

The success of the k-means algorithms strongly depends on the choice of starting cluster centers. To date, different clustering algorithms have been developed which are modifications of the k-means algorithm and these modifications differ from each other on the way the starting cluster centers are chosen.

These modifications include the following algorithms:

1. Lloyd algorithm. This clustering algorithm is the version of the k-means algorithm. It was introduced in [114].

2. Forgy algorithm. This algorithm is an alternative of least-squares algorithm for clustering [115].

3. MacQueen algorithm. This clustering algorithm introduced in [116] is similar to the Forgys algorithm. The difference is in the last stage where the MacQueen algorithm moves the center points to the mean of their Voronoi set.

4. Hartigan algorithm. This algorithm was introduced in [117] and it is another version of the k-means algorithm. The difference is in the stopping criteria used in these algorithms.

5. k-means++ algorithm. This algorithm is the version of the k-means algorithm and was introduced in [118]. It uses a special procedure for initialization of cluster centers using data points. The k-means++ algorithm is among most efficient algorithms for clustering. It can be applied to large data sets.

6. X-means algorithm. This algorithm is an improvement of the original k-means algorithm [119]. It uses a special procedure for initialization of cluster centers.

All these algorithms can be applied to solve clustering algorithms in large data sets. They are fast and can provide real time clustering. However, results presented in [120] demonstrate that all these algorithms, except the k-means++ algorithm are highly inaccurate in large data sets. Even multi-start versions of these algorithms are inaccurate and cannot be considered as alternatives to many other algorithms for clustering in large data sets. The k-means++ algorithm is not always efficient in data sets with not separated clusters.

2.3.3 Partial/merge k-means algorithm

In the paper [44], the authors introduce the partial/merge k-means algorithm which processes the overall set of points in cells, and merges the results of the partial k-means steps into an overall cluster representation. The partial k-means and the merge k-means are implemented as data stream operators that are adaptable to available computing resources such as volatile memory and processors by parallelizing and cloning operators, and by computing k-means on partitions of data that can be fit into memory.

Partial/merge k-means re-runs the k-means several times to get better result at each partition. However, this algorithm is sensitive to the size of partitioning in massive data sets and can end up at a solution which is significantly different from the global solution.

2.3.4 Parallel k-means algorithm

In [43], the authors develop a parallelized version of the k-means algorithm. This allows them to apply the k-means algorithm to very large data sets. The authors use a network of homogeneous workstations with Ethernet network and message-passing for communication between processors. In an Ethernet network, all communications consist of packets transmitted on a shared serial bus available to all processors and a master-slave single program multiple data approach (SPMD) is used.

The concept is to distribute processing of k-means on k machines which result in a satisfactory time complexity. On the other hand for k clusters we have to configure exactly k machines and every time rerun the k-means form the starting cluster centers. However, due to memory limitation this algorithm may not be efficient for massive data set.

2.3.5 BIRCH algorithm

One of the earliest clustering methods for large data sets is BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [38]. This algorithm adopts the notion of clustering feature to summary description of clustering properties, which is demonstrated that it is especially suitable for very large data sets. The BIRCH incrementally and dynamically clusters incoming multi dimensional metric data points to try to produce the best quality clustering with the available resources (i.e., available memory and I/O time constraints). It utilizes measurements that capture the natural closeness of data. These measurements can he stored and updated incrementally in a height balanced tree.

2.3.6 CADD algorithm

Clustering Algorithm based on Density and adaptive Density-reachable (CADD) is proposed in [42]. According to the notion and enlightenment of BIRCH, the author in this paper proposes an incremental clustering algorithm based on definitions of

subcluster similarity for very large spatial data sets. The incremental clustering algorithm is simple and efficient, and has good performance especially for very large spatial data sets. Similar to the Clustering Feature CF, a SubCluster Feature SCF is also presented by a triple of numbers, which gives out the statistic description of a subcluster.

Since the CADD algorithm is an extension of the BIRCH algorithm in some extend, which has argued that unlike BIRCH, it can detect clusters with arbitrary shape and size, however the same shortcomings of the former algorithm is still unsolved.

2.3.7 Patch clustering algorithm

Batch clustering (like k-means) requires all training data to be stored in the main memory which becomes infeasible for very large or massive data sets. The paper [45] proposes a simple and efficient strategy for k-means clustering with restricted buffer where data are processed consecutively in patches of predefined size (Patch Clustering). In [46] the same strategy is transferred to Neural Gas Network.

Apart from memory reduction, patch clustering allows a reduction of time because of the faster convergence of the separate patch clustering. This can be explored even further by introducing parallelization into the procedure. The parallelization can be done in a way such that almost no communication over head is included.

2.3.8 Grid based clustering

The Grid File [47] is a multidimensional data structure, which adapts gracefully to the distribution of patterns X in the value space of the domains of X. The Grid Structure is a main memory data structure. It lacks the external disk storage support and the data manipulation facilities of the original Grid File. The Grid Structure consists of d scales (one for each dimension), the grid directory (a d-dimensional array) and b data blocks. Each scale is a 1-dimensional array. A value of this array represents a $(d-1)$-dimensional hyperrectangle, which partitions the value space.

The grid directory is a d-dimensional dynamic array and represents the grid partition produced by the d scales. The data blocks contain the stored patterns. Each element of the grid directory refers to a data block. It is possible that two or

more directory elements reference the same data block. The value space defined by the union of the directory elements referencing the data block i is called block region V_{Bi}. Each block region is always shaped like a d-dimensional rectangular box.

Conventional cluster algorithms calculate a distance based on a dissimilarity metric (e.g. Euclidean distance, etc.) between patterns and cluster centers. The patterns are clustered accordingly to the resulting dissimilarity index. The Grid Clustering algorithm organize the value space containing the patterns. For more than one dimensional data a variation of the multidimensional data structure of the Grid File will be used, which is called Grid Structure.

The patterns are treated as points in a d-dimensional value space and are randomly inserted into the Grid Structure. These points are stored according to their pattern values, preserving the topological distribution. The Grid Structure partitions the value space and administrates the points by a set of surrounding rectangular shaped blocks. In the literature many grid based clustering algorithms [47, 49–51] are proposed.

2.3.9 Incremental clustering

Usually the massive data set can not fit into the available main memory, therefore the entire data matrix is stored in a secondary memory and data items are transferred to the main memory one at a time for clustering. Only the cluster representations are stored in the main memory to alleviate the space limitations.

Incremental clustering is based on the assumption that it is possible to consider patterns one at a time and assign them to existing clusters. A new data item is assigned to a cluster without affecting the existing clusters significantly. Typically, they are non-iterative. So their time requirements are also small. The major advantage of the incremental clustering algorithms is their limited space requirement since the entire data set is not necessary to store in the memory. Therefore, these algorithms are well suited for very large or massive data sets.

2.3.10 Subspace clustering

Unlike feature selection and feature transformation methods which examine the data set as a whole, subspace clustering algorithms localize their search and are able to

uncover clusters that exist in multiple, possibly overlapping subspaces in a massive data set. Just as with feature selection, subspace clustering requires a search method and an evaluation criteria. In addition, subspace clustering must somehow limit the scope of the evaluation criteria so as to consider different subspaces for each cluster.

Subspace clustering must evaluate features on only a subset of the data, representing a cluster. They must use some measure to define this context. We refer to this as a measure of locality. Some authors categorized the subspace clustering algorithms into two groups of Bottom-up and Top-down based on how they determine a measure of locality with which to evaluate subspaces. A very good survey of subspace clustering can be found in [52, 53, 55, 56, 58, 60].

2.4 Evolutionary algorithms for clustering

Evolutionary algorithms have been widely used to design clustering algorithms. They have been applied either directly to clustering problems, considering these problems as global optimization problems or in combination with other clustering algorithms to improve the quality of solutions obtained by the latter algorithms.

Evolutionary algorithms are easy to implement and they have both local and global search properties. These properties explain why these algorithms are attractive for solving clustering problems. However, in large data sets evolutionary algorithms become inaccurate and may require large computational effort. On the other side these algorithms can be used to generate good strating cluster centers for other clustering algorithms.

The paper [75] applies the tabu search algorithm for solving clustering problems. Comparison of clustering algorithms based on different evolutionary techniques is presented in [76]. The simulated annealing method for clustering is developed in [85, 95, 98] and a branch and bound algorithm for clustering is studied in [87, 91]. The paper [121] presents an algorithm based on the combination of the k-means and genetic algorithms.

Evolutionary algorithms can be used to generate good starting cluster centers distributed over all search space. The combination of these algorithms with other clustering may lead to the design efficient and accurate clustering algorithms for large data sets. Here we briefly describe four evolutionary algorithms which have been used to design clustering algorithms:

1. Genetic Algorithm
2. Artificial Bee Colony (ABC) Optimization
3. Particle Swarm Algorithm
4. Ant Colony Algorithm

2.4.1 Genetic Algorithm

D. Goldberg developed a computational algorithm based on the Darwin theory rule: "the strongest species that survives" and "the survival of an organism can be maintained through the process of reproduction, crossover and mutation". This is called Genetic Algorithm (GA). It is used to get solution to an optimization problem in a natural fashion. This solution is called a chromosome which is made up of genes. To get the optimum solution (maximum/minimum) of the optimization problem by GA, the chromosomes will undergo a process called fitness function. This fitness function is used to measure the suitability of a solution generated by GA with the problem [39, 40].

2.4.2 Artificial Bee Colony Algorithm

Basturk and Karaboga introduced an Artificial Bee Colony (ABC) algorithm for numerical optimization problems based on the foraging behavior of honey bees for [41]. The ABC model is consisted of three kinds of bees colony: employed bees, onlookers and scouts. It is assumed that for each food source there is only one artificial employed bee. It can be said that the number of employed bees in the colony is equal to the number of food sources around the hive. The employed bees function like that they go to their food source and come back to hive and dance on this area. When food source of an employed bee has been abandoned, it becomes a scout bee which then starts searching a new food source. On the other hand, onlooker bees watch the dances of employed bees and select the food sources depending on dances [41]. The ABC algorithm is used to solve both unconstrained and constrained optimization problems.

We describe the ABC algorithm for unconstrained optimization problems. In this case the number of the employed bees is equal to the number of solutions in the population. At the first step, a randomly distributed initial population (food

source positions) is generated. After initialization, the population is subjected to repeat the cycles of the search processes of the employed, onlooker, and scout bees, respectively. An employed bee produces a modification on the source position in her memory and discovers a new food source position. The ABC algorithm [41] is explained by the colony of artificial bees which are divided into three groups of bees: employed bees, onlookers and scouts, respectively. First half majority of the bee colony consists of the employed artificial bees and the second half contains the onlookers. There is only one employed bee for each food source. The employed bee whose food source has been ended by the bees becomes a scout. In mathematical equation representation, we may express the above statement as:

- No. of employed bees = No. of food sources around the hive.
- No. of the employed bees or the onlooker bees = No. of solutions in the population.

In the ABC algorithm, the position of a food source shows a possible solution to the optimization problem and the nectar amount of a food source relates to the quality (fitness) of the associated solution. In the initialization step, the ABC produces a randomly distributed initial population $P(G = 0)$ of SN solutions (food source positions), where SN denotes the size of population. Each solution $x_i, i = 1, \ldots, SN$ is a D-dimensional vector, where D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles $(C = 1, 2, \ldots, MCN)$, of the search processes of the employed bees, the onlooker bees and scout bees [41]. Here MCN is the maximum number of cycles.

An employed bee generates an alteration on the position (solution) in her memory which depends upon the local information (visual information) and checks the nectar amount (fitness value) of the new source (new solution). If the nectar amount of the new one (position) is higher than that of the previous one, the bee memorizes the new position and disregards the old one. Or else she holds the previous position in her memory. When all employed bees finish the search process, they share the nectar information of the food sources and their position information with the onlooker bees in the dance area.

An onlooker bee evaluates the nectar information taken from all employed bees and selects a food source with a probability associated to its nectar amount. As in the case of the employed bee which makes a modification on the position in her

memory and tests the nectar amount of the candidate source. If, its nectar is evaluated higher than that of the previous one, the bee memorizes the new position and not remembers the old position. An artificial onlooker bee decides a food source, based on the the probability value related with that food source. Details of this algorithm can be found in [41].

2.4.3 Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [61], inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called p_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called g_{best}. When a particle takes part of the population as its topological neighbors, the best value is a local best, called l_{best}. Details of this algorithm can be found in [61, 62, 65].

2.4.4 Ant Colony Optimization

The ant colony optimization (ACO) search process was initially described, and has been used to address problems in many different fields, including telecommunication, transportation and forestry. ACO algorithms are based on the mechanics of ants searching for food from a nest area. It was initially suggested for route management problems. This algorithm is inspired by observation of real ants.

Individually, each ant is blind, frail and almost insignificant. Yet, by being able to co-operate with each other, the colony of ants demonstrates complex behaviour. One of these is the ability to get the closest route to a food source or some other interesting landmark. This is done by laying down special chemicals called "pheromones." As more ants use a particular trail, the pheromone concentration on it increases, hence attracting more ants. In our example, an artificial ant is placed randomly in each city and, during each iteration, chooses the next city to go to (see, [68, 70, 71], for details).

2.5 Data mining software

In this section we briefly describe publicly available data mining packages.

2.5.1 WEKA

WEKA (an abbreviation for Waikato Environment for Knowledge analysis) is data mining software, which is developed by the Department of Computer Science, University of Waikato, New Zealand [36] with the funding of the New Zealand government since 1993. It is open source application software written in the Java programming language [36].

This was first used in agriculture industries for a work bench for machine learning tools, to determine the necessary success factors. Secondly, it was trained to develop new methods and then assess their efficiency. This has led to a data mining application that possesses many learning algorithms, pre-processing and post-processing tools and a perceptive graphical user interface [36]. WEKA, an out of the box application (that can be used immediately after installation), is widely used in research and education and industry for data mining applications. There are many methods of interfacing with the WEKA software in order to challenge the data min-

ing problem [36]. In our case, it is then practical that the user begins the data mining procedure by using the application's "explorer" graphical user interface. The user interface shows a well-defined menu which includes the following tasks [36]:

- Data pre-processing;
- Supervised data classification;
- Clustering (or unsupervised classification);
- Associate rules;
- Attribute selections;
- Visualization.

These tasks roughly imitate the steps shown in the Knowledge Discovery in Databases process. For the collection phase, WEKA can connect to databases, although it does not allow for a storage solution itself. It is required to collect data and store it elsewhere.

2.5.2 R

R is a language and environment for statistical computing and graphics. R language is widely used among statisticians and data miners for developing statistical and data analysis software. It includes the following features [113]:

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either onscreen or on hard copy, and
- A well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.

2.6 Summary

This chapter provides the literature review of algorithms for solving clustering problems in very large data sets. First, we briefly discuss main steps for data preprocessing. Then we consider clustering algorithms including those based on the evolutionary algorithms. Finally, we briefly describe publicly available data mining software containing clustering algorithms.

Chapter 3

Optimization models and algorithms of clustering

In this chapter we consider optimization models of the clustering problem and discuss algorithms for its solution. These algorithms are based on different optimization techniques and an incremental approach is used for their design.

Definition of the clustering problem is given in Chapter 2. We use the squared Euclidean distance function to define the similarity in this problem. Clusters are identified by their centers (called also centroids).

Suppose that a data set A containing finite number of points $a^1,\ldots,a^m$ in the n-dimensional space $\mathbb{R}^n$ is given. Our aim is to compute $k > 0$ clusters in this data set. This problem is also known as the k-clustering problem.

3.1 Combinatorial model of the clustering problem

Denote the set of k clusters in the set A by $\bar{A} = (A^1,\ldots,A^k)$ and a set of all possible k-partitions of the set A by $\tilde{A}$. Then *the combinatorial model* can be given as:

$$\text{minimize } \Psi_k(\bar{A}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} d(x^j, a^i) \tag{3.1.1}$$

subject to

$$\bar{A} = (A^1,\ldots,A^k) \in \tilde{A}. \tag{3.1.2}$$

Here x^j is the center of the cluster $A^j,\ j = 1,\ldots,k$ which can be found by solving the following optimization problem:

$$\text{minimize } \frac{1}{|A^j|}\sum_{a\in A^j} d(x,a) \text{ subject to } x \in \mathbb{R}^n. \tag{3.1.3}$$

Here $|\cdot|$ stands for the cardinality of a set. If the squared Euclidean distance is used for the similarity measure then the center x^j can be found explicitly as follows:

$$x^j = \frac{1}{|A^j|}\sum_{a\in A^j} a,\ j = 1,\ldots,k. \tag{3.1.4}$$

Note that in this formulation decision variables are nonempty subsets of the set A and therefore optimization algorithms cannot be directly applied to solve the problem (3.1.1)-(3.1.2). The combinatorial model can be used to solve clustering problems only in very small data sets.

3.2 Integer programming model of clustering problems

The k-clustering problem is formulated as the following optimization problem (see [84, 97]):

$$\text{minimize } \psi_k(x,w) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} w_{ij}\,\|x^j - a^i\|^2 \tag{3.2.1}$$

subject to

$$\sum_{j=1}^{k} w_{ij} = 1,\ i = 1,\ldots,m, \tag{3.2.2}$$

$$w_{ij} \in \{0,1\},\ i = 1,\ldots,m,\ j = 1,\ldots,k. \tag{3.2.3}$$

Here w_{ij} is the association weight of pattern a^i with the cluster j. w_{ij} is 1 if pattern a^i is allocated to the cluster j and 0 otherwise. We can see that w is an $m\times k$-matrix. The function ψ_k is called *a cluster function.*

Cluster centers x^j are computed using binary coefficients $w_{ij},\ i = 1,\ldots,m,\ j = 1,\ldots,k$:

$$x^j = \frac{\sum_{i=1}^{m} w_{ij}a^i}{\sum_{i=1}^{m} w_{ij}},\ \ j = 1,\ldots,k. \tag{3.2.4}$$

In fact this optimization model of the clustering problem contains continuous variables $x^j \in \mathbb{R}^n,\ j = 1,\ldots,k$ which are cluster centers and also binary variables $w_{ij},\ i = 1,\ldots,m,\ j = 1,\ldots,k$, however, continuous variables are expressed via binary variables using (3.2.8). Therefore in this problem binary variables are decision variables and for this reason the problem (3.2.1)-(3.2.3) is called the integer programming model of the clustering problem.

In the next chapter we will develop algorithms for identifying dense areas in a data set. This algorithm will allow us to reduce the number of data points by removing data points from some neighborhood of a given point and assigning it a weight depending on the number of removed points. Such an approach necessitate to consider data sets where points may have different weights. We formulate optimization models for such data sets.

Now we assume that each data point $a^i \in A,\ i = 1,\ldots,m$ has own weight $c_i \geq 0$. Then the integer programming model (3.2.1)-(3.2.3) of the k-clustering problem can be reformulated as follows:

$$\text{minimize } \bar{\psi}_k(x,w) = \frac{1}{m}\sum_{i=1}^{m} c_i \sum_{j=1}^{k} w_{ij}\,\|x^j - a^i\|^2 \tag{3.2.5}$$

subject to

$$\sum_{j=1}^{k} w_{ij} = 1,\ i = 1,\ldots,m, \tag{3.2.6}$$

$$w_{ij} \in \{0,1\},\ i = 1,\ldots,m,\ j = 1,\ldots,k. \tag{3.2.7}$$

In this case cluster centers x^j are computed as:

$$x^j = \frac{\sum_{i=1}^{m} c_i w_{ij} a^i}{\sum_{i=1}^{m} c_i w_{ij}},\ \ j = 1,\ldots,k. \tag{3.2.8}$$

3.2.1 Nonsmooth optimization models of clustering problems

Nonsmooth nonconvex optimization formulation of the clustering problem is as follows (see [78–80, 84]):

$$\text{minimize } \ f_k(x) \quad \text{subject to } x = (x^1,\ldots,x^k) \in \mathbb{R}^{n\times k}, \tag{3.2.9}$$

where

$$f_k(x^1,\ldots,x^k)=\frac{1}{m}\sum_{i=1}^{m}\min_{j=1,\ldots,k}\|x^j-a^i\|^2. \tag{3.2.10}$$

The objective function f_k is called *a cluster function.* This function is convex when $k=1$. It is also continuously differentiable in this case since the minimum operation is not used. The function f_k is both nonsmooth and nonconvex when $k>1$ and both these properties stem from the use of minimum operation.

The number of local minimizers of the objective function f_k increases significantly as the number of clusters k and the number of data points m increase. Function under minimum are simple convex quadratic functions, however, the use of minimum operation leads to appearance of many local minimizers when k is relatively large.

The objective function f_k can be represented as a difference of two convex functions as follows:

$$f_k(x)=f_k^1(x)-f_k^2(x),$$

where

$$f_k^1(x)=\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\|x^j-a^i\|^2,$$

$$f_k^2(x)=\frac{1}{m}\sum_{i=1}^{m}\max_{j=1,\ldots,k}\sum_{t=1,t\neq j}^{k}\|x^t-a^i\|^2.$$

This leads to *the nonsmooth difference of convex model* of the clustering problem.

If the data set A contains points a^i with weights $c_i,\ i=1,\ldots,m$ then the nonsmooth optimization model (3.2.9)-(4.3.6) can be reformulated as:

$$\text{minimize}\ \ \bar{f}_k(x)\ \ \text{ subject to } x=(x^1,\ldots,x^k)\in\mathbb{R}^{n\times k}, \tag{3.2.11}$$

where

$$\bar{f}_k(x^1,\ldots,x^k)=\frac{1}{m}\sum_{i=1}^{m}c_i\min_{j=1,\ldots,k}\|x^j-a^i\|^2. \tag{3.2.12}$$

The expression for the difference of convex components of this function are:

$$\bar{f}_k^1(x)=\frac{1}{m}\sum_{i=1}^{m}c_i\sum_{j=1}^{k}\|x^j-a^i\|^2,$$

$$\bar{f}_k^2(x) = \frac{1}{m}\sum_{i=1}^{m} c_i \max_{j=1,\ldots,k} \sum_{t=1,t\neq j}^{k} \|x^t - a^i\|^2.$$

3.2.2 Comparison of two optimization models

Comparing the integer programming and nonsmooth optimization models of the clustering problems we can come to the following conclusions:

1. The objective function ψ_k depends only on binary variables w_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,k$, however, the objective function f_k depends only on continuous variables $x^1,\ldots,x^k$. It is easy to deal with continuous variables than integer and in particular, binary variables.

2. The number of variables in the integer programming model (3.2.1)-(3.2.3) is $m \times k$ whereas this number in nonsmooth optimization model (3.2.9) is $n \times k$ and the number of variables does not depend on the number of instances. It should be noted that in many real-world data sets the number of instances m is significantly greater than the number of features n. This means that the number of variables in the integer programming model can easily reach millions in large data sets however, this number will not change with increase of the number of data points in the nonsmooth optimization model.

3. The function ψ_k is continuously differentiable with respect to w. Since the function f_k is represented as a sum of minima functions it is nonsmooth for $k > 1$.

4. Problem (3.2.1)-(3.2.3) is integer programming problem and problem (3.2.9) is nonsmooth global optimization problem. These problems are equivalent in the sense that their global minimizers coincide (see [84]).

Items 1 and 2 can be considered as advantages of the nonsmooth optimization formulation (3.2.9) of the clustering problem.

3.2.3 The auxiliary clustering problem

The clustering problem is a global optimization problem and in large data sets the objective function in this problem has many local minimizers. Conventional global optimization methods are very time-consuming when one applies to solve

such problems. Local search methods demand significantly less computational time and computer memory than global optimization methods, however they can get only local solutions which can be significantly different from global ones. Global solutions or solutions close to them provide the best cluster distribution of a data set with least number of clusters.

The success of local search methods strongly depends on the set of starting cluster centers. Therefore, it is imperative to develop a special procedure to generate good starting cluster centers. One such procedure will be introduced in the next chapter. This procedure uses the so-called auxiliary clustering problem which we define below. For $k > 1$ this problem is defined under assumption that the solution to $(k-1)$-clustering problem is known.

Let $x^1, \ldots, x^{k-1},\ k \geq 2$ be a solution to the $(k-1)$-clustering problem. Denote by d^i_{k-1} the distance between $a^i,\ i = 1, \ldots, m$ and the closest cluster center among $k-1$ centers $x^1, \ldots, x^{k-1}$:

$$d^i_{k-1} = \min\left\{\|x^1 - a^i\|^2, \ldots, \|x^{k-1} - a^i\|^2\right\}. \tag{3.2.13}$$

We will also use the notation d^a_{k-1} for $a \in A$. Define the following function:

$$\bar{f}_k(y) = \frac{1}{m}\sum_{i=1}^{m} \min\left\{d^i_{k-1}, \|y - a^i\|^2\right\},\ y \in \mathbb{R}^n. \tag{3.2.14}$$

We call $\bar{f}_k$ *the k-th auxiliary cluster function*. This function is nonsmooth and as a sum of minima of convex functions it is, in general, nonconvex. Moreover, the function is locally Lipschitz and directionally differentiable. It is obvious that

$$\bar{f}_k(y) = f_k(x^1, \ldots, x^{k-1}, y),\ \forall y \in \mathbb{R}^n.$$

A problem:

$$\text{minimize } \bar{f}_k(y) \ \text{ subject to } y \in \mathbb{R}^n \tag{3.2.15}$$

is called *the k-th auxiliary clustering problem.*

Now consider the case when each data point $a^i \in A$ has a weight $c_i \geq 0,\ i = 1, \ldots, m$. In this case we have

$$d^i_{k-1} = c_i \min\left\{\|x^1 - a^i\|^2, \ldots, \|x^{k-1} - a^i\|^2\right\}.$$

the function $\bar{f}_k$ can be rewritten as:

$$\bar{f}_k(y) = \frac{1}{m}\sum_{i=1}^{m} \min\left\{d_{k-1}^i, c_i\|y-a^i\|^2\right\},\ y \in \mathbb{R}^n.$$

3.2.4 Optimization algorithms for clustering problems

Optimization methods, both deterministic and stochastic, have been applied to develop different algorithms for solving clustering problems and especially, for solving the minimum sum-of-squares clustering problems. These algorithms can be categorized into the following groups:

1. *Clustering algorithms based on deterministic optimization techniques.* The clustering problem is a global optimization problem. Both global and local search methods were applied to solve this problem. These methods include the dynamic programming, the interior point method, the cutting plane method, the branch and bound and the neighborhood search methods [87, 88, 90, 91, 94]. Other deterministic optimization techniques include the discrete gradient, truncated codifferential and hyperbolic smoothing methods which were applied to solve the minimum sum-of-squares clustering problem using its nonsmooth optimization formulation [82, 86, 100, 101]. In [96], the clustering problem is formulated as a nonlinear programming problem for which a tight linear programming is constructed via the reformulation-linearization technique. This construction is embedded within a specialized branch-and-bound algorithm to solve the problem to global optimality. A clustering algorithm based on a variant of the generalized Benders decomposition, denoted as the global optimum search, is developed in [99].

2. *Clustering algorithms based on metaheuristics.* Some metaheuristics were applied to solve the clustering problem including tabu search, simulated annealing and genetic algorithms [75, 76, 85, 95, 98]. Metaheuristics can efficiently deal with both integer and continuous variables and therefore they can be applied to solve the clustering problem using its integer programming formulation (3.2.1)-(3.2.3). However, it is also well-known that metaheuristics are not efficient for solving clsutering problems in very large data sets. In such data sets they require huge computational effort.

3. *Heuristics.* These algorithms were specifically designed to solve the clustering problem. The k-means and its variations are representatives of such heuristics [89, 97]. In these algorithms starting cluster centers are generated randomly either among data points or from the search space. Clustering algorithms belonging to this class are very sensitive to the choice of starting cluster centers.

4. *Heuristics based on the incremental approach.* These algorithms start with the computation of the center of the whole data set A and attempt to optimally add one new cluster center at each stage. In order to solve Problem (3.2.9) for $k > 1$ these algorithms start from an initial state with the $k-1$ centers for the $(k-1)$-clustering problem and the remaining k-th center is placed in an appropriate position. The global k-means and modified global k-means are representatives of these algorithms [77, 81, 90, 93].

Numerical results demonstrate that most of algorithms mentioned in items 1) and 2) are not always efficient to solve the clustering problems in large data sets. For example, some deterministic methods and metaheuristics cannot efficiently solve clustering problems in data set with thousands of data points. These methods require considerably more computational effort than heuristic algorithms in such data sets. They can be applied to solve clustering problems in large data sets in combination with other fast algorithms.

Heuristic clustering algorithms have shown to be efficient algorithms. Their typical representative, k-means algorithm, can deal with very large data sets. However, this algorithm is very sensitive to the choice of starting points. It can calculate only local minimizers of the clustering problem and in large data sets these local minimizers might be significantly different from the global minimizers. The multi-start versions of these algorithms are not alternatives to more accurate and efficient algorithms based on various optimization techniques.

Since the clustering is a global optimization problem the use of local methods for its solution should be combined with algorithms for finding good starting cluster centers. Algorithms based on an incremental approach are able to get such starting points. Incremental clustering algorithms have been developed over the last decade [77, 81, 90, 92, 93]. Numerical results demonstrate that these algorithms are efficient for solving clustering problems in large data sets. However, in very large data sets containing hundreds of thousands and millions of data points these

algorithms may require prohibitively large computational time. Therefore, special algorithms should be developed to accurately and efficiently solve clustering problems in such data sets.

Next we describe several incremental and optimization based clustering algorithms which will be used for comparison in Chapters 5, 6 and 7. All these algorithms compute clusters incrementally that is they start with one cluster center which is the center of the whole data set A and gradually add a new cluster center. Main difference between these algorithms is in the way they compute the starting point for the next cluster center. For given $k > 1$ assume that the solution to the $(k-1)$-clustering problem is known and we show how each algorithm solves the k-clustering problem.

Global k-means algorithm (GKM). The GKM algorithm is introduced in [93]. In this algorithm all data points are considered as a starting point for the k-th cluster center. Each data point is added to $(k-1)$ cluster centers and the k-means algorithm starts from these points to solve k-clustering problem. This means that one gets new m solutions to the k-clustering problem. The solution with the smallest value of the objective function in clustering problem is chosen as a solution to the k-clustering problem. Although such an algorithm is accurate however it is not efficient even for data sets containing tens thousands of data points. Therefore, in its implementation not all data points but the data point providing largest decrease of the objective function is chosen as a starting point for the k-th cluster center.

Modified Global k-means algorithm (MGKM). The MGKM algorithm was introduced in [77]. In this algorithm data points providing the decrease of the objective function more than some threshold are chosen as a starting points to solve the auxiliary clustering problem. Then the solution to the auxiliary clustering problem with the smallest value of the auxiliary clustering function is chosen as the starting point for the k-th cluster center. This point is added to $k-1$ cluster centers to form a starting point for the k-clustering problem and the k-means algorithm is applied to solve the problem starting from this starting point.

The multi-start modified global k-means algorithm (MS-MGKM) was developed in [104]. This algorithm is an extension of the MGKM algorithm. In this algorithm a special procedure is introduced to generate a set of starting cluster centers. Data points and the auxiliary clustering problem are used to generate these centers. The k-means algorithm is applied starting from each of these points and

previous $k-1$ cluster centers to solve the k-clustering problem. The best solution with lowest value of the clustering function is accepted as a solution to the k-th clustering problem.

Difference of convex model based clustering algorithm (DCClust). This algorithm was developed in [122]. It is based on the difference of convex model of the clustering problem. The nonsmooth optimization algorithm was introduced to solve this problem. A special procedure is applied to get good starting points for cluster centers.

Algorithm based on the Difference of convex algorithm (DCA). This algorithm is considered in [122]. It is also based on the difference of convex model of the clustering problem. The Difference of Convex Algorithm, introduced in [106, 107], is applied to solve optimization problems both clustering and auxiliary clustering problems. A special procedure is used to generate starting points for cluster centers.

3.3 Summary

In this chapter three models of the hard unconstrained clustering problem is considered. The comparison of optimization models is presented. It is shown that the nonsmooth optimization model allows one to significantly reduce the number of decision variables and to avoid using binary variables. These models also were introduced for data sets where each data point has own weight.

Different algorithms for solving the clustering problems are discussed and compared. It is noted that most of these algorithms are not applicable to solve the clustering problems in very large data sets. Finally, more detailed discussion on clustering algorithm, which are used for comparison using numerical results, is provided.

Chapter 4

Fast incremental clustering algorithms

In this chapter we introduce fast clustering algorithms. These algorithms are applicable to solve clustering problems in very large data sets. Most important components of these algorithms are the procedure for finding good starting cluster centers, a special procedure to reduce computational efforts and a procedure to reduce the size of a data set. We start with description of the procedure for finding starting cluster centers. This procedure is described for data sets where each data point has a weight.

4.1 Computation of starting points

An algorithm proposed in this section is an extension of the algorithm from [104] for data sets with weights.

Let a data set $A = \{a^1, \ldots, a^m\} \subset \mathbb{R}^n,\ m \geq 1$ be given. Moreover, assume that each data point a^i has a weight $c_i \geq 0$. Let $l > 1$ and the solution $(x^1, \ldots, x^{l-1}) \in \mathbb{R}^{(l-1)n}$ to the $(l-1)$-partition problem is known. Consider the following two sets:

$$S_1 = \left\{ y \in \mathbb{R}^n :\ c_i \|y - a^i\|^2 \geq d^i_{l-1},\ \forall i \in \{1, \ldots, m\} \right\},$$

$$S_2 = \left\{ y \in \mathbb{R}^n :\ \exists i \in \{1, \ldots, m\} \text{ such that } c_i \|y - a^i\|^2 < d^i_{l-1} \right\}.$$

The set S_1 contains all points $y \in \mathbb{R}^n$ which do not attract any point from the set A and the set S_2 contains all points $y \in \mathbb{R}^n$ which attract at least one point from A. It

is obvious that cluster centers $x^1,\ldots,x^{l-1} \in S_1$. Since the number l of clusters is less than the number of data points in the set A all data points which are not cluster centers belong to the set S_2 (because such points attract at least themselves) and therefore this set is not empty. Note that $S_1 \bigcap S_2 = \emptyset$ and $S_1 \bigcup S_2 = \mathbb{R}^n$. It is clear that

$$\bar{f}_l(y) \leq \frac{1}{m}\sum_{i=1}^{m} d_{l-1}^i,\ \forall y \in \mathbb{R}^n$$

and

$$\bar{f}_l(y) = f_{l-1}(x^1,\ldots,x^{l-1}) = \frac{1}{m}\sum_{i=1}^{m} d_{l-1}^i,\ \forall y \in S_1$$

that is the l-th auxiliary cluster function is constant on the set S_1 and any point from this set is a global maximizer of this function. In general, a local search method will terminate at any of these points. Therefore, starting points for solving Problems (3.2.9) and (3.2.15) should not be chosen from the set S_1. In this section, we design a special procedure which allows one to select starting points from the set S_2.

Take any $y \in S_2$. Then one can divide the set A into two subsets as follows:

$$\bar{B}_1(y) = \left\{a \in A :\ c_a\|y-a\|^2 \geq d_{l-1}^a\right\},$$

$$\bar{B}_2(y) = \left\{a \in A :\ c_a\|y-a\|^2 < d_{l-1}^a\right\}.$$

Here c_a is a weight of the point $a \in A$. Notice that $\bar{B}_1(y) = B_2(y) \bigcup B_3(y)$ and $\bar{B}_2(y) = B_1(y)$. The set $\bar{B}_2(y)$ contains all data points $a \in A$ which are closer to the point y than to their cluster centers and the set $\bar{B}_1(y)$ contains all other data points. Since $y \in S_2$ the set $\bar{B}_2(y) \neq \emptyset$. Furthermore, $\bar{B}_1(y) \bigcap \bar{B}_2(y) = \emptyset$ and $A = \bar{B}_1(y) \bigcup \bar{B}_2(y)$. Then

$$\bar{f}_l(y) = \frac{1}{m}\left(\sum_{a \in B_1(y)} d_{l-1}^a + \sum_{a \in B_2(y)} c_a\|y-a\|^2\right).$$

The difference $z_l(y)$ between the value of the l-th auxiliary cluster function at y and the value $f_{l-1}(x^1,\ldots,x^{l-1})$ for the $(l-1)$-clustering problem is:

$$z_l(y) = \frac{1}{m}\sum_{a \in \bar{B}_2(y)} \left(d_{l-1}^a - c_a\|y-a\|^2\right)$$

which can be rewritten as

$$z_l(y) = \frac{1}{m}\sum_{a\in A}\max\left\{0, d^a_{l-1} - c_a\|y-a\|^2\right\}. \tag{4.1.1}$$

The difference $z_l(y)$ shows the decrease of the value of the l-th cluster function f_l comparing with the value $f_{l-1}(x^1,\ldots,x^{l-1})$ if the point $(x^1,\ldots,x^{l-1},y)$ is chosen as the cluster center for the l-clustering problem.

If a data point $a \in A$ is the cluster center then this point belongs to the set S_1, otherwise it belongs to the set S_2. Therefore we choose a point y from the set $A \setminus S_1$. We take any $y = a \in A \setminus S_1$, compute $z_l(a)$ and introduce the following number:

$$z^1_{max} = \max_{a\in A\setminus S_1} z_l(a). \tag{4.1.2}$$

Let $\gamma_1 \in [0,1]$ be a given number. We compute the following subset of A:

$$\bar{A}_1 = \left\{a \in A \setminus S_1 :\ z_l(a) \geq \gamma_1 z^1_{max}\right\}. \tag{4.1.3}$$

If $\gamma_1 = 0$ then $\bar{A}_1 = A \setminus S_1$ and if $\gamma_1 = 1$ then the set $\bar{A}_1$ contains data points with the largest decrease z^1_{max} (the global k-means algorithm from [93] uses such data points as the starting point for the l-th cluster center).

For each $a \in \bar{A}_1$ we compute the set $\bar{B}_2(a)$ and its center $c(a)$. We replace the point $a \in \bar{A}_1$ by the point $c(a)$ because the latter is better representative of the set $\bar{B}_2(a)$ than the former. Denote by $\bar{A}_2$ the set of all such centers. For each $c \in \bar{A}_2$ we compute the number $z^2_l(c) = z_l(c)$ using (4.3.3). Finally, we compute the following number:

$$z^2_{max} = \max_{c\in\bar{A}_2} z^2_l(c). \tag{4.1.4}$$

The number z^2_{max} represents the largest decrease of the values $f_l(x^1,\ldots,x^{l-1},c)$ among all centers $c \in \bar{A}_2$ comparing with the value $f_{l-1}(x^1,\ldots,x^{l-1})$.

Let $\gamma_2 \in [0,1]$ be a given number. We define the following subset of $\bar{A}_2$:

$$\bar{A}_3 = \left\{c \in \bar{A}_2 :\ z^2_l(c) \geq \gamma_2 z^1_{max}\right\}. \tag{4.1.5}$$

If $\gamma_2 = 0$ then $\bar{A}_3 = \bar{A}_2$ and if $\gamma_2 = 1$ then the set $\bar{A}_3$ contains only centers c with the largest decrease of the cluster function f_l. If take $\gamma_1 = 0$ and $\gamma_2 = 1$ then we get the scheme for finding starting points used in the modified global k-means algorithm

[77].

All points from the set $\bar{A}_3$ are considered as starting points for solving the auxiliary clustering problem (3.2.15). This problem is nonconvex and it is important to get good starting points if one applies a local method for its solution. We use all data points for the computation of the set $\bar{A}_3$ and therefore this set contains starting points from different parts of the data sets. Such a strategy allows to get either global or near global solutions to Problem (3.2.9) (as well as to Problem (3.2.15)) using only local methods.

Applying the k-means algorithm and using starting points from $\bar{A}_3$ we get a set of local minimizers of Problem (3.2.15). Since the k-means algorithm starting from different points can arrive to the same solution, the number of local minimizers found is no greater than the cardinality of the set $\bar{A}_3$. We denote by $\bar{A}_4$ the set of local minimizers of Problem (3.2.15) obtained using the k-means algorithm and starting points from $\bar{A}_3$. Define

$$\bar{f}_l^{min} = \min_{y \in \bar{A}_4} \bar{f}_l(y). \tag{4.1.6}$$

Let $\gamma_3 \in [1, \infty)$ be a given number. Compute the following set:

$$\bar{A}_5 = \left\{ y \in \bar{A}_4 : \bar{f}_l(y) \leq \gamma_3 \bar{f}_l^{min} \right\}. \tag{4.1.7}$$

If $\gamma_3 = 1$ then $\bar{A}_5$ contains the best local minimizers of the function $\bar{f}_l$ obtained using starting points from the set $\bar{A}_3$. If γ_3 is sufficiently large then $\bar{A}_5 = \bar{A}_4$.

We use each point from the set $\bar{A}_5$ as a starting point $\bar{y}$ for the l-th cluster center. That is we use more than one starting point to solve Problem (3.2.9). The best solution found is accepted as a solution to the l-partition problem in Step 5. Thus, an algorithm for finding starting points to solve Problem (3.2.9) can be summarized as follows.

Algorithm 2. Algorithm for finding the set of starting points.

Input: The solution $(x^1, \ldots, x^{l-1})$ to the $(l-1)$-clustering problem.

Output: The set of starting points for the l-th cluster center.

Step 0. (Initialization). Select $\gamma_1, \gamma_2 \in [0, 1]$ and $\gamma_3 \in [1, \infty)$.

Step 1. Compute z_{max}^1 using (4.1.2) and the set $\bar{A}_1$ using (4.1.3).

Step 2. Compute z^2_{max} using (4.1.4) and the set $\bar{A}_3$ using (4.1.5).

Step 3. Compute the set $\bar{A}_4$ of local minimizers of the auxiliary clustering problem (3.2.15) using starting points from the set $\bar{A}_3$.

Step 4. Compute $\bar{f}_l^{min}$ using (4.1.6) and the set $\bar{A}_5$ using (4.1.7). $\bar{A}_5$ is the set of starting points for the l-th cluster center.

4.2 Algorithm for reduction of data points

Most very large data sets contain dense areas and points in such areas are very close to each other. Contribution of these points to the objective function in the clustering problem is almost the same and therefore if two points are very close to each other one of these points can be removed by giving more weight to another point.

The vicinity of each point depends on the number of points in the data set and the number of attributes in this data set, that is, this vicinity is defined depending on the size of the data set and the user provided tolerance. The tolerance should be the same for all data sets.

Denote this tolerance by $\varepsilon > 0$.

Algorithm 3. Algorithm for finding key points.

Input: The data set A and tolerance $\varepsilon > 0$.

Output: The reduced data set $\bar{A}$.

Step 0. Set $A_0 = A$ and $\bar{A} = \emptyset$.

Step 1. Take any $a \in A_0$. Compute the set

$$B(a) = \left\{ b \in A_0 : \|a - b\|^2 \leq \varepsilon \right\}.$$

Step 2. Update the sets

$$A_0 = A_0 \setminus B(a), \quad \bar{A} = \bar{A} \cup \{a\}$$

and compute the weight c_a of a as: $c_a = |B(a)|$.

Step 3 (Stopping criterion). If $A_0 = \emptyset$ then stop. Otherwise go to Step 1.

Algorithm 3 allows one to identify dense areas in the data set A, replace it by its representative and compute the weight of this representative point. It is obvious

that for any $x = (x^1, \ldots, x^k) \in \mathbb{R}^{nk}$

$$f_k(x^1, \ldots, x_k) \leq \frac{1}{m} \sum_{a \in \bar{A}} c_a \min_{j=1,\ldots,k} \|x^j - a\|^2 \tag{4.2.1}$$

On the other side for each $a \in \bar{A}$ and for any $y \in \mathbb{R}^n$ we have

$$c_a \|a - y\|^2 \leq \sum_{b \in B(a)} \|b - y\|^2 + c_a \varepsilon. \tag{4.2.2}$$

In this case

$$f_k(x^1, \ldots, x_k) = \frac{1}{m} \sum_{a \in \bar{A}} \sum_{b \in B(a)} \min_{j=1,\ldots,k} \|x^j - b\|^2$$

Then it follows from (4.2.2) that

$$\sum_{a \in \bar{A}} c_a \min_{j=1,\ldots,k} \|x^j - a\|^2 \leq \sum_{a \in \bar{A}} \sum_{b \in B(a)} \min_{j=1,\ldots,k} \|x^j - b\|^2 + m\varepsilon$$

Here we note that $m = \sum_{a \in \bar{A}} c_a$. Thus, we have that for any $x = (x^1, \ldots, x^k) \in \mathbb{R}^{nk}$

$$\sum_{a \in \bar{A}} c_a \min_{j=1,\ldots,k} \|x^j - a\|^2 \leq f_k(x^1, \ldots, x^k) + \varepsilon. \tag{4.2.3}$$

Define the cluster function for the data set $\bar{A}$ as follows:

$$f_{ck}(x^1, \ldots, x^k) = \frac{1}{m} \sum_{a \in \bar{A}} c_a \min_{j=1,\ldots,k} \|x^j - a\|.$$

The k-clustering problem for the data set $\bar{A}$ is:

$$\text{minimize } f_{ck}(x) \text{ subject to } x = (x^1, \ldots, x^k) \in \mathbb{R}^{nk}, \tag{4.2.4}$$

Combining (4.2.1) with (4.2.3) leads to the following result:

$$f_k(x^1, \ldots, x^k) \leq f_{ck}(x^1, \ldots, x^k) \leq f_k(x^1, \ldots, x^k) + \varepsilon \tag{4.2.5}$$

for all $x = (x^1, \ldots, x^k) \in \mathbb{R}^{nk}$.

Let $x^* \in \mathbb{R}^{nk}$ be a solution to the k-th clustering problem (3.2.9) and x^{**} be a

solution to the problem (4.2.4). This means that

$$f_k(x^*) \leq f_k(x) \text{ for all } x \in \mathbb{R}^{nk},$$

$$f_{ck}(x^{**}) \leq f_{ck}(x) \text{ for all } x \in \mathbb{R}^{nk}.$$

Then it follows from (4.2.5) that

$$f_k(x^*) \leq f_k(x^{**}) \leq f_{ck}(x^{**}) \leq f_{ck}(x^*) \leq f_k(x^*) + \varepsilon.$$

Thus, we have

$$f_k(x^*) \leq f_{ck}(x^{**}) \leq f_k(x^*) + \varepsilon.$$

This means that accuracy of a clustering algorithm will depend on the tolerance $\varepsilon > 0$. As it was mentioned above this tolerance depends on the size of a data set (the number of data points and the number of attributes). The following simple procedure is proposed to define the tolerance ε.

First, we compute the centroid of the whole data set A and then compute the value $\hat{f}_1$ of the cluster function f_1. Define the tolerance ε as follows:

$$\varepsilon = \frac{n\hat{f}_1}{m}\varepsilon_0$$

where m is the number of data points, n is the number of attributes. Here ε_0 is a tolerance and it is the same for all data sets. We suggest to set $\varepsilon_0 = 10^{-6}$.

4.3 Reduction of computational effort

In this section we will discuss some approaches to reduce the amount of computations in the incremental clustering algorithms. Clustering algorithms use an affinity (or distance) matrix at each iteration. Even in moderately large data sets this matrix cannot be stored in memory and therefore should be repeatedly computed at each iteration. This makes clustering algorithms very time consuming. An incremental approach provides opportunity to decrease the computational effort in such algorithms and to avoid to compute the whole affinity matrix at each iteration [81, 104, 110].

We consider two schemes to reduce the amount of computational effort. Both schemes are applicable to incremental algorithms. We suggest to use the distances

between data points and cluster centers instead of the affinity matrix. Since the number of clusters is significantly less than the number of data points the former matrix is much smaller than the latter one. We describe these schemes for the data set A and they can be described for the reduced data set $\bar{A}$ in the same way.

Assume that for given $k \geq 2$ the solution $x^1, \ldots, x^{k-1}$ to the $(k-1)$-clustering problem is known. Let $v_{il} = \|x^i - a^l\|^2$ be the squared distance between the data point a^i, $i = 1, \ldots, m$ and the cluster center x^l, $l = 1, \ldots, k-1$. Then we can consider an $m \times (k-1)$ matrix $V_{k-1} = (v_{il})$, $i = 1, \ldots, m$, $l = 1, \ldots, (k-1)$. We also consider the vector $D_{k-1} = (d^1_{k-1}, \ldots, d^m_{k-1})$ of m components where $k-1$ is the squared distance between the data point a^i and its cluster center in the $k-1$-partition. Note that the matrix V_{k-1} and the vector D_{k-1} are available after the $k-1$-th iteration [81, 104].

The first approach to reduce the computational effort is simple. Let a data point $a^i \in A$ be given and $x^{l(i)}$ is its cluster center. Here $l(j) \in (1, \ldots, k-1)$. For a given u and data point a^i if

$$v_{il}(j) \geq (1 + \frac{1}{\sqrt{(u)}})^2 d^j_{k-1}. \tag{4.3.1}$$

$$\|a^i - a^j\| \geq \|a^i - x^{l(j)}\| - \|a^j - x^{l(j)}\| \geq \frac{1}{\sqrt{(u)}} \|a^j - x^{l(j)}\|. \tag{4.3.2}$$

$\|a^i - a^j\|^2 \geq d^j_{k-1}$ and therefore $a^j \notin S^u(a^i)$. This condition allows us to reduce the number of pairwise distance computations.This reduction becomes substantial as the number of clusters increases We introduce the following set

$$R^u(a^i) = a^j \in A : v_{il}(j) < (1 + \frac{1}{\sqrt{(u)}})^2 d^j_{k-1} \tag{4.3.3}$$

Results described above show that $S^u(a^i) \subset R^u(a^i)$. Then we can use the set $R^u(a^i)$ instead of the set A to compute the value of the function $\bar{f}^u_k$ in Step 2 of Algorithm 1. In this case we may not get the exact value of this function, however, it gives a good approximation to the exact value. Moreover, one can take

$$W \in (1, (1 + \frac{1}{\sqrt{(u)}})^2]. \tag{4.3.4}$$

Consider the set $R^u(a^i) = a^j \in A : \|a^i - a^j\|^2 < W d^j_{k-1}$ and then replace A by $R^u(a^i)$ for the computation of the function $\bar{f}^u_k$. This will further reduce the amount

of computational effort in Step 2 of Algorithm 1. In numerical experiments in the next section we take $w = 1.5$.

The second approach is based on the fact that data points which are very close to previous cluster centers cannot be considered as candidates to be starting cluster center to minimize the auxiliary function. At the $k-1$-th iteration, we compute a squared averaged radius of each cluster A^l, $l = 1,\ldots,k-1$:

$$r_l^2 = \frac{1}{|A^l|}\sum_{a\in A^l}\|x^l - a\|^2. \tag{4.3.5}$$

and a squared maximum radius $r_l^{max} = \max_{a\in A^l}\|x^l - a\|^2$. Here $|\cdot|$ is the cardinality of a set. Consider the following numbers

$$r_l^{max} = \frac{r_l^{max}}{|r_l^2|} \geq 1, \beta = \varepsilon(\alpha - 1), \varepsilon = 0.001, r_{lk} = 1 + \beta_l(k-1), l = 1,\ldots,(k-1). \tag{4.3.6}$$

It is clear that $\gamma_{lk} \geq 1$ where as: $l = 1,\ldots,(k-1)$: We consider the following subset of the cluster A^l:

$$\bar{A}^l = a \in A^l : \|x^l - a\|^2 \geq \gamma_{lk} r_l^2 \tag{4.3.7}$$

In other words we remove from the cluster A_l all points for which $\|x^l - a\|^2 < \gamma_l r_l^2$. In incremental approach the clusters are becoming more stable as their number k increases. Therefore we also increase the numbers γ_{lk} as k increases. Consider the set

$$\bar{A} = \cup_{l=1,\ldots,(k-1)}\bar{A}^l \tag{4.3.8}$$

Only data points $a \in |barA$ are considered as the candidates to be starting points for minimizing the auxiliary function $\bar{f}_k$. One can see that the use of the above described approaches allow us to avoid the computation of the whole affinity matrix. Thus, Steps 2 and 3 of Algorithm 1 can be rewritten as follows: [81, 104, 110].

Step 2u. For each $a \in P(u(t))) \cap \bar{A}$ compute the set $S^{u(t)}(a)$, its center c and the value $\bar{f}_{(k,a)}^{u(t)} = \bar{f}_k^{u(t)}(c)$ of the function $\bar{f}_k^{u(t)}$ at the point c over the set $R_W^u(a^i)$ [81, 104, 110]. Step 3u. Compute

$$\bar{f}_{(k,min)}^{u(t)} = \min_{a\in P(u(t)))\cap A}\bar{f}_{(k,a)}^{u(t)}(\bar{a}) = \operatorname*{argmin}_{a\in P(u(t)))\cap A}\bar{f}_{(k,a)}^{u(t)}, \tag{4.3.9}$$

[81, 93, 110, 111]. the corresponding center c and the set $S^{u(t)}(c)$, knowing the

solution for the $(k-1)$-partition problem. In order to do so, first, we compute a starting point for the k-th cluster center (Step 2 of Algorithm 2). This is done by minimizing the auxiliary cluster function. Since this function is nonconvex and it has many local solutions we use several starting points, scattered over the data set, to get its global or near global minimizer (unlike in the modified global k-means algorithm where one starting point is used). We achieve this by introducing the parameter u in the auxiliary function as in [81]. For different values of the parameter u we select the data point which attracts the cluster providing the lowest value for the function [81, 104, 110]. (Step 2u of Algorithm 1) [81, 110, 111]

In order to compute the cluster attracted by a given data point one should compute the whole affinity matrix. Here we apply the first scheme to reduce the computational effort by excluding data points which lie nearby cluster centers from the $(k-1)$-partition. This allows us to exclude these data points from: (a) the list of points which can attract large clusters and (b) the list of points which can be attracted by non-excluded data points. In data sets with good cluster structure this allows us to significantly reduce the number of candidates for starting points and the number of data points which can be attracted by a given candidate. This means that using the incremental nature of the algorithm we avoid any computation of the affinity matrix.

For each value of the parameter u we obtain a starting point for solving problem. Again when solving this problem we exclude from our computations those data points which are nearby their cluster centers. As a result we get a set of local minimizers of problem (Steps 3u and 4 of Algorithm 1). We evaluate the auxiliary cluster function at these local minimizers still excluding data points nearby their cluster centers from our computations. Then we choose the best local minimizer as a starting point for the k-th cluster center (Step 6 of Algorithm 1). We combine this starting point for the k-th cluster center with the $k-1$ centers from the previous iteration to form a starting point for solving the k-partition problem. Then we apply the k-means algorithm starting from this point and get k-partition of the data set (Step 3 of Algorithm 2). In this step we apply the triangle inequalities for distances to reduce the computational effort used by the k-means algorithm. This makes it necessary to store in the memory the matrix which contains distances between data points and cluster centers from the $(k-1)$-st iteration. This matrix is much smaller than the affinity matrix.

4.4 An incremental clustering algorithm

In this section we present an incremental algorithm for solving Problem (3.2.9). This problem is a global optimization problem and it may have many local solutions, however only global solutions or solutions close to global ones provide the best cluster distribution of a data set with the least number of clusters. Clustering in very large data sets is out of reach of conventional global optimization methods. Therefore, clustering algorithms which can get at least near global solutions are of interest. To be efficient such algorithms should employ local search optimization methods and involve special procedures to generate good starting points.

The incremental approach allows one to design such algorithms. It should be noted that there are two types of incremental algorithms for clustering. Algorithms of the first type use the data incrementally while algorithms of the second type compute clusters incrementally. We consider the second type of the incremental algorithms. These algorithms compute clusters incrementally starting from one cluster which is the whole data set and gradually adds one cluster at each iteration.

The use of the incremental approach allows one to design efficient procedures for generating starting cluster centers. Cluster centers from the previous iteration are considered as good candidates to be starting cluster centers at the current iteration. Next, we design a quite general incremental algorithm for solving clustering problems.

Algorithm 4. An incremental clustering algorithm.

Input: The data set $A = \{a^1, \ldots, a^m\}$.

Output: The set of k cluster centers $\{x^1, \ldots, x^k\}$.

Step 1. (Initialization). Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Set $l := 1$.

Step 2. (Stopping criterion). Set $l := l + 1$. If $l > k$ then stop. The k-partition problem has been solved.

Step 3. (Computation a set of starting points for the next cluster center). Apply Algorithm 2 to compute the set $\bar{A}_5$ defined by (4.1.7).

Step 4. (Computation a set of cluster centers). For each $\bar{y} \in \bar{A}_5$ take $(x^1, \ldots, x^{l-1}, \bar{y})$ as a starting point, solve Problem (3.2.9) and get a solution $(\hat{y}^1, \ldots, \hat{y}^l)$. Denote by $\bar{A}_6$ a set of all such solutions.

Step 5. (Computation of the best solution). Compute

$$f_l^{min} = \min\left\{f_l(\hat{y}^1,\ldots,\hat{y}^l):\ (\hat{y}^1,\ldots,\hat{y}^l) \in \bar{A}_6\right\}$$

and the collection of cluster centers $(\bar{y}^1,\ldots,\bar{y}^l)$ such that

$$f_l(\bar{y}^1,\ldots,\bar{y}^l) = f_l^{min}.$$

Step 6. (Solution to the l-partition problem). Set $x^j := \bar{y}^j,\ j = 1,\ldots,l$ as a solution to the l-th partition problem and go to Step 2.

One can see that Algorithm 6 in addition to the k-partition problem solves also all intermediate l-partition problems where $l = 1,\ldots,k-1$. Steps 3 and 4 are the most important steps of this algorithm. The success of the algorithm heavily depends on Step 3. In these two steps one solve optimization problems: the auxiliary clustering problem and clustering problem. Optimization methods (local search or global search) and heuristic algorithms such as the k-means algorithm can be applied to solve these problem. In next section we consider two such algorithms: one algorithm is based on the heuristic algorithms and another algorithm is based on the optimization technique. We will describe these algorithms for a data set whose points have weights.

4.5 Hyperbolic smoothing of cluster functions

In this section we define the hyperbolic smoothing of the cluster function f_k and the auxiliary cluster function $\bar{f}_k$. We start with the brief definition of hyperbolic smoothing functions. Consider the following maximum function:

$$\varphi(x) = \max\{0,x\},\ x \in \mathbb{R}. \tag{4.5.1}$$

The hyperbolic smoothing function approximating the function (4.5.1) can be defined as:

$$\phi_\tau(x) = \frac{x+\sqrt{x^2+\tau^2}}{2}. \tag{4.5.2}$$

Here $\tau > 0$ is called a *precision* or *smoothing* parameter. The hyperbolic smoothing functions and their properties were studied in [4, 100, 101]. In particular, the

following proposition is true.

Proposition 1. *The function $\phi_\tau(x)$ has the following properties:*

1. *$\phi_\tau(\cdot)$ is an increasing convex C^∞ function;*
2. *$\varphi(x) < \phi_\tau(x) \leq \varphi(x) + \frac{\tau}{2}$, $\forall \tau > 0$.*

The hyperbolic smoothing functions for more general maximum functions were studied in [9, 105]. Consider the following maximum function:

$$\psi(x) = \max_{j=1,\ldots,p} \psi_j(x)$$

where $p \geq 1$ and functions $\psi_j,\ j = 1,\ldots,p$ are continuously differentiable. We introduce the following function using functions $\psi_j,\ j = 1,\ldots,p$ and an additional variable $t \in \mathbb{R}$:

$$\Psi(x,t) = t + \sum_{j=1}^{p} \max(0, \psi_j(x) - t).$$

It is clear that

$$\psi(x) = \Psi(x, \psi(x)). \tag{4.5.3}$$

Applying (4.5.2) we can write the hyperbolic smoothing function $\bar{F}_\tau(x,t)$ for the function Ψ as

$$\bar{F}_\tau(x,t) = t + \frac{1}{2} \sum_{j=1}^{p} \left(\psi_j(x) - t + \sqrt{(\psi_j(x) - t)^2 + \tau^2} \right).$$

Then using (4.5.3) one can write the hyperbolic smoothing function $F_\tau(x,t)$ for the maximum function ψ as follows:

$$F_\tau(x,t) = t + \frac{1}{2} \sum_{j=1}^{p} \left(\psi_j(x) - t + \sqrt{(\psi_j(x) - t)^2 + \tau^2} \right) \tag{4.5.4}$$

where $t = \psi(x)$.

Hyperbolic smoothing for minimum functions can be defined in a similar way. For the function

$$\bar{\varphi}(x) = \min\{0, x\},\ x \in \mathbb{R}$$

we have

$$\bar{\varphi}(x) = -\max\{0, -x\}$$

and therefore the hyperbolic smoothing function $\bar{\Phi}_\tau$ for $\bar{\varphi}$ is as follows:

$$\bar{\Phi}_\tau(x) = \frac{1}{2}\left(x - \sqrt{x^2+\tau^2}\right),\ x \in \mathbb{R}. \tag{4.5.5}$$

Now consider the following minimum function

$$\theta(x) = \min_{j=1,\ldots,p} \theta_j(x)$$

where $p \geq 1$ and functions $\theta_j,\ j = 1,\ldots,p$ are continuously differentiable. Then

$$\theta(x) = -\max_{j=1,\ldots,m} -\theta_j(x)$$

and applying (4.5.4) we get that the hyperbolic smoothing function $Q_\tau(x,t)$ for the function θ as:

$$Q_\tau(x,t) = t + \frac{1}{2}\sum_{j=1}^{p}\left(\theta_j(x) - t - \sqrt{(\theta_j(x)-t)^2+\tau^2}\right). \tag{4.5.6}$$

Here

$$t = -\max_{j=1,\ldots,p} -\theta_j(x) = \min_{j=1,\ldots,p} \theta_j(x).$$

Now we can define hyperbolic smoothing functions to both the clustering function f_k and the auxiliary cluster function $\bar{f}_k$. We start with the function $\bar{f}_k$. Since

$$\bar{f}_k(y) = \frac{1}{m}\left[\sum_{i=1}^{m} d_{k-1}^i + \sum_{i=1}^{m} \min(0, \|y-a^i\|^2 - d_{k-1}^i)\right]$$

applying (4.5.5) we get that the hyperbolic smoothing function $U_{k,\tau}$ to $\bar{f}_k$ is:

$$U_{k,\tau}(y) = \frac{1}{2m}\sum_{i=1}^{m}\left(d_{k-1}^i + \|y-a^i\|^2 - \sqrt{(\|y-a^i\|^2 - d_{k-1}^i)^2+\tau^2}\right),\ y \in \mathbb{R}^n.$$

In order to define the hyperbolic smoothing function for f_k consider the following functions:

$$v_i(x^1,\ldots,x^k) = \min_{j=1,\ldots,k} \|x^j - a^i\|^2,\ i = 1,\ldots,m.$$

Applying (4.5.6) we get the following smoothing function for v_i

$$Q_\tau^i(x^1,\ldots,x^k,t_i) = t_i + \frac{1}{2}\sum_{j=1}^{k}\left(\|x^j - a^i\|^2 - t_i - \sqrt{(\|x^j - a^i\|^2 - t_i)^2 + \tau^2}\right)$$

with

$$t_i = \min_{j=1,\ldots,k} \|x^j - a^i\|^2. \tag{4.5.7}$$

Then it is easy to formulate the smoothing function $V_{k,\tau}$ for the function f_k:

$$V_{k,\tau}(x^1,\ldots,x^k,t) = \frac{1}{m}\sum_{i=1}^{m}\left[t_i + \frac{1}{2}\sum_{j=1}^{k}\left(\|x^j - a^i\|^2 - t_i - \sqrt{(\|x^j - a^i\|^2 - t_i)^2 + \tau^2}\right)\right]$$

where $t = (t_1,\ldots,t_m)$ and $t_i,\ i = 1,\ldots,m$ is defined by (4.5.7).

Take any sequence $\{\tau_p\}$ such that $\tau_p \downarrow 0$ as $k \to \infty$. Then Problem (3.2.9) can be replaced by the sequence of smooth problems as follows (see [9]):

$$\text{minimize } V_{k,\tau_p}(x^1,\ldots,x^k,t) \text{ subject to } x^1,\ldots,x^k \in \mathbb{R}^{n\times k}. \tag{4.5.8}$$

Similarly, Problem (3.2.15) can be replaced by the sequence of the following smooth problems:

$$\text{minimize } U_{k,\tau_p}(y) \text{ subject to } y \in \mathbb{R}^n. \tag{4.5.9}$$

The convergence of the hyperbolic smoothing method was studied in [9, 11].

4.5.1 Computational complexity of the fast modied global k-means algorithm

In order to select an initial point for the next cluster center in the modied global k-means algorithm one needs $O(m^2 2 + mt)$ distance calculations. Here t is the number of iterations by Algorithm 1. Then the estimation for the total number of distance calculations at the k-th iteration of Algorithm 2 is $O(mkT + m^2 + mt)$. Here T is the number of iterations by Algorithm 2. Thus, in order to compute k cluster centers the modied global k-means algorithm requires $O(mkT + km + kmt)$ distance calculations. To get k cluster centers the fast modied global k-means algorithm(without complexity reduction schemes) requires O(p(mk2T + km2 + kmt)) distance calculations. Here p is the cardinality of the set U [81, 93, 110, 111] Steps

2u and 3u require $O(m1+m1t)$ distance calculations. Here m1 is the number of data points in the set $m1 \in P(u(t))) \cap A$ and $m1 \leqslant m$, m becomes smaller and smaller as the number of clusters increases. Therefore the fast modied global k-means algorithm, with Steps 2 and 3 in Algorithm 1 replaced by Steps 2u and 3u, requires $O(p(mk^2T + km1 + km1t))$ distance calculations [81, 93, 110]. The fast global k-means algorithm from [81, 93, 110] requires $O(mkT + km)$ and the fast global k-means clustering algorithm from [81,104,110] requires $O(mm^2k+mkk1+mn)$ distance calculations to generate k cluster centers. Here $m2 \leq m$, $k1 \leq k$ and n is the number of attributes. Comparing with other global k-means algorithms and taking into account that the number p is small (in most cases $p \leqslant 5$) and $m1 \leqslant m$, we can see that the fast modied global k-means algorithm with complexity reduction schemes requires less computational effort than the global and modied global k-means algorithms. The fast global k-means algorithm has similar computational complexity with the algorithm from [81, 104, 110].

4.5.2 Incremental Algorithms for fast modified global k-means

The existing incremental algorithms in cluster analysis can be divided into the following classes:

1. Single pass incremental algorithms: Algorithms in which new data points are added at each iteration and cluster centers are improved accordingly [81, 104, 110].

2. Fast Global Incremental Algorithms: Algorithms where clusters are built incrementally adding one cluster center at a time [81, 104, 110].

In this section, a new version of the revised global k-means algorithm is suggested. The algorithms described in this section belong to the second class. These algorithms are not directly applicable to solve clustering problems in vast data sets. In order to solve k- partition clustering problem these algorithms start from one cluster center (centroid of the dataset) and compute cluster centers incrementally adding a new center at each iteration. However, this algorithm uses signicantly more computational effort than other incremental algorithms.

In the new version we reduce the amount of computational effort by:

1. Removing data points which are close to cluster centers found in the previous iteration. Thus we use the whole dataset only at the first iteration when we compute the centroid of the data set [81, 110].

2. Using the triangle inequality for distances to avoid unnecessary computations [81, 110].

3. Introducing a scheme to generate starting points from different parts of the data set to minimize the auxiliary function [81, 110].

4. Applying k-means algorithm,starting from these points to minimize the auxiliary cluster function and the best solution is selected as a starting point for the next cluster center [81, 104, 105].

The proposed algorithm is applicable to only data sets with numeric attributes. Clustering algorithms for categorical data sets can be found, for example,in [81, 109, 110]. These algorithms are capable of getting either global or near global solutions for clustering problems in many data sets.

4.6 A smooth incremental clustering algorithm

In this section we describe an incremental algorithm for solving Problem (3.2.9). Incremental clustering algorithms build clusters dynamically adding one cluster center at a time. The smooth incremental clustering algorithm for finding the k-partition of the set A proceeds as follows:

Algorithm 5. A smooth incremental algorithm for clustering.

Step 1. (Initialization). Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Set $l := 1$.

Step 2. (Stopping criterion). Set $l := l + 1$. If $l > k$ then stop. The k-partition problem has been solved.

Step 3. (Computation of the next cluster center). Select any starting point $y^0 \in \mathbb{R}^n$ and solve Problem (4.5.9). Let $\bar{y} \in \mathbb{R}^n$ be a solution to this problem.

Step 4. (Refinement of all cluster centers). Take $(x^1, \ldots, x^{l-1}, \bar{y})$ as a starting point and solve Problem (4.5.8). Let $(y^1, \ldots, y^k)$ be a solution to this problem.

Step 5. (Solution to the l-partition problem). Set $x^j := y^j,\ j = 1, \ldots, l$ as a solution to the l-th partition problem and go to Step 2.

An algorithm to get the starting point for the fast modified global k-means algorithm for clustering.

Step 1. Set $t := 0$.

Step 2. Set $t := t + 1$. If $t > p$. then go to step 6. Otherwise $u(t) \in \cup$

Step 3. For each $a \in P(u(t))) \cap \bar{A}$ compute the set $S^{u(t)}(a)$, its center c and the value $\bar{f}^{u(t)}_{(k,a)} = \bar{f}^{u(t)}_{k}(c)$ of the function $\bar{f}^{u(t)}_{k}$ at the point c over the set $R^{u}_{W}(a^i)$.

*Step 4.*Compute

$$\bar{f}^{u(t)}_{(k,min)} = \min_{a \in P(u(t)))\cap A} \bar{f}^{u(t)}_{(k,a)}(\bar{a}) = \operatorname*{argmin}_{a \in P(u(t)))\cap A} \bar{f}^{u(t)}_{(k,a)}, \tag{4.5.10}$$

the corresponding center c and the set $S^{u(t)}(c)$.

Step 5. Recompute the set $S^{u(t)}(c)$ and its center until no more data points escape or return to this set. Let $\bar{c}(u(t))$ be the nal value for the center c. Compute the value $\bar{f}_{(k,t)}$ of the auxiliary function $\bar{f}_k$ at the point $\bar{c}(u(t))$.

Step 6. Go to Step 1.

Step 7. Compute $\bar{f}_{(k,min)} = \min_{t \in 1,\ldots,p} \bar{f}_{(k,t)}$
and $t_0 \in (1,\ldots,p)$ such that $\bar{f}_{(k,t_0)} = \bar{f}_{(k,min)}$. Set $\bar{c}_f = \bar{c}(u(t_0)).\bar{c}_f$ is a starting point for the k-th cluster center.

A fast modified global k-means algorithm for clustering.

Step 1. (Initialization).Select a tolerance $\varepsilon > 0$.

Step 2. (Initialization).Compute the center $x^1 \in \mathbb{R}^n$ of the set A of the set A. Let f^1 be the corresponding value of the objective function. Set $k = 1$.

Step 3. (Computation of the next cluster center). Set $k = k + 1$.Let $x^1, ..., x^k - 1$ be the cluster centers for $(k-1)$-partition problem. Apply the Algorithm-1 to get a starting point $\bar{y} \in \mathbb{R}^n$ for the k-th cluster center.

Step 4.(Refinement of all cluster centers). Take $(x^1, ..., x^k - 1, \bar{(y)})$ as a new starting point, apply algorithm to solve k-partition problem. Let $(y^1, ..., y^k)$, be a solution to this problem and f^k be the corresponding value of the objective function.

Step 5. (Computation of the best solution) Keep the best k partition(objective function) obtained and its centers $x^1, ..., x^k - 1$.

Step 6. (Stopping criterion). If $\frac{f^k - 1 f^k}{f^1} < \varepsilon$,then stop, otherwise set $x^i = y^i$ and go to Step 2.

Remark 2. One can see that Algorithm 5 in addition to the k-partition problem solves also all intermediate l-partition problems where $l = 1, \ldots, k-1$. Steps 3 and 4 are the most important steps of Algorithm 5. Both (4.5.8) and (4.5.9) are global optimization problems. The success of an incremental clustering algorithm heavily depends on Step 3. The right choice of the starting point for the l-th cluster center in Step 3 may lead to the finding of the global or near global minimizers of the clustering function in Step 4. In the next section we will design an algorithm for finding such starting points.

Remark 3. Both (4.5.8) and (4.5.9) are smooth optimization problems and one can use local search algorithms of smooth optimization to solve them. Therefore we call this algorithm the smooth incremental clustering algorithm. In our numerical experiments, we apply the quasi-Newton method with the BFGS updates to solve problems (4.5.8) and (4.5.9).

4.7 A modified smooth incremental clustering algorithm

Now we can modify Algorithm 5 applying Algorithm 2 in Step 3.

Algorithm 6. A modified smooth incremental algorithm for clustering.

Step 1. (Initialization). Compute the center $x^1 \in \mathbb{R}^n$ of the set A. Set $l := 1$.

Step 2. (Stopping criterion). Set $l := l + 1$. If $l > k$ then stop. The k-partition problem has been solved.

Step 3. (Computation a set of starting points for the next cluster center). Apply Algorithm 2 to compute the set $\bar{A}_5$ defined by (4.1.7).

Step 4. (Computation a set of cluster centers). For each $\bar{y} \in \bar{A}_5$ take $(x^1, \ldots, x^{l-1}, \bar{y})$ as a starting point and solve Problem (4.5.8). Let $(\hat{y}^1, \ldots, \hat{y}^l)$ be a solution to this problem. Denote by $\bar{A}_6$ a set of all such solutions.

Step 5. (Computation of the best solution). Compute

$$f_l^{min} = \min \left\{ f_l(\hat{y}^1, \ldots, \hat{y}^l) : \ (\hat{y}^1, \ldots, \hat{y}^l) \in \bar{A}_6 \right\}$$

and the collection of cluster centers $(\bar{y}^1, \ldots, \bar{y}^l)$ such that

$$f_l(\bar{y}^1, \ldots, \bar{y}^l) = f_l^{min}$$

Step 6. (Solution to the l-partition problem). Set $x^j := \bar{y}^j,\ j = 1, \ldots, l$ as a solution to the l-th partition problem and go to Step 2.

Chapter 5

Computational results: small data sets

In this chapter we present and discuss computational results using small size data sets. All data sets contain only numeric attributes and they do not contain missing values. First, we give a brief description of data sets, then present results. These results include optimal values of the cluster function obtained by each algorithm and CPU time required by them. The following algorithms are used for comparison: the global k-means algorithm (GKM), the multi-start modified global k-means algorithm (MS-MGKM), the multi-start k-means algorithm (MS-KM), the difference of convex clustering algorithm (DCA), the clustering algorithm based on the difference of convex representation of the cluster function and nonsmooth optimization (DCClust) and two algorithms proposed in this thesis: the fast multi-start modified global k-means algorithm without weights (FMS-MGKM2) and with weights (FMS-MGKM). The description of these algorithms can be found in Chapter 4.

The number of starting points in the MS-KM algorithm is set to 500. Algorithms MS-MGKM, DCA and DCClust use the algorithm for computation of starting cluster centers described in the previous chapter. The implementation of FMS-MGKM and FMS-MGKM2 algorithms was also discussed in the previous chapter. CPU time in all tables are in seconds. In all data sets up to 25 clusters are computed.

5.1 Data sets

The brief description of small data sets is given in Table 5.1. In this table we include the number of instances (m, the number of attributes (n) and the total number of entries (N_e) for each data set. The number of instances in these data sets is less than ten thousand and the number of attributes is ranging from 5 to 41.

Table 5.1: Small size data sets

No.	Data sets	m	n	N_e
1	Wilt	4839	5	24195
2	Wine Quality	4898	12	58776
3	Waveform Generator	5000	41	205000
4	Turkiye Student Evaluation	5820	28	162960
5	Drug data sets yprop 41	8885	22	195470
6	Combined Cycle Power Plant	9568	4	38272
7	Gesture Phase Segmentation	9900	18	178200

- Wilt is a data set which contains information of Root-wilt-disease (RWD), caused by phytoplasma, in Pine trees, in Jiangsu Province and collected by Remote sensing. This remote sensing study is using a multiscale object-based classification method for detecting diseased trees in high-resolution multi spectral satellite imagery. This data set has six attributes but only five attributes contributes to clustering as one attribute is categorical [13, 83].

- Wine Quality Data Set contains the information of two data sets, related to red and white vinho verde wine samples, from the north of Portugal. The aim is to model physicochemical tests [14, 83] to predict wine quality based. The eleven attributes which take part in physicochemical tests are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

- Waveform Database Generator data-set contains information of three classes of noise waves generated , each of which class wave is generated from a combination of 2 of 3 “base” waves. Each instance is generated frequency added noise (mean 0 and variance 1) in each attribute. This data set has 41

attributes but only 40 attributes take part in clustering experiments, as one attribute is categorical [15, 83].

- Turkiye Student Evaluation data-set consists of evaluation scores made available by students of Gazi University in Ankara (Turkey). There are total 33 attributes but 28 are of course specific questions and additional 5 attributes are about named, instructor ID, class room number, number of repeat, attendance and number of difficulties. These additional are removed before clustering as these are categorical [16, 83].

- Drug data set "yes on Proposition 41" is abbreviated as yprop-41, is a cancer drug data set which gets a striking difference in the behavior of cancer-drug targets as compared with targets of non-cancer drugs, it has 267 attributes, all binary and categorical values are removed before data clustering [6].

- Combined Cycle Power Plant data set contains data points gathered from a Combined Cycle Power Plant (when the plant was tuned to work with full load), over 6 years (2006-2011). It has five attributes, hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. The attribute related to the output is removed before the clustering experiments on plant input variable data sets [83].

- Gesture Phase Segmentation is a data set which is made up of features extracted from 7 videos with people suffering from gesticulating problems, the focus in this study is at Gesture Phase Segmentation. This data set contains 50 attributes divided into two files: 18 raw files and 32 processed files. We used only attributes with numeric values of the raw files. The input file data is extracted from 18 XML files which have eighteen features [21, 83].

5.2 Results

Results for small data sets are given in Tables 5.2-5.15.The best results in all the tables are highlighted using the bold font. Results for the Wilt data set presented in Tables 5.2 and 5.3 show that the proposed algorithms almost reach the best results obtained other algorithms, however, CPU time used by these algorithms is less than

Table 5.2: Results with Wilt data set: Cluster function values, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	$FMS-MGKM2$
2	**78554702.33**	78554709.00	**78554702.33**	**78554702.33**	78554702.68	78554744.95	78555651.30
3	59442307.84	59443256.00	59443242.92	**59077391.41**	59443266.72	59442349.22	59443351.15
5	33381076.98	33381076.00	33380360.93	**32680810.26**	33380388.32	33380787.51	33380655.41
7	22137549.10	22138268.00	22137315.23	**21751701.08**	22137345.77	22137564.46	22138055.82
10	14030481.79	14037919.00	**14030458.96**	15258276.19	14073744.17	14030571.59	14073364.05
12	11314144.17	11312393.00	**11312331.46**	13223502.32	11314145.25	11312441.47	11314573.84
15	9173076.52	**9172478.00**	9182731.42	10841596.98	9172492.64	9172525.66	9185524.01
17	**8115173.80**	8124023.00	8115119.56	9960732.14	8115187.48	8115241.14	8128690.80
20	**6998566.24**	7025471.00	7003329.45	9143588.30	7004011.21	7002936.48	7012471.99
22	6455609.47	6464891.00	6451519.29	7370285.87	6455617.36	6451857.14	**6432172.46**
25	5756465.36	5802688.00	**5755493.87**	5957685.96	5756478.70	5771804.81	5780406.52

Table 5.3: Results for Wilt data set: CPU time in seconds, (this Table corresponds to the Figure 5.1), (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	3.07	1.34	0.81	6.19	**0.59**	0.91	1.41
3	4.20	2.45	1.68	8.27	**1.47**	1.75	2.47
5	7.69	4.77	5.02	15.30	**3.77**	4.44	5.44
7	9.88	6.83	8.60	19.31	**5.55**	6.11	6.97
10	15.37	**10.67**	32.03	31.08	11.72	11.67	12.53
12	19.08	**13.13**	50.78	38.24	21.50	16.55	19.20
15	21.95	**16.23**	63.32	44.04	30.77	18.98	21.83
17	28.39	**18.25**	108.08	57.03	41.69	23.48	24.80
20	31.64	**21.09**	147.31	63.35	71.08	26.28	27.67
22	33.43	**23.23**	179.76	67.13	75.88	28.58	30.52
25	37.25	**26.16**	235.17	75.40	111.20	31.92	33.42

that used by other algorithms (except the GKM algorithm). There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Results for the Wine Quality data set are presented in Tables 5.4 and 5.5. These results demonstrate that the proposed algorithms are not as accurate as some other algorithms, however, their results quite close to the best results obtained by other algorithms. The proposed algorithms require less CPU time than other algorithms (except the GKM algorithm).It is true that in some data sets GKM is faster than the proposed algorithms, however all these data sets are small data sets. In large data sets especially in very large data sets, GKM is not comparable with the proposed algorithms both in the sense of accuracy and efficiency. The algorithm FMS-MGKM2 is more accurate and uses less CPU time than the FMS-MGKM algorithm in this

Table 5.4: Results for Wine Quality data set: Cluster function values, (The best results are highlighted).

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	4169863.00	4169864.00	4169863.00	**4169862.95**	4169863.12	4173774.50	4173011.96
3	2748617.00	2748617.00	2748617.00	**2748616.89**	2748617.48	2751017.69	2751778.45
5	1753079.00	1753105.00	**1753071.00**	1753079.21	1753079.50	1757998.82	1757365.87
7	1379225.00	1380989.00	**1379222.00**	1379224.93	1379224.97	1387195.04	1385208.72
10	1043490.00	1047523.00	**1042472.00**	1064727.29	1042527.88	1047497.41	1044720.46
12	902114.70	908470.10	**901751.10**	940903.15	902021.51	914542.43	918006.52
15	748865.00	**748590.80**	749185.50	762507.19	750167.62	763623.17	755429.79
17	675497.90	674520.70	**672146.90**	690590.25	675530.36	686225.49	676926.77
20	**585058.10**	588463.60	584682.30	607520.21	584614.53	593040.49	586838.73
22	540633.40	540511.00	**540222.00**	569023.90	541354.81	549616.03	542023.25
25	488982.30	489235.90	**486233.20**	523750.07	488107.33	496960.96	492118.96

Table 5.5: Results for Wine Quality data set: CPU time in seconds,(this Table corresponds to the Figure 5.3), (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	$FMS-MGKM2$
2	2.03	1.88	2.16	4.07	1.33	**1.14**	1.77
3	4.67	3.55	5.31	9.19	2.86	**2.39**	3.22
5	12.08	7.31	20.89	25.16	7.83	**6.56**	9.14
7	17.22	10.84	43.95	34.15	17.25	**10.20**	11.58
10	25.16	**16.28**	89.67	50.05	36.55	16.89	16.44
12	29.36	**19.30**	136.83	58.34	65.09	21.06	20.20
15	34.06	**24.25**	220.09	68.67	99.47	29.73	24.72
17	36.92	27.30	263.31	73.34	109.47	33.36	**26.59**
20	43.55	**32.13**	520.13	87.66	159.78	36.77	34.20
22	49.55	**35.56**	742.44	99.75	199.59	41.47	38.00
25	54.11	**40.38**	944.05	109.34	252.63	49.44	45.83

data set.

Results for the Waveform Generator data set presented in Tables 5.6 and 5.7. These results show that the proposed algorithms are more accurate than other algorithms in this data set and they use less CPU time. There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Results for the Turkiye Student Evaluation data set, given in Tables 5.8 and 5.9, show that the proposed algorithms fail to produce accurate results in this data set, however, they require significantly less CPU time than other algorithms. The FMS-MGKM2 algorithm is more accurate and requires less CPU time than the FMS-MGKM algorithm.

Tables 5.10 and 5.11 contains results on the Drug yprop 41 data set. These

Table 5.6: Results for Waveform Generator data set: Cluster function values, (The best results are highlighted).

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	2.64E+05	2.64E+05	2.64E+05	**235000**	2.61E+05	2.61E+05	2.61E+05
3	2.30E+05	2.30E+05	2.30E+05	**227591.6**	2.28E+05	2.28E+05	2.28E+05
5	2.13E+05	2.13E+05	2.13E+05	2.11E+05	2.11E+05	**211241.61**	2.11E+05
7	2.05E+05	2.06E+05	2.05E+05	2.04E+05	2.04E+05	**204063.67**	2.04E+05
10	2.01E+05	2.01E+05	2.00E+05	199462.48	**199460.15**	1.99E+05	1.99E+05
12	1.99E+05	1.99E+05	1.99E+05	1.98E+05	1.98E+05	**197844.09**	1.98E+05
15	1.97E+05	1.98E+05	1.97E+05	1.98E+05	1.96E+05	**195940.42**	1.96E+05
17	1.96E+05	1.98E+05	1.96E+05	1.97E+05	1.95E+05	**194988.81**	1.95E+05
20	1.95E+05	1.97E+05	1.95E+05	1.95E+05	1.94E+05	**193594.93**	1.94E+05
22	1.94E+05	1.96E+05	1.95E+05	1.94E+05	1.93E+05	**192751.66**	1.93E+05
25	1.93E+05	1.96E+05	1.95E+05	1.94E+05	1.93E+05	**192169.3**	1.92E+05

Table 5.7: Results for Waveform Generator data set: CPU time in seconds,(this Table corresponds to the Figure 5.5), (The best results are highlighted)

	Clustering time values by the following Algorithms						
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
	Waveform Generator						
2	4.92	7.23	8.06	1.89	**4.44**	5.88	6.73
3	8.98	14.31	14.97	17.92	**9.03**	9.18	10.02
5	18.66	27.98	35.88	36.67	22.23	**16.73**	17.28
7	29.92	42.09	59.91	59.09	39.36	**25.77**	26.23
10	52.36	63.59	140.39	104.70	144.27	**46.65**	46.81
12	67.09	77.81	293.11	134.05	224.91	58.80	**58.78**
15	91.16	96.98	480.77	182.80	318.66	80.32	**79.73**
17	108.92	111.86	584.70	217.22	406.89	90.35	**89.56**
20	132.23	130.75	921.27	264.27	506.92	108.59	**107.31**
22	144.69	144.30	1138.66	290.00	660.19	120.40	**118.66**
25	163.58	163.16	1965.91	328.30	699.52	136.65	**134.25**

results show that both the FMS-MGKM and FMS-MGKM2 algorithms produce results which are very close to the best results obtained by all algorithms. However, these two algorithms use significantly less CPU time than other algorithms. There is no any significant difference between the performances of the FMS-MGKM and FMS-MGKM2 algorithms.

Results for the Combined Cycle Power Plant data set are presented in Tables 5.12 and 5.13. One can see in this data set both proposed algorithms produce results which are close to the best results obtained other algorithms. Again these two algorithms use less CPU time than other algorithms and again there is no any significant difference between the performances of the FMS-MGKM and FMS-MGKM2 algorithms.

Tables 5.14 and 5.15 present results for the Gesture Phase Segmentation data set.

Table 5.8: Results for Turkiye Student Evaluation data set: Cluster function values, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	130969.15	130969.00	**130807.50**	130807.52	130969.23	135968.75	139959.09
3	81781.85	89800.00	**81781.83**	81781.83	81781.91	86920.25	90621.85
5	60648.18	60648.00	**60647.47**	60647.47	60648.25	69869.18	71870.32
7	54722.64	54055.00	54724.32	**54023.71**	54723.85	64444.01	58660.34
10	49208.80	**48802.00**	48884.38	49124.25	49209.11	59580.30	54960.44
12	46912.68	46463.00	**46437.43**	47464.04	46953.48	58881.23	53083.43
15	44578.11	**43939.00**	44074.83	45346.34	44567.35	55592.27	51010.33
17	43182.14	42686.00	**42638.70**	44438.78	43157.76	54430.62	48703.83
20	41658.83	41459.00	**41107.82**	43350.21	41600.85	52461.64	47477.00
22	40984.65	40501.00	**40480.12**	42264.53	40959.66	50560.46	46776.43
25	39940.88	**39448.00**	39506.32	41572.60	39999.61	49929.37	44839.15

Table 5.9: Results for Turkiye Student Evaluation data set: CPU time in seconds,(this Table corresponds to the Figure 5.7), (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	3.28	6.39	4.13	6.18	3.22	1.63	**1.16**
3	6.25	12.48	8.52	12.36	6.47	3.23	**1.75**
5	10.83	24.45	17.21	21.04	11.80	6.88	**2.95**
7	14.86	36.14	30.59	29.05	17.58	10.66	**4.09**
10	20.39	54.08	56.99	41.53	26.56	16.25	**5.61**
12	24.20	66.48	91.09	48.52	33.13	19.52	**6.86**
15	29.56	84.66	162.48	59.33	60.33	23.81	**8.64**
17	33.34	97.72	229.98	66.00	77.91	26.59	**9.66**
20	38.64	116.83	309.94	77.81	110.02	30.98	**11.45**
22	41.86	128.83	344.36	83.55	122.06	33.41	**12.50**
25	47.61	147.38	431.67	95.78	195.66	37.58	**14.25**

The proposed FMS-MGKM and FMS-MGKM2 algorithms obtain results which are close to the best results obtained other algorithms and they use less CPU time than other algorithms. There is no any significant difference between the performances of the FMS-MGKM and FMS-MGKM2 algorithms.

Figures 5.2,5.4,5.6,5.8, present the dependence of the number of distance functions evaluations on the number of clusters for four small size data sets. These figures clearly show that the proposed algorithms: FMS-MGKM and FMS-MGKM2 require significantly less distance function evaluations than any other clustering algorithms used in comparison. One can see similar dependence also for other small size data sets.

Figures 5.1, 5.3, 5.5, 5.7 and 5.9 present the dependence of CPU time on the number of clusters for five small size data sets. This figure clearly show that the proposed algorithms: FMS-MGKM and FMS-MGKM2 in most of datasets, require

Table 5.10: Results for Drug yprop 41 data set: Cluster function values, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**1414.58**	**1414.58**	**1414.58**	**1414.58**	1415.60	**1414.58**	**1414.58**
3	1188.44	1188.44	1188.44	1188.44	1188.57	1188.44	**1188.44**
5	998.95	**998.93**	998.94	**998.93**	1000.67	**998.93**	**998.93**
7	901.66	901.72	901.57	**900.68**	903.43	901.53	901.53
10	799.27	809.27	**799.25**	807.92	801.93	809.27	809.27
12	755.99	762.94	756.07	759.35	757.79	759.78	759.77
15	704.09	707.07	**703.38**	703.60	707.51	704.36	704.36
17	675.07	680.35	**675.05**	675.90	678.12	678.26	678.25
20	639.15	643.41	**639.02**	642.63	639.39	640.82	641.29
22	617.23	619.78	**617.09**	619.29	619.35	620.09	617.53
25	588.53	589.42	**588.34**	595.96	591.36	589.10	589.11

Table 5.11: Results for Drug yprop 41 data set:CPU time in seconds, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	16.56	13.25	7.23	33.50	**4.94**	8.98	9.20
3	32.16	26.64	16.31	65.41	**9.94**	13.48	13.72
5	55.61	51.77	35.42	112.20	**19.69**	21.91	21.84
7	103.13	77.19	118.92	280.16	**31.47**	34.47	34.30
10	158.80	116.00	226.61	318.45	57.81	50.97	**50.42**
12	190.03	138.41	278.38	381.33	74.52	63.78	**62.84**
15	246.64	171.81	480.19	495.34	100.13	78.08	**76.14**
17	293.67	195.39	777.34	588.53	123.00	87.34	**85.33**
20	337.08	229.56	1027.81	674.47	166.00	108.39	**107.34**
22	371.31	253.81	1255.80	744.69	185.98	118.22	**116.83**
25	418.23	290.33	2175.34	838.98	234.34	135.84	**130.61**

significantly less CPU time than any other clustering algorithms used in comparison. One can see similar dependence also for other small size data sets.

Table 5.16 demonstrates the overall performance of the proposed algorithm.

Table 7.13 demonstrates the overall performance of the proposed algorithm with a with a breakdown of the numbers for different k values .

5.3 Summary

In this chapter, we applied the proposed algorithms to solve the clustering problem in seven small data sets. These data sets contain between 4800 and 10000 instances. We computed up to 25 clusters and compared results with other incremental clustering algorithms as well as with the multi-start k-means algorithm.Overall exactly five times, the proposed algorithms require less CPU time than other algorithms.

Table 5.12: Results for Combined Cycle Power Plant data set: Cluster function values, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	2428026.00	2428026.00	2428026.00	**2428025.99**	2428026.25	2428032.47	2428064.02
3	**1686857.00**	**1686857.00**	**1686857.00**	1686857.45	1686857.79	1686863.39	1686892.82
5	1130678.00	1139016.00	1130673.00	**1130667.59**	1130750.09	1139024.63	1139046.97
7	894119.50	894119.50	**894103.30**	894105.62	894124.45	894124.29	894145.67
10	**697473.20**	697483.00	**697473.80**	753577.61	697491.97	697483.52	697496.49
12	617462.40	616323.20	**616305.50**	655172.08	616373.74	616839.87	616856.44
15	529446.00	529558.10	**529077.50**	550913.50	529472.63	529110.57	529108.20
17	492065.30	493070.60	**491084.80**	511649.21	492065.64	491632.69	491657.81
20	443155.30	446557.30	**443119.80**	533811.29	443329.38	446620.44	446501.62
22	421878.50	420930.00	**420860.90**	436805.32	421289.98	421261.28	420964.49
25	393242.10	394367.90	392503.90	407027.14	**392122.17**	393973.67	393922.74

Table 5.13: Results for Combined Cycle Power Plant data set: CPU time in seconds,(this Table corresponds to the Figure 5.9), (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	5.38	4.3	2.92	11.28	**2.78**	4.31	4.219
3	10.17	7.36	4.94	20.39	**4.91v**	6.13	6.109
5	19.11	13.53	10.22	38.55	10.38	**10.20**	10.25
7	24.72	19.13	15.64	49.87	14.86	**13.88**	14.031
10	41.5	27.72	31.66	84.52	25.27	21.27	**20.86**
12	50.7	33.25	61.41	102.85	37.7	26.52	**25.28**
15	61.03	41.23	99.8	123.01	64.39	32.06	**31.45**
17	64.14	46.45	128	129.98	71.63	36.16	**35.52**
20	76.09	54.91	173	152.88	105.03	43.02	**42.16**
22	79.91	59.63	192	160.77	127.56	46	**45.20**
25	86.83	66.53	281.23	175.08	171.98	51.19	**49.98**

Results show that in most data sets the proposed algorithms get solutions which are very close to the best solutions found by all algorithms used in our computations. Only data set where the proposed algorithms failed to produce accurate results was Turkiye Student Evaluation data set.

In most cases the proposed algorithms require less, and in some instances significantly less, CPU time than other algorithms used for comparison. In most cases, we did not observe any significant difference between algorithms with weights and without weights both in the sense of accuracy and used computational effort.

Overall, results from this chapter allow us to conclude that the proposed clustering algorithms are accurate and efficient clustering algorithms in small data sets used in computations.

Table 5.14: Results for Gesture Phase Segmentation data set: Cluster function values, (The best results are highlighted).

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**63910.85**	**63910.85**	**63910.85**	**63910.85**	63910.96	**63910.85**	63911.07
3	45381.58	45381.58	45381.58	**32445.45**	45381.74	45381.58	45381.79
5	**30489.75**	31172.83	**30489.75**	**30489.75**	30491.07	31172.83	31174.04
7	**23188.67**	**23188.67**	**23188.67**	23195.82	23190.50	23188.67	23189.15
10	18102.91	18102.91	**17963.28**	18135.51	18104.48	18102.91	18103.30
12	**15812.57**	15812.69	15820.64	16222.81	15814.39	15812.68	15813.07
15	13341.09	13419.43	**13341.04**	14688.70	13344.82	13419.43	13419.72
17	**12155.19**	12245.28	12167.16	14180.72	12219.40	12245.52	12245.81
20	10926.95	10954.84	**10842.72**	13197.47	10888.97	10909.53	10910.01
22	10237.48	10264.94	**10148.45**	12950.69	10236.51	10265.34	10265.82
25	9303.73	9274.61	**9234.66**	12610.59	9312.64	9274.59	9275.24

Table 5.15: Results for Gesture Phase Segmentation data set: CPU time in seconds, (The best results are highlighted)

k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**7.08**	10.55	8.81	14.16	9.17	9.81	9.81
3	**12.25**	20.39	15.70	24.01	15.80	15.08	14.95
5	23.56	42.80	36.52	47.55	28.97	23.80	**23.36**
7	32.95	63.66	57.44	66.58	41.70	32.77	**32.76**
10	47.50	95.41	138.52	95.33	65.31	46.50	**45.34**
12	57.19	115.69	208.64	114.95	77.42	54.23	**53.34**
15	71.73	146.47	360.30	143.14	106.31	66.41	**65.97**
17	81.33	168.05	411.81	163.41	119.66	74.41	**74.31**
20	95.30	199.88	662.81	192.44	166.20	88.47	**88.23**
22	103.92	221.44	790.84	206.55	190.30	97.64	**97.44**
25	117.23	252.27	1083.89	234.22	223.70	109.56	**109.45**

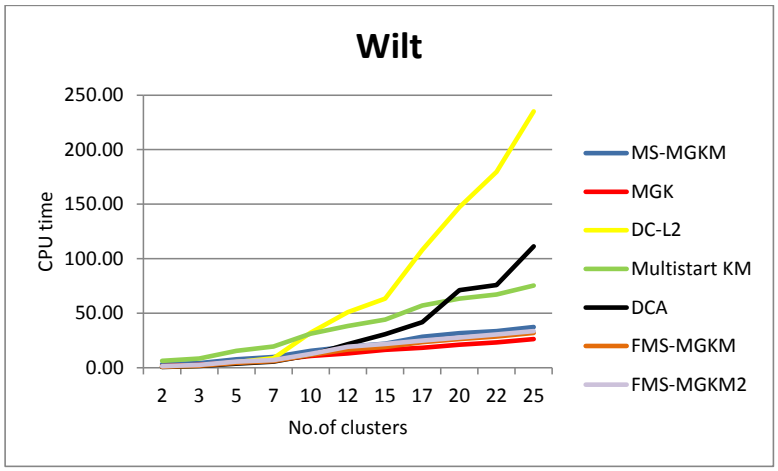

Figure 5.1: The CPU time vs the number of clusters: Wilt data set,(this Figure corresponds to Table. 5.3)

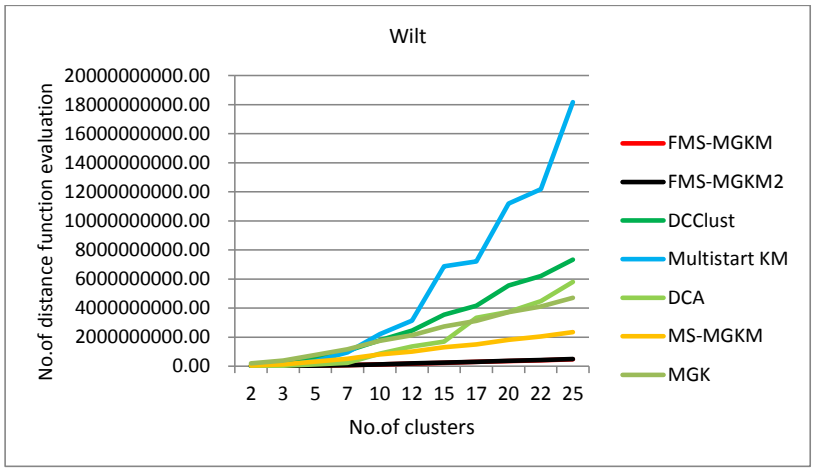

Figure 5.2: The number of distance function evaluations vs the number of clusters: Wilt data set

Figure 5.3: The number CPU time vs the number of clusters: Wine Quality data set,(this Figure corresponds to Table. 5.5)

Figure 5.4: The number of distance function evaluations vs the number of clusters: Wine Quality data set

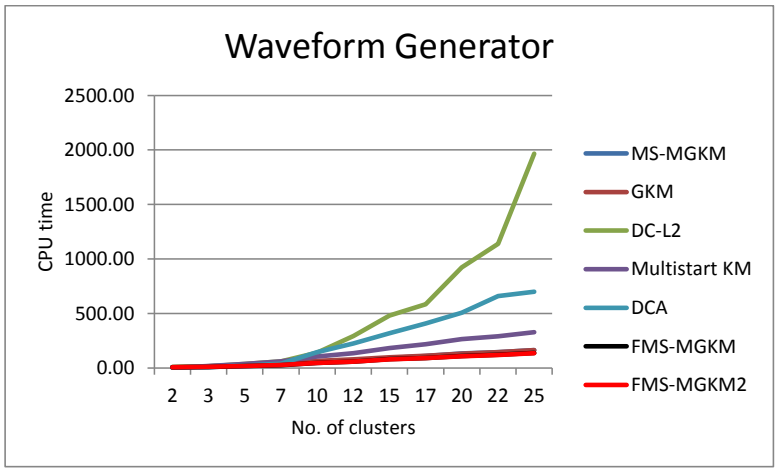

Figure 5.5: The CPU time vs the number of clusters: Waveform data set,(this Figure corresponds to Table. 5.7)

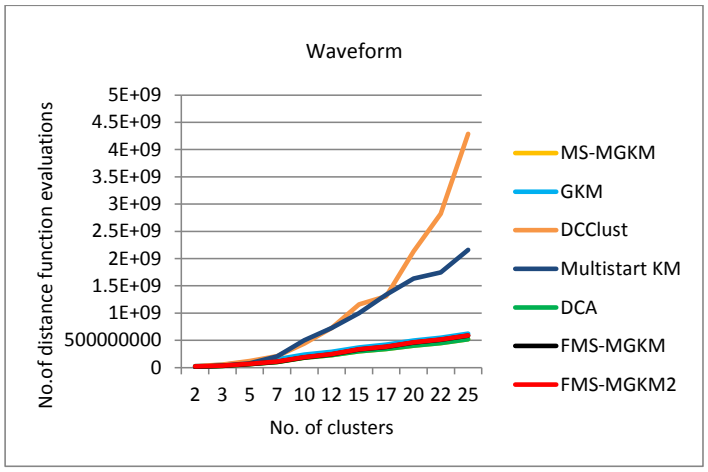

Figure 5.6: The number of distance function evaluations vs the number of clusters: Waveform data set

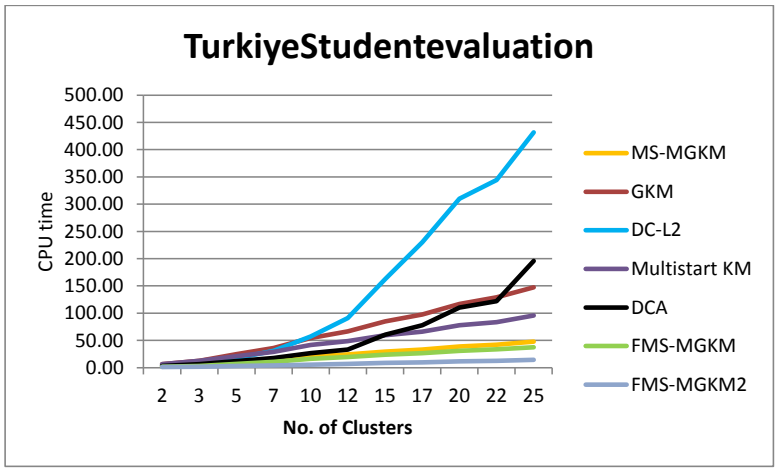

Figure 5.7: The CPU time vs the number of clusters: Turkiye Students evaluations data set,(this Figure corresponds to Table. 5.9)

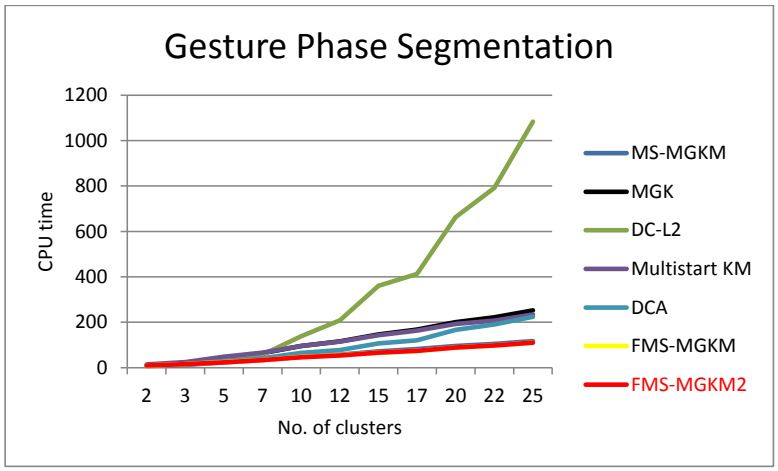

Figure 5.8: The number of distance function evaluations vs the number of clusters: Phase Gesture data set,(this Figure corresponds to Table. 5.15).

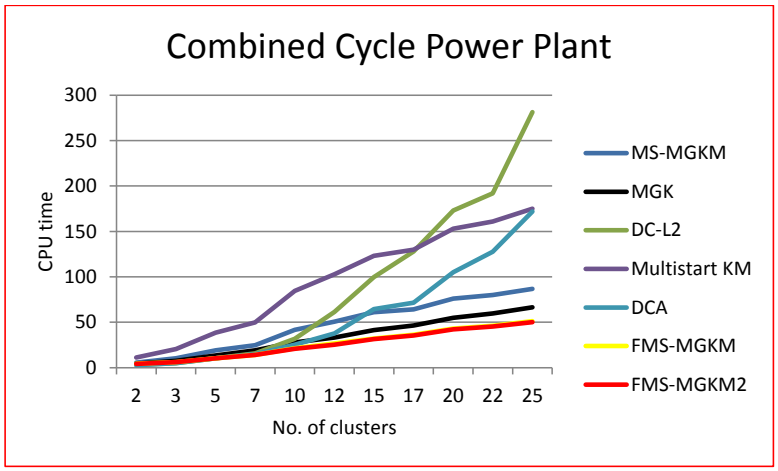

Figure 5.9: The number of clusters vs the CPU time: Combined Cycle Power Plant data set,(this Figure corresponds to Table. 5.13).

Table 5.16: Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms.

No.	**Parameters/Approach**	**Proposed algorithm**
1	Efficiency	**Produced 5 times the best (first) and few times the second best result.**
2	Computation	Requires low computations
3	Performance	Fast and accurate in comparison with the existing algorithms (finds similar objective function values but superior in CPU time)

Table 5.17: Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in terms of its efficiency,when tested over five small data sets,in comparison with other algorithms, with a breakdown of the numbers for different k values

K.	**No. of times the proposed algorithms shows the best results for given k**	**No. of times the proposed algorithm shows the second best results for given k**
2	2	1
3	2	2
5	2	2
7	4	3
10	4	3
12	5	2
15	5	2
17	5	2
20	5	2
22	5	2
25	5	2

Chapter 6

Computational results: medium size data sets

In this chapter we present and discuss computational results using medium size data sets. All data sets contain only numeric attributes and data sets do not contain missing values. First, we give a brief description of data sets and then present computational results. These results include optimal values of the cluster function obtained by each algorithm and CPU time required by them.

The following algorithms are used for comparison: the GKM algorithm, the MS-MGKM algorithm, the MS-KM algorithm, the DCA algorithm, the DCClust algorithm and two newly proposed algorithms: the FMS-MGKM and FMS-MGKM2 algorithms. The description of these algorithms are given in Chapter 4.

The number of starting points in the MS-KM algorithm is set to 400. Algorithms MS-MGKM, DCA and DCClust use the algorithm for computation of starting cluster centers described in the previous chapter. CPU time in all tables are in seconds. In all data sets up to 25 clusters are computed.

6.1 Data sets

The brief description of medium size data sets is given in Table 6.1. In this table we include the number of instances (m), the number of attributes (n) and the total number of entries (N_e) for each data set. The number of instances in these data sets is between ten thousand and one hundred thousand and the number of attributes is ranging from 2 to 128.

Table 6.1: Description of medium data size data sets

Sr.No.	Data sets	m	n	N_e
1	Gas Sensor Array Drift	13910	128	1780480
2	D15112 in TSPLIB	15112	2	30224
3	Letter Recognition	20000	16	320000
4	Chess (King-Rook vs. King)	28056	3	84168
5	Online News popularity	39644	38	1506472
6	Bank Marketing	45211	7	316477
7	Tamilnadu Electricity s Board Hourly Readings	45781	2	91562
8	KEGG Metabolic Relation Network	53413	20	1068260
9	Shuttle Landing Control	58000	10	580000
10	Jester Collaborative Filtering	73421	101	7415521
11	Programmed Logic Array	85900	2	171800
12	Sensit-vehicle-acoustic	98528	51	5024928

Some explanations on these data sets follow.

- Gas Sensor Array Drift data set contains information from 16 chemical sensors employed, which are used in simulations for drift return in a discrimination task of 6 gases at different levels of concentrations. The categorial features are removed and the remaining 128 attributes are used in clustering [22, 83].

- D15112 is a data set which contains information of mathematical problems related to the traveling salesman problem, it shows a large TSP instances in TSPLIB. it consists of 15,112 cities of Germany (D = Deutschland). The data set was contributed to TSPLIB by Andre Rohewe. it contains only two attributes [27, 83].

- Letter Recognition is a database of character image features which is used to identify each of the 26 capital letters in the English alphabet. The alphabet images depends on 20 different fonts and each letter among these 20 fonts was randomly deformed to produce a file of 20,000 unique stimuli and each of it is converted into 16 primitive numerical attributes which were then scaled

between values from 0 through 15. All sixteen attributes are used for clustering [29, 83].

- Chess (King-Rook vs. King) database is generated by Michael Bain and Arthur van Hoff of the Turing Institute, Glasgow, UK, to compute complex domains of Chess endgames countable and enumerable values for White King and Rook against Black King(KRK). All the database enumerable values are produced in a single iterative process utilizing the "standard backup" algorithm [24, 83].

- Online News popularity data set expresses a set of heterogeneous attributes of the articles published by Mashable in a period of two years, to predict the number of stocks in social networks (popularity). In the data file, the columns which contain complete zero values are removed as well as categorical values are removed and a data input file of thirty eight features is used in clustering [26, 83].

- Bank Marketing data is associated with direct marketing campaigns based on phone calls of a Portuguese banking institution, used to forecast if the client will subscribe a term deposit or not. The following features of the bank client data: age (numeric), type of job, marital status, education credit in default, has housing loan, personal loan, contact communication, last contact month, last contact day of the week, last contact duration, campaign: number of contacts performed during this campaign, days: number of days that passed by after the client was last contacted from a previous campaign (numeric: 999 means client was not previously contacted), previous: number of contacts performed before this campaign and for this client (numeric), outcome: outcome of the previous marketing campaign (categorical: "failure", "nonexistent", "success"), emp.var.rate: employment variation rate - quarterly indicator (numeric), cons.price.idx: consumer price index - monthly indicator (numeric), cons.conf.idx: consumer confidence index - monthly indicator (numeric), euribor3m: euribor 3 month rate - daily indicator (numeric), nr.employed: number of employees - quarterly indicator (numeric) are input to predict the desired output variable. In the clustering experiments, categorical and completely zero column values are removed from the input data file [20, 83].

- KEGG Metabolic Relation Network contains information data, which is used to examine, which pathways and associated functions are likely to be encoded in the genome. In the computer representation of relation network, substrate and product compounds are considered as edges while enzyme and genes are placed as nodes. Only 20 features of Relation Networks are used in clustering and categorical attributes were removed [8, 83].

- TamilNado Electriciy Board Hourly Reading data set shows a state-of-the-art survey of Tamil Nadu Electricity Board Data set structure and real time electricity readings of load demand for residential, commercial, industrial and agriculture in Tamil Nadu Around Thanajvur [7, 83].

- Shuttle Landing Control database contains information of a partially reusable low Earth orbital spacecraft system operated by the U.S, reported by Michie, D. in 1988. The report states that Burke's group utilized RULEMASTER to produce intelligible and under standable rules for deciding the conditions under which an auto landing would be better for manually controlling the spacecraft [10, 83].

- Programmed Logic Array (pla85900, Johnson) is a database created by Bell Labs in 2006. It along with a computer code certify the optimality of a solution to the 85,900-city traveling salesman problem. This is a largest instances data set in the TSPLIB collection of challenge problems [12, 83].

- Jester Collaborative Filtering Data set consists of 4.1 Million continuous ratings (-10.00 to 10.00) of 100 jokes from 73,421 users all gathered in a time between April 1999 and May 2003. It recommends you jokes, based on your ratings of previous jokes, by utilizing a collaborative filtering algorithm known as Eigentaste. For clustering purpose, the categorial attribute are removed [6, 35].

- Sensit-vehicle-acoustic data set contains information of vehicle acoustic signals which are considered as unwanted traffic noise, collected by wireless sensor networks [6, 33].

6.2 Results

Numerical results with medium size data sets are presented in Tables 6.2-6.13. The best results in all the tables are highlighted using the bold font.

Table 6.2: Results for Gas Sensor Array Drift data set,(this Table corresponds to the Figure 6.1), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**7.91E+13**	**7.91E+13**	**7.91E+13**	**7.91E+13**	**7.91E+13**	8.04E+13	8.04E+13
3	**5.02E+13**	**5.02E+13**	**5.02E+13**	**5.02E+13**	**5.02E+13**	5.07E+13	5.07E+13
5	**3.22E+13**	3.23E+13	3.23E+13	**3.22E+13**	3.23E+13	3.39E+13	3.39E+13
7	**2.25E+13**	**2.25E+13**	**2.25E+13**	**2.25E+13**	**2.25E+13**	2.40E+13	2.40E+13
10	**1.66E+13**	**1.66E+13**	**1.66E+13**	1.70E+13	**1.66E+13**	1.81E+13	1.81E+13
12	**1.41E+13**	**1.41E+13**	**1.41E+13**	1.55E+13	**1.41E+13**	1.60E+13	1.60E+13
15	**1.13E+13**	1.14E+13	1.14E+13	1.28E+13	1.14E+13	1.36E+13	1.36E+13
17	1.02E+13	**1.01E+13**	**1.01E+13**	1.12E+13	1.01E+13	1.24E+13	1.24E+13
20	8.92E+12	8.85E+12	8.85E+12	**1.03E+13**	8.86E+12	1.15E+12	1.15E+12
22	8.13E+12	8.18E+12	8.14E+12	9.77E+12	8.15E+12	**1.09E+12**	**1.09E+12**
25	7.27E+12	7.27E+12	7.27E+12	8.83E+12	7.28E+12	**1.02E+12**	**1.02E+12**
CPU time (in seconds)							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	103.19	174.66	**88.3**	209.89	101.32	116.69	122.97
3	264.52	346.02	236.8	531.64	249.82	**177.06**	186.11
5	604.47	690.28	654.95	1213.59	667.97	**316.44**	326.84
7	931.2	1038.91	1312.55	1866.47	1329.57	**460.03**	471.72
10	1445.64	1556.23	2254.56	2893.48	2271.58	**699.39**	702.95
12	1794.22	1895.47	3068.98	3588.81	3088	**853.28**	858.64
15	2334.22	2412.06	4471.97	4676.03	4490.99	1087.63	**1080.61**
17	2677.39	2761.03	5301.02	5009.2	5320.04	1252.38	**1231.91**
20	3227.33	3281.16	7027.03	6456.66	7048.05	1486.91	**1461.98**
22	3619.2	3656.97	7862.98	7251.23	7884	1648.83	**1616.75**
25	4148.95	4214.33	9319.44	8305.23	9340.46	1885.59	**1849.03**

Results for the Gas Sensor Array Drift data set presented in Table 6.2 show that although the proposed algorithms are not accurate in this data set they use significantly less CPU time than other algorithms. There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Table 6.3 contains results for the D15112 data set. One can see that the proposed algorithms produce results which are close to the best results among all algorithms, however, they use significantly less CPU time than most other algorithms. Again in this data set there is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms.

Results for the Letter Recognition data set presented in Tables 6.4 show that, overall, the proposed algorithms almost reach the best results obtained by all algo-

Table 6.3: Results for D15112 data set, (this Table corresponds to the Figure 6.11), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**3.68E+11**	**3.68E+11**	**3.68E+11**	**3.68E+11**	**3.68E+11**	**3.68E+11**	**3.68E+11**
3	**2.53E+11**	**2.53E+11**	**2.53E+11**	**2.53E+11**	**2.53E+11**	**2.53E+11**	**2.53E+11**
5	**1.33E+11**	**1.33E+11**	**1.33E+11**	**1.33E+11**	**1.33E+11**	**1.33E+11**	**1.33E+11**
7	**9.32E+10**	**9.32E+10**	**9.32E+10**	**9.32E+10**	**9.32E+10**	**9.32E+10**	**9.32E+10**
10	6.49E+10	6.54E+10	**6.45E+10**	**6.45E+10**	**6.45E+10**	6.54E+10	6.54E+10
12	**5.45E+10**	**5.45E+10**	**5.45E+10**	**5.45E+10**	**5.45E+10**	**5.45E+10**	**5.45E+10**
15	**4.31E+10**	4.32E+10	4.32E+10	**4.31E+10**	4.32E+10	4.32E+10	4.32E+10
17	3.80E+10	3.80E+10	**3.78E+10**	3.84E+10	**3.78E+10**	3.80E+10	3.80E+10
20	3.25E+10	**3.23E+10**	3.24E+10	3.26E+10	3.25E+10	**3.23E+10**	**3.23E+10**
22	**2.90E+10**	2.92E+10	**2.90E+10**	2.92E+10	2.91E+10	2.91E+10	2.91E+10
25	**2.53E+10**	**2.54E+10**	2.53E+10	2.54E+10	2.54E+10	2.54E+10	2.54E+10
CPU time (in seconds)							
2	15.02	11.05	**6.27**	36.58	6.46	7.55	7.53
3	31.72	18.22	**10.23**	66.47	13.63	14.59	14.61
5	42.92	26.8	**15.58**	85.28	16.21	21.44	21.14
7	56.13	34.49	**24.23**	112.69	29.9	27.22	26.95
10	73.67	45.04	42.43	152.58	50.45	35.41	**35.19**
12	86.97	52.09	73.21	176.47	79.5	40.83	**40.63**
15	110.56	63.2	114.63	233.06	118.61	50.45	**50.28**
17	123.13	77.24	179.79	246.14	182.65	56.09	**55.94**
20	145.73	89.53	266.64	292.39	274.94	65.00	**64.88**
22	159.67	96.02	291.96	359.52	294.43	72.11	**72.00**
25	185.22	106.27	471.79	370.36	489.68	79.66	**79.56**

rithms, however, CPU time used by these algorithms is significantly less than that used by other algorithms. In this data set, the performance of the FMS-MGKM2 algorithm is better than that of the FMS-MGKM algorithm in this data set both in sense of accuracy and the used computational time.

Based on results for the Chess (King-Rook vs. King) data set given in Tables 6.5 we can conclude the FMS-MGKM algorithm failed to produce good results. Although the FMS-MGKM2 algorithm is not accurate in this data set, however its results much better than that by the FMS-MGKM algorithm. We have completely different picture on the used computational time. Both the FMS-MGKM and FMS-MGKM2 algorithms are extremely fast. They are significantly better than other algorithms in this sense.

We can see from results for the Online News popularity data set, presented in Table 6.6, that although the proposed algorithms are not as accurate as other algorithms they require significantly less computational time. There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Results for the Bank Marketing data set presented in Table 6.7 show that the

Table 6.4: Results for Letter Recognition data set, (this Table corresponds to the Figure 6.9), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**1.38E+06**	**1.38E+06**	**1.38E+06**	**1.38E+06**	**1.38E+06**	**1.38E+06**	**1.38E+06**
3	**1.25E+06**	**1.25E+06**	**1.25E+06**	**1.25E+06**	**1.25E+06**	**1.25E+06**	**1.25E+06**
5	**1.10E+06**	**1.10E+06**	**1.10E+06**	**1.10E+06**	**1.10E+06**	1.08E+06	**1.10E+06**
7	9.72E+05	9.72E+05	9.72E+05	**9.70E+05**	9.72E+05	9.73E+05	9.73E+05
10	**8.58E+05**	**8.58E+05**	**8.58E+05**	**8.58E+05**	**8.58E+05**	8.78E+05	8.63E+05
12	8.04E+05	8.05E+05	8.05E+05	**8.02E+05**	8.04E+05	8.23E+05	8.05E+05
15	**7.44E+05**	7.48E+05	7.48E+05	7.62E+05	7.54E+05	7.49E+05	7.51E+05
17	**7.15E+05**	7.20E+05	7.16E+05	7.16E+05	7.19E+05	7.23E+05	7.18E+05
20	6.76E+05	6.77E+05	6.76E+05	**6.74E+05**	6.80E+05	6.82E+05	6.77E+05
22	6.52E+05	6.60E+05	6.56E+05	**6.50E+05**	6.54E+05	6.68E+05	6.55E+05
25	6.24E+05	6.32E+05	6.24E+05	**6.22E+05**	6.24E+05	6.32E+05	6.25E+05
CPU time (in seconds)							
2	55.92	48.11	**19.77**	119.38	53.61	38.33	36.44
3	99.22	99.42	**40.72**	200.88	105.20	61.56	58.50
5	219.14	200.81	**85.63**	446.09	213.56	105.20	94.83
7	325.56	299.72	**134.02**	654.31	353.83	144.58	140.95
10	498.11	440.17	216.00	1002.30	571.50	205.44	**194.92**
12	584.56	522.45	278.60	1195.72	718.08	241.38	**227.67**
15	756.83	655.75	381.58	1414.16	994.70	300.38	**275.45**
17	841.92	735.80	450.09	1690.45	1263.03	334.45	**309.45**
20	970.95	865.25	563.60	1898.00	1569.30	387.88	**359.83**
22	1054.42	945.13	668.15	2110.33	1991.63	426.88	**391.48**
25	1177.03	1076.86	813.44	2367.39	2410.45	486.67	**441.86**

proposed algorithms reach good accuracy and they use less CPU time than other algorithms (except the DCClust algorithm). The FMS-MGKM2 algorithm is more efficient than the the FMS-MGKM algorithm in this data set.

Table 6.8 contains results for the TamilNadu Electricity Board Hourly Reading data set. In this data set both proposed algorithms achieved similarly very good results in the sense of accuracy, they obtained the best result. Computational time used by these algorithms are similar to that of most other algorithms. These means that only small portion of data points was removed by the algorithm for data point reduction.

Results for the KEGG Metabolic Relation Network data set are given in Table 6.9. These results show that the proposed FMS-MGKM algorithm failed to solve the clustering problem in this data set. The FMS-MGKM2 algorithm was more successful, its results are acceptable in sense of accuracy. However, both algorithms are very fast in this data set in comparison with other algorithms.

Results for the Shuttle Control data set presented in Table 6.10 show that the proposed algorithms reach good accuracy in this data set, however, they use significantly less CPU time than other algorithms. There is no any significant difference

Table 6.5: Results for Chess (King-Rook vs. King) data set, (this Table corresponds to the Figure 6.10), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**2.01E+05**	**2.01E+05**	**2.01E+05**	2.06E+05	2.02E+05	2.38E+05	2.17E+05
3	**1.37E+05**	**1.37E+05**	**1.37E+05**	**1.37E+05**	**1.37E+05**	1.82E+05	1.49E+05
5	8.49E+04	8.48E+04	**8.47E+04**	8.49E+04	8.49E+04	1.65E+05	9.66E+04
7	6.77E+04	6.69E+04	**6.66E+04**	6.77E+04	6.67E+04	1.35E+05	7.90E+04
10	5.26E+04	5.24E+04	**5.22E+04**	5.25E+04	5.25E+04	8.17E+04	6.28E+04
12	4.72E+04	4.68E+04	4.75E+04	**4.70E+04**	4.70E+04	6.65E+04	5.59E+04
15	**4.02E+04**	4.04E+04	4.03E+04	4.09E+04	4.06E+04	5.71E+04	4.44E+04
17	3.75E+04	3.66E+04	**3.60E+04**	3.76E+04	3.66E+04	5.28E+04	4.01E+04
20	3.39E+04	3.23E+04	**3.10E+04**	3.28E+04	3.14E+04	4.50E+04	3.55E+04
22	3.13E+04	2.95E+04	**2.83E+04**	3.04E+04	2.84E+04	4.32E+04	3.31E+04
25	2.76E+04	2.68E+04	**2.59E+04**	2.74E+04	2.64E+04	3.81E+04	3.02E+04
CPU time (in seconds)							
2	37.32	29.97	10.94	74.03	37.81	0.08	**0.06**
3	68.03	57.05	21.09	136.06	65.61	0.08	**0.08**
7	147.05	138.52	66.72	294.39	164.02	0.09	**0.09**
10	200.34	194.88	102.85	400.00	239.34	0.13	**0.11**
12	233.39	231.88	129.81	466.11	288.72	0.14	**0.13**
15	288.35	286.13	176.23	576.70	363.47	0.17	**0.16**
17	326.12	321.56	205.02	653.21	420.56	0.19	**0.17**
20	389.68	375.17	258.62	779.11	504.61	0.23	**0.22**
22	431.37	409.91	294.16	864.75	555.70	0.27	**0.23**
25	487.49	461.73	359.19	973.58	641.50	0.31	**0.28**

between the accuracy of the FMS-MGKM and FMS-MGKM2 algorithms, however, the latter algorithm a bit faster than the former one.

Table 6.11 contain results for the Jester Collaborative Filtering data set. Both proposed algorithms achieved good accuracy in this data set, however, they use significantly less CPU time than other algorithms. There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Results for the Programmed Logic Array (Pla85900) data set are presented in Tables 6.12. These results show that the proposed algorithms demonstrate very similar performance with other algorithms both in sense of accuracy and used computational time.

Table 6.13 contains results obtained using the Sensit-vehicle-acoustic data set. Both newly developed algorithms achieved, in many cases, best results and they used significantly less computational time than other algorithms. There is no any significant difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set.

Table 6.14 demonstrates the overall performance of the proposed algorithm.

Table 6.6: Results for Online News popularity data set, (this Table corresponds to the Figure 6.12), (The best results are highlighted)

Cluster function values							
k	*MS − MGKM*	*GKM*	*DCClust*	*MS − KM*	*DCA*	*FMS − MGKM*	FMS-MGKM2
2	3.88E+14	9.54E+14	**3.70E+14**	**3.70E+14**	3.71E+14	3.88E+14	3.88E+14
3	2.57E+14	5.92E+14	2.48E+14	**2.41E+14**	2.49E+14	2.57E+14	2.57E+14
5	**1.30E+14**	3.11E+14	8.91E+14	1.96E+14	8.92E+14	**1.30E+14**	**1.30E+14**
7	8.05E+13	1.80E+14	**6.00E+13**	1.72E+13	6.01E+13	8.50E+13	8.50E+13
10	5.10E+13	1.17E+14	3.69E+13	**1.62E+13**	3.70E+13	5.10E+13	5.10E+13
12	3.91E+13	9.67E+13	2.69E+13	**1.57E+13**	2.70E+13	3.91E+13	3.91E+13
15	3.03E+13	7.76E+13	2.06E+13	**1.50E+13**	2.07E+13	3.06E+13	3.06E+13
17	2.69E+13	6.99E+13	1.78E+13	**1.47E+13**	1.79E+13	2.74E+13	2.73E+13
20	2.35E+13	5.98E+13	**1.44E+13**	1.46E+13	1.45E+13	2.36E+13	2.36E+13
22	2.14E+13	5.56E+13	1.25E+13	1.45E+13	1.26E+13	**2.13E+13**	2.14E+13
25	**1.93E+13**	4.96E+13	1.03E+13	1.45E+13	1.04E+13	1.94E+13	1.94E+13
CPU time (in seconds)							
2	443.00	786.23	**43.15**	925.17	50.82	369.41	377.86
3	1048.39	1714.05	**118.55**	2113.28	126.22	575.02	569.27
5	2126.63	3325.65	**476.57**	4458.38	484.24	977.17	958.17
7	3445.95	4870.41	**908.77**	7089.89	916.44	1344.66	1312.02
10	4272.92	7151.20	1875.44	80963.63	1883.11	1864.13	**1830.45**
12	5236.14	8695.73	2602.61	10651.50	2608.28	2224.38	**2185.64**
15	6142.14	10978.93	4191.36	12374.05	4197.03	2737.66	**2702.80**
17	6874.69	12489.18	5536.69	13811.38	5542.36	3048.47	**3017.25**
20	7832.80	14775.26	7515.88	15695.09	7525.55	3538.70	**3498.67**
22	8472.58	16293.56	8908.39	17042.16	8918.06	3905.84	**3870.59**
25	9580.64	18608.83	11348.37	19164.05	11358.04	4359.69	**4359.64**

Figures 6.2, 6.4, 6.6, 6.8 and 6.13 illustrate dependence of the number of distance functions evaluations on the number of clusters. These figures clearly show that the proposed algorithms: FMS-MGKM and FMS-MGKM2 require significantly less distance function evaluations than any other clustering algorithms used in comparison. The similar picture exists also for all other data sets used in our computational experiments.

Figures 6.1, 6.3,6.5, 6.7,6.9,6.10, 6.11 and 6.12 present the dependence of CPU time on the number of clusters for eight medium size data sets. These figure clearly show that the proposed algorithms: FMS-MGKM and FMS-MGKM2 require significantly less CPU time in most of the datasets than any other clustering algorithms used in comparison. One can see similar dependence also for other very large size data sets.

Table 7.13 demonstrates the overall performance of the proposed algorithm with a with a breakdown of the numbers for different k values .

Table 6.7: Results for Bank Marketing data set, (this Table corresponds to the Figure 6.3), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**2.02E+11**	**2.02E+11**	**2.02E+11**	**2.02E+11**	**2.02E+11**	**2.02E+11**	**2.02E+11**
3	**1.13E+11**	**1.13E+11**	**1.13E+11**	**1.13E+11**	**1.13E+11**	**1.13E+11**	**1.13E+11**
5	4.88E+10	4.88E+10	4.92E+10	4.88E+10	**4.87E+10**	4.88E+10	4.88E+10
7	2.89E+10	2.89E+10	3.10E+10	2.89E+10	**2.88E+10**	2.89E+10	2.89E+10
10	**1.64E+10**	1.67E+10	**1.64E+10**	**1.64E+10**	**1.64E+10**	1.67E+10	1.67E+10
12	1.24E+10	1.31E+10	1.23E+10	**1.22E+10**	1.23E+10	1.31E+10	1.31E+10
15	9.18E+09	9.49E+09	9.51E+09	**9.17E+09**	9.18E+09	9.49E+09	9.49E+09
17	7.72E+09	7.73E+09	7.74E+09	**7.62E+09**	7.72E+09	7.73E+09	7.73E+09
20	6.34E+09	6.43E+09	6.43E+09	**6.24E+09**	6.34E+09	6.43E+09	6.43E+09
22	5.59E+09	**5.65E+09**	5.66E+09	5.59E+09	5.58E+09	5.65E+09	5.65E+09
25	4.77E+09	4.91E+09	4.78E+09	**4.76E+09**	4.85E+09	4.91E+09	4.91E+09
CPU time (in seconds)							
2	100.28	332.22	**11.78**	200.02	39.78	292.83	239.13
3	468.61	727.7	**314.16**	936.2	414.16	1017.59	782.42
5	**1226.9**	1671.78	2849.81	2451.9	2949.81	1802.44	1390.03
7	**1773.37**	2672.81	4534	3416.75	4634	2295.33	1802.22
10	3017.89	3597.41	16875.19	5905.77	16975.19	**2706.55**	2171.48
12	3412.24	4050.63	17456.58	6694.48	17556.58	2979.72	**2423.81**
15	3765.88	4683.03	19556.11	7401.76	19656.11	3290.7	**2719.34**
17	4087.87	5036.25	21287.75	8045.73	21387.75	3471.8	**2895.53**
20	4284.6	5495.91	40045.36	8439.2	40045.36	3685.16	**3104.16**
22	4461.33	5825.25	47980.75	8792.66	48980.75	3898.69	**3312.17**
25	4684.6	6200.42	54144.16	9239.2	55144.16	4041.97	**3453.53**

6.3 Summary

In this chapter, we applied the proposed algorithms to solve the clustering problem in twelve medium size data sets. These data sets contain between 13900 and 100000 data points. We computed up to 25 clusters in these data sets and compared results with other incremental clustering algorithms and also with the multi-start k-means algorithm.Overall exactly ten times, the proposed algorithms require less CPU time than other algorithms.

With some exceptions the proposed in this thesis algorithms demonstrated good performance in the sense clustering accuracy. In most data sets the proposed algorithms get solutions which are very close to the best solutions found by all algorithms used in our computational experiments.

In most cases the proposed algorithms require less, and in some instances significantly less, CPU time than other algorithms. In some cases, we did not see any significant difference between algorithms with weights and without weights both in the sense of accuracy and used computational effort, however in some other cases the algorithm without weights demonstrated better performance than the algorithm

Table 6.8: Results for TamilNadu Electricity Board Hourly Reading data set, (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**4.77E+03**	**4.77E+03**	**4.77E+03**	**4.77E+03**	**4.77E+03**	**4.77E+03**	**4.77E+03**
3	**3.03E+03**	**3.03E+03**	**3.03E+03**	**3.03E+03**	**3.03E+03**	**3.03E+03**	**3.03E+03**
5	**1.62E+03**	**1.62E+03**	**1.62E+03**	**1.62E+03**	**1.62E+03**	**1.62E+03**	**1.62E+03**
7	1.18E+03	**1.16E+03**	**1.16E+03**	1.39E+03	1.17E+03	1.16E+03	1.16E+03
10	7.86E+02	7.82E+02	**7.80E+02**	7.80E+02	7.95E+02	7.82E+02	7.86E+02
12	6.54E+02	**6.51E+02**	**6.51E+02**	6.54E+02	6.56E+02	**6.51E+02**	**6.51E+02**
15	**5.11E+02**	5.13E+02	5.12E+02	5.18E+02	5.19E+02	5.13E+02	5.13E+02
17	4.52E+02	4.51E+02	**4.50E+02**	**4.50E+02**	4.59E+02	4.51E+02	4.51E+02
20	3.86E+02	**3.84E+02**	**3.84E+02**	**3.84E+02**	3.87E+02	**3.84E+02**	**3.84E+02**
22	3.48E+02	3.49E+02	3.48E+02	**3.46E+02**	3.50E+02	**3.46E+02**	**3.46E+02**
25	**3.03E+02**	**3.03E+02**	**3.03E+02**	**3.03E+02**	3.04E+02	**3.03E+02**	**3.03E+02**
CPU time (in seconds)							
2	133.19	109.34	**26.50**	293.88	40.69	111.78	127.50
3	265.47	269.80	**49.81**	565.59	72.17	216.58	237.28
5	399.50	413.73	**91.35**	828.52	127.81	303.22	319.97
7	560.08	534.94	**138.42**	981.35	190.41	376.92	389.77
10	800.73	850.47	**214.81**	1029.00	285.48	567.88	581.50
12	955.64	1025.88	**263.20**	1930.72	347.08	646.75	670.73
15	1198.48	1237.75	**382.06**	2030.06	437.89	779.23	797.42
17	1357.23	1351.02	**430.97**	2720.63	496.25	852.20	868.42
20	1584.73	1509.08	**533.13**	3208.16	585.17	951.56	965.00
22	1701.97	1612.92	**598.36**	3407.88	646.16	1032.05	1040.39
25	1884.77	1778.19	**728.41**	3804.59	738.09	1133.31	1149.30

with weights.

Overall, results from this chapter allow us to conclude that the proposed clustering algorithms are accurate and efficient clustering algorithms in most medium size data sets.

Table 6.9: Results for KEGG Metabolic Relation Network data set, (this Table corresponds to the Figure 6.5), (The best results are highlighted)

Cluster function values							
k	*MS − MGKM*	*GKM*	*DCClust*	*MS − KM*	*DCA*	*FMS − MGKM*	FMS-MGKM2
2	**1.14E+09**	**1.14E+09**	**1.14E+09**	1.35E+09	**1.14E+09**	1.37E+09	1.31E+09
3	**4.90E+08**	**4.90E+08**	**4.90E+08**	1.10E+09	**4.90E+08**	1.15E+09	5.76E+08
5	**1.88E+08**	**1.88E+08**	**1.88E+08**	**1.88E+08**	**1.88E+08**	1.94E+08	2.26E+08
7	1.20E+08	1.20E+08	1.21E+08	**1.16E+08**	1.20E+08	1.26E+08	1.44E+08
10	**6.35E+07**	**6.35E+07**	6.36E+07	8.28E+07	**6.35E+07**	8.89E+07	7.63E+07
12	**4.78E+07**	**4.78E+07**	4.81E+07	7.53E+07	4.80E+07	8.49E+07	6.46E+07
15	**3.51E+07**	3.66E+07	**3.51E+07**	6.96E+07	3.55E+07	7.65E+07	4.31E+07
17	2.99E+07	3.19E+07	**2.96E+07**	6.81E+07	3.00E+07	7.38E+07	3.62E+07
20	2.56E+07	2.55E+07	**2.51E+07**	6.63E+07	2.55E+07	7.27E+07	3.28E+07
22	2.26E+07	**2.25E+07**	2.28E+07	6.53E+07	2.26E+07	7.23E+07	2.87E+07
25	1.95E+07	1.97E+07	1.96E+07	6.21E+07	**1.93E+07**	7.20E+07	2.55E+07
CPU time (in seconds)							
2	49.23	345.22	39.14	98.55	38.31	36.89	**29.11**
3	221.19	777.47	95.75	446.73	113.80	60.52	**48.80**
5	1426.56	2001.05	319.46	2854.78	525.98	132.44	**86.16**
7	1782.48	2900.36	806.51	3563.83	2555.22	204.30	**119.84**
10	3185.13	4637.73	1605.47	6387.84	4483.98	298.84	**176.89**
12	4317.17	5399.95	2229.33	8621.16	5855.23	359.33	**215.27**
15	5693.55	6547.58	3300.64	11546.15	8401.22	452.45	**306.52**
17	6204.28	7436.16	3941.13	12832.34	9502.72	511.45	**357.81**
20	6726.13	8556.98	4963.48	13924.35	13264.66	588.45	**406.45**
22	7425.95	9392.09	5773.16	15289.94	15120.80	638.84	**444.63**
25	8402.20	10499.50	7261.91	16458.01	21249.03	708.59	**500.11**

Table 6.10: Results for Shuttle Control data set, (The best results are highlighted)

Cluster function values							
k	*MS − MGKM*	*GKM*	*DCClust*	*MS − KM*	*DCA*	*FMS − MGKM*	FMS-MGKM2
2	**2.13E+09**	2.81E+09	**2.13E+09**	**2.13E+09**	2.14E+09	**2.13E+09**	**2.13E+09**
3	**1.09E+09**	1.51E+09	**1.09E+09**	2.18E+09	1.10E+09	**1.09E+09**	**1.09E+09**
5	**7.25E+08**	8.39E+08	7.26E+08	1.00E+09	7.36E+08	**7.25E+08**	**7.25E+08**
7	**4.34E+08**	4.45E+08	4.37E+08	6.83E+08	4.47E+08	**4.34E+08**	**4.34E+08**
10	**2.83E+08**	3.02E+08	2.85E+08	5.42E+08	2.95E+08	**2.83E+08**	**2.83E+08**
12	**2.14E+08**	2.42E+08	2.21E+08	5.36E+08	2.31E+08	**2.14E+08**	**2.14E+08**
15	**1.53E+08**	1.71E+08	1.60E+08	4.59E+08	1.70E+08	**1.53E+08**	**1.53E+08**
17	**1.24E+08**	1.41E+08	1.30E+08	4.38E+08	1.40E+08	1.27E+08	1.27E+08
20	**1.02E+08**	1.07E+08	1.06E+08	4.08E+08	1.07E+08	1.06E+08	1.06E+08
22	**9.06E+07**	9.32E+07	9.57E+07	3.69E+08	9.67E+07	9.41E+07	9.41E+07
25	7.70E+07	**7.55E+07**	8.12E+07	3.57E+08	8.22E+07	8.00E+07	8.00E+07
CPU time (in seconds)							
2	17.06	118.47	8.38	17.06	**8.38**	99.53	97.38
3	80.09	235.39	25.69	330.25	**22.43**	115.03	111.13
5	277.08	527.89	84.92	706.53	**149.93**	156.89	150.5
7	407.5	878.97	309.22	1027.36	384.26	223.77	**215.19**
10	646.91	1293.28	688.1	1583.8	808.49	388.25	**369.64**
12	862.22	1620.07	1095.82	1724.14	1648.26	457.53	**435.25**
15	1176.7	1995.72	2276.91	2537.52	2394.41	641.52	**606.28**
17	1476.47	2244.56	2600.52	3014.47	2696.06	814.81	**768.59**
20	2064.61	2483.18	3251.2	4186.13	3333.29	1062	**1000.98**
22	2555.3	2688.62	3556.58	5220.25	3849.22	1235.16	**1165.31**
25	2990.11	2997.96	4735.23	6054.34	4925.58	1568.45	**1478.25**

Table 6.11: Results for Jester Collaborative Filtering data set,(The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**6.97E+09**	6.99E+09	**6.97E+09**	6.99E+09	6.99E+09	6.99E+09	6.99E+09
3	**5.02E+09**	**5.02E+09**	**5.02E+09**	5.84E+09	5.04E+09	**5.02E+09**	**5.02E+09**
5	4.03E+09	**3.81E+09**	**3.81E+09**	4.17E+09	3.91E+09	**3.81E+09**	**3.81E+09**
7	**3.53E+09**	**3.53E+09**	**3.53E+09**	3.66E+09	3.63E+09	**3.53E+09**	**3.53E+09**
10	**3.33E+09**	**3.33E+09**	**3.33E+09**	3.49E+09	3.53E+09	**3.33E+09**	**3.33E+09**
12	**3.26E+09**	**3.26E+09**	**3.26E+09**	3.42E+09	3.46E+09	**3.26E+09**	**3.26E+09**
15	**3.20E+09**	3.22E+09	**3.20E+09**	3.36E+09	3.40E+09	**3.20E+09**	**3.20E+09**
17	3.16E+09	3.21E+09	**3.16E+09**	3.23E+09	3.26E+09	**3.16E+09**	**3.16E+09**
20	3.12E+09	3.18E+09	**3.12E+09**	3.25E+09	3.18E+09	**3.12E+09**	**3.12E+09**
22	3.09E+09	3.11E+09	3.09E+09	3.18E+09	3.17E+09	**3.08E+09**	**3.08E+09**
25	3.00E+09	**2.90E+09**	**2.90E+09**	2.96E+09	2.93E+09	3.05E+09	3.05E+09
CPU time (in seconds)							
2	7390.63	5332.91	4960.29	14904.25	4960.42	**725.3**	738.34
3	13554.22	10583.34	9828.61	27231.44	9828.09	**1063.45**	1079.94
5	22788.06	21296.45	18593.62	45699.12	16005.22	1656.19	**1655.92**
7	32650.73	31507.22	22888.66	65424.47	22888.66	2301.27	**2279.03**
10	49149.63	47093.31	31001.19	98422.25	35054.64	3414.3	**3363.97**
12	60054.28	57172.88	42633.53	120231.56	42633.53	4148.95	**4088.61**
15	76696.36	72105.86	55303.64	153515.72	55303.64	5266.88	**5193.36**
17	87952	81937.81	66124.03	176027	66124.03	6044.86	**5968.53**
20	990123.33	990716.33	80735.33	1980369.66	87125.33	7149.61	**7087.94**
22	100935.33	100935.33	85935.33	201993.66	90938.23	7875.16	**7830.49**
25	116331.13	103735.33	96735.33	232785.26	98425.42	8937.85	**8904.57**

Table 6.12: Results for Programmed Logic Array (Pla85900) data set and (this Table corresponds to the Figure 6.6), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**3.75E+15**	**3.75E+15**	**3.75E+15**	3.79E+15	**3.75E+15**	**3.75E+15**	**3.75E+15**
3	**2.28E+15**	**2.28E+15**	**2.28E+15**	2.29E+15	**2.28E+15**	**2.28E+15**	**2.28E+15**
5	**1.34E+15**	**1.34E+15**	**1.34E+15**	1.45E+15	**1.34E+15**	**1.34E+15**	**1.34E+15**
7	**9.71E+14**	**9.71E+14**	**9.71E+14**	9.82E+14	**9.71E+14**	**9.71E+14**	**9.71E+14**
10	**6.83E+14**	**6.83E+14**	**6.83E+14**	6.87E+14	**6.83E+14**	**6.83E+14**	**6.83E+14**
12	5.76E+14	5.75E+14	5.75E+14	**5.73E+14**	5.76E+14	5.75E+14	5.75E+14
15	**4.61E+14**	4.63E+14	4.62E+14	**4.61E+14**	4.62E+14	4.63E+14	4.63E+14
17	**4.10E+14**	**4.10E+14**	**4.10E+14**	4.21E+14	4.15E+14	**4.10E+14**	**4.10E+14**
20	3.52E+14	3.53E+14	3.52E+14	3.70E+14	**3.50E+14**	3.51E+14	3.51E+14
22	**3.20E+14**	3.23E+14	**3.20E+14**	3.32E+14	3.21E+14	**3.20E+14**	**3.20E+14**
25	2.87E+14	2.85E+14	**2.83E+14**	2.88E+14	**2.83E+14**	2.85E+14	2.85E+14
CPU time (in seconds)							
2	960.67	435.28	**80.08**	950.67	113.67	329.75	344.81
3	1715.31	1247.08	**153.48**	1701.31	216.05	793.08	822.44
5	2413.36	1836.36	**307.22**	2210.32	457.05	1220.67	1253.33
7	2965.28	2301.94	**470.89**	2861.10	723.38	1563.53	1597.02
10	4012.20	3081.13	**728.34**	4012.20	1171.80	2138.47	2171.05
12	4464.11	3605.94	**930.08**	4361.61	1503.06	2539.84	2571.34
15	5703.02	4264.78	**1278.09**	5801.22	2034.44	3026.88	3056.53
17	6224.19	4672.17	**1580.19**	6325.79	2469.42	3338.78	3367.19
20	6878.20	5299.33	**2021.56**	7171.26	3106.67	3804.59	3830.86
22	7407.25	5654.97	**2273.89**	7512.16	3624.02	4043.47	4068.27
25	8048.31	6177.55	**2725.47**	8328.73	4337.30	4394.53	4418.09

Table 6.13: Results for Sensit-vehicle-acoustic data set, (this Table corresponds to the Figure 6.7), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	1.38E+05	1.38E+05	1.38E+05	**1.34E+05**	1.38E+05	1.38E+05	1.38E+05
3	**1.17E+05**	**1.17E+05**	**1.17E+05**	**1.17E+05**	**1.17E+05**	**1.17E+05**	**1.17E+05**
5	9.99E+04	1.00E+05	1.00E+05	**9.72E+04**	1.00E+05	1.00E+05	1.00E+05
7	**8.85E+04**	**8.85E+04**	**8.85E+04**	8.98E+04	**8.85E+04**	**8.85E+04**	**8.85E+04**
10	7.80E+04	**7.70E+04**	7.76E+04	7.88E+04	7.76E+04	7.80E+04	7.80E+04
12	7.37E+04	**7.11E+04**	7.16E+04	7.23E+04	7.16E+04	7.20E+04	7.20E+04
15	6.51E+04	**6.48E+04**	**6.48E+04**	6.70E+04	**6.48E+04**	**6.48E+04**	**6.48E+04**
17	6.15E+04	6.15E+04	6.15E+04	**6.13E+04**	6.15E+04	6.15E+04	6.15E+04
20	5.76E+04	**5.74E+04**	5.85E+04	5.77E+04	5.86E+04	5.85E+04	5.85E+04
22	5.58E+04	**5.53E+04**	5.58E+04	5.60E+04	5.65E+04	5.62E+04	5.63E+04
25	5.36E+04	**5.27E+04**	5.32E+04	5.33E+04	5.32E+04	5.31E+04	5.31E+04
CPU time (in seconds)							
2	3378.84	3605.23	**2083.05**	5146.63	2749.48	3352.36	3253.19
3	6843.64	6978.44	**4636.83**	10254.62	5918.64	5104.09	5004.67
5	13616.49	13561.53	8850.87	22167.38	11728.61	8357.94	**8213.02**
7	19859.77	20092.96	12588.49	32667.7	17347.13	**11221.48**	11272.48
10	27818.33	29752.99	17690.15	44740.51	23930.06	15573.59	**15546.59**
12	32925.62	36165.92	21069.23	53408.19	28703.89	**18193.03**	18195.03
15	40301.25	45550.5	26085.07	64300.29	36885.17	**22009.23**	22162.23
17	44881.44	51392.56	29406.47	71400.6	42482.89	24516.42	**24513.42**
20	51451.7	60160.35	34480.01	81000.37	50448.47	28213.06	**28112.06**
22	55820.03	65867.82	37829.54	87114.16	55033.02	30667.98	**30660.98**
25	62103.61	74362.01	43139.39	96159.36	61835.38	34575.41	**34528.41**

Table 6.14: Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms.

No.	Parameters/Approach	Proposed algorithm
1	Efficiency	10 times gives the best result and few times gives the second best result
2	Computation	Requires low memory and small amount of calculations
3	Performance	Fast and accurate in comparison with the existing algorithms (finds similar objective function values but much superior in CPU time)

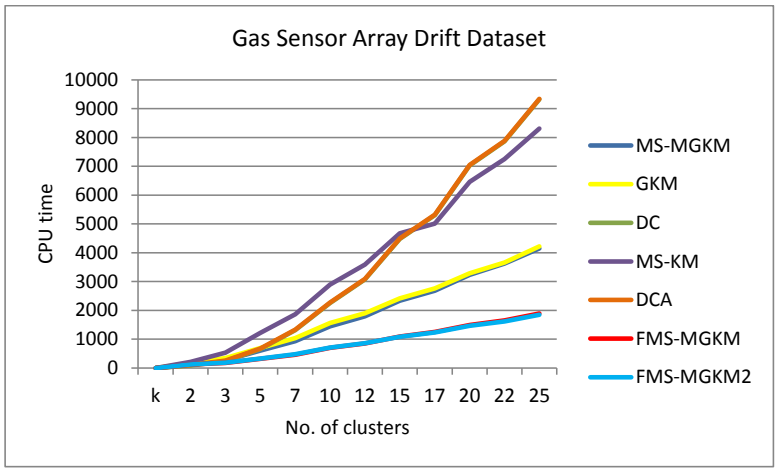

Figure 6.1: The CPU time vs the number of clusters: Drift data set,(this Figure corresponds to Table. 6.2)

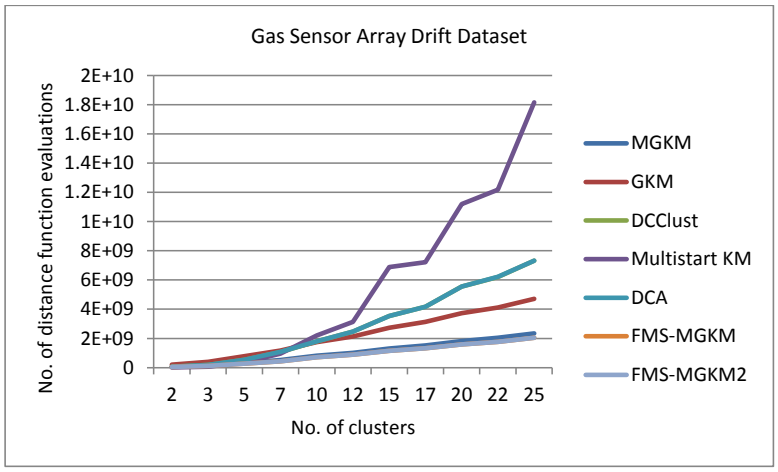

Figure 6.2: The number of distance function evaluations vs the number of clusters: Drift data set

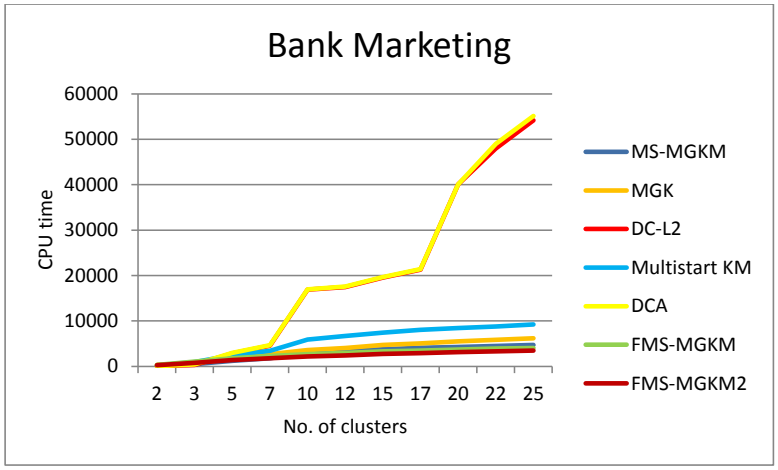

Figure 6.3: The CPU time vs the number of clusters: Bank Marketing data set,(this Figure corresponds to Table. 6.7)

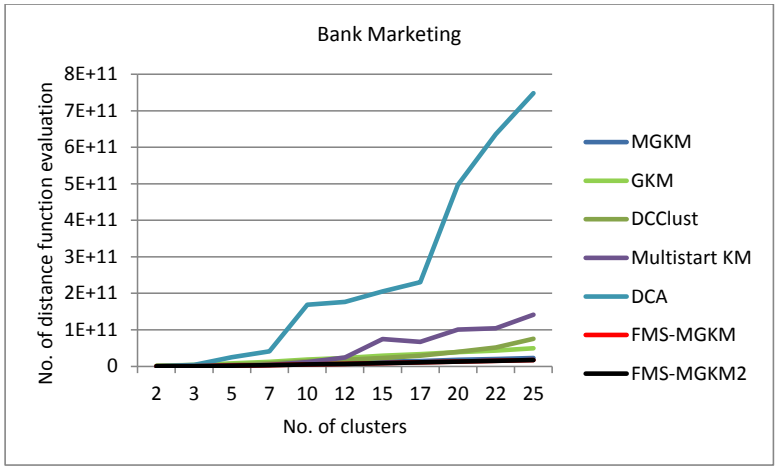

Figure 6.4: The number of distance function evaluations vs the number of clusters: Bank Marketing data set

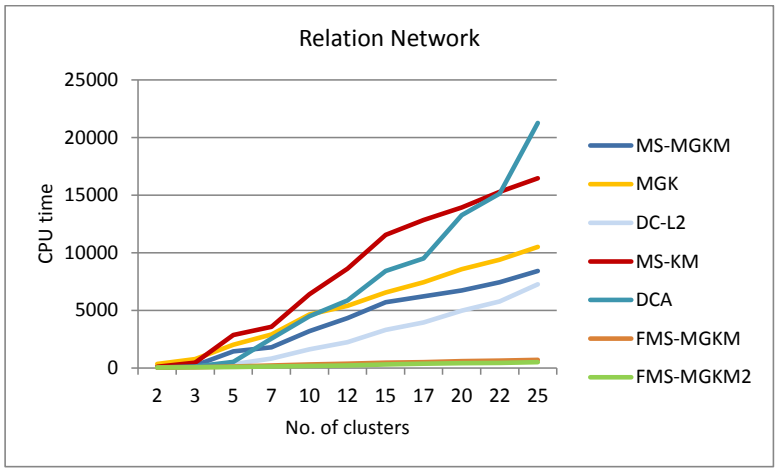

Figure 6.5: The CPU time vs the number of clusters: Relation Network data set, (this Figure corresponds to Table. 6.9)

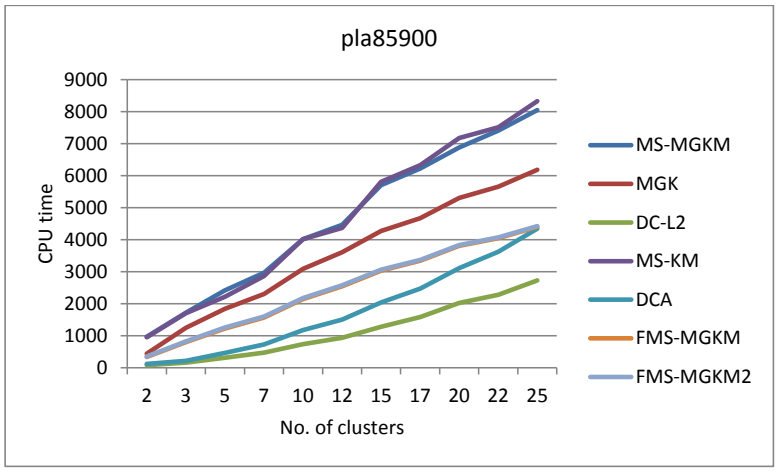

Figure 6.6: The CPU time vs the number of clusters: Programmed Logic Array (Pla85900) data set,(this Figure corresponds to Table. 6.12)

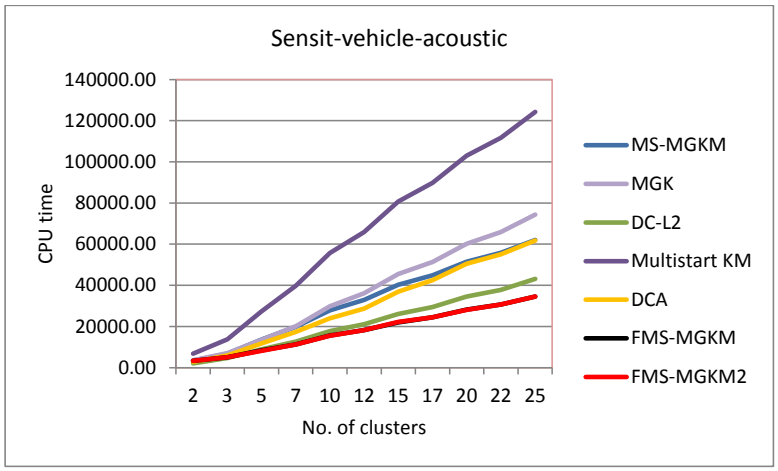

Figure 6.7: The CPU time vs the number of clusters: Sensit-vehicle-acoustic data set, (this Figure corresponds to Table. 6.13)

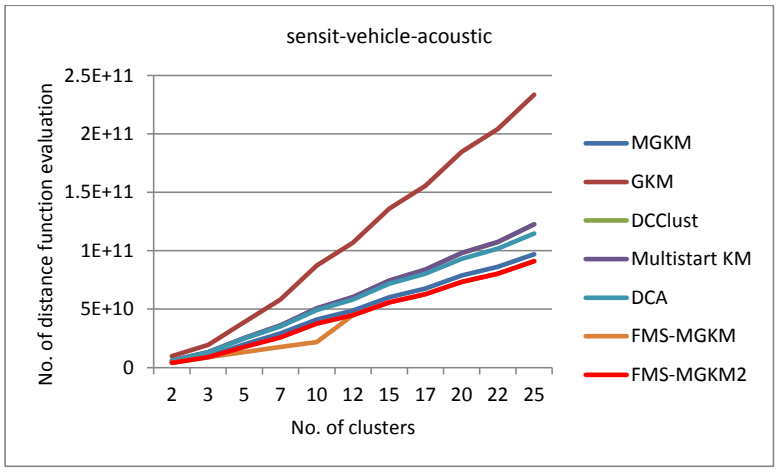

Figure 6.8: The number of distance function evaluations vs the number of clusters: Sensit-vehicle-acoustic data set

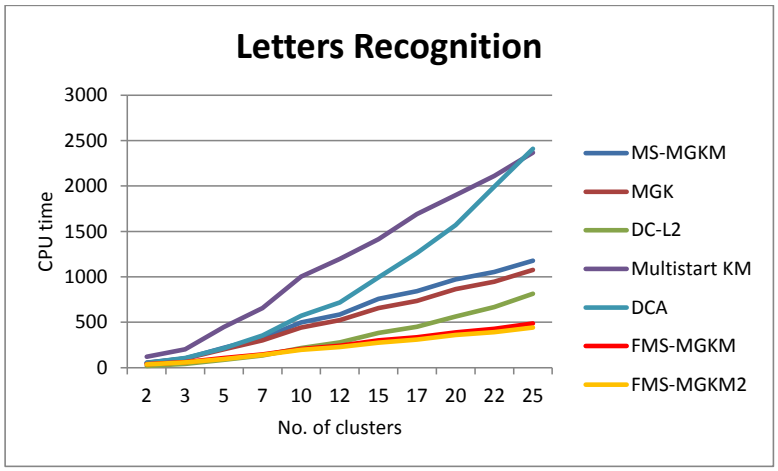

Figure 6.9: The CPU time vs the number of clusters: Letters data set, (this Figure corresponds to Table. 6.4)

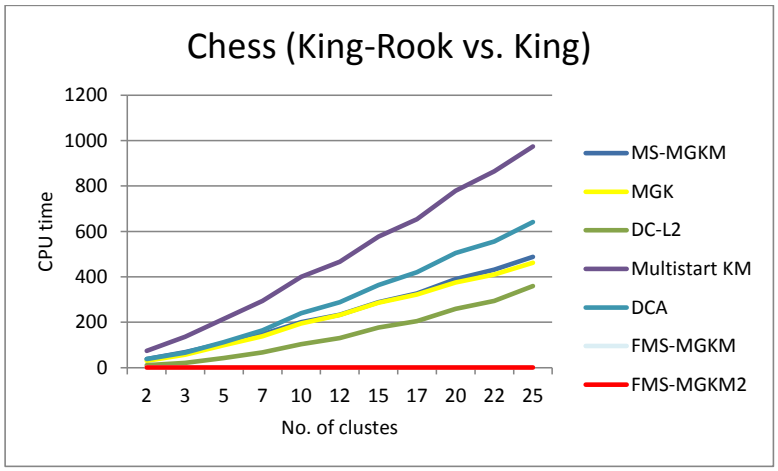

Figure 6.10: The CPU time vs the number of clusters: Chess (King-Rook vs. King) data set,(this Figure corresponds to Table. 6.5)

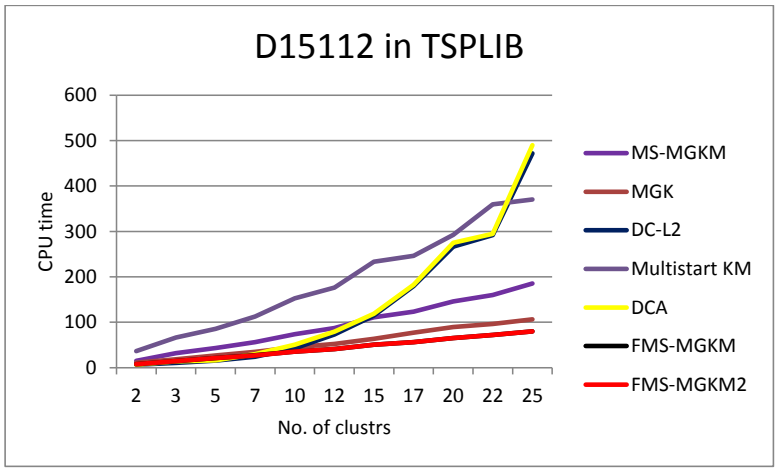

Figure 6.11: The CPU time vs the number of clusters: D15112 data set,(this Figure corresponds to Table. 6.3)

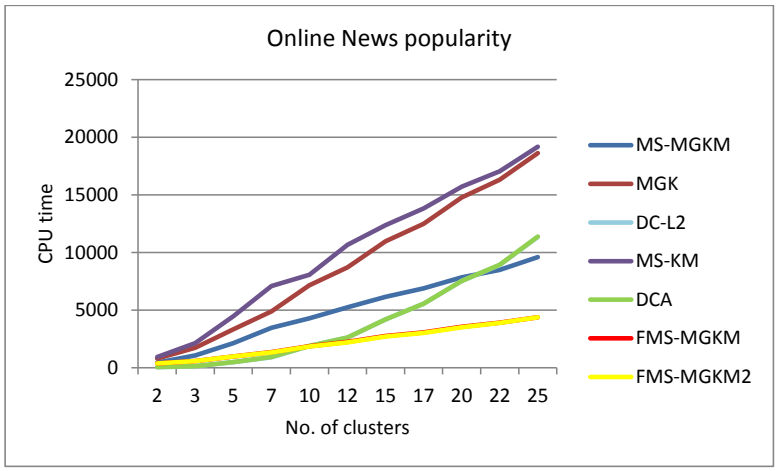

Figure 6.12: The CPU time vs the number of clusters: Online popularity data set ,(this Figure corresponds to Table. 6.6)

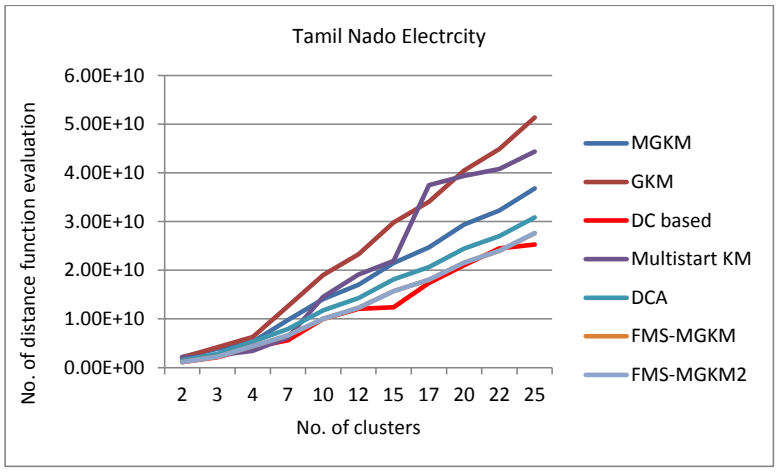

Figure 6.13: The number of distance function evaluations vs the number of clusters: Tamilnadu data set

Table 6.15: Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in comparison with other algorithms, with a breakdown of the numbers for different k values

K.	**No. of times the proposed algorithm shows the best results for given k**	**No. of times the proposed algorithm shows the second best results for given k**
2	3	0
3	3	2
5	5	2
7	6	3
10	10	0
12	10	0
15	10	0
17	10	0
20	10	0
22	10	0
25	10	0

Chapter 7

Computational results: large data sets

In this chapter we present and discuss computational results using very large data sets. All data sets contain only numeric attributes and data sets do not contain missing values. We give a brief description of data sets and then present computational results. These results include optimal values of the cluster function obtained by each algorithm and CPU time required by them. The following algorithms are used for comparison: the GKM algorithm, the MS-MGKM algorithm, the MS-KM algorithm, the DCA algorithm, the DCClust algorithm and two newly proposed algorithms: the FMS-MGKM and FMS-MGKM2 algorithms. The description of these algorithms are given in Chapter 4.

The number of starting points in the MS-KM algorithm is set to 200. In all data sets up to 25 clusters are computed. The computational time by all algorithms is restricted by 30 hours. If an algorithm uses more than 30 hours this is considered as "failure" and in tables it is shown with the letter "F". Algorithms MS-MGKM, DCA and DCClust use the algorithm for computation of starting cluster centers described in the previous chapter. CPU time in all tables are in seconds.

7.1 Data sets

The brief description of very large data sets is given in Table 7.1. In this table we include the number of instances (m, the number of attributes (n) and the total number of entries (N_e) for each data set. The number of instances in these data sets

is from one hundred thousand to over one million and the number of attributes is ranging from 2 to 23.

Table 7.1: The brief description of large data sets.

No.	Data sets	n	m	N_e
1	Shuttle2 Mldata	101500	10	1015000
2	Localization Person Activity	164860	4	494580
3	Online Video Characteristics and Trans coding Time	168268	5	1177876
4	Ijcnn1	191681	23	4408663
5	Skin-non-skin segmentation	245057	4	980228
6	Artificial-2state-sequence-data	250000	14	3500000
7	Cod Coma	488565	9	4397085
8	Online Retail Data Set	541910	2	2167640
9	Algebra 2005 2006 train	809694	3	2429082
10	Phones Accelerometer	1048576	4	6291456

- Localization Data Data for Person Activity database consists of data information of demos of five people implementation various activities. Each person put on four sensors (tags) on ankle left, ankle right, belt and chest respectively whilst going through the same situation five times. [59, 83].

- Online Video Characteristics and Transcoding Time Data set, consists of a million randomly sampled video instances listing 10 fundamental video characteristics. Cluster analysis, which is reported in this work ,is carried on the bitrate(video bitrate in Kbits),height(in pixle), width(in pixles), frame rate and estimated frame rate of the data set, gives insight statistics on characteristics of consumer fundamental online videos, which can be more used to model and optimize parts of a multimedia processing systems. Attributes YouTube video id, duration, bitrate(total in Kbits) and categorical URL are removed from data input file [57, 83].

- Ijcnn1 is a data set which contains information on protein sequence identification. Five categorical features are removed from data set [54, 83].

- Online Retail data set is a multinational data set which consists of all the trades and transactions information, which is happening between 01.12.2010 and 09.12.2011 for a UK-based and registered online retail (which is not a store). Categorical attributes are removed before clustering [63, 83].

- Skin-non skin Segmentation data set is built by using information over Blue, Green and Red color space. Skin and Non skin data set is created by utilizing skin textures of face images of the peoples from a variety of age group, genders, and races [66, 83].

- Cod Comma data set is a collection of data about the possible causes of having lower energy of transition in the regions of the coma samples of Cod [6, 108].

- Algebra training data set provides the information about Algebra courses for the entire school for the year of 2005-2006 [6].

- Phones accelerometer data set contains information of the tilting motion and orientation of a mobile phone, which is collected by phone sensors [6].

7.2 Results

Numerical results with medium size data sets are presented in Tables 7.2-7.11. The best results in all the tables are highlighted using the bold font.

Results for the shuttle2 data set data set presented in Table 7.2 show that the proposed in this thesis algorithms are not accurate in this data set. On the same time they use significantly less CPU time than other algorithms. The FMS-MGKM algorithm failed to produce a good quality solution in this data set, however the FMS-MGKM2 algorithm produced much better solution.

Results for the Localization Data for Person Activity data set presented in Table 7.3 show that the new algorithms produce the best or close to the best solution. Their CPU time is comparable with that of by other algorithms. It is easy that there is no any difference between the performance of the FMS-MGKM and FMS-MGKM2 algorithms in this data set. This is due to the fact that the algorithm for reduction of the number of data points removed only very few points from the data set.

Table 7.4 contains clustering results for the Online Video Characteristics and Transcoding Time data set. These results demonstrate that the proposed algorithm

Table 7.2: Results for shuttle2 data set, (this Table corresponds to the Figure 7.1), (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	1.37E+04	1.37E+04	1.37E+04	**1.36E+04**	1.37E+04	1.38E+04	1.43E+04
3	6.78E+03	6.78E+03	6.78E+03	**6.66E+03**	6.78E+03	6.80E+03	7.98E+03
5	3.78E+03	3.78E+03	3.78E+03	**3.66E+03**	3.78E+03	3.79E+03	6.87E+03
7	2.54E+03	2.54E+03	2.54E+03	**2.42E+03**	2.54E+03	2.83E+03	3.90E+03
10	1.60E+03	1.60E+03	1.60E+03	**1.48E+03**	1.60E+03	1.77E+03	2.51E+03
12	1.31E+03	1.30E+03	1.30E+03	**1.18E+03**	1.30E+03	1.62E+03	2.44E+03
15	1.03E+03	1.02E+03	**1.02E+03**	9.08E+03	1.03E+03	1.38E+03	1.72E+03
17	9.01E+03	8.97E+03	8.85E+03	7.80E+03	8.89E+03	**1.01E+03**	1.27E+03
20	7.43E+03	7.43E+03	7.43E+03	6.26E+03	7.47E+03	9.50E+02	**1.21E+02**
22	6.74E+03	6.61E+03	6.60E+03	5.44E+03	6.64E+03	8.23E+02	**1.19E+02**
25	5.78E+02	5.65E+03	5.65E+02	**4.48E+02**	5.69E+02	7.38E+02	7.70E+01
CPU time (in seconds)							
2	330.11	738.58	223.34	353.11	235.47	6.53	**4.75**
3	545.52	1449.91	488.08	698.52	500.21	7.05	**4.92**
5	1561.59	3204.28	1132.44	1714.59	1144.57	8.11	**5.30**
7	2385.18	4595.58	1789.53	2538.18	1801.66	9.27	**5.78**
10	3758.77	6702.84	2859.94	3911.77	2872.07	10.95	**6.56**
12	4378.61	8027.61	3650.63	4531.61	3661.76	12.13	**6.97**
15	5069.55	9761.14	4833.31	5222.55	4844.44	13.88	**7.61**
17	5489.00	10846.52	5607.42	5642.00	5618.55	15.00	**8.34**
20	6052.54	12458.61	6848.88	6205.54	6866.01	16.84	**9.16**
22	6411.14	13535.42	7672.48	6564.14	7689.61	18.09	**9.75**
25	6975.05	15135.48	8944.22	7128.05	8961.35	20.00	**10.81**

achieved much better results than other algorithms. This is due to the fact that the reduced data set allows one to easily get better cluster structure in the data set. The proposed algorithms use significantly less CPU time than other algorithms in this data set. The FMS-MGKM2 algorithm performs better than the FMS-MGKM algorithm.

Clustering results for the artificial-2state-sequence-data data set are given in Table 7.5. The FMS-MGKM2 algorithm produces high quality solutions which are very close to the best solutions obtained by all algorithms. The FMS-MGKM algorithm also was quite successful however its solutions are not close to the best solutions. One can see that there is a huge difference between the computational time used by the proposed algorithms and other algorithms.

Table 7.6 presents clustering results for the Skin-non skin Segmentation data set. One can see that both proposed algorithms are able to produce good quality solutions, their performance is quite similar. We can also see that these two algorithms require significantly less CPU time than other algorithms.

All other algorithms used in comparison failed to solve clustering problems in the Cod Coma data set in given time frame. This can be seen from Table 7.7. The

Table 7.3: Results for Localization Data for Person Activity data set, (this Table corresponds to the Figure 7.6),(The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**1.04E+05**	**1.04E+05**	**1.04E+05**	**1.04E+05**	**1.04E+05**	**1.04E+05**	**1.04E+05**
3	**7.72E+04**	**7.72E+04**	**7.72E+04**	**7.72E+04**	7.73E+04	**7.72E+04**	**7.72E+04**
5	**5.60E+04**	**5.60E+04**	**5.60E+04**	**5.60E+04**	5.61E+04	**5.60E+04**	**5.60E+04**
7	**4.45E+04**	**4.45E+04**	**4.45E+04**	**4.45E+04**	4.46E+04	**4.45E+04**	**4.45E+04**
10	**3.34E+04**	**3.34E+04**	**3.34E+04**	**3.34E+04**	3.35E+04	**3.34E+04**	**3.34E+04**
12	**3.03E+04**	**3.03E+04**	3.04E+04	**3.03E+04**	3.05E+04	3.04E+04	3.04E+04
15	**2.63E+04**	**2.63E+04**	**2.63E+04**	**2.63E+04**	2.67E+04	**2.63E+04**	**2.63E+04**
17	**2.41E+04**	**2.41E+04**	**2.41E+04**	**2.41E+04**	2.44E+04	**2.41E+04**	**2.41E+04**
20	**2.17E+04**	**2.17E+04**	**2.17E+04**	**2.17E+04**	2.20E+04	2.20E+04	2.20E+04
22	**2.05E+04**	**2.05E+04**	2.06E+04	**2.05E+04**	2.07E+04	2.06E+04	2.06E+04
25	1.93E+04	1.93E+04	**1.89E+04**	1.93E+04	1.92E+04	**1.89E+04**	**1.89E+04**
CPU time (in seconds)							
2	2232.33	881.00	**345.56**	2232.33	419.53	1931.73	1931.73
3	3808.05	1761.28	**758.77**	3808.05	915.08	3313.06	3313.06
5	7080.29	3501.96	**1622.44**	7080.29	1891.91	6345.42	6345.42
7	8604.84	5264.86	**2417.47**	8604.84	2747.83	7467.47	7467.47
10	10427.61	7875.70	**3649.28**	10427.61	4057.70	9365.45	9365.45
12	12135.58	9631.03	**4466.06**	12135.58	4953.22	10786.44	10786.44
15	13943.59	12244.38	**5769.06**	13943.59	6372.55	12598.80	12598.80
17	14756.54	13984.60	**6766.88**	14756.54	7339.22	13695.44	13695.44
20	17201.09	16608.13	**8321.16**	17201.09	8783.89	15341.31	15341.31
22	18218.41	18360.74	**9337.19**	18218.41	9706.11	16414.44	16414.44
25	19180.95	21017.56	**10972.97**	19180.95	11073.14	18289.28	18289.28

FMS-MGKM2 algorithm was able to get the best solution or the solution close to the best one. The FMS-MGKM algorithm also produced good quality of solutions however, they are always worse than those obtained by the FMS-MGKM2 algorithm. CPU time used by both these algorithms is too little for data sets of this size.

All clustering algorithms, except the FMS-MGKM and FMS-MGKM2 algorithms, failed to solve clustering problems in the Online Retail data set. However, we can see from Table 7.8 that the FMS-MGKM algorithm also could not produce any reasonable solutions, except for small number of clusters. CPU time used by these two algorithms is extremely small for this type data sets.

Results for the Algebra training data set are presented in Table 7.9. Again we can see that all clustering algorithms, except algorithms FMS-MGKM and FMS-MGKM2, failed to solve clustering problems in this data set. The FMS-MGKM and FMS-MGKM2 algorithms demonstrated the similar performance, although the FMS-MGKMs algorithm is more accurate than the FMS-MGKM2 algorithms. Again CPU time used by these two algorithms is too small for this type of data sets.

We can see from Table 7.10 that again, all clustering algorithms, except algo-

Table 7.4: Results for Online Video Characteristics and Transcoding Time data set, (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	1.50E+11	1.50E+11	1.50E+11	1.50E+11	1.50E+11	**5.23E+10**	5.47E+10
3	1.11E+11	1.11E+11	1.11E+11	1.11E+11	1.11E+11	3.39E+10	**3.36E+10**
5	6.07E+10	6.07E+10	6.07E+10	6.07E+10	6.07E+10	**1.39E+10**	1.97E+10
7	4.52E+10	4.41E+10	4.52E+10	4.52E+10	4.52E+10	**1.09E+10**	1.46E+10
10	3.29E+10	3.31E+10	3.29E+10	3.29E+10	3.29E+10	**8.42E+09**	9.11E+09
12	2.66E+10	2.66E+10	2.80E+10	2.80E+10	2.80E+10	7.93E+09	**7.53E+09**
15	2.22E+10	2.23E+10	2.22E+10	2.22E+10	2.22E+10	6.61E+09	**6.39E+09**
17	2.01E+10	2.06E+10	2.02E+10	2.02E+10	2.01E+10	6.26E+09	**5.94E+09**
20	1.80E+10	1.81E+10	1.80E+10	1.80E+10	1.80E+10	6.00E+09	**5.12E+09**
22	1.68E+10	1.69E+10	1.68E+10	1.68E+10	1.68E+10	5.84E+09	**3.96E+09**
25	1.52E+10	1.52E+10	1.53E+10	1.53E+10	1.52E+10	5.65E+09	**3.43E+09**
CPU time (in seconds)							
2	2092.48	1337.49	164.48	1.19	181.45	33.61	**32.77**
3	4597.52	3216.40	595.33	885.42	558.89	64.75	**50.92**
5	12067.36	9667.24	1905.78	1619.05	1748.17	115.86	**87.20**
7	17126.50	17756.36	3575.25	4306.89	3181.94	161.08	**128.11**
10	25216.92	24233.99	6334.16	6736.84	5863.72	207.75	**171.52**
12	29452.44	26135.13	8219.20	11510.56	8530.20	234.81	**191.97**
15	34253.50	30080.46	10893.52	15480.39	11648.61	273.25	**224.61**
17	38335.75	31701.26	12678.55	20071.36	14227.36	295.80	**247.73**
20	42799.13	34718.60	15805.55	23572.92	18354.52	330.20	**283.53**
22	45797.11	36475.42	17910.77	28682.36	22050.39	356.55	**312.30**
25	49395.06	38985.68	22464.30	32905.17	44945.09	389.80	**342.02**

rithms FMS-MGKM and FMS-MGKM2, failed to solve clustering problems in this data set. The FMS-MGKM and FMS-MGKM2 algorithms demonstrated the similar performance both in the sense of accuracy and used computational time. Again CPU time used by these two algorithms is too small for this type of data sets.

Results for the Ijcnn1 data set are presented in Table 7.11. The proposed algorithms outperform other algorithms in sense of clustering accuracy. Moreover, they use significantly less computational time than other algorithms. There is no any significant difference between the performance of the algorithms FMS-MGKM and FMS-MGKM2 in this data set.

Figures 7.2, 7.3, 7.4 and 7.5 illustrate dependence of the number of distance functions evaluations on the number of clusters. These figures clearly show that the proposed algorithms: FMS-MGKM and FMS-MGKM2 require significantly less distance function evaluations than any other clustering algorithms used in comparison. The similar picture exists also for all other data sets used in our computational experiments.

Figures 7.1, 7.6, 7.7, 7.7,7.8,7.9 and 7.10 present the dependence of CPU time on the number of clusters for seven very large data sets. These figure clearly show

Table 7.5: Results for artificial-2state-sequence-data set,(The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**5.21E+15**	**5.21E+15**	**5.21E+15**	**5.21E+15**	**5.21E+15**	6.07E+15	**5.21E+15**
3	**2.31E+15**	**2.31E+15**	**2.31E+15**	**2.31E+15**	**2.31E+15**	2.45E+15	**2.31E+15**
5	**8.33E+14**	**8.33E+14**	**8.33E+14**	**8.33E+14**	**8.33E+14**	1.26E+15	8.34E+14
7	**4.25E+14**	**4.25E+14**	**4.25E+14**	4.26E+14	**4.25E+14**	5.47E+14	4.26E+14
10	2.09E+14	2.09E+14	**2.08E+14**	2.09E+14	2.09E+14	2.82E+14	2.09E+14
12	**1.45E+14**	**1.45E+14**	**1.45E+14**	**1.45E+14**	**1.45E+14**	1.59E+14	**1.45E+14**
15	9.31E+13	9.31E+13	**9.28E+13**	9.31E+13	9.31E+13	1.21E+14	9.35E+13
17	7.25E+13	7.25E+13	**7.24E+13**	7.25E+13	7.25E+13	9.70E+13	**7.23E+13**
20	**5.25E+13**	**5.25E+13**	**5.25E+13**	**5.25E+13**	**5.25E+13**	6.48E+13	5.26E+13
22	**4.31E+13**	**4.31E+13**	**4.31E+13**	**4.31E+13**	**4.31E+13**	5.35E+13	4.36E+13
25	3.36E+13	3.36E+13	3.36E+13	3.36E+13	3.36E+13	3.86E+13	**3.33E+13**
CPU time (in seconds)							
2	7374.89	**8.88**	2881.07	7374.89	5101.94	14.41	14.03
3	11385.53	14.93	5354.55	11385.53	9336.02	14.48	**14.13**
5	22678.83	26.36	11038.74	22678.83	17076.05	14.73	**14.36**
6	28351.42	34.35	13968.88	28351.42	23429.53	15.08	**14.67**
10	49232.34	56.94	24217.78	49232.34	33815.97	15.77	**15.28**
12	58995.98	68.61	29630.69	58995.98	39959.72	16.31	**15.8**
15	71898.74	84.12	37735.92	71898.74	51655.33	17.27	**16.73**
17	80706.49	94.15	43352.38	80706.49	58650.28	17.98	**17.47**
20	51968.23	109.97	51968.23	51968.23	69267.66	19.27	**18.72**
22	58295.53	120.26	58295.53	58295.53	80291.74	20.2	**19.67**
25	68716.39	136.35	68716.39	68716.39	96241.24	21.77	**21.25**

that the proposed algorithms: FMS-MGKM and FMS-MGKM2 for most of the datasets, require significantly less CPU time than any other clustering algorithms used in comparison. One can see similar dependence also for other very large size data sets.

Table 7.12 demonstrates the overall performance of the proposed algorithm.

Table 7.13 demonstrates the overall performance of the proposed algorithm with a with a breakdown of the numbers for different k values .

7.3 Summary

In this chapter, we applied the proposed algorithms to solve the clustering problem in ten large and very large data sets. These data sets contain between 100000 and over 1000000 instances. We computed up to 25 clusters and compared results with other incremental clustering algorithms as well as with the multi-start k-means algorithm.Overall exactly nine times, the proposed algorithms require less CPU time than other algorithms.

Table 7.6: Results for Skin-non skin Segmentation data set,(this Table corresponds to the Figure 7.7),(The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	**1.32E+09**	**1.32E+09**	**1.32E+09**	**1.32E+09**	**1.32E+09**	1.58E+09	1.57E+09
3	**8.94E+08**	**8.94E+08**	**8.94E+08**	**8.94E+08**	**8.94E+08**	1.09E+09	1.12E+09
5	5.18E+08	**5.02E+08**	**5.02E+08**	5.18E+08	**5.02E+08**	6.21E+08	6.08E+08
7	3.61E+08	3.63E+08	3.63E+08	3.61E+08	3.63E+08	**4.51E+08**	4.59E+08
10	**2.51E+08**	**2.51E+08**	**2.51E+08**	**2.51E+08**	**2.51E+08**	3.41E+08	3.55E+08
12	**2.14E+08**	**2.14E+08**	**2.14E+08**	**2.14E+08**	2.15E+08	2.75E+08	2.72E+08
15	1.71E+08	**1.70E+08**	**1.70E+08**	**1.70E+08**	**1.70E+08**	2.23E+08	2.16E+08
17	1.50E+08	**1.48E+08**	**1.48E+08**	1.50E+08	**1.48E+08**	2.01E+08	2.02E+08
20	**1.26E+08**	1.28E+08	1.28E+08	**1.26E+08**	1.28E+08	1.67E+08	1.62E+08
22	**1.15E+08**	1.17E+08	1.16E+08	**1.15E+08**	1.17E+08	1.61E+08	1.52E+08
25	**1.02E+08**	1.03E+08	1.03E+08	**1.02E+08**	1.03E+08	1.52E+08	1.34E+08
CPU time (in seconds)							
2	4230.51	5010.19	846.02	8331.03	970.31	**216.58**	262.89
3	7633.08	10768.66	1484.54	15136.16	1825.25	**308.36**	333.52
5	12313.42	16479.37	2596.12	24496.84	3380.70	451.61	**451.41**
7	14563.29	20957.30	4045.54	28996.58	5135.22	581.38	**566.94**
10	18130.61	27357.77	5607.72	36131.22	7451.09	727.02	**684.06**
12	19813.58	31226.22	6960.31	39497.16	8941.08	841.56	**787.48**
15	22034.95	37881.38	8701.50	43939.90	11946.05	994.64	**911.73**
17	23426.09	41655.43	9987.12	46722.18	13862.67	1080.00	**983.20**
20	25633.29	46495.40	11539.28	51136.58	16675.11	1215.69	**1095.92**
22	26796.21	49691.59	12664.96	53462.42	18759.27	1349.86	**1170.95**
25	28461.48	54499.17	14575.80	56792.96	22161.98	1497.52	**1276.97**

We can note that the performance of all clustering algorithms used in this comparison are different from that in small and medium size data sets. First, the algorithm which do not use data points weights is more accurate than the algorithm which uses such weights. There is any significant between these algorithms in large data sets.

In some data sets these two algorithms were only algorithms which were able to produce a solution in a reasonable time (less than 30 hours). Other algorithms failed to produce any solution in a reasonable time in some cases. This algorithm cannot be directly applied to solve clustering algorithms in very large data sets.

In most cases the proposed algorithms require less, and in some instances significantly less, CPU time than other algorithms used for comparison. Overall, results from this chapter allow us to conclude that the proposed clustering algorithms are accurate and efficient clustering algorithms in very large data sets used in computations.

Table 7.7: Results for Cod Coma data set,(The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	9.78E+15	Fail	9.78E+15	9.57E+15	9.78E+15	1.17E+16	**9.82E+15**
3	4.36E+15	Fail	4.36E+15	4.85E+15	4.36E+15	4.65E+15	**4.39E+15**
5	1.58E+15	Fail	1.58E+15	1.67E+15	1.58E+15	2.56E+15	**1.59E+15**
7	Fail	Fail	8.11E+14	Fail	8.11E+14	1.43E+15	**8.20E+14**
10	Fail	Fail	4.01E+14	Fail	4.01E+14	4.81E+14	**4.10E+14**
12	Fail	Fail	2.80E+14	Fail	2.80E+14	3.09E+14	**2.84E+14**
15	Fail	Fail	1.80E+14	Fail	1.80E+14	2.32E+14	**1.83E+14**
17	Fail	Fail	1.41E+14	Fail	1.41E+14	1.56E+14	**1.42E+14**
20	Fail	Fail	1.03E+14	Fail	1.03E+14	1.29E+14	**1.03E+14**
22	Fail	Fail	Fail	Fail	Fail	1.18E+14	**8.57E+13**
25	Fail	Fail	Fail	Fail	Fail	7.65E+13	**6.56E+13**
CPU time (in seconds)							
2	23146.93	Fail	3985.09	23143.33	3985.09	16.23	**16.11**
3	36958.81	Fail	8093.16	36848.71	8093.16	16.31	**16.19**
5	89271.76	Fail	15605.59	89373.66	15605.59	16.56	**16.45**
7	Fail	Fail	22772.67	Fail	22772.67	16.92	**16.81**
10	Fail	Fail	33702.10	Fail	33702.10	17.61	**17.56**
12	Fail	Fail	41018.95	Fail	41018.95	18.20	**18.17**
15	Fail	Fail	52138.73	Fail	52138.73	19.31	**19.27**
17	Fail	Fail	60086.86	Fail	60086.86	20.16	**20.13**
20	Fail	Fail	70953.01	Fail	70953.01	21.66	**21.56**
22	Fail	Fail	Fail	Fail	Fail	22.77	**22.67**
25	Fail	Fail	Fail	Fail	Fail	24.63	**24.52**

Table 7.8: Results for Online Retail data set, (The best results are highlighted)

Cluster function values		
k	$FMS-MGKM$	FMS-MGKM2
2	**3.09E+10**	1.01E+11
3	**2.98E+10**	7.97E+10
5	2.87E+10	**1.70E+10**
7	2.84E+10	**1.85E+10**
10	2.79E+10	**3.82E+09**
12	2.78E+10	**3.31E+09**
15	2.78E+10	**1.39E+09**
17	2.78E+10	**1.31E+09**
20	2.78E+10	**1.23E+09**
22	2.78E+10	**1.15E+09**
25	2.77E+10	**7.08E+08**
CPU time (in seconds)		
2	0.34	**0.23**
3	0.41	**0.27**
5	0.58	**0.36**
7	0.80	**0.48**
10	1.23	**0.75**
12	1.53	**1.02**
15	2.06	**1.41**
17	2.50	**1.70**
20	3.13	**2.25**
22	3.61	**2.70**
25	4.47	**3.38**

Table 7.9: Results for Algebra training data set, (this Table corresponds to the Figure 7.10), (The best results are highlighted)

Cluster function values		
k	$FMS-MGKM$	FMS-MGKM2
2	2.77E+16	**1.64E+16**
3	1.07E+16	**7.24E+15**
5	3.22E+15	**2.70E+15**
7	1.58E+15	**1.40E+15**
10	7.51E+14	**6.74E+14**
12	5.23E+14	**4.69E+14**
15	3.33E+14	**3.05E+14**
17	2.70E+14	**2.38E+14**
20	1.82E+14	**1.73E+14**
22	1.52E+14	**1.43E+14**
25	1.16E+14	**1.09E+14**
CPU time (in seconds)		
2	7.48	**6.97**
3	7.58	**7.02**
5	7.81	**7.27**
7	8.19	**7.64**
10	8.97	**8.41**
12	9.63	**9.08**
15	10.83	**10.33**
17	11.80	**11.30**
20	13.39	**12.81**
22	14.61	**13.89**
25	16.61	**16.00**

Table 7.10: Results for Phones Accelrometer data set, (this Table corresponds to the Figure 7.8), (The best results are highlighted)

Cluster function values		
k	$FMS-MGKM$	FMS-MGKM2
2	4.09E+15	**2.32E+15**
3	1.39E+15	**9.93E+14**
5	4.32E+14	**3.76E+14**
7	2.10E+14	**1.92E+14**
10	9.69E+13	**9.02E+13**
12	6.54E+13	**6.21E+13**
15	4.14E+13	**4.00E+13**
17	3.27E+13	**3.15E+13**
20	2.38E+13	**2.34E+13**
22	**1.92E+13**	1.97E+13
25	1.50E+13	**1.47E+13**
CPU time (in seconds)		
2	9.41	**7.67**
3	9.50	**7.73**
5	9.80	**7.94**
7	10.16	**8.25**
10	11.02	**8.92**
12	11.77	**9.50**
15	13.13	**10.55**
17	14.16	**11.39**
20	16.00	**12.86**
22	17.38	**13.98**
25	19.80	**15.86**

Table 7.11: Results for Ijcnn1 data set, (The best results are highlighted)

Cluster function values							
k	$MS-MGKM$	GKM	$DCClust$	$MS-KM$	DCA	$FMS-MGKM$	FMS-MGKM2
2	2.58E+05	2.68E+05	2.58E+05	2.58E+05	2.58E+05	**1.04E+05**	**1.04E+05**
3	2.40E+05	2.50E+05	2.40E+05	2.40E+05	2.40E+05	**7.72E+04**	**7.72E+04**
5	2.03E+05	2.13E+05	2.03E+05	2.03E+05	2.03E+05	**5.60E+04**	**5.60E+04**
7	1.65E+05	1.75E+05	1.65E+05	1.65E+05	1.66E+05	**4.45E+04**	**4.45E+04**
10	1.11E+05	1.21E+05	1.11E+05	1.11E+05	1.11E+05	**3.34E+04**	**3.34E+04**
12	8.97E+04	9.98E+04	8.97E+04	8.97E+04	8.97E+04	3.04E+04	**2.99E+04**
15	8.29E+04	9.30E+04	8.11E+04	8.29E+04	8.14E+04	**2.63E+04**	**2.63E+04**
17	7.72E+04	8.73E+04	7.55E+04	7.72E+04	7.70E+04	**2.41E+04**	**2.41E+04**
20	6.87E+04	7.88E+04	6.71E+04	6.87E+04	6.78E+04	2.20E+04	**2.17E+04**
22	6.31E+04	7.32E+04	6.22E+04	6.31E+04	6.11E+04	**2.06E+04**	**2.06E+04**
25	5.64E+04	6.65E+04	5.62E+04	5.64E+04	5.78E+04	**1.89E+04**	**1.89E+04**
CPU time (in seconds)							
2	2683.84	2715.26	2340.75	2693.84	3880.77	**2379.20**	2401.53
3	6354.87	6386.29	6161.59	6354.87	8695.66	**3931.09**	4090.08
5	13018.66	13050.08	12755.78	13028.66	18086.95	8643.22	**7613.30**
7	18659.83	18691.25	20808.58	18651.83	27319.53	10332.44	**9383.66**
10	26296.98	26328.40	30904.59	26296.98	40431.33	12844.81	**11722.19**
12	30949.80	30981.22	36311.08	30959.81	49553.89	14270.36	**13574.55**
15	39028.28	39059.70	44229.14	39028.28	64377.05	16554.77	**15192.22**
17	44673.13	44704.55	49372.45	44773.13	74991.75	17808.69	**16480.17**
20	52863.93	52895.35	57229.23	52863.93	92377.83	19783.08	**18605.63**
22	58058.72	58090.14	62480.70	58058.72	105163.36	20822.22	**19487.72**
25	65849.21	65880.63	70285.09	66849.21	120830.17	22524.67	**21193.34**

Table 7.12: Supporting Metrics/Table for counting how many cases the proposed algorithm produces the best(first) and the second best result in comparison with other algorithms.

No.	**Parameters/Approach**	**Proposed algorithm**
1	Efficiency	**9 times gives the best result and few times gives the second best result**
2	Computation	Requires low memory and small amount of calculations
3	Performance	Fast and accurate in comparison with the existing algorithms (finds similar objective function values but much superior in CPU time)

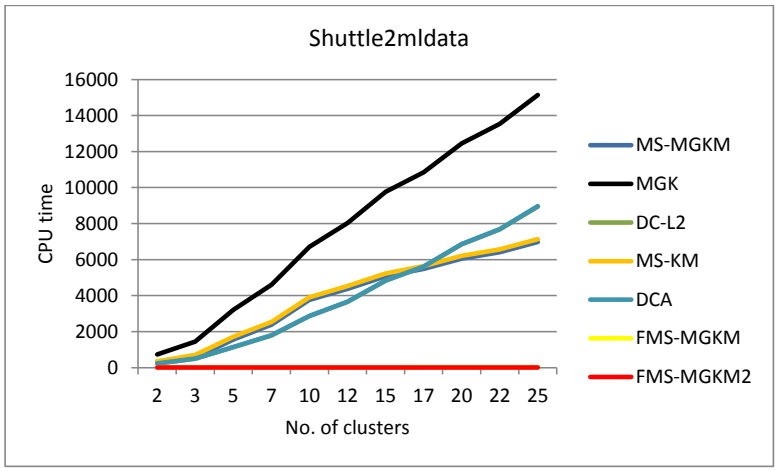

Figure 7.1: The CPU time vs the number of clusters: Shuttle2 Mldata data set, (this Figure corresponds to Table. 7.2)

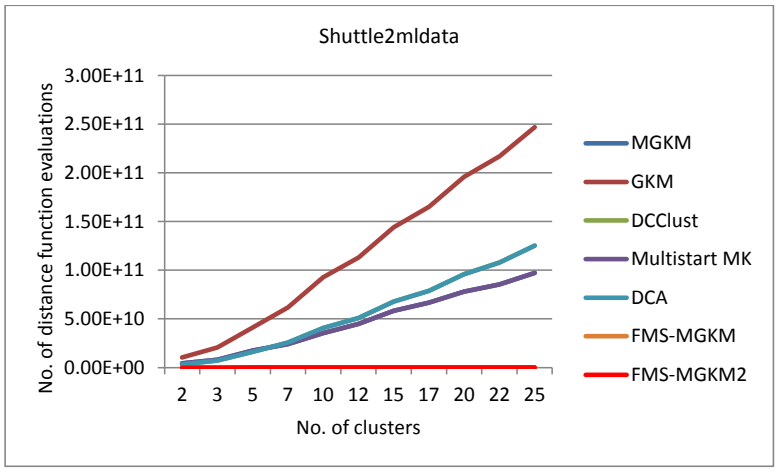

Figure 7.2: The number of distance function evaluations vs the number of clusters: Shuttle2 Mldata data set

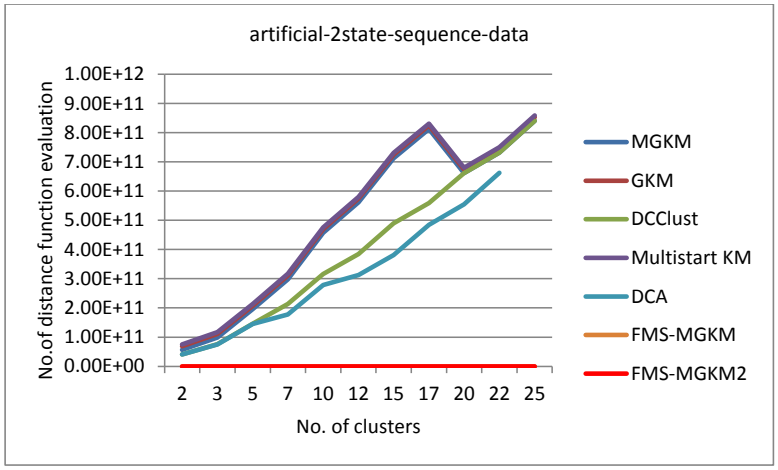

Figure 7.3: The number of distance function evaluations vs the number of clusters: Artificial-2state-sequence-data data set

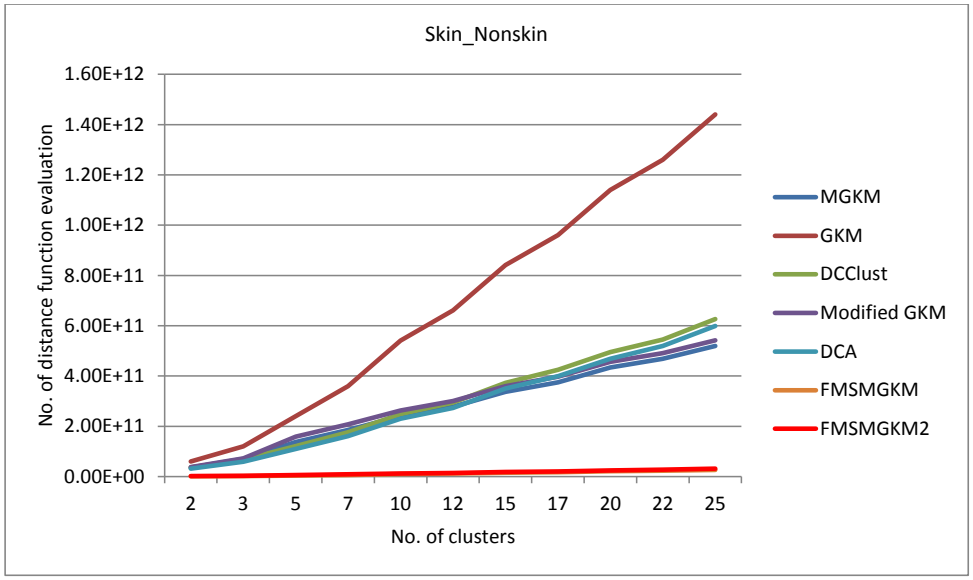

Figure 7.4: The number of distance function evaluations vs the number of clusters: Skin-non-skin segmentation data set

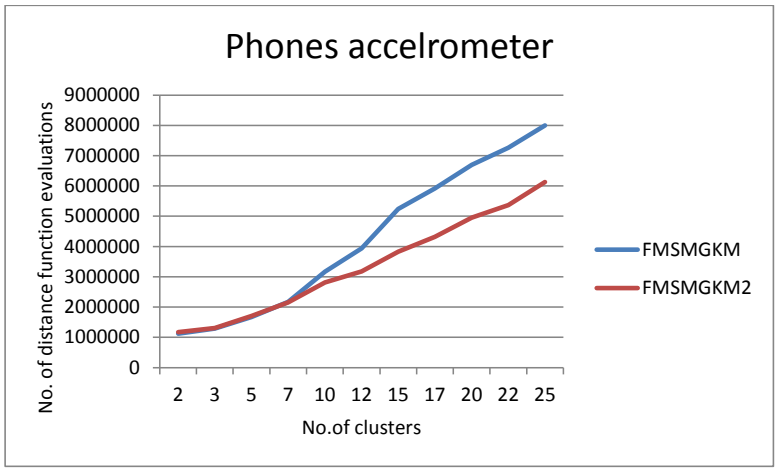

Figure 7.5: The number of distance function evaluations vs the number of clusters: Phones Accelerometer data set

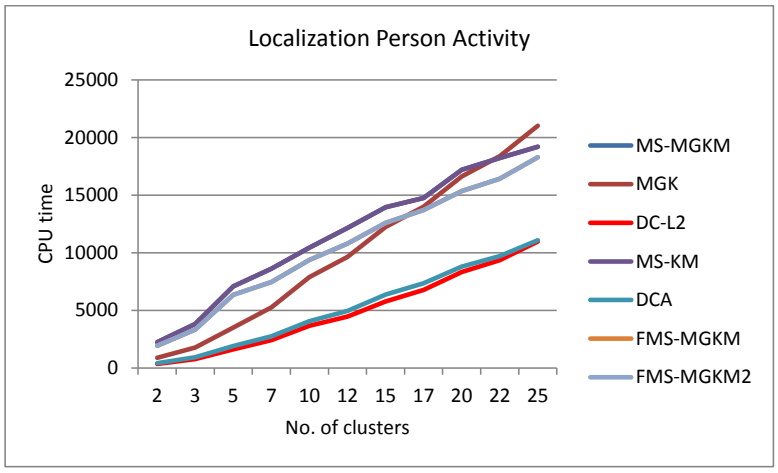

Figure 7.6: The CPU time vs the number of clusters: Localization for person Activity data set, (this Figure corresponds to Table. 7.3)

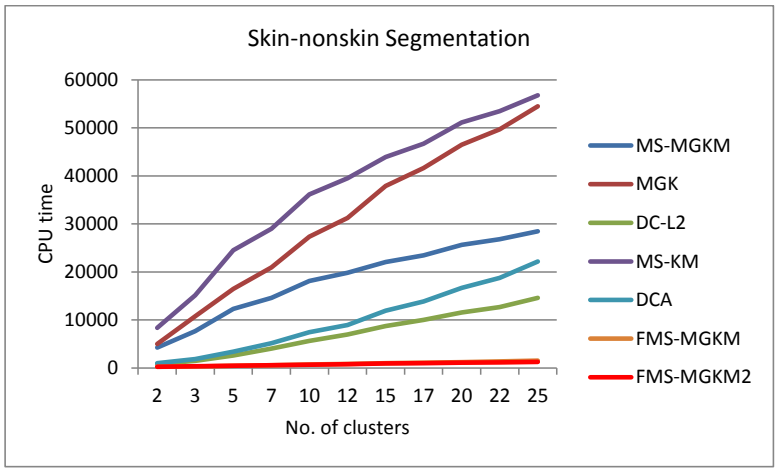

Figure 7.7: The CPU time vs the number of clusters:Skin-non-skin segmentation data set, (this Figure corresponds to Table. 7.6)

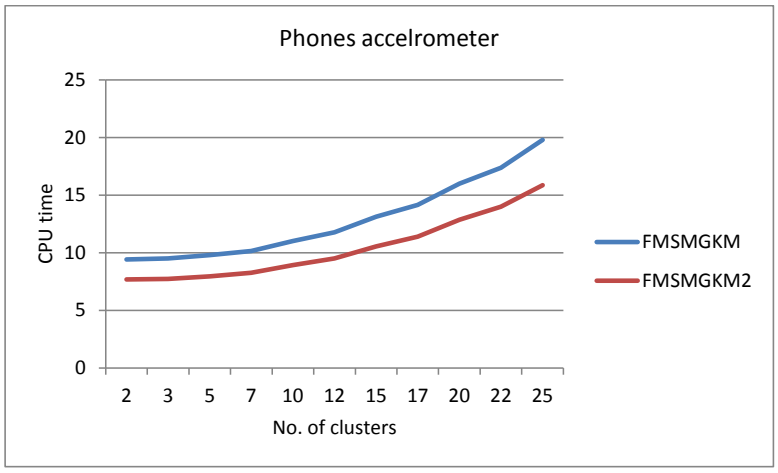

Figure 7.8: The CPU time vs the number of clusters:Phones Accelerometer data set (this Figure corresponds to Table. 7.10)

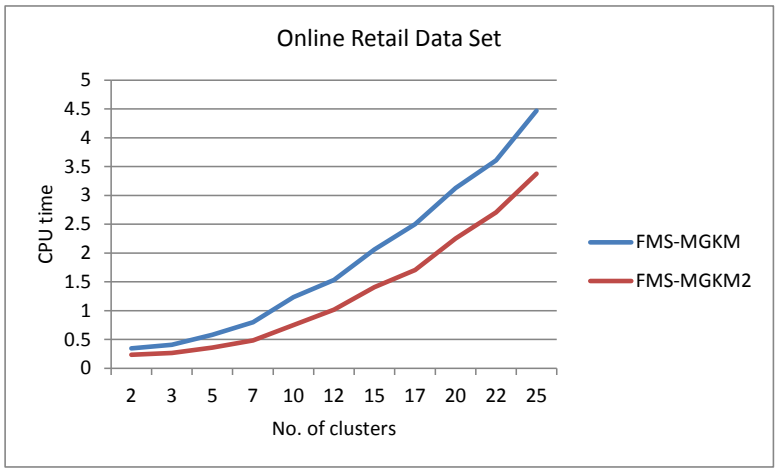

Figure 7.9: The CPU time vs the number of clusters:Online Retail Dataset,(this Figure corresponds to Table. 7.8)

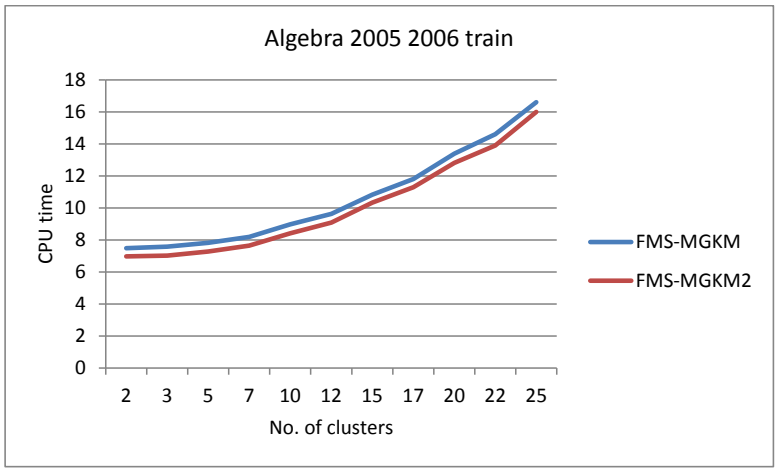

Figure 7.10: The CPU time vs the number of clusters:Algebra 2005 2006 train Dataset, (this Figure corresponds to Table. 7.9)

Table 7.13: Supporting Metrics/Table for counting how many cases out of how many cases the proposed algorithms achieve the best(first) and the second best result in comparison with other algorithms, with a breakdown of the numbers for different k values

K.	**No. of times the proposed algorithm shows the best results for given k**	**No. of times the proposed algorithm shows the second best results for given k**
2	6	3
3	9	0
5	9	0
7	9	0
10	9	0
12	9	0
15	9	0
17	9	0
20	9	0
22	9	0
25	9	0

Chapter 8

Conclusions and future work

In this thesis fast and accurate clustering algorithms have been studied to solve the minimum sum-of-squares clustering problems in very large data sets. In this study the term "very large data set" means that the data set contains hundreds of thousands or millions points and/or maximum hundreds of attributes. However, we assume that this data set can be stored in random access memory of a computer.

Most clustering algorithms, including those based on optimization techniques, and their applicability to very large data sets have been discussed. It has been noted that most of these algorithms are not applicable to solve clustering problems in such data sets. Because they may require a large computational effort and may not produce any solution in a reasonable time. Therefore, there is a need to develop new clustering algorithms which are real time and accurate algorithms for clustering in very large data sets.

According to objectives of this study we introduced the algorithm to reduce the number of data points in a data set. This algorithm reads the whole data only twice. This algorithm identifies dense areas in a data set and replace these areas by their representative points by assigning them weights. Weights of representative points are defined as the number of points from some its vicinity. This leads to generation of data sets with weights and clustering algorithms are modified to solve clustering problems in such data sets.

Most clustering algorithms are based on local search techniques and therefore, their success strongly depends on the choice of starting cluster centers. Using the nonconvex nonsmooth optimization model of the clustering problem we developed the algorithm to generate good starting cluster centers. This algorithm use data

points and then generates starting cluster centers from the whole search space.

Using the algorithm for reduction of data points and the algorithm for generating starting cluster centers we design new clustering algorithms for solving clustering algorithm in very large data sets. These algorithms are based on the incremental approach, they build clusters gradually starting from one cluster which is the whole data sets. New clustering algorithms are extensions of optimization based incremental clustering algorithms for data sets with weights. These algorithms involve also some schemes to reduce computational effort.

The implementation of the proposed algorithms is discussed and algorithms were evaluated using small size (with tens of thousands of data points), medium size (with up to a few hundreds of thousands of data points) and very large (with several hundreds of thousands and millions of data points) data sets. These data sets contains from very few to more than one hundred attributes.

Results clearly demonstrate new algorithms are very fast even in very large data sets. Results also demonstrate that these algorithms are real-time clustering algorithms. Their results were compared with those obtained using several center-based clustering algorithms based on optimization techniques.

In conclusion, we can say that this research demonstrate how the existing clustering algorithms can be scaled up to solve clustering problems in very large data sets. This research develops algorithms which are accurate, efficient and real-time clustering algorithms for very large data sets.

In this thesis we did not consider the problem of clustering in data sets which cannot be stored in the random access memory of a computer. These problems can be considered as directions of future research. Two main directions can be identified here:

1. Most clustering algorithms have good potential for parallelization. The use of many processors in supercomputers will significantly accelerate the convergence of such algorithms. Here we can mention two possibilities for parallelization. One possibility is that to divide the data set into many pieces and pass each piece to one processor and to solve the clustering separately for each piece of data. Then special techniques should be developed to merge clustering results from each processor.

2. This direction does not assume the presence of many processors, that is, in

this case the clustering problem is solved using normal computer not super-computer. The idea is to consider the data set as a streamline data set, divide it into many pieces and at each iteration the clustering problem for one piece-wise of data. In this case special techniques should be developed to identify most informative points at each iteration to pass them to the next iteration.

Bibliography

[1] R. Dubes and A.K. Jain, Clustering techniques: the user's dilemma, *Pattern Recognition*, 8(1976), 247-260.

[2] K.R. Rao, Data Mining and Clustering Techniques, *DRTC Workshop on Semantic Web,* 11(2003), 8-10.

[3] P. Hanjoul, D. Peeters, A comparison of two dual-based procedures for solving the *p*-median problem, *European Journal of Operational Research,* 20(1985), 387-396.

[4] A.E. Xavier and A.A.F.D. Oliveira, Optimal covering of plane domains by circles via hyperbolic smoothing, *Journal of Global Optimization,* 31(2005), 493-504.

[5] UCI repository of machine learning databases, http://www.ics.uci.edu/mlearn/MLRepository.html.

[6] Machine learning repository mldata.ord, http://mldata.org/repository/data/

[7] T. M. Usha , Knowledging on Tamil Nadu electricity board (TNEB) and electricity load demand forecasting by Gaussian processes using real time data,Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), (2013).

[8] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski and T. Ideker, Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res,* 13(2003),2498–2504.

[9] A.M. Bagirov, A. Al Nuaimat and N. Sultanova, Hyperbolic smoothing method for minimax problems, *Optimization,* 2012.

[10] D.Michie,The Fifth Generation's Unbridged Gap:In Rolf Herken (Ed.) The Universal Turing Machine: A Half-Century Survey,*Oxford University Press,*1988, 466-489.

[11] R. Xu and D. Wunsch, Survey of clustering algorithms,*IEEE Transactions on Neural Networks,* 16(2005), 645-678.

[12] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/pla85900.tsp

[13] B.Johnson, R.Tateishi, N. Hoan, A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees,*International Journal of Remote Sensing,* 34 (2013), 6969-6982.

[14] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.In Decision Support Systems, *Elsevier,*47(2009),547-553.

[15] Leo Breiman, Jerome H. Friedman, Adam Olshen, Jonathan Stone. Classification and Regression Trees, 1984.

[16] Gunduz, G. Fokoue, E., 2013.

[17] D. Webb, Efficient piecewise linear classifiers and applications, A PhD Thesis, Federation University, Science Technology and Engineering Faculty,2010.

[18] Pnar Tfekci, Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, *International Journal of Electrical Power and Energy Systems,* 60(2014),126-140.

[19] M. Kantardzic, *Data Mining: Concepts, Models, Methods and Algorithms.* John Wiley & Sons, Inc., NY, 2002.

[20] S. Moro, P. Cortez and P. Rita. A Data Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, *Elsevier,* 62(2014) 22-31.

[21] R. C. B.Madeo, C. A. M.Lima , S. M.Feres . Gesture Unit Segmentation using Support Vector Machines: Segmenting Gestures from Rest Positions,*Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC),* 2013,46-52.

[22] A.Vergara,S.Vembu,T.Ayhan,L. Homer, Chemical gas sensor drift compensation using classifier ensembles, Sensors and Actuators *Chemical,*2012.

[23] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[24] M. Bain, Learning optimal chess strategies,*Institute for New Generation Computer Technology, Tokyo, Japan,*1994.

[25] A. Asuncion and D. Newman, UCI machine learning repository, 2007.

[26] K. Fernandes, P. Vinagre and P. Cortez, A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News,*Proceedings of the 17th EPIA - Portuguese Conference on Artificial Intelligence,*2015.

[27] G. Reinelt, TSP-LIB-A Traveling Salesman Library, ORSA J. Comput. 3(1991), 319-350.

[28] R. Kimball. The data warehouse toolkit: practical techniques for building dimensional data warehouses, *John Wiley & Sons, Inc., NY,USA,* 1996.

[29] P. W. Frey and D. J. Slate,Letter Recognition Using Holland-style Adaptive Classifiers, *Machine Learning,* 6(1991).

[30] J. Einbinder, K. Scully, R. Pates, J. Schubart and R. Reynolds, Case study: a data warehouse for an academic medical center, *J. Health Inf. Manag.,* 15(2001),165-175.

[31] G. Batista and M. Monard, An analysis of four missing data treatment methods for supervised learning, *Applied Artificial Intelligence,* 17(2003),519-533.

[32] S. Zhang, C. Zhang and Q. Yang, Data preparation for data mining. *Applied Artificial Intelligence,* 17(2002),375-381.

[33] M. Duarte and Y. H. Hu, Vehicle classification in distributed sensor networks,*Journal of Parallel and Distributed Computing,* 64(2004),826-838.

[34] H. Liu and H. Motoda, *Instance Selection and Construction for Data Mining.* Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[35] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins, A Constant Time Collaborative Filtering Algorithm: Information Retrieval, 4(2001), 133-151.

[36] I. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques,* Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

[37] L. Shalabi, Z.Shaaban, and B. Kasasbeh, Data mining: A preprocessing engine. *Journal of Computer Science,* 2(2006),735-739.

[38] T. Zhang and R. Ramakrishnan, BIRCH: An Efficient Data Clustering Databases Method for Very Large Databases. *ACM SIGMOD Record,* 1(1996),103-114.

[39] D.E. Goldberg and K. Deb, A comparison of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms,* edited by G. J. E. Rawlins,1991,69-93.

[40] M. Gen and R. Cheng, Genetic algorithms & engineering design. Wiley, New York, 1997.

[41] B. Basturk and D. Karaboga, An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization, *IEEE Swarm Intelligence Symposium,* 2006, 12 - 14.

[42] H.D. Meng, Y.C. Song, F.Y. Song and S.L.Wang, Clustering for Complex and Massive Data, *2009 International Conference on Information Engineering and Computer Science*, 2009, 1–4.

[43] S. Kantabutra and A. Couch, Parallel K-means clustering algorithm on NOWs. *NECTEC Technical journal,* 1(2000),243-247.

[44] S. Nittel, K. Leung, and A. Braverman, Scaling clustering algorithms for massive data sets using data streams. *In Proceedings of the 19th International Conference on Data Engineering,* March, 2003,5-8.

[45] F. Farnstrom, J. Lewis, and C. Elkan, Scalability for Clustering Algorithms Revisited. *ACM SIGKDD Explorations Newsletter,* 2(2000),51–57.

[46] N. Alex, A. Hasenfuss and B. Hammer, Patch clustering for massive data sets. *Neurocomputing,* 72(2009), 1455-1469.

[47] J. Nievergelt and H. Hinterberger, The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on,* 9(1984), 38-71.

[48] H. Nagesh, S. Goil and A. Choudhary, Adaptive Grids for Clustering Massive Data Sets. *In Proceedings of the 1st SIAM ICDM, Chicago, IL,* 477(2001), 1–17.

[49] E. Schikuta, Grid-Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets, *Proceedings of the 13th International Conference on Pattern Recognition,* 2(1996),101–105.

[50] E. Schikuta and M. Erhart. BANG-clustering: A novel grid-clustering algorithm for huge data sets. *Advances in Pattern Recognition,* 1998,867-874.

[51] E. Wmma, A new shifting grid clustering algorithm, *Pattern Recognition,* 37(2004),503-514.

[52] K. Woo, GETIT: a fast and intelligent subspace clustering algorithm using dimension voting, *Information and Software Technology,* 46(2004),255-271.

[53] L. Parsons and E. Haque, Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter,* 6(2004),90–105.

[54] Chih-Chung Chang and Chih-Jen Lin,IJCNN 2001 challenge: Generalization ability and text decoding. In Proceedings of IJCNN. IEEE, 2001.

[55] H. Nagesh, S. Goil and A. Choudhary, A scalable parallel subspace clustering algorithm for massive data sets. *Proceedings 2000 International Conference on Parallel Processing,*,2000, 477–484.

[56] S. Goil, H. Nagesh and A. Choudhary, MAFIA: Efficient and scalable subspace clustering for very large data sets. *In ICDE (International Conference on Data Engineering),* Germany, 2001.

[57] Tewodors Deneke, Analysis and Transcoding Time Prediction of Online Videos Multimedia (ISM), *IEEE International Symposium,*2015,14-16.

[58] C.H. Cheng, A.W. Fu and Y. Zhang. Entropy-based subspace clustering for mining numerical data. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining,* 1999, 84-93.

[59] B. Kaluza, V. Mirchevska, E. Dovgan, M. Lustrek, M. Gams, An Agent-based Approach to Care in Independent Living, International Joint Conference on Ambient Intelligence (AmI-10), Malaga, Spain, 1999.

[60] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc and J. Park, Fast algorithms for projected clustering. *ACM SIGMOD Record,* 28(1999),61-72.

[61] J. Kennedy and R.C. Eberhart, Particle swarm optimization. *Proc. IEEE int'l conf. on neural networks,*1(1995), 1942-1948.

[62] R.C. Eberhart and Y. Shi, Comparison between genetic algorithms and particle swarm optimization. *Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf.,* Springer-Verlag, Berlin, San Diego, CA., 1998.

[63] Daqing Chen, Sai Liang Sain, and Kun Guo, Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining, *Journal of Database Marketing and Customer Strategy Management,*19 (2012), 197-208.

[64] S.Jeffrey, A collection of data sets used in the book "Analyzing Categorical Data," New York, 2003.

[65] Q. Bai, Particle swarm optimization algorithm, *Computer and Information System,* 2010.

[66] Rajen B. Bhatt, Gaurav Sharma, Abhinav Dhall, Santanu Chaudhury, Efficient skin region segmentation using low complexity fuzzy decision tree model, *IEEE-INDICON,* 2009,16-18.

[67] M. Vahdat, A. Ghio, L. Oneto, D. Anguita, M. Funk, M. Rauterberg, Advances in learning analytics and educational data mining, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2015).

[68] M. Dorigo, and L.M. Gambardella, Ant colonies for the traveling salesman problem. *BioSystems,* 43(1997), 73-81.

[69] Guo.Chenjuan, Yu Ma, Bin Yang and S. Jensen, Evaluating models of vehicular environmental impact, SIGSPATIAL/GIS 2012, 269-278.

[70] M. Dorigo and Ch. Blum, Ant Colony optimization theory: A survey, *Science Direct,* 344(2005), 243-278.

[71] Zar Chi Su and May Aye Khan, Solving Travel Salesman Problem using improved ant colony optimization algorithm, *International Journal of Information and Education Technology,* 5, 2011.

[72] X-S. Yang and S. Deb, Cuckoo search via Levy flights. In: *Proceedings of World Congress on Nature & Biologically Inspired Computing.IEEE Publications,*2009, 210-214.

[73] R.B. Payne, M.D. Sorenson and K. Klitz, *The Cuckoos.* Oxford University Press, New York, 2005.

[74] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere, The Million Song Dataset, *In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[75] K.S.Al-Sultan, A tabu search approach to the clustering problem, *Pattern Recognition,*1995, 28(9), 1443-1451.

[76] K.S.Al-Sultan, M.M.Khan, Computational experience on four algorithms for the hard clustering problem, *Pattern Recognition Letters,*1996, 17, 295–308.

[77] A.M.Bagirov, Modified global *k*-means algorithm for sum-of-squares clustering problems, *Pattern Recognition,*2008, 41(10), 3192–3199.

[78] A.M.Bagirov, A.M.Rubinov,J. Yearwood, A global optimisation approach to classification, *Optimization and Engineering,*2002, 3(2), 129–155.

[79] A.M.Bagirov, A.M.Rubinov,N.V. Soukhoroukova,J. Yearwood, Supervised and unsupervised data classification via nonsmooth and global optimization, *TOP: Spanish Operations Research Journal,*2003, 11(1), 1–93

[80] A.M.Bagirov, and J.Ugon,An algorithm for minimizing clustering functions, Optimization, 2005,54(4-5),351-368.

[81] A.M.Bagirov, J.Ugon, D.Webb, Fast modified global *k*-means algorithm for sum-of-squares clustering problems,*Pattern Recognition,2011,* 44, 866–876.

[82] A.M. Bagirov,J. Yearwood, A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems,*European Journal of Operational Research,* 170(2006), 578–596.

[83] C.Blake,E. Keogh,C.J. Merz, UCI Repository of machine learning databases: [http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[84] A.Rizzi, M.Vichi,H.H. Bock, Clustering and Neuralnetworks: Advances in Data Science and Classification,*Springer-Verlag,*,1998,265–277.

[85] D.E.Brown,C.E. Entail, A practical application of simulated annealing to the clustering problem, *Pattern Recognition,*,1992, 25, 401–412.

[86] V.F. Demyanov, A.M. Bagirov, A.M. Rubinov, A method of truncated codifferential with application to some problems of cluster analysis, *Journal of Global Optimization,*2002, 23(1), 63–80.

[87] G.Diehr, Evaluation of a branch and bound algorithm for clustering, *SIAM J. Scientific and Statistical Computing*,1985, 6, 268–284.

[88] P.Hansen,B. Jaumard, Cluster analysis and mathematical programming, *Mathematical Programming*,1997, 79(1-3), 191–215.

[89] P.Hansen,N. Mladenovic, *J*-means: a new heuristic for minimum sum-of-squares clustering, *Pattern Recognition,*2001 4, 405–413.

[90] P.Hansen,N. Mladenovic, Variable neighborhood decomposition search, *Journal of Heuristic*,2001, 7, 335–350.

[91] W.L.G.Koontz,P.M. Narendra,K, Fukunaga, A branch and bound clustering algorithm, *IEEE Transactions on Computers,*1975, 24, 908–915.

[92] J,Z.C.Lai,T.J. Huang, Fast global k-means clustering using cluster membership and inequality, *Pattern Recognition,*2010, 43(3), 731–737.

[93] A.Likas,M. Vlassis,J. Verbeek, The global *k*-means clustering algorithm, *Pattern Recognition,*2003, 36, 451–461.

[94] O.du.Merle,P. Hansen,B. Jaumard,N. Mladenovic, An interior point method for minimum sum-of-squares clustering, *SIAM J. on Scientific Computing,*2001, 21, 1485–1505.

[95] S.Z.Selim,K,S. Al-Sultan, A simulated annealing algorithm for the clustering, *Pattern Recognition,*1992, 24(10), 1003–1008.

[96] H.D.Sherali and J.Desai, A global optimization RLT-based approach for solving the hard clustering problem, *Journal of Global Optimization,*2005, 32, 281–306.

[97] H.Spath, Cluster Analysis Algorithms, *Ellis Horwood Limited, Chichester* ,1980.

[98] L.X.Sun,Y.L. Xie,X.H. Song,J.B. Wang,R.Q. Yu, Cluster analysis by simulated annealing, *Computers and Chemistry*,1994, 18, 103–108.

[99] Meng Piao Tan, James R. Broach, Christodoulos A. Floudas, A novel clustering approach and prediction of optimal number of clusters: global optimum search with enhanced positioning, *Journal of Global Optimization,*2007, 39(3), 323–346.

[100] A.E.Xavier, The hyperbolic smoothing clustering method, *Pattern Recognition,*,2010, 43(3), 731–737.

[101] A.E.Xavier and V.L.Xavier, Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions, *Pattern Recognition,*,2011, 44(1), 70–77.

[102] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Advances in knowledge discovery and data mining,in: American Association for Artificial Intelligence,1996, 1-34.

[103] Petar Ristoski, Heiko Paulheim,Semantic Web in data mining and knowledge discovery: A comprehensive survey,*Science, Services and Agents on the World Wide Web,* 36 (2016), 1-22.

[104] B. Ordin and A.M.Bagirov, A heuristic algorithm for solving the minimum-sum of squares clustering problems, *J. Global Optim,* 61(2015), 341–361.

[105] A.M.Bagirov, B.Ordin, G.Ozturk, A.E.Xavier, An incremental clustering algorithm based on hyperbolic smoothing, *Comput. Optim. Appl,* 61(2015), 219–241.

[106] L.T.H.An, H.V.Ngai and P.D.Tao, Exact penalty and error bounds in DC programming, *J.Global Optim,*2012, 52(3), 509–535.

[107] P.D.Tao and L.T.H.An, Convex analysis approach to DC programming:theory,algorithms and applications, Act a Math.Vietnam,1997,22(1),289–355.

[108] P.H.Wolfe,Finding the nearest point in a polytope,Math.Progr,1976,11(2),128–149.

[109] A. Agresti, Categorical Data Analysis, Wiley, NY, 1990.

[110] J.Z.C. Lai, T.-J. Huang, Fast global k-means clustering using cluster membership and inequality, *Pattern Recognition,*2010, 43(3), 731–737.

[111] E.Hansen, B.K.Ngai,N. Cheung, Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering, *Journal of Classication,*2005, 22(2), 287–310.

[112] R. Cattral, F. Oppacher, D. Deugo, Evolutionary Data Mining with Automatic Rule Generalization, *Recent Advances in Computers, Computing and Communications,* 2002,296-300.

[113] http://www.r-project.org.

[114] S. Lloyd, Least squares quantization in pcm, *IEEE Transactions on Information Theory*,1982, 28(2),129–137.

[115] E.W. Forgy, Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*,1965,21, 768–769.

[116] J.B. MacQueen, Some methods for classification and analysis of multivariate observations. In: L.M.L. Cam, J. Neyman (eds.) *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability,* 1(1967),281297.

[117] J.A. Hartigan and M.A.Wong, Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*,1979, 28(1), 100–108.

[118] D. Arthur and S. Vassilvitskii, k-means++: the advantages of careful seeding. In: Bansal, N. and Pruhs, K. and Stein, C. (ed.) SODA 07 *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms,* 2007,1027–1035.

[119] D. Pelleg and A. Moore, X-means: Extending k-means with efficient estimation of the number of clusters. In: Langley, P. (ed.) ICML00 Proceedings of the Seventeenth International Conference on Machine Learning, *Morgan Kaufmann Publishers Inc,* 2000,727–734.

[120] A.M. Bagirov and E. Mohebi, Nonsmooth optimization based clustering algorithms,

[121] Md A. Rahman and Md Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with k-means, *Knowledge-Based Systems,*2014, 71, 345–365.

[122] A.M. Bagirov, S. Taheri and J. Ugon, Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems, *Pattern Recognition,*,2016, 53, 12–24.