Data clustering using the Bees Algorithm and the Kd-Tree structure

A thesis Submitted to Cardiff University

For the degree of

Doctor of Philosophy

By

Hasan Al-Jabbouli

Intelligent Systems Research Laboratory

Manufacturing Engineering Centre

Cardiff University

United Kingdom

2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Data clustering has been studied intensively during the past decade. The K-means and C-means algorithms are the most popular of clustering techniques. The former algorithm is suitable for 'crisp' clustering and the latter, for 'fuzzy' clustering. Clustering using the K-means or C-means algorithms generally is fast and produces good results. Although these algorithms have been successfully implemented in several areas, they still have a number of limitations. The main aim of this work is to develop flexible data management strategies to address some of those limitations and improve the performance of the algorithms.

The first part of the thesis introduces improvements to the K-means algorithm. A flexible data structure was applied to help the algorithm to find stable results and to decrease the number of nearest neighbour queries needed to assign data points to clusters. The method has overcome most of the deficiencies of the K-means algorithm.

The second and third parts of the thesis present two new clustering algorithms that are capable of locating near optimal solutions efficiently. The proposed algorithms combine the simplicity of the K-means algorithm and the C-means algorithm with the capability of a new optimisation method called the Bees Algorithm to avoid local optima in crisp and fuzzy clustering, respectively.

Experimental results for different data sets have demonstrated that the new clustering algorithms produce better performances than those of other

algorithms based upon combining an evolutionary optimisation tool and the K-means and C-means clustering methods.

The fourth part of this thesis presents an improvement to the basic Bees Algorithm by applying the concept of recursion to reduce the randomness of its local search procedure.

The improved Bees Algorithm was applied to crisp and fuzzy data clustering of several data sets. The results obtained confirm the superior performance of the new algorithm.

In The Name of Allah,

The Most Gracious, The Most Merciful

ACKNOWLEDGMENTS

First of all I thank Allah (My Lord) the all high, the all great who made it possible for me to complete this work.

I am privileged to have Professor D. T. Pham as my supervisor. The high standard of his research has always been an inspiration and a goal to me. I am deeply grateful to him for his consistent encouragement, invaluable guidance and strong support during the course of this study. His thoughtful advice and constant support extended to me will always be remembered.

Grateful acknowledgement of my funding and support must be made to my home country Syria and the Syrian Ministry of Higher Education.

I am also very grateful to all the members of the Manufacturing Engineering Centre for their friendship and help.

My most sincere gratitude and appreciation go to my dear wife "H. Aldaqa" for her patience, continuous encouragement and support over the past difficult years. Thanks as well to Allah for his gifts; my beloved daughter "Hala" is truly one of them.

I am deeply indebted to my parents, my grandmother and all the members of my family who gave me continuous support and encouragement throughout my life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ART	Adaptive Resonance Theory
BS	Binary Splitting
CCIA	Cluster Centre Initialisation Algorithm
CLARA	Clustering LARge Applications
CLARANS	Clustering Large Applications based upon RANdomised Search
CLIQUE	CLustering In QUEst
DBMSDC	Density-Based Multi-Scale Data Condensation
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DENCLUE	DENsity-based CLUstEring
DM	Data Mining
DSBS	Direct Search Binary Splitting
EM	Expectation Maximisation
EPFCM	Evolutionary Programming based FCM
FCM	Fuzzy Clustering Method
FFCM	Fast Fuzzy Clustering Method
GA	Genetic Algorithm
IFCM	Improved Fuzzy C-Means
KDD	Knowledge Discovery in Databases
Kd-tree	K-dimension tree
KKZ	Katsavounidis, Kuo, and Zhang
MCAR	Missing Completely At Random
MR-Kdtree	Multiple Resolution K-dimension tree
NGH	NeiGHbours

OPTICS	Ordering Points To Identify the Clustering Structure
PAM	Partitioning Around Medoids
PCA	Principle Component Analysis
QPSO	Quantum-behaved Particle Swarm Optimisation
R-Bees	Recursive Bees
SCS	Simple Cluster Seeking
SOM	Self Organising Map
STING	STatistical INformation Grid

CHAPTER 1

Introduction

1.1 Background

Large amounts of data are produced every day as a result of activities in such different areas as engineering, marketing, medicine, education, etc. Even in just one area, such as engineering, data can come from multiple sources, be of many different types and be in numerous formats.

Ten years ago the main challenge was how to save and handle these data and as a result data warehouses (Ponniah 2001) were introduced. A subsequent and important challenge was how to refine and analyse the data, to extract useful information and knowledge from it to support decision making. The amount of data to be analysed is greater than any human can cope with and this led to the development of data mining.

Data mining (Han and Kamber 2006) is a complex topic, and links with many other disciplines such as computer science, engineering and manufacturing. It adds value to computational techniques from statistics, machine learning and pattern recognition.

Data clustering, which is the division of data into groups of similar objects (Zhijie et al. 2005), is a data mining and machine learning task. It is often used as a pre-processing technique in data mining. The clustering problem is difficult and has been addressed in many contexts and by researchers in many disciplines.

In general, data clustering is divided into crisp and fuzzy clustering. Here, the term "crisp clustering" is used to indicate a clustering process in which each data point belongs to one cluster, while "fuzzy clustering" indicates a clustering process in which each data point can belong to more than one cluster at the same time.

The K-means and C-means algorithms are the best known ones for crisp and fuzzy clustering, respectively. These methods suffer from a number of disadvantages, such as instability and, in the case of the K-means algorithm, the large numbers of neighbourhood searches required. There is also, trapping at local optima, which is a problem with both algorithms.

The main aim of this work is to improve the performance of the K-means and C-means clustering algorithms by introducing an improved data structure and other techniques to overcome some of the associated disadvantages.

1.2 Research Aim and Objectives

The overall aim of this research was to test the hypothesis that a new efficient data structure and optimisation tool called the Bees Algorithm can be used to accelerate and enhance the process of clustering in both the K-means and C-means algorithms.

The main research objectives were as follows:

- To overcome the instability and improve the neighbourhood search in the K-means algorithm by using an improved Kd-tree as an efficient data structure for data clustering.
- To avoid the local optima problem of the K-means and C-means algorithms by combining a novel optimisation method called the Bees Algorithm with both of them.
- To improve the local search procedure of the Bees Algorithm by using the concept of recursion and applying the proposed method to crisp and fuzzy clustering.

1.3 Research Methodology

To reach the above objectives, the following methodology was adopted:

- Literature Review - where related papers for each topic were studied to identify research trends and potential solutions.
- Experimentation - where the performance of each of the proposed algorithms was tested and compared against that of other clustering methods to assess their effectiveness.

1.4 Thesis Organisation

This thesis is organised as follows:

Chapter 2 briefly reviews data mining and gives an introduction to data clustering. Clustering algorithms are described and their advantages and disadvantages are discussed.

Chapter 3 presents a new clustering algorithm using an improved data structure based on the Kd-tree to overcome a number of the K-means algorithm's disadvantages.

Chapter 4 introduces the application of a novel optimisation technique called the Bees Algorithm to clustering. A new improved clustering method based on a combination of the Bees Algorithm and the K-means algorithm to solve the problem of local optima in crisp clustering is described.

The Bees Algorithm again is used with the C-means algorithm to solve the problem of local optima in fuzzy clustering. The combined algorithm is described in Chapter 5.

Chapter 6 describes an improvement to the local search procedure of the Bees Algorithm by using the concept of recursion. The chapter gives the results obtained in applying the algorithm to clustering.

Chapter 7 concludes the thesis and proposes directions for further research.

Appendix A describes all the datasets used in the research.

CHAPTER 2

Literature Review

2.1 Data Mining

Data Mining (DM) is the extraction of knowledge from data (Han and Kamber 2006) and it is considered as a step in the process of Knowledge Discovery in Databases (KDD) (Figure 2.1). It may also be defined as the process of automatically searching in large volumes of data for patterns to extract hidden facts and predictive information.

DM is used in companies and organisations to determine the relationships amongst internal and external factors. For example, DM can enable companies to determine the impact of customer satisfaction on sales by extracting useful information that help in decision making (Lejeune 2001).

There are many other applications of DM, such as: Fraud Detection (Kirkosa et al. 2007), Credit Risk Analysis (Kotsiantis 2007), Sky Survey Cataloguing (Fayyad et al. 1996a) and Text Mining (Feldman and Sanger 2006), etc.

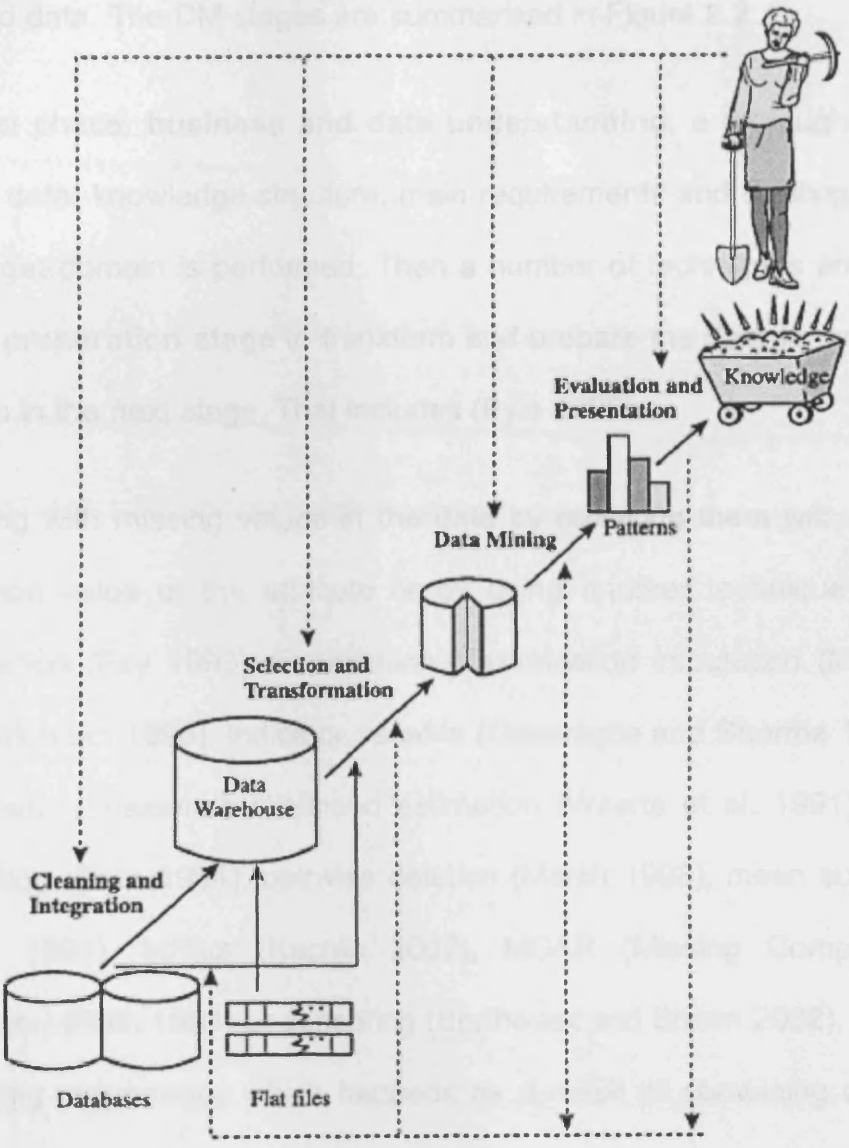


Figure 2.1 – Data mining as a step in the process of KDD (Fayyad et al. 1996b)

DM algorithms are used to perform difficult tasks associated with many challenges that vary between the nature, the size and the level of noise in the processed data. The DM stages are summarised in Figure 2.2.

In the first phase, **business and data understanding**, a thorough study of available data, knowledge structure, main requirements and existing systems in the target domain is performed. Then a number of techniques are used in the **data preparation stage** to transform and prepare the data for knowledge extraction in the next stage. That includes (Pyle 1999):

- Dealing with missing values in the data by replacing them with the most common value of the attribute or by using another technique such as imputation (Fay 1993), Expectation Maximisation imputation (McLachlan and Krishnan 1996), indicator variable (Garavaglia and Sharma 1998), full information maximum likelihood estimation (Waarts et al. 1991), listwise detection (Roth 1994), pairwise deletion (Marsh 1998), mean substitution (Roth 1994), MPlus (Kaplan 2002), MCAR (Missing Completely At Random) (Roth 1994) or censoring (Berthouex and Brown 2002).
- Avoiding redundancy which happens as a result of combining data from different sources. This can be detected by correlation analysis (Arnold and Wollenberg 2006).
- Applying a suitable noise handling technique to reduce the noise which might exist in the data (Teng 2001).

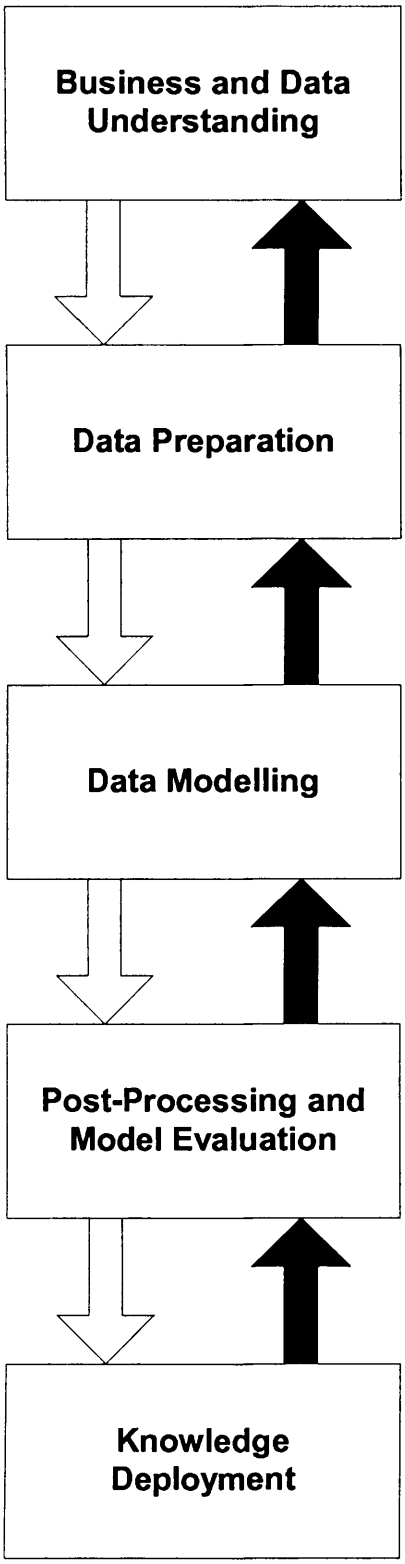


Figure 2.2 - The process model of DM (Chapman et al. 2000)

- Standardising or normalising the values of the attributes in the dataset according to the requirements of the next stage. This takes place by making the average value of the attribute equal to 0 and its standard deviation to 1 in the case of standardisation, or making its value in the range of 0 to 1 in the case of normalisation.
- Dividing continuous attributes into discrete, equal or variable intervals, if required.
- Extracting or constructing a number of features when there is a need to improve the overall performance of the DM process.
- Filtering or reducing the number of dimensions by keeping the attributes of rich information and eliminating useless ones. This can be performed by applying a feature selection technique (Liu and Motoda 2007).
- Filtering or reducing data instances to decrease the amount of data to be handled by using a data sampling method (Ardilly and Tillé 2006) or other instance reduction technique (Fodor 2002).

In the **data modelling stage**, various mining functions are applied according to the main requirements which were defined in the first stage. This can be performed by applying one or more of DM tasks where each of these tasks can utilise a number of different techniques. The selection of the appropriate technique is important to achieve the main purpose and is reliant on the experience of the DM expert. DM tasks include (Fayyad et al. 1996b):

i. **Classification** is defined as a function that maps data items into a number of pre-defined classes or groups. Classifiers have two stages (Nguyen 2004): the training stage in which the classifier learns from a set of

examples forming the target dataset and the final stage in which the classifier is used to classify the rest of the data. This function has many applications in: medical image analysis (Kim et al. 2008), toxicogenomics (Hamadeh and Afshari 2004), Geostatistics, speech recognition (Miyajima et al. 2000), handwriting recognition (Biem 2006), biometric identification (Jain et al. 2004), spam filtering (Sahami et al. 1998), internet search engines (Yeh et al. 2008) and credit scoring (Baesens et al. 2002).

ii. **Regression** is the process of learning a function to map data items to real-valued prediction variables (Abonyi and Feil 2007). It starts with a dataset of known target values and develops a formula based on observed data to predict the required information. In other words, it is used to develop a mathematical formula which can predict future behaviour based on observed data. For example, a regression could be used to predict the value of a car based on its specifications, obtained from observed data for many other types of cars. Regression also has many applications in industrial organisational psychology (Bobko 2001) and household expenditure (Branscum et al. 2007).

iii. **Clustering** is a common descriptive task which seeks to identify a finite set of categories or clusters to describe the data so that similar data points belong to the same cluster (Han and Kamber 2006). In other words, the main aim of this function is to find groups of high quality where inter-clusters are similar while intra-clusters are dissimilar. Clustering, unlike classification, is used to segment data into groups which were not previously specified. More details about data clustering are mentioned in next section.

iv. **Association** is a function that aims to discover the rules of relationships amongst co-occurring items (Han and Kamber 2006). In other words it finds a model that describes significant dependencies between variables which might exist at the structural or the quantitative level. The structural level provides information about the local dependence of variables on each other, while the quantitative level explains the strengths of the dependencies using a numerical scale. Association has many applications in market basket analysis (Berry and Linoff 2004) and in the discovery of business trends (Dong and Li 1999).

Modelling stage is often performed in a trial and error manner.

Post-processing and evaluation stage, in which a validation and refining of the proposed results is performed, is coupled with the modelling stage. Both of them can be repeated many times.

The final DM solution is deployed in the **knowledge deployment stage**. This can be made over the web or any local network based on the requirements of the solution owner.

Data clustering is one of the most important tasks of the data mining module, and this research focuses on improvements to its algorithms as a primary objective.

2.2 Data Clustering

The main goal of clustering is the determination of the intrinsic groupings in a set of unlabeled data, which makes the history of data clustering as old as the history of mankind. The clustering problem, which is also called grouping, occurs in many disciplines because of its broad appeal in manufacturing and its importance as a main discipline within DM tasks. Clustering has been defined in several ways:

- "Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters)" (Jain et al. 1999).
- "Clustering is finding groups within a certain set of data in which each group contains objects similar to each other and different from those of other groups" (Han and Kamber 2006).

Data clustering is also known by a number of other names, including: clustering, cluster analysis, botryology, numerical taxonomy, typological analysis and automatic classification.

Although clustering has its main usage in DM, it also has an immense number of applications in many fields including: image analysis (Kersten et al. 2005), bioinformatics (Au et al. 2005) and market segmentation (Kuo et al. 2002). Data clustering also plays an important role in the field of information retrieval (Smith et al. 1997). It is also extremely useful in scientific and engineering analysis (Au et al. 2005; Fisher et al. 1993; Kersten et al. 2005; Kuo et al. 2002; Xu 2005), as well as in many other applications such as: data cleaning,

exploratory analysis (Klosgen and Zytkow 2002) and in web mining (Xu 2005; Zamir et al. 1997).

Similarity of elements within the same cluster is the most important measurement in data clustering. It is often assessed according to a distance measure; a number of elements are assessed as similar depending on the distance between them. Because of its popularity, the Euclidean distance is adopted as a distance measurement parameter in this research. There are other possible metrics (Anderberg 1973; Diady and Simon 1976; Jain and Dubes 1988) which can be used in data clustering, including the Manhattan distance (Krause 1986), Mahalanobis distance (Mahalanobis 1936) and Hamming distance (Hamming 1950).

The sum of the square of the distance from each data point to the centre of the cluster to which it belongs is the most used criterion in cluster analysis (Aloise 2009; Merle et al. 2000). Clustering algorithms try to divide the dataset into '*k*' clusters and minimise this sum (Equation 2.1). The less the value of this sum the better is the clustering.

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2 \qquad \text{Equation 2.1}$$

Where:

$\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ in the cluster (j) and the cluster centre c_j.

n_j is the number of data points in the cluster c_j.

2.3 Data Clustering Algorithms

Many clustering algorithms have been developed. It is difficult to have a clear classification for them because of their similarities and the ways they overlap, but it is possible to divide them into five categories (Pham and Afify 2007), see Figure 2.3: Partitioning methods, Hierarchical methods, Density-based methods, Grid-based methods and other methods.

2.3.1 Partitioning Methods

The common idea of these algorithms is that they separate data points into '*k*' disjoint clusters (Jain et al. 1999). First they create an initial set of '*k*' partitions, where the parameter '*k*' is the number of partitions to construct; then they use an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. Typical partitioning methods include K-means (Jain and Dubes 1988; MacQueen 1967), K-medoids (Kaufman and Rousseeuw 2005), Expectation Maximisation (Dellaert 2002; McLachlan and Krishnan 1996) and their improvements.

The time required for these algorithms to give results is typically of the order: *O(N log N)*. This makes them very efficient clustering algorithms for small datasets. The main disadvantages of these methods are: the initialisation stage which can affect the results of the clustering, the need to specify the parameter '*k*' which is the number of clusters and that they can become trapped at local optima.

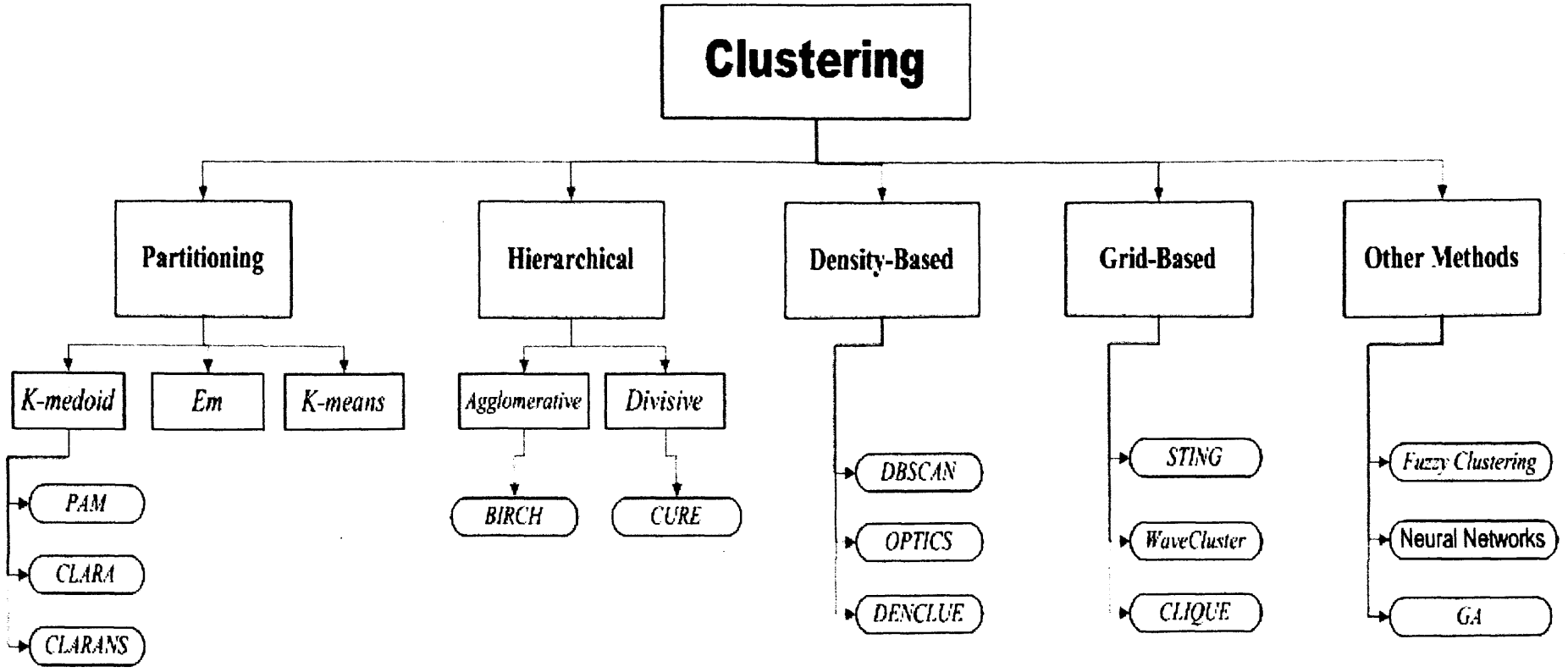


Figure 2.3 – A classification of the main clustering techniques (Pham and Afify 2007)

2.3.1.1 The K-means Algorithm

The K-means algorithm is used to group objects into '*k*' number of sets based on their attributes (features). This is achieved by minimising '*E*', which is defined by Equation 2.1. The main steps of the algorithm are listed in Figure 2.4.

Advantages:

- With a large number of variables, the K-means algorithm is almost computationally faster than other clustering algorithms (if '*k*' is small).
- It can produce tighter clusters compared to hierarchical clustering algorithms, especially if the clusters are globular.

Disadvantages:

- It is almost difficult to predict the value of '*k*', which is the only parameter needed for this algorithm.
- It does not work well with non-globular clusters.
- Different initial partitions for the data can lead to different final clusters.

More details about this algorithm and its improvements are discussed in Section 2.4.

1. Place '*k*' points into the space represented by the objects that are being clustered. These points represent initial group centres.
2. Assign each object to the group that has the closest centre.
3. When all objects have been assigned, recalculate the positions of the '*k*' centres.
4. Repeat Steps 2 and 3 until the centres no longer move.

Figure 2.4 – Main steps of the K-means algorithm (MacQueen 1967)

2.3.1.2 The K-medoids Algorithm

The K-medoids algorithm (Kaufman and Rousseeuw 2005) is similar to the K-means algorithm. It chooses the most central '*k*' data points in each cluster to represent the clusters and calls them "medoids". The algorithm then starts a loop of operations to form the best clusters. All of these steps are listed in Figure 2.5.

Advantages: It is very useful for small datasets.

Disadvantages: It is a time consuming method.

A number of different improvements have been developed for the K-medoids algorithm such as PAM (Partitioning Around Medoids) (Kaufman and Rousseeuw 2005), CLARA (Clustering LARge Applications) (Kaufman and Rousseeuw 2005), CLARANS (Clustering Large Applications based upon RANdomised Search) (Ng and Han 1994) and the improved CLARANS (Ester et al. 1995).

PAM starts by selecting random medoids and then repeatedly swaps each medoid with one of the other data points, to see if this can improve the clustering quality (Kaufman and Rousseeuw 2005). PAM has the drawback that it is inefficient with large datasets (Han et al. 2001). To overcome this disadvantage, CLARA was proposed in (Kaufman and Rousseeuw 2005). It applies PAM to samples of the dataset and then gives the best clustering of these samples. Another more efficient clustering algorithm, CLARANS, that uses random search was proposed in (Ng and Han 1994) and then improved in (Ester et al. 1995).

Repeat

1. Assign each data point to the most similar (closest) medoid.
2. Select new medoids list candidates randomly.
3. Calculate the total cost of using the candidates as medoids.
4. If the cost of using new medoids is less than the cost of using current medoids then use the new set of medoids.

Until (There is no change in medoids)

Figure 2.5 – Main steps of the K-medoids algorithm (Han and Kamber 2006)

2.3.1.3 Expectation Maximisation Algorithm (EM)

EM (Dellaert 2002) is an iterative optimisation method used in statistics to maximise the posterior probability of the parameters θ given the data U, marginalising over the hidden variables J over the space τ (Equation 2.2):

$$\theta^* = \arg\max_{\theta} \sum_{J \in \tau^n} p(\theta, J \mid U) \qquad \text{Equation 2.2}$$

The algorithm alternates between two major steps: the expectation step (E-step) in which it computes the probability of a data point 'x' belonging to a cluster 'j' (Equation 2.3) based on the estimated parameters; and the maximisation step (M-step) in which it computes the maximum likelihood estimations of the parameters (Equation 2.4) by maximising the expected likelihood found on the E-step.

The algorithm (Figure 2.6) starts by initialising the distribution parameters and then it repeats the two steps (E and M) by using the parameters found in the M-step to begin another E-step until convergence or the termination condition is met.

$$p(x) = \sum_{j=1}^{k} p_j p(x \setminus j) \qquad \text{Equation 2.3}$$

where 'p_j' is the mixture weight for cluster 'j', and '$p(x \setminus j)$' is the mixture model for cluster 'j'.

$$L = \sum_{n=1}^{N} \log p(x_n) = \sum_{n=1}^{N} \log\left(\sum_{j=1}^{k} p_j p(x_n \setminus j) \right)$$ Equation 2.4

where '*N*' is the number of data points.

Advantages: The algorithm is applicable to both categorical and numerical attributes and at the same time it has a strong statistical basis (Pham and Afify 2007).

Disadvantages: This method can easily be trapped at local minima. To overcome this problem, the algorithm has to scan the dataset many times.

McLachlan and Krishnan discussed the convergence properties of the EM in (McLachlan and Krishnan 1996). A new scheme for the EM algorithm was proposed in (Bradley et al. 1998b), in which the authors applied the EM steps to samples of the dataset instead of dealing with the whole dataset. The data points in each sample are compressed to produce a number of representative statistics which are used to update the parameters when the next sampling is considered (Pham and Afify 2007).

Initialise parameters.

Repeat

1. E-step: Compute '*p(x)*' for all '*x*' in the dataset using Equation 2.3.
2. M-step: Compute the maximum likelihood estimation of the parameters and update the Gaussian distributions for each cluster '*k*' using Equation 2.4.

Until (Convergence)

Figure 2.6 – Main steps of the EM algorithm (Pham and Afify 2007)

2.3.2 Hierarchical Methods

Hierarchical methods refer to a technique in clustering which creates a hierarchical decomposition of the given set of data points' clusters recursively. The methods can be classified as being either agglomerative (bottom-up) clustering or divisive (top-down) clustering.

In agglomerative clustering (Johnson 2006), Figure 2.7a, each data point is assigned to a single cluster, so that if there are '*N*' data points, then '*N*' clusters will be created with one data point in each one.

Then a series of grouping processes takes place by finding the closest pair of clusters and mixing them together into one cluster until all data points are grouped in one single cluster or until a specific threshold is met. These steps are summarised in Figure 2.7b.

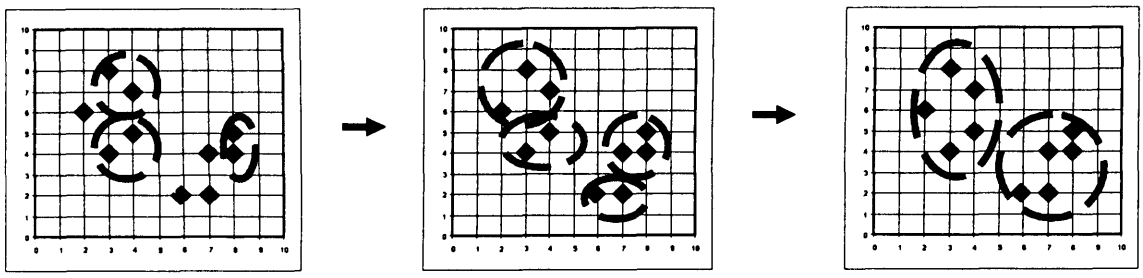


Figure 2.7a - Agglomerative Clustering

Repeat:

1. Assigning each data point to a cluster
2. Mix the closest two clusters into one cluster

Until (there is one single cluster)

Figure 2.7b – Main steps of agglomerative clustering (Jain et al. 1999)

In divisive clustering (Jain et al. 1999), Figure 2.8a, all data points are assigned to one single cluster. Then a series of division processes takes place by finding the maximum distance between any two data points and taking these two data points as seeds for two new clusters which will be created by dividing the parent cluster into two. All other data points are placed into one of the new created clusters based on the closest seed point. The whole process is repeated until each data point belongs to only one cluster or until a specific threshold is met. These steps are summarised in Figure 2.8b.

Advantages: Hierarchical algorithms are useful and informative for data display by producing an ordering of the data points. At the same time they do not require the number of clusters to be set as a parameter and the algorithm can be stopped at any time.

Disadvantages: In spite of their simplicity, hierarchical algorithms need to perform a large number of "nearest-neighbours" queries for the points in the dataset. If the data has '*d*' dimensions and there are '*n*' points in the dataset, the cost of a single iteration is *O(kdn)* where '*k*' is the number of clusters. As one would have to run several iterations, it is generally not feasible to run the algorithm for a large number of data points. Apart from this, the hierarchical clustering algorithms can never undo what was done previously.

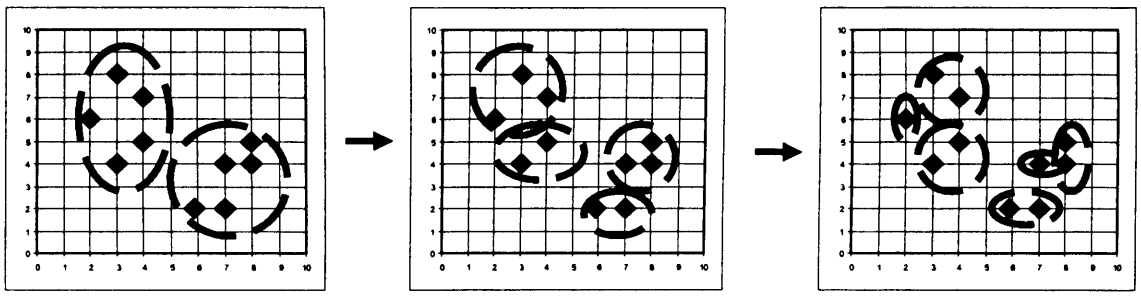


Figure 2.8a - Divisive Clustering

Repeat:

1. Assigning all data point to a one cluster.
2. Find the farthest two points and take them as seeds for division.
3. Divide the cluster into two and assign other data points to clusters based on the closest seed point.

Until (each data point is in a single cluster)

Figure 2.8b – Main steps of divisive clustering (Jain et al. 1999)

2.3.3 Grid-based Methods

These methods form a grid structure and perform the clustering on it by quantising the attribute space into a specific number of cells (Han and Kamber 2006).

One of the most famous algorithms in this category is STING (STatistical INformation Grid) (Wang et al. 1997) which uses the grid cell, Figure 2.9a, to store a number of useful statistical information to be used in the clustering process. It starts (Figure 2.9b) by dividing the spatial area into rectangular cells, then partitioning each cell into a specific number of smaller cells according to the required level of resolution. It then calculates a set of parameters for each cell. Then for each layer, starting from the layer with the smallest number of cells and for each cell in the layer, the algorithm computes a confidence interval. It then removes the irrelevant cells in each layer.

Another two famous algorithms in this category are "WaveCluster" (Sheikholeslami et al. 1998) which uses the wavelet transform to perform the clustering and CLIQUE (CLustering In QUEst) (Agrawal et al. 2005) which is used for clustering in high-dimensional data spaces depending on a combination of grid and density-based approaches.

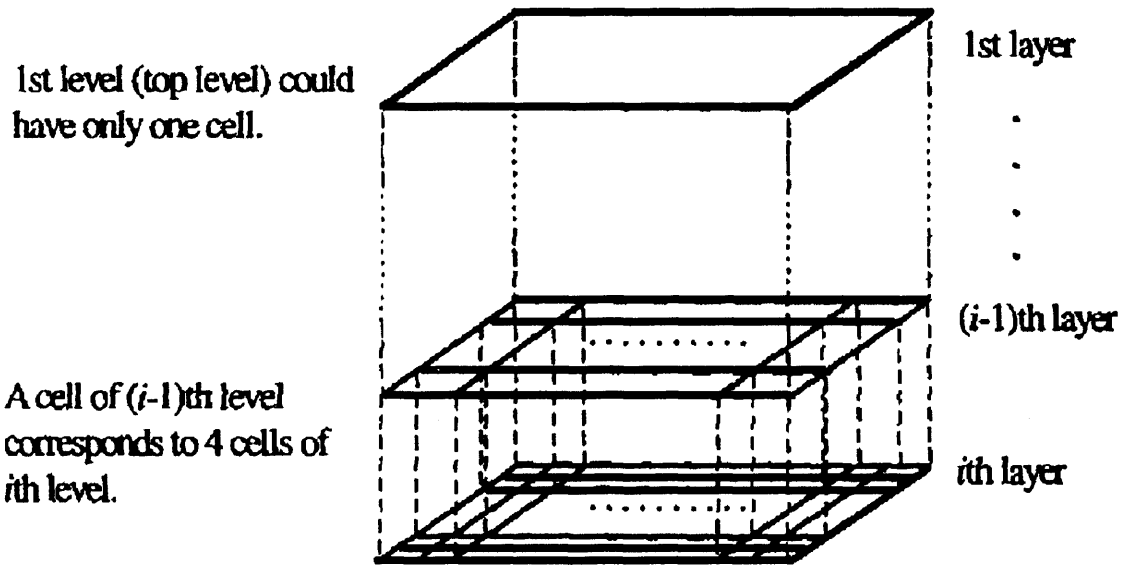


Figure 2.9a – Hierarchical structure in STING (Wang et al. 1997)

1. Determine a layer to begin with.
2. For each cell of this layer

   Calculate the confidence interval of probability that this cell is relevant.
3. From the interval calculated above, label the cell as relevant or not relevant.
4. If (this layer is the bottom layer) Then Go to Step 6. Else Go to Step 5.
5. Go down the hierarchy structure by one level.

   Go to Step 2 for those cells that form the relevant cells of the Higher level layer.
6. If (the specification is met) Then Go to Step 8. Else go to Step 7.
7. Retrieve those data points falling into the relevant cells and do further processing.

   Return the results that meet the requirement of the query. Go to Step 9.
8. Find the regions of relevant cells and return those regions that meet the requirement of the query. Go to Step 9.
9. Stop.

Figure 2.9b – Main steps of the STING algorithm (Wang et al. 1997)

In WaveCluster (Sheikholeslami et al. 1998), Figure 2.10a, clusters are formed by dealing with the data points as multi-dimensional signals representing the feature space using a single processing technique called the wavelet transform method. This method resolves each signal into different frequency sub-groups and finds the high and low frequency groups which represent clusters and outliers respectively.

The CLIQUE algorithm, Figure 2.10b, divides the data space into separate rectangular cells where each cell contains a set of data points. It determines the dense cells, which have a number of data points greater than a specified number and applies a depth-first search algorithm to find sets of connected high density cells to create the clusters.

Advantages: These methods take a short processing time and are able to handle outliers (Pham and Afify 2007). The complexity is $O(k)$ where 'k' is the number of cells in the lowest layer.

Disadvantages: They have the limitations that they are less effective when there is large number of dimensions and it is difficult to detect clusters of non-horizontal or vertical boundaries.

1. Quantise feature space, and then assign objects to the units.
2. Apply wavelet transform on the feature space.
3. Find the connected components (clusters) in the sub-bands of transformed feature space, at different levels.
4. Assign label to the units.
5. Make the lookup table.
6. Map the objects to the clusters.

Figure 2.10a – Main steps of the WaveCluster algorithm (Sheikholeslami et al. 1998)

1. Identification of subsets that contain clusters.
2. Identification of clusters.
3. Generation of minimal description for the clusters.

Figure 2.10b – Main steps of the CLIQUE algorithm (Agrawal et al. 2005)

2.3.4 Density-based Methods

These methods define the clusters as an arbitrary shape or region of data points within the data space, where each cluster is separated from other clusters by noise data which are low density regions (Han and Kamber 2006).

The most famous algorithm in this category is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al. 1996). In this algorithm, the neighbourhood within a specific '*Eps*' radius of any data point in a cluster has at least a specific number '*MinPts*' of data points. The algorithm (Figure 2.11) repeats in each iteration, selecting any random point '*p*' that does not belong to any cluster and making it the core of a cluster. It then expands this cluster by finding any density-reachable point from '*p*' (Ester et al. 1996).

Advantages: It is less sensitive to outliers and can deal with a large number of data points. It is not affected by the shape of the cluster. Another interesting advantage is that it does not require the number of clusters in the data as a priori parameter.

Disadvantages: The algorithm is sensitive to the parameters '*Eps*' and '*MinPts*' while it is very difficult to determine them. It does not perform well on datasets with varying densities or with large dimensions.

```
1. C = 0
2. For each unvisited point P in dataset D
3. Mark P as visited
4. Find P Neighbours (N) within Eps radius
5. If (Size of(N) < MinPts) Then
       mark P as NOISE
   Else
       C = next cluster
       // Expanding the cluster
       add P to cluster C
       For each point PP in N
           If PP is not visited Then
               Mark PP as visited
               Find PP Neighbours (NN) within Eps
               radius
           If NN >= MinPts Then
                   N = N joined with NN
           If PP is not yet member of any cluster Then
                   add PP to cluster C
```

Figure 2.11 – Main steps of the DBSCAN algorithm (Ester et al. 1996)

OPTICS (Ordering Points To Identify the Clustering Structure) is a method developed to overcome the weakness of DBSCAN in determining the *Eps*, and '*MinPts*' parameters (Ankerst et al. 1999). It produces an ordering of data points to visualise the clustering results of lower values of '*Eps*' and '*MinPts*', so making it easier to compute them (Pham and Afify 2007).

Despite its simpler way of determining '*Eps*' and '*MinPts*', OPTICS suffers when dealing with datasets of large number of dimensions. To help overcome this, a new technique called DENCLUE (DENsity-based CLUstEring) has been introduced (Hinneburg and Keim 1998). This algorithm starts by defining the overall density of points as the sum of a set of mathematical functions called influence functions, which describe the impact of a data point on its neighbours. Then it defines the clusters mathematically by finding the local maxima of the overall density function. Although it requires pre-defined parameters to start its work, this algorithm has a number of powerful advantages, being fast and robust to noise (Pham and Afify 2007).

2.3.5 Other Methods

Other algorithms which do not belong to any of the above mentioned four categories have been developed to deal with data clustering problems. These include: artificial neural network approaches (Carpenter and Grossberg 1991; Carpenter et al. 1991; Kohonen 2001), Genetic Algorithm (GA)-based clustering (Krishna and Murty 1999; Maulik and Bandyopadhyay 2000) and Fuzzy clustering (Bezdek 1981; Dunn 1973; Jang et al. 1997).

2.3.5.1 Artificial Neural Network Approaches

Several artificial neural networks have been used for clustering. One of them is the SOM (Self Organising Map) (Kohonen 2001), which consists of (Figure 2.12) an input layer that has neurons connected to each neuron in the single output layer (called the Kohonen layer) via connections of variable weights. This network can preserve the distance between clusters in a dataset, so it can provide a two dimensional visualising of the data. In SOM, each neuron in the Kohonen layer represents one of the patterns which the network can recognise, so that similar patterns are represented by adjacent neurons while different patterns are represented by distant neurons.

Another widely applicable neural network-based algorithm is called the Adaptive Resonance Theory (ART) (Carpenter and Grossberg 1991). This network (Figure 2.13) uses unsupervised learning to perform the clustering by selecting a cluster unit for each presented pattern and adjusting the weights to let the cluster unit learn the pattern, so that the degree of similarity of similar patterns is controlled by a vigilance parameter reset mechanism.

Using this similarity criterion, the network adds the input to one cluster if it is similar to members of that cluster; otherwise it creates a new cluster. In general, beside the time required for the network training, another main disadvantage of these algorithms is the difficulty in finding the suitable learning parameters for the network.

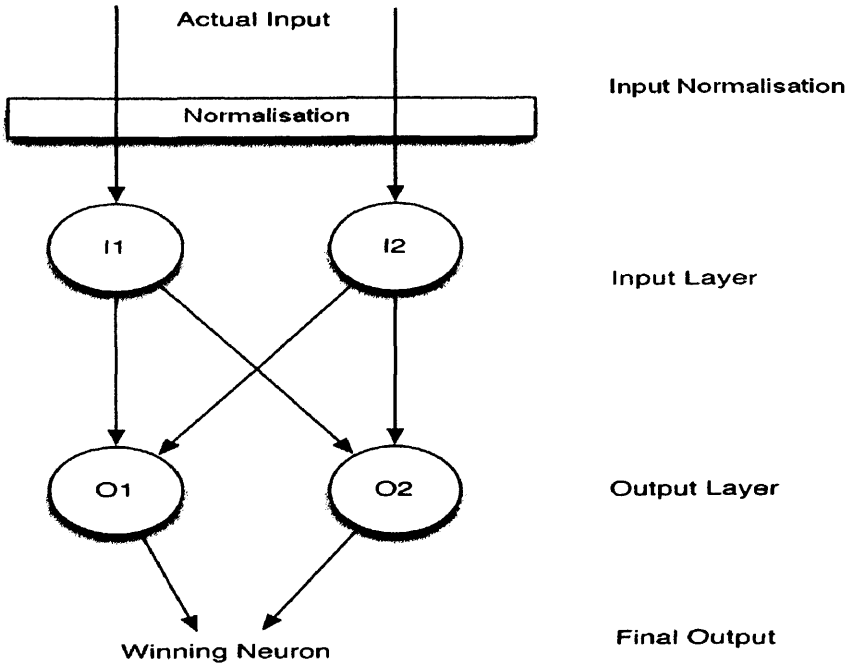


Figure 2.12 – A typical SOM structure (Kohonen 2001)

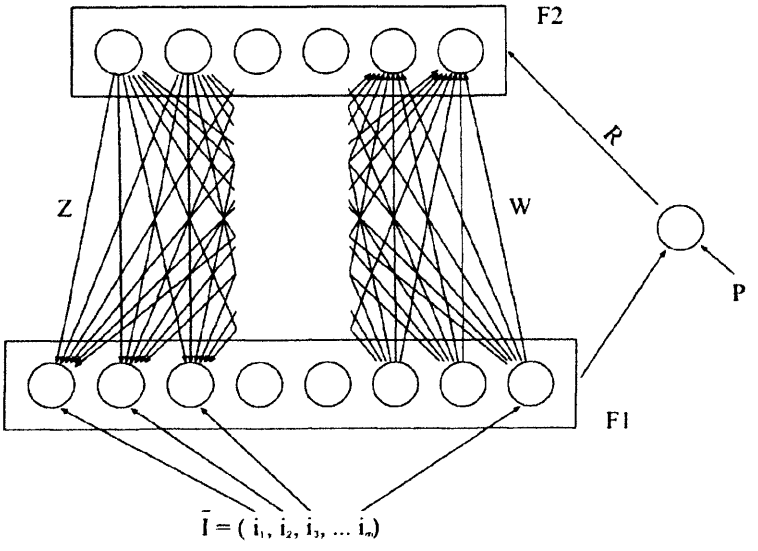


Figure 2.13 – Basic ART structure (Carpenter et al. 1991)

2.3.5.2 Genetic Algorithm Approaches

The capability of the Genetic Algorithm (GA) (Davis 1991), which is a well-known evolutionary technique, was applied (Sheikh et al. 2008) to evolve the proper number of clusters (Bandyopadhyay and Maulik 2002; Kudova 2007; Song and Park 2006) and to provide appropriate clustering (Hall et al. 1999; Krishna and Murty 1999; Lin et al. 2005; Lu et al. 2004; Maulik and Bandyopadhyay 2000) .

The centres of the clusters are represented by a string of bits (chromosome). A collection of such chromosomes is called a population. Each population represents different solutions in the search space. The solution is reached using the stochastic search which is the core of the GA.

GA-based clustering starts by generating an initial population. Then the algorithm (Figure 2.14) repeats the process of evaluating the solution for the current population and generating a new population based on improvements to the current population, using a number of operations such as random mutation and cross-over. More details about the usage of GA for clustering are given in Chapter 4.

1. Choose an initial population
2. Evaluate the fitness of each individual in the population
3. Repeat
    a. Select best-ranking individuals to reproduce.
    b. Breed a new generation through cross-over and/or mutation (genetic operations) and give birth to offspring.
    c. Evaluate the individual fitnesses of the offspring
    d. Replace worst ranked part of population with offspring
4. until termination: (time limit or sufficient fitness achieved)

Figure 2.14 – Basic steps of the GA algorithm (Maulik and Bandyopadhyay 2000)

2.3.5.3 Fuzzy Clustering

The idea of fuzzy clustering was initially introduced in 1969 by Ruspini (Ruspini 1969). Most of the clustering algorithms produce crisp clusters, where each data point belongs to one cluster at a time. In fuzzy clustering, each data point can belong to more than one cluster at the same time. In other words, the value of the membership function for all data points in crisp clustering (Figure 2.15a) has the value '1' (belong to the cluster) or '0' (Doesn't belong), while in fuzzy clustering the value of the membership function (Figure 2.15b) could be any value less than '1' and greater than '0'.

The most popular algorithm in fuzzy clustering is the fuzzy C-means (FCM). This algorithm was developed by Dunn in 1973 (Dunn 1973) and then improved by Bezdek in 1981 (Bezdek 1981). After determining the number of the clusters as a parameter to the algorithm, the FCM (Figure 2.16) tries to produce fuzzy clusters and generates the membership degree of each data point for each of these clusters. It starts by generating initial centres randomly and then repeatedly updates both the centres of clusters and the membership degree for each data point.

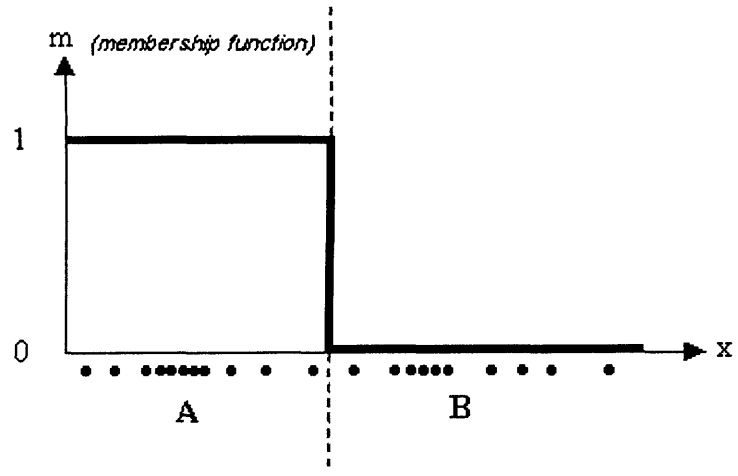


Figure 2.15a – Membership function in Crisp Clustering (Matteucci 2003)

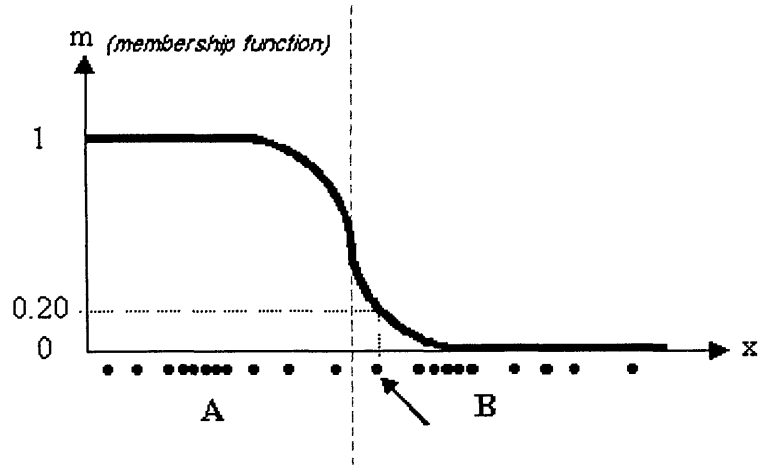


Figure 2.15b – Membership function in Fuzzy Clustering (Matteucci 2003)

Repeat

1. Initialise the membership values $U = [u_{ij}]$ matrix, $U^{(0)}$

2. At k-step: Calculate the centres vectors $C^{(k)} = [c_j]$ with $U^{(k)}$

$$C_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

3. Update $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

Until ($\|U^{(k+1)} - U^{(k)}\| < \sigma$)

Figure 2.16 – Basic steps of the C-Means algorithm (Bezdek 1981)

Where:

K: is the iteration step

σ : is a termination criterion between 0 and 1

$U^{(k)}$: is the value of U at the k^{th} iteration

m : is the fuzziness degree (any real number greater than 1).

c : is the number of clusters.

2.4 Approaches

Because of their simplicity and popularity, the K-means and C-means algorithms will be at the core of this research.

2.4.1 Approaches to the K-means Algorithm

As explained earlier (Figure 2.4), the K-means algorithm starts by generating a random set of centres based on the required number of clusters, one centre for each cluster. Each data point is assigned to the cluster of the nearest centre. The algorithm then represents each cluster by the mean value of the data points which belong to it and reassigns data points to these clusters. This process is repeated until a specified criterion is met or convergence occurs.

This algorithm is very efficient and simple. However, it has a number of disadvantages, which are summarised as follows (Jain et al. 1999; Pham and Afify 2007):

Sensitivity: It is sensitive to noise and outliers.

Choice of initial centres: A bad choice of initial centres can have a great impact on both the performance and the distortion (E in Equation 2.1).

Large number of nearest neighbours (ngh) queries: It needs to perform a large number of (*ngh*) queries for the points in the dataset. This means that it performs many distance calculations (between points and centres). The cost

of each single iteration is *O(kdn)*, where '*k*' is the number of clusters, '*d*' is the number of dimensions and '*n*' is the number of data points in the dataset.

Instability: The K-means is an unstable algorithm and can give different clustering results for each run because of the random initial clustering.

Trapping into local optima: The performance of the K-means algorithm can be measured by calculating the distortion. The lower the value of the distortion, the better is the clustering. In other words, the algorithm aims to minimise the distortion but sometimes it gets trapped at a local optimum.

Slow movement in last iterations: In the last several iterations, the centres move very little but at the same time many calculations are taking place, so these last iterations are costly with a little outcome.

Dealing with large datasets: The performance of the clustering algorithm is related to the volume of the dataset. In most cases it takes a long time to obtain a good solution when dealing with large datasets because of the large number of *ngh* queries.

The number of clusters: The number of clusters should be initially defined, but when this number is wrong the algorithm is forced to give unexpected results.

Many developments for the K-means algorithm have been proposed (Babu and Murty 1993; Ball and Hall 1965; Bezdek 1981; Cheung 2003; Krishna and Murty 1999; Kuo et al. 2005; Zhang 2001). They vary in the estimation of the

dissimilarity, the techniques used to calculate the mean and in the choice of the preliminary separation.

Also many improvements have been made to overcome the speed of the convergence (Al-Daoud et al. 1995; Bottou and Bengio 1995; Estivill-Castro and Yang 2004; Fritzke 1997; Kanungo et al. 2002; Patané and Russo 2001; Pelleg and Moore 1999, 2000). Other improvements have focused on the initialisation process (Al-Daoud and Roberts 1996; Bradley and Fayyad 1998; Epter and M. Krishnamoorthy 1999; Yang and Luo 2005), while other researchers have proposed an incremental version of the K-means (Pham et al. 2004a). Other extensions have been introduced to enable the algorithm to deal with categorical data (Al-Daoud et al. 1995; Bottou and Bengio 1995; Estivill-Castro and Yang 2004; Fritzke 1997; Huang 1998; Kanungo et al. 2002; Patané and Russo 2001; Pelleg and Moore 1999, 2000); or large datasets (Bradley et al. 1998a; Chen et al. 2005; Farnstrom et al. 2000; Jinlan et al. 2005; Pham et al. 2004b).

2.4.2 Approaches to the C-means algorithm

As explained in Figure 2.16, the C-means algorithm repeats a number of operations to find the best fuzzy clusters. Or in other words, the algorithm tries to find the best membership degree of each data point in all clusters.

This algorithm is very efficient and it has many applications in different areas. However, it has a number of disadvantages, which are summarised as follows:

Sensitivity: It is sensitive to outliers.

Choice of initial centres: A bad choice of initial centres can affect the performance of the algorithm.

Trapping into local optima: The algorithm aims to generate the best membership values of data points which give the best clustering but sometimes it gets trapped at a local optimum.

Large number of computations: It needs to perform a large number of calculations to obtain a solution.

Time cost: The algorithm takes a long time to converge.

The number of clusters: The number of clusters should be initially defined in the algorithm.

To increase its robustness to outliers, modifications to the FCM using *Lp* norm distances was proposed (Hathaway et al. 2000).

An Improved Fuzzy C-Means (IFCM) (Kaiqi et al. 2008) was proposed to solve the problem of the choice of initial cluster centres and to reduce the computational complexity of the FCM. This algorithm uses the Quick Subtractive Clustering (QSC) for getting initial cluster centres. And it uses the mapping from pixel space to eigenvector space for modifying the object function therefore reducing the computational complexity.

Another algorithm, called the Modified Fast Fuzzy C-means Algorithm for Image Segmentation, was also proposed (Guo et al. 2009) to overcome the problem of initialising clusters centres. It uses the cluster centres obtained by the sample density as the initial centres.

To overcome the speed of convergence in the FCM, a new algorithm that decreases the number of distance calculations and called FFCM (Fast Fuzzy Clustering Method) (Al-Zoubi et al. 2007) was proposed.

In the purpose of increasing the accuracy of the segmentation in case of mixed noises and increasing the processing speed, the modified FCM algorithm was proposed (Szilágyi1 et al. 2007). This algorithm extracts a scalar feature value from the neighbourhoods of each pixel, using a filtering technique that deals with both spatial and gray level distances.

GA (Alata et al. 2008) was used to determine the optimum number of clusters. Another algorithm called EPFCM (Evolutionary Programming based FCM) (Donga et al. 2009) was also proposed to find the number of clusters dynamically.

2.5 Proposed Solutions

The main aim of this research is to accelerate and enhance the process of crisp and fuzzy clustering in the K-means and C-means algorithms. This will depend on using an efficient data structure called the Kd-tree (K-Dimension Tree) and a new optimisation tool called the Bees Algorithm (Pham et al. 2006).

Chapter 3 introduces a new version of the K-means algorithm, based on the Kd-tree, to overcome a number of the K-means limitations. The proposed algorithm produces good clustering results without randomness.

Chapter 4 and Chapter 5 present solutions for the local optima problem in both the K-means and C-means algorithms, based on a novel optimisation algorithm called the Bees Algorithm.

Chapter 6 introduces an improvement to the Bees-based clustering algorithms used in Chapters 4 and 5.

2.6 Summary

This chapter reviews the main aspects of DM and its main steps. The data clustering process is described and its main algorithms are discussed briefly. The K-means and C-means algorithms, which are the most common and well-known data clustering algorithms, are reviewed with their main advantages and disadvantages. Research directions to help overcome these disadvantages are given.

CHAPTER 3

Improvements to the K-means Algorithm based on the Kd-tree

3.1 Preliminaries

The K-means algorithm (Jain and Dubes 1988; MacQueen 1967) is a well-known and attractive clustering algorithm because of its simplicity and convergence properties.

However, as mentioned in Chapter 2, one of the drawbacks of the algorithm is its instability. This means that it produces different results each time because of the randomness of the initialisation stage. This randomness affects both the number of iterations needed to find a solution and the final solution itself. As a good initialisation process of the centres usually leads to a good final solution.

Another drawback of the K-means algorithm is the large number of neighbour searches required to find the data points within one cluster (to find data points which are close to each centre).

This chapter gives an overview of existing techniques to deal with these problems. It also proposes a new method based on the Kd-tree data structure to overcome these problems.

The chapter is organised as follows: Section 3.2 reviews the main methods used to improve the initialisation of the K-means algorithm. Section 3.3 introduces the Kd-tree and summarises its usage in data clustering. Section 3.4 describes the proposed algorithm in detail. Experimental results and the analysis of the proposed algorithm are presented in Section 3.5 and Section 3.6 respectively. Section 3.7 summarises the chapter.

3.2 Review

Many methods have been suggested to overcome the drawback of the initialisation stage of the K-means algorithm. One of the earliest methods was proposed in (Anderberg 1973). The method chooses '*k*' instances randomly from the database to be used as seeds. This increases the likelihood of choosing a point near the centre of a cluster simply as that is where the highest density of points is located. However, there is no guarantee that the chosen seeds will be near the centre of the same cluster. Also, the repeated runs of this random generation to obtain a solution make it a time consuming method.

Another strategy was introduced in (MacQueen 1967). It depends on choosing the first '*k*' points from a random sequence in the database as preliminary

seeds. After choosing these seeds, the next data point in the database is assigned to the cluster which is represented by the nearest seed. An updating process moves the centre of that cluster to be the mean of all points assigned to it. This process is repeated for all other data points in the database. The main disadvantage of this strategy is that it takes a very long time with large databases, as the mean vector must be calculated every time a new instance is added.

The SCS (Simple Cluster Seeking) method (Redmond and Heneghan 2007; Tou and Gonzalez 1977) sets the first instance in the database as the first seed. If the distance to the next point in the database is greater than a defined threshold, then this next point will be selected as a second seed. The method then repeats the same steps with next point where the distance to the next seed will be calculated based on the last selected seed. This process will be repeated until all other '*k*' seeds are selected. The main weakness of this method is its dependence on the order of points in the database and the threshold value.

The Binary Splitting (BS) method was proposed in (Linde et al. 1980). It tends to be a greedy technique, finding the first centre as (k=1) and using the main steps of the K-means algorithm. It adds a small error '*e*' to the centre '*c*' and splits it into two (*c*+*e*) and (*c*-*e*). The K-means steps are then run again until convergence. The cycle of splitting and convergence is repeated until the required number of centres is reached or until each cluster contains only one point. The quality of this algorithm depends on the value of '*e*'. Moreover, the

process of performing the steps of the K-means algorithm after each splitting increases the complexity of the algorithm.

In 1990, Kaufman and Rousseeuw introduced a new seeding method (Kaufman and Rousseeuw 2005). This method selects the first seed as the most central point and the next seed as the one which gives the greatest reduction in the distortion, which is the sum of all distances from each data point to the centre of its cluster. The computation cost for choosing each seed makes this algorithm unsuitable for large datasets.

The Near Optimal Seed Selection method, based on genetic programming, was suggested in (Babu and Murty 1993). This algorithm aims to optimise the seed selection by generating populations each of which consists of various seed selections. The fitness, which is the distortion of the population, is calculated by using the same steps as the K-means algorithm with the proposed seeding mechanism until convergence. The algorithm generates new populations based on the fittest parts of the current population. This process is repeated a pre-specified number of times. The main drawback of this method is that the results vary significantly with the choice of crossover and mutation probabilities (Jain et al. 1999). Also, the process of running the K-means steps every time to calculate the fitness of each generation makes this method infeasible with large datasets.

In 1993, Huang and Harris proposed the Direct Search Binary Splitting (DSBS) algorithm (Huang and Harris 1993). This algorithm was an enhanced form of the BS algorithm through the use of Principal Component Analysis

(PCA), which improves the clustering quality by choosing the 'ε' vector of the BS deterministically.

The KKZ (Katsavounidis, Kuo, and Zhang) algorithm (Katsavounidis et al. 1994) was proposed in 1994. It starts by choosing a point on the edge of the data as a seed. The furthest point from this seed is selected to be the second seed. The third seed is the point in the dataset whose total distance from the first two points is a maximum. The process of choosing the furthest point from its nearest seed is repeated until reaching to the required number of seeds. In addition to being costly in terms of time, any noise in the data can lead to unexpected initialisations.

A new method, which depends on dividing the data space into '*M*' disjoint sub-spaces, was proposed in (Al-Daoud and Roberts 1996). In each sub-space, K_j seeds are generated randomly and placed where the number of data points in the ' j^{th} ' sub-space is ' N_j '. This method was improved by dividing the space into two disjoint volumes and placing seeds on a regular grid in each sub-space. The number of seeds in each sub-space is related to its density. Good results were presented for this method, but only for two-dimensional spaces.

In 1997, creating a cloud of '*k*' seeds around the mean of all points was suggested (Thiesson et al. 1997) by disarranging the mean of the points '*k*' times to generate '*k*' seeds.

Another technique was introduced in 1998 (Bradley and Fayyad 1998). It splits the data into 10, or so, sub-sets and runs the K-means steps on each of

them. The results of these 10 runs which are 10 '*k*' centre points are used as an input to another instance of the K-means algorithm, which runs 10 times. Each of these runs is initialised using the '*k*' final centre locations from one of the 10 sub-sets runs. The resulting '*k*' centre locations from this run are used to initialise the K-means algorithm for the whole dataset.

In 2004, a new method called the Cluster Centre Initialisation Algorithm (CCIA) was introduced (Khan and Ahmad 2004). This method uses the DBMSDC (Density-Based Multi-Scale Data Condensation), which estimates the density of the data at a point and sorts points according to their densities. A point is chosen from the top of the list, and all points within a radius inversely proportional to the density at that point are pruned. Then a next point from non-pruned list is selected. The method repeats the same steps until the remaining points become equal to a pre-specified number.

A number of algorithms with an incremental approach to improve the K-means algorithm have been introduced. One of them was proposed in 2000 (Pelleg and Moore 2000). The algorithm starts with a small number of clusters and then it inserts new centres in suitable positions. Despite its efficiency, this algorithm does not guarantee that the distortion will be minimised after inserting a new centre.

Nguyen (Nguyen 2004) also suggested an improved mechanism for centre insertion by adding a small value to the current centre and inserting the new centre close to the one which we want to duplicate. This algorithm generates good clustering but it takes a long time to be applied to the whole dataset, so it is not suitable for large datasets.

3.3 The Kd-tree and Data Clustering

The Kd-tree (Bentley 1980; Friedman et al. 1977) is a multi-dimensional binary tree used for organising data points in the k-dimensional space in order to decrease the time of the nearest neighbour search. It consists of a number of nodes and leaves where the root node contains all data points. Each node in the tree has two child nodes, while leaves are without child nodes. Data points, or references to them, are stored in leaves only.

The Kd-tree uses splitting planes that are perpendicular to one of the coordinate system axes, chosen so that it goes through one of the points in the tree. There are several variants of the Kd-tree according to the stopping criterion or the applied splitting plan.

The idea of using the Kd-tree for clustering was firstly introduced by Moore (Moore 1999). It was used for estimating the parameters of a mixture of Gaussian clusters to reduce the cost of the EM-based clustering.

To speed up the K-means algorithm and make it tractable for large datasets, the blacklisting algorithm (Pelleg and Moore 1999; Pelleg and Moore 2000) was presented. This algorithm updates data points in each cluster in bulk instead of updating each data point in each cluster separately. It is based on a variant of the Kd-tree called MRKd-tree (Multiple Resolution K-dimension tree) (Moore and Lee 1998).

In 2002, the filtering algorithm (Kanungo et al. 2002) was introduced, the algorithm is an efficient variant of the K-means using the Kd-tree. It uses the

Kd-tree as a data structure for storing multi-dimensional data points. Each node of this tree is represented by a box containing a set of data points. The root node is the box which contains all of the points set, while a leaf is a node which contains one point. The splitting criterion used in this algorithm depends on hyper-planes splitting orthogonally to the longest side of the node through the median coordinate of the associated points.

The filtering algorithm starts by creating the Kd-tree for the given data points. For each node in the tree, the algorithm maintains a set of candidate centres. The candidate centres for the root consist of all '*k*' centres. The algorithm propagates for each node candidates which are the closest to the mid-point of this node. A filtering process then starts to select the nearest centre from these candidates by filtering (pruning) any centre which is not closer to any part of the node than any others. If there is no nearest centre for all points in the node, then this process will recur on its children.

To improve the filtering algorithm, the fast greedy algorithm and the restricted filtering algorithm were proposed in (Hussein 2002). The former modified the boundaries of the node and the latter added a new threshold for the direct computation of distance and for Kd-tree splitting.

Aiming to increase the number of seeds until '*k*' seeds are found, the global K-means algorithm (Likas et al. 2003) was proposed. This algorithm employs the Kd-tree to use the centres of '*k*' created buckets as seeds for the K-means algorithm. It uses a variant of the Kd-tree of a splitting criterion that splits each bucket along the direction of the maximum variance.

Recently, Redmond and Heneghan described their method for initialising the K-means algorithm (Redmond and Heneghan 2007). The Kd-tree is used to perform a density estimation of the data. The algorithm then chooses '*k*' seeds for the K-means algorithm in the highest density nodes of the Kd-tree. This method uses a variation of the Kd-tree which splits data along the median.

In this chapter, improvements to the basic Kd-tree structure are reported. The improved Kd-tree is used as a data structure to improve the choice of initial centres and to reduce the number of the nearest neighbour searches. It was also combined with an efficient centre insertion technique to propose an incremental operation that overcomes the instability problem of K-means algorithm.

3.4 The Proposed Algorithm

3.4.1 Conventions

For convenience, in this chapter, the information in a dataset is presented in tree leaves, where the leaf is a node without any children. Each cluster is presented in a set of leaves.

Each node is represented by <*Node_id, Hmax, Hmin*> where the '*Node_id*' is just a sequential ID, '*Hmax*' and '*Hmin*' are the upper and lower bounds of the node dimensions.

3.4.2 The Improved Kd-tree

The proposed algorithm depends on the operation of an inline construction for an improved variant of the Kd-tree. In this variant, the region of each node is represented by a hyper-rectangle identified by two vectors '*Hmax*' and '*Hmin*'.

When creating a new node, the algorithm calculates a number of attributes for the node and uses them in the next steps to decrease the mathematical calculations required in each step. The new suggested structure includes a number of additional attributes compared to those used in the filtering and fast greedy algorithms to make the computing process faster. Each node in the proposed structure has the following attributes:

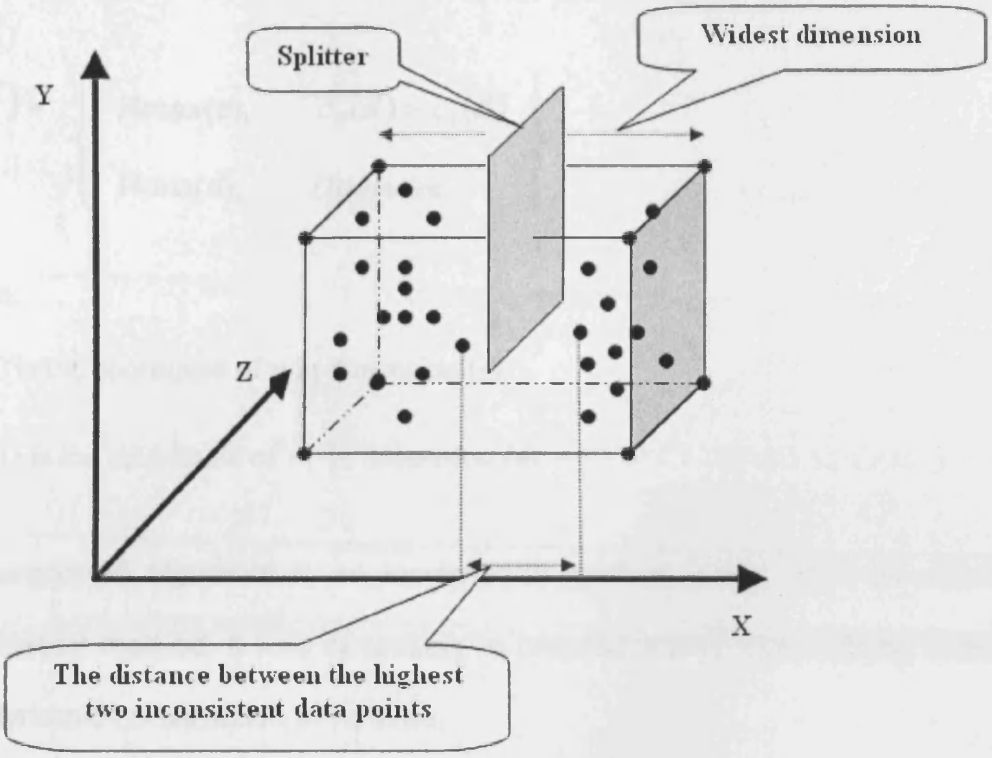
- node_id: a unique number for each node in the tree.
- cluster_id: the ID of the cluster to which the node belongs.
- density: the density of the node.

- items_number: the number of items in the node.
- total_sum: the sum of all items in the node.
- HMax: the array of the upper bounds of the node dimensions.
- HMin: the array of the lower bounds of the node dimensions.
- items: a list of items' indexes in the node.
- left_child: a pointer to the left child of the node.
- right_child: a pointer to the right child of the node.

These attributes of each node need to be computed just once when the node is created.

This tree is further improved by implementing a new splitting method that is suitable for clustering. The splitting point is selected to be the average of the farthest two neighbouring data points in that dimension (Figure 3.1a), and happens to the widest dimension every time the algorithm wants to split a leaf.

The mechanism of assigning a leaf '*L*' to a cluster is simple (Figure 3.1b). This mechanism, is illustrated in detail in (Kanungo et al. 2002). It depends on the extreme point '*p*' in the direction $\vec{v} = c_2 - c_1$ of the leaf '*L*' and the two clusters' centres 'c_1' and 'c_2'. By calculating the distance from '*p*' to 'c_1' and 'c_2', it will be known which of 'c_1' and 'c_2' is closer to '*L*' and whether all the data points in '*L*' belong to the cluster of centre 'c_1' or 'c_2'.



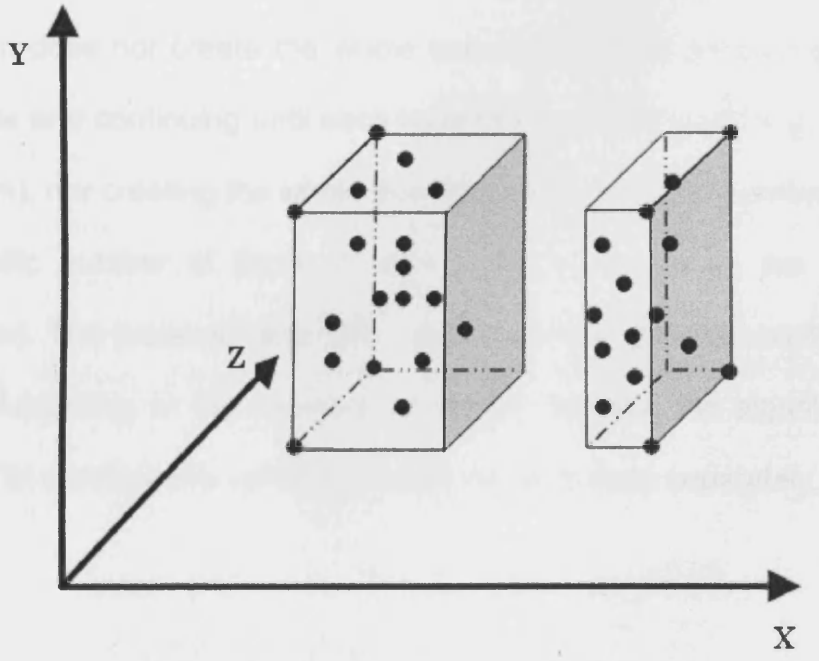


Figure 3.1a – The proposed new splitting method for a node.

The extreme point '*p*' can be determined as the corner of the leaf, as follows:

$$p(d) = \begin{cases} \text{Hmax}(d), & c_2(d) > c_1(d) \\ \text{Hmin}(d), & \textit{Otherwise} \end{cases}$$

Where:

$p(d)$: is the coordinate of p in dimension (d).

$c_i(d)$: is the coordinate of c_i in dimension (d).

The proposed algorithm is an incremental method and it uses an efficient initialisation method. It was necessary to propose a new tree building method of a dynamic construction in runtime.

Unlike the filtering, restricted and fast greedy algorithms, the proposed algorithm does not create the whole tree starting from all data points in the root node and continuing until each node has one data point (e.g., the filtering algorithm), nor creating the whole tree with nodes until a pre-defined threshold (a specific number of items in each node) is met (e.g., the fast greedy algorithm). The proposed algorithm starts with one node that contains all data points. According to the required number of clusters, the algorithm decides whether to continue the splitting process for each node separately, or to stop.

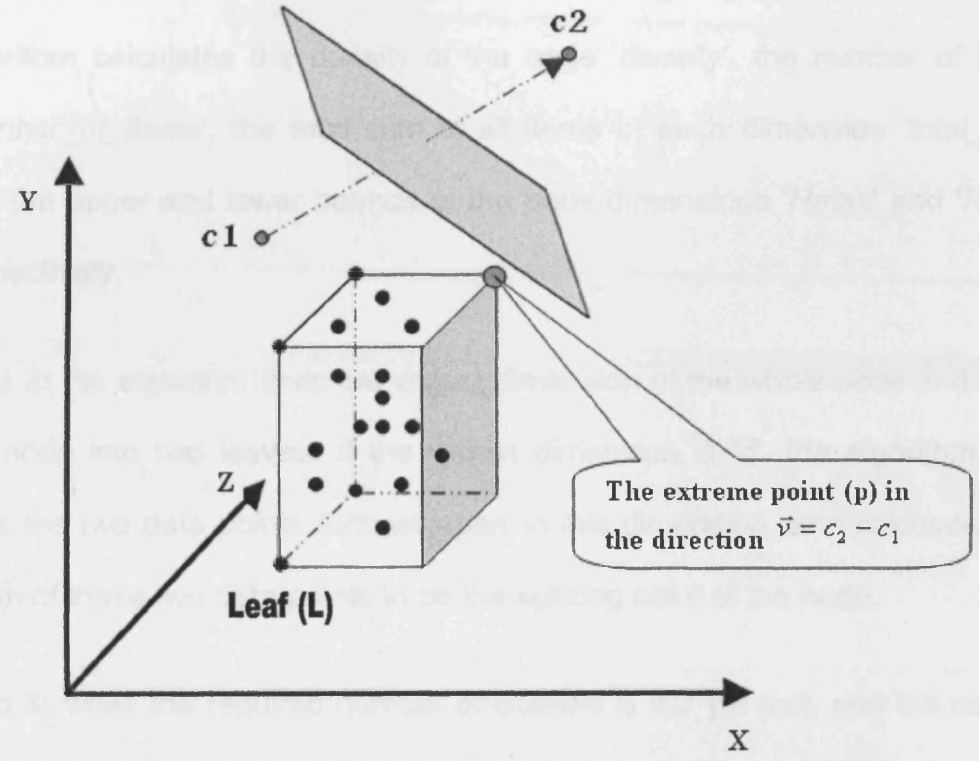


Figure 3.1b – The mechanism of assigning a leaf to a cluster

3.4.3 Algorithm Description

The main steps of the proposed algorithm, see Figure 3.1c, are as follows:

Step 1: the procedure starts by adding references to all data points in one node which is the root of the tree. When creating any node in the tree, the algorithm calculates the density of the node '*density*', the number of items '*number_of_items*', the total sum of all items in each dimension '*total_sum*' and the upper and lower bounds of the node dimensions '*Hmax*' and '*Hmin*', respectively.

Step 2: the algorithm finds the widest dimension of the whole node and splits the node into two leaves. If the widest dimension is '*d*', the algorithm then finds the two data points furthest apart in this dimension, and calculates the mean of these two data points to be the splitting point of the node.

Step 3: while the required number of clusters is not yet met, and the current number of clusters is equal to, or more than, the current number of leaves, then the lowest density leaf is split in step 3.1.

Step 3.1: the algorithm splits the lowest density leaf to produce more leaves to insert the new centre in one of them.

Step 3.2: the algorithm inserts a new centre to be the mean of the highest density leaf of the tree.

Step 3.3: the main steps of the K-means algorithm are applied using the created tree.

1. Construct the first node of the tree and add all data points to that node.
2. Split the tree according to the widest dimension.
3. while(current_clusters_number < required_clusters_number) do
    3.1. if (current_clusters_number >= number_of_leaves) then
        Split the lowest density leaf once according to the widest dimension.
    3.2. Insert a new centre in the highest density leaf.
    3.3. Repeat
        3.3.1. Assign leaves to centres.
        3.3.2. Move centres to means.
    Until (no more movement)
    3.4. If (any centre has no leaves) then
        3.4.1. Eliminate this centre
4. End while

Figure 3.1c – The basic steps of the proposed Kd-tree-based algorithm

The algorithm assigns all leaves to an appropriate centre and then each centre is moved to be the mean of all points of the leaves which belong to it.

Step 3.4: If any centre is inserted with no leaves belonging to it, the algorithm eliminates that centre.

3.5 Experiments

The algorithm was applied to three artificial datasets: data1, data2 and data3. It was also applied to real datasets: Iris (Asuncion and Newman 2007), Crude Oil (Johnson and Wichern 2001) and Vowel (Grabmeier and Rudolph 2002). The main characteristics of these real datasets are summarised in Appendix A.

The results of the proposed algorithm were compared to those of the K-means algorithm. The clustering criterion '*E*' (Equation 2.1 in chapter 2) was used to evaluate the performance of the algorithms. The smaller the value of this metric, the better the clustering results. All algorithms were executed many times. The average, minimum and maximum values of '*E*' were noted.

Table 3.1 shows the values of the parameters used in this test for each algorithm. Figure 3.2 shows the value of the distortion '*E*' in each level of the proposed algorithm when applied to the datasets.

Figures 3.3a, b, c, d, e and f show the stability of the proposed algorithm compared to that of the K-means algorithm by giving the value of '*E*' obtained

by the K-means and the proposed algorithm for five runs for data1, data2, data3, Iris, Crude Oil and Vowel datasets, respectively.

The value of the distortion of the proposed algorithm was compared to that of the K-means algorithm in Tables 3.2a and b.

As the proposed algorithm is also intended to reduce the number of nearest neighbour searches, Tables 3.3a and b list the number of distance calculations between data points and cluster centres in each dataset for both the K-means and the proposed algorithms:

All the tests were conducted on a Pentium-IV 2.40GHz machine with 1024MB RAM. Microsoft Visual Studio version 6 C compiler on Windows XP was used for the tests.

Algorithm	Parameters	Value
The K-means Algorithm	Maximum number of iterations	1000
	Number of clusters for each dataset	According to the dataset
The Proposed Algorithm	Number of clusters for each dataset	According to the dataset

Table 3.1 - Parameters used in the clustering experiments

Number of Clusters	Data 1	Data 2	Data 3	Iris	Crude Oil	Vowel
1	134.3	2053.3	899.8	291.4	608.6	2.537×10^6
2	70.36	1588	864.6	128.1	420.8	2.214×10^6
3	47.61	1583	-	98.64	278.9	2.090×10^5
4	-	1151	-	-	-	2.053×10^5
5	-	1068	-	-	-	2.038×10^5
6	-	989	-	-	-	1.519×10^5
7	-	989	-	-	-	-
8	-	975.9	-	-	-	-
9	-	971.1	-	-	-	-

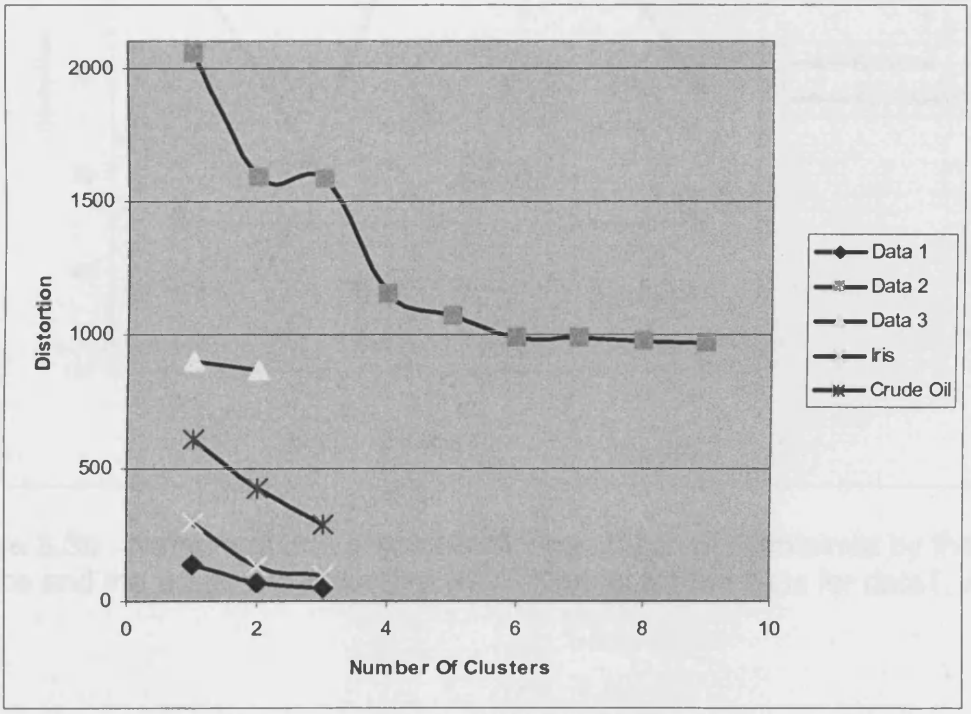


Figure 3.2 – Numerical and graphical representation of E in each level of the proposed algorithm as a function of the number of clusters

Run	1	2	3	4	5
The K-means Algorithm	62.00	51.01	67.17	63.98	54.81
The Proposed Algorithm	47.61	47.61	47.61	47.61	47.61

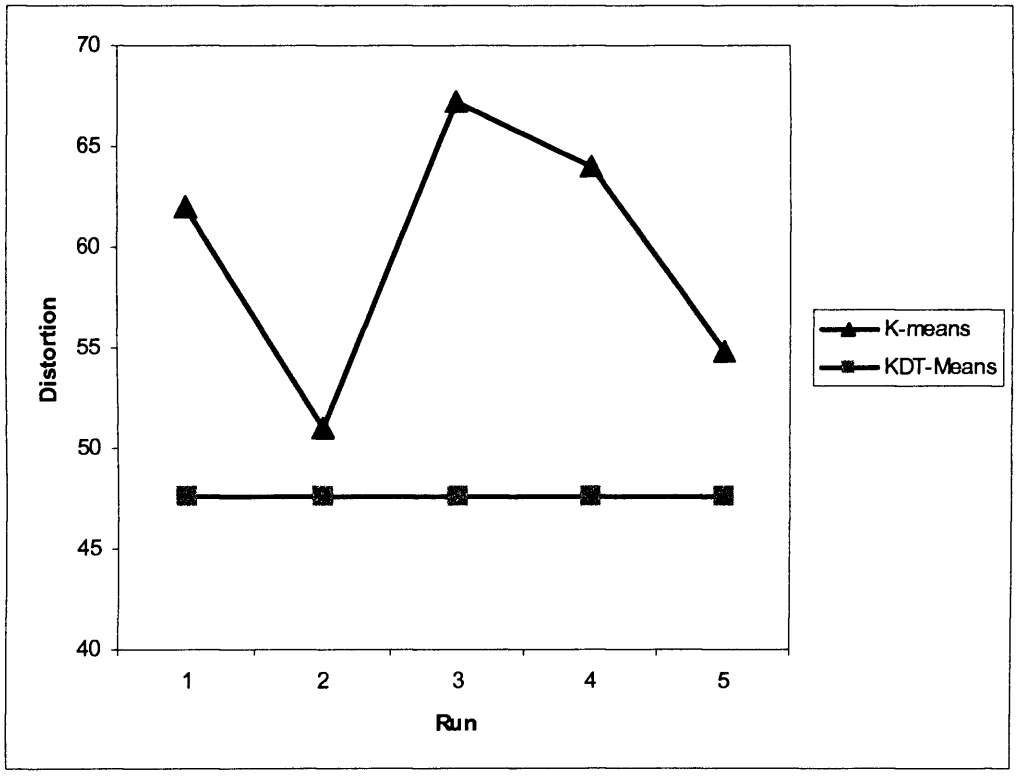


Figure 3.3a – Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for data1, when k=3.

Run	1	2	3	4	5
The K-means Algorithm	917.4	950.6	908.9	1006	976.2
The Proposed Algorithm	971.2	971.2	971.2	971.2	971.2

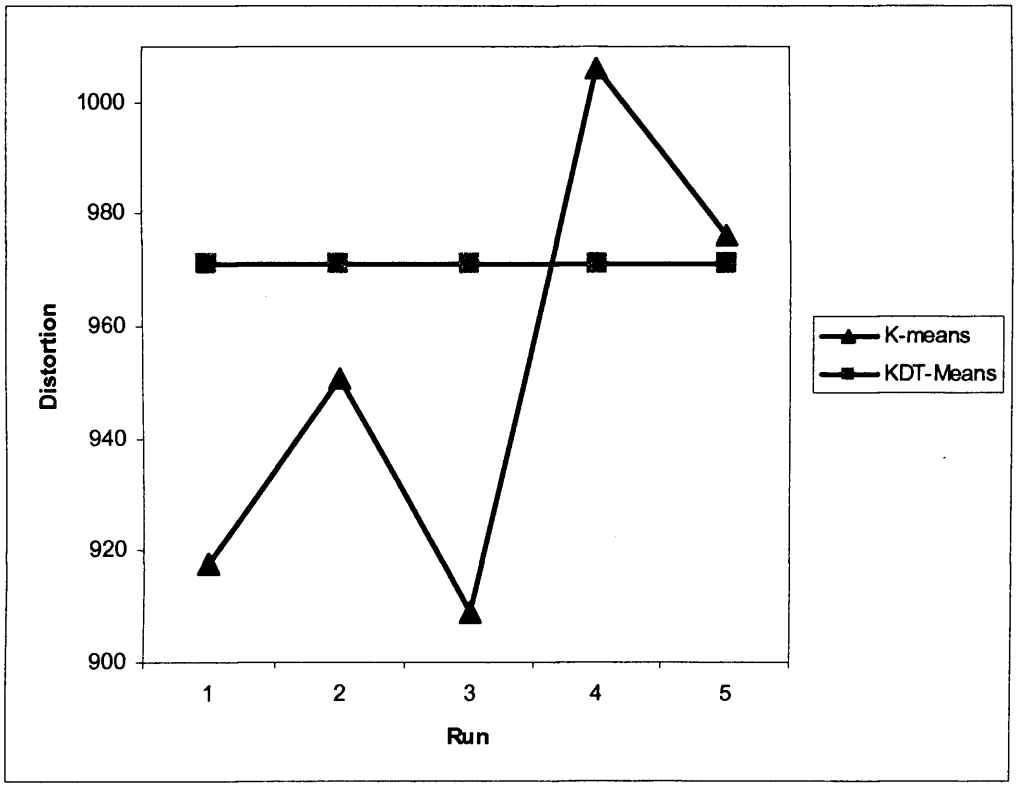


Figure 3.3b – Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for data2, when k=9.

Run	1	2	3	4	5
The K-means Algorithm	1203	1105	1239	1224	1054
The Proposed Algorithm	864.6	864.6	864.6	864.6	864.6

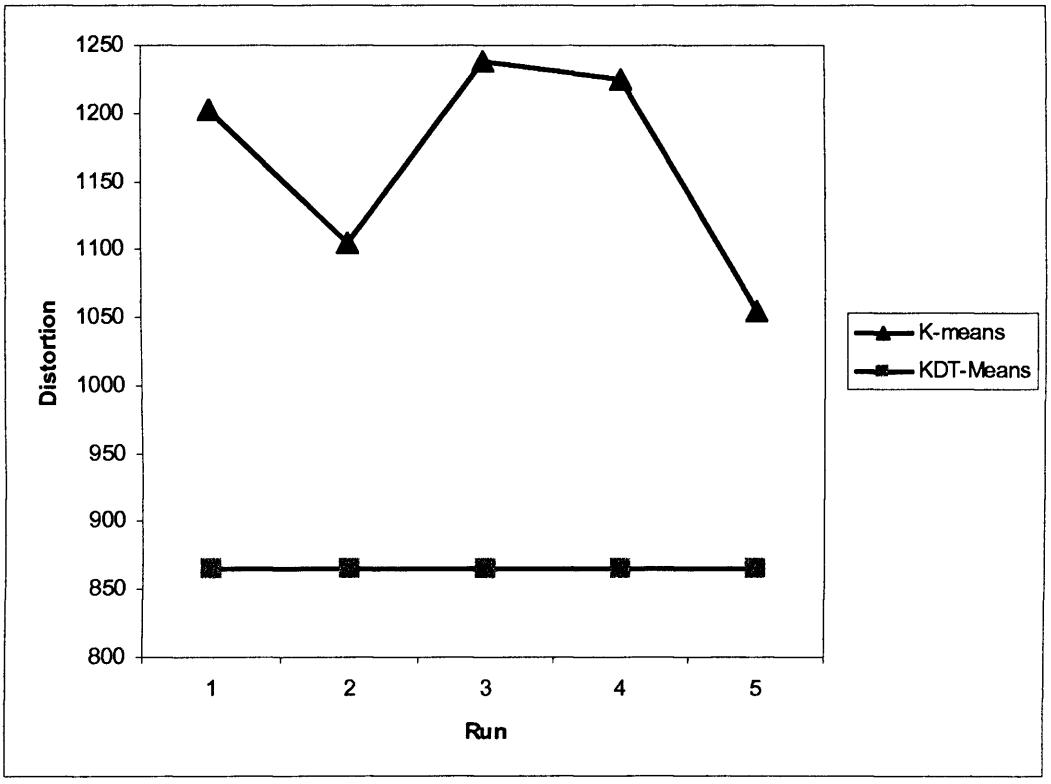


Figure 3.3c – Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for data3, when k=2.

Run	1	2	3	4	5
The K-means Algorithm	97.22	97.20	122.9	124.0	97.20
The Proposed Algorithm	98.64	98.64	98.64	98.64	98.64

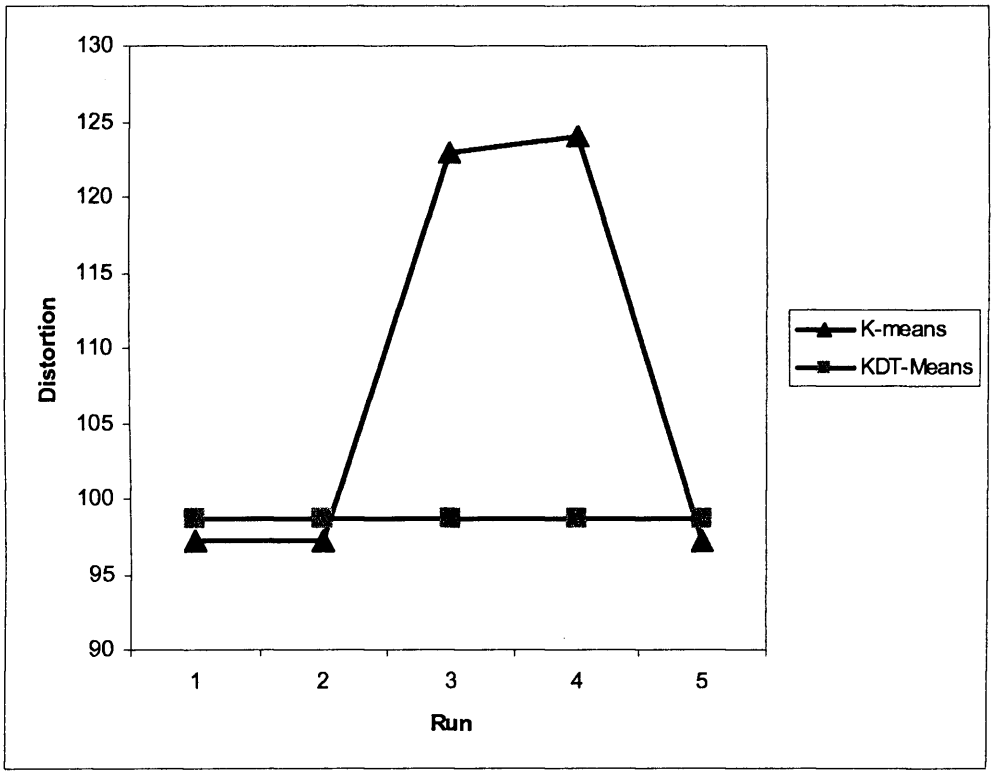


Figure 3.3d – Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for Iris, when k=3.

Run	1	2	3	4	5
The K-means Algorithm	279.7	279.7	279.5	279.6	279.7
The Proposed Algorithm	278.9	278.9	278.9	278.9	278.9

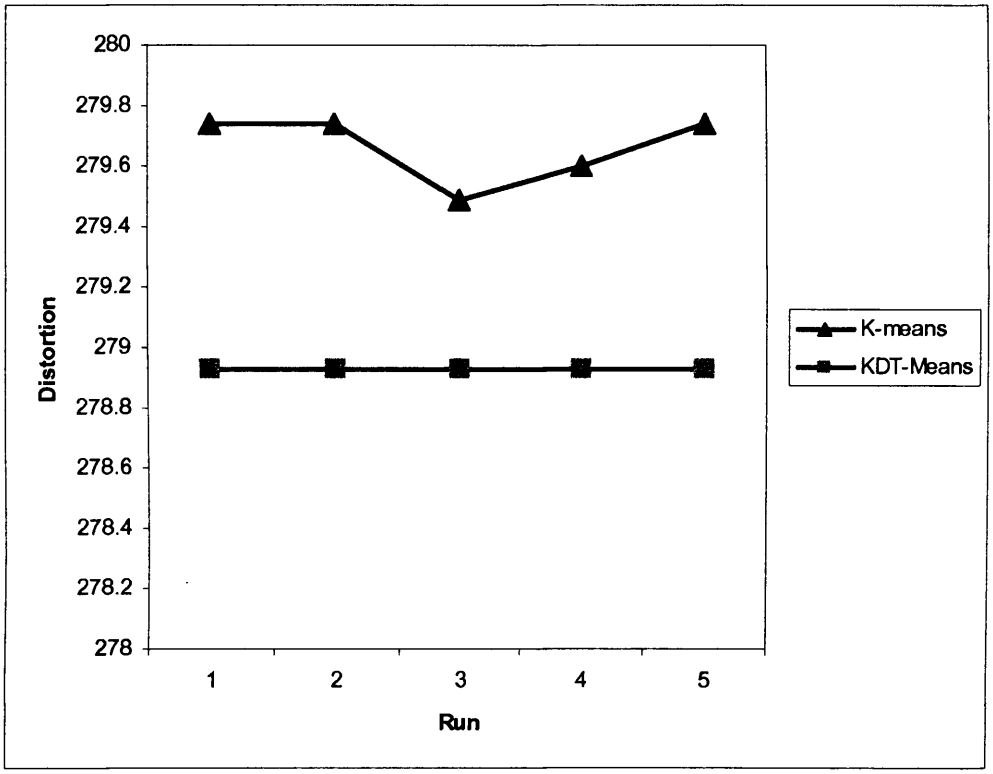


Figure 3.3e - Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for Crude Oil, when k=3.

Run	1	2	3	4	5
The K-means Algorithm	1.553×10^5	1.494×10^5	1.600×10^5	1.611×10^5	1.594×10^5
The Proposed Algorithm	1.519×10^5	1.519×10^5	1.519×10^5	1.519×10^5	1.519×10^5

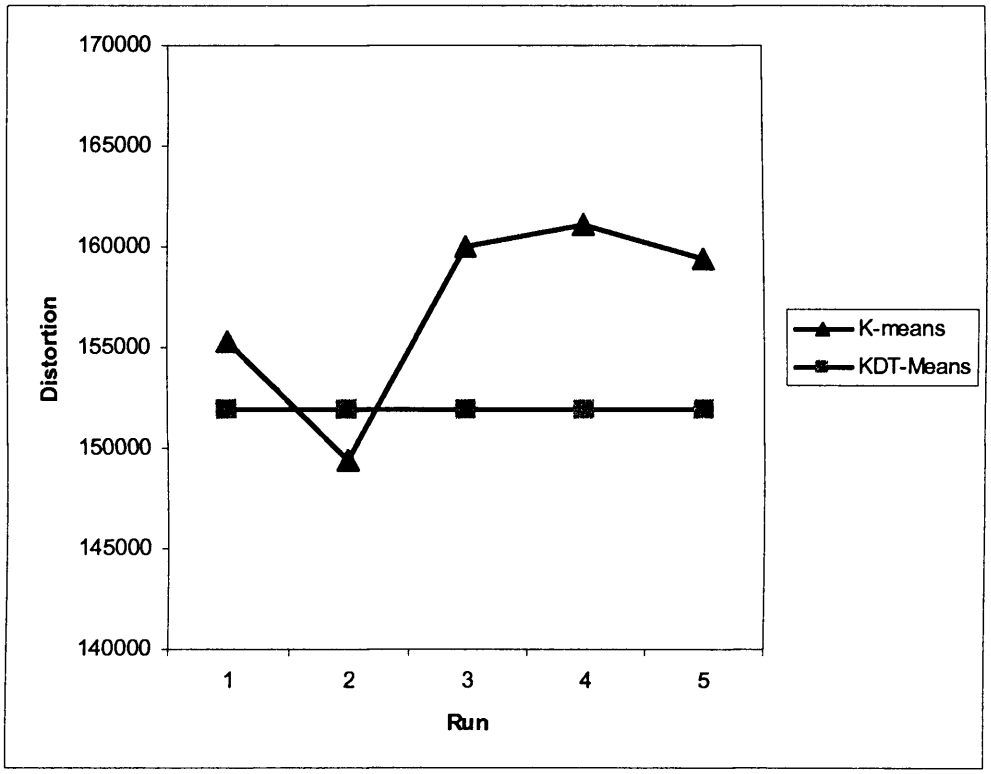


Figure 3.3f – Numerical and graphical representation of *E* obtained by the K-means and the proposed Algorithm (KDT-Means) for five runs for Vowel, when k=6.

Dataset	Algorithm	Mean	Min.	Max.
Data 1	The K-means Algorithm	59.79	51.01	67.17
	The Proposed Algorithm	47.61	47.61	47.61
Data 2	The K-means Algorithm	951.8	908.9	1006
	The Proposed Algorithm	971.2	971.2	971.2
Data 3	The K-means Algorithm	1165	1054	1239
	The Proposed Algorithm	864.6	864.6	864.6

Table 3.2a – Clustering results on the artificial datasets: data1, data2, and data3

Dataset	Algorithm	Mean	Min.	Max.
Iris	The K-means Algorithm	107.7	97.20	124.0
	The Proposed Algorithm	98.64	98.64	98.64
Crude Oil	The K-means Algorithm	279.6	279.5	279.7
	The Proposed Algorithm	278.9	278.9	278.9
Vowel	The K-means Algorithm	1.538×10^5	1.493×10^5	1.611×10^5
	The Proposed Algorithm	1.519×10^5	1.519×10^5	1.519×10^5

Table 3.2b – Clustering results on the real datasets: Iris, Crude Oil and Vowel

Algorithm \ Dataset	Data1	Data2	Data3
K-means	1368	24300	6000
The proposed algorithm	228	9000	2000

Table 3.3a – Number of nearest neighbour (ngh) queries in the clustering algorithms for the artificial datasets

Algorithm \ Dataset	Iris	Crude Oil	Vowel
K-means	1800	1176	20904
The proposed algorithm	450	168	5226

Table 3.3b – Number of nearest neighbour (ngh) queries in the clustering algorithms for the real datasets

3.6 The Algorithm Performance

As can be seen from the experiments, the proposed clustering algorithm outperforms the K-means algorithm in most cases and it gives stable results in all cases. It gives a lower value of the distortion '*E*', which means better clustering solutions. For example, for the Crude Oil dataset, it produced an optimum mean value that is better than the K-means algorithm in all runs.

The stability of the proposed algorithm can be seen in Figures 3.3a, b, c, d, e and f. The algorithm gives the same value of the distortion in 5 different runs, which means overcoming for the problem of instability.

The proposed algorithm suggests a significant improvement as the number of leaves is dynamic and related to the clustering process. This mechanism of tree construction reduces the space complexity drastically. It should be noted that if the average height of the Kd-tree reduces by just one, which happens because of the incremental mechanism of the algorithm, the number of nodes to be expanded is reduced to almost half. Also the '*numberOfLeaves*' is invariably less than the '*numberOfPoints*', and only in the worst case they are equal to each other. As a result of that, it is clear that the usage of the proposed Kd-tree structure reduces the time needed to assign data points to clusters.

The complexity of the proposed algorithm is affected by the number of clusters. The higher the number of clusters, the greater is the time the algorithm takes. Another advantage of the proposed algorithm is that it can

deal with large datasets (large numbers of data points) because of its ability to deal with data points in bulks.

Compared to other Kd-tree-based clustering algorithms, the proposed Kd-tree structure has an inline building method. This reduces the complexity of the algorithm considerably by not running the splitting process and the K-means steps on all leaves of the pre-built tree (as in the blacklisting, filtering and fast greedy algorithms) but by running them on a limited number of leaves of the dynamic tree only. In other words, in the blacklisting and filtering algorithms a whole Kd-tree is built before running the main algorithm steps. This means a longer running time and higher complexity for the algorithm.

Also in the fast greedy algorithm, in spite of the improvement of the tree building process, by using a pre-defined threshold to stop the Kd-tree splitting, it is still not fully inline building. Also the tree must be built before running the main algorithm and it is difficult to specify the stopping threshold if there is no previous knowledge about the dataset.

The complexity is also reduced considerably by not having to find a set of points that could act as an appropriate set for the insertion of a new centre (as in the fast greedy algorithm) and then assigning leaves to centres. But by inserting the new centre directly to be the mean of the highest density leaf, where a better new centre can be added, as has already been proved in (Kanungo et al. 2002).

At the same time, the quality of the solution is generally better. This is achieved by using the new proposed splitting method which increases the possibility of grouping similar data points together in the same leaf.

3.7 Summary

This chapter discusses a new Kd-tree-based clustering method. The proposed algorithm is used to overcome the instability and the large number of nearest neighbour searches of the K-means algorithm.

The results for the proposed algorithm have been compared with those obtained by the K-means algorithm. It has been experimentally demonstrated that the proposed algorithm gives better stable clustering solutions.

It was also shown that the proposed Kd-tree structure improved the process needed to assign data points to clusters so that the number of neighbourhood searches decreased.

CHAPTER 4

Improvements to the K-means Algorithm based on the Bees Algorithm

4.1 Preliminaries

As mentioned in Chapter 2, one of the most popular clustering methods is the K-means algorithm because of its simplicity and computational efficiency. The algorithm involves search and optimisation. One of the major disadvantages of this method is its tendency to converge to local optima.

This chapter proposes a clustering method to avoid local optima and find global solutions to the clustering problem. The algorithm integrates the simplicity of the K-means algorithm with the capability of the Bees Algorithm (Pham et al. 2006b) for locating near optimal solutions efficiently.

The Bees Algorithm performs a version of a neighbourhood search combined with a random search in a way that is reminiscent of the food foraging behaviour of swarms of honey bees. The algorithm has been successfully applied to different optimisation problems, including the training of neural networks for control charts pattern recognition (Pham et al. 2006a) and wood defects identification (Pham et al. 2006f).

The chapter is organised as follows: Section 4.2 reviews the problem of being trapped into a local optimum and different clustering methods used to overcome this problem. Section 4.3 describes the foraging behaviour of bees and the core ideas of the Bees Algorithm. The proposed clustering method is explained in detail in Section 4.4. Results of different clustering experiments are reported in Section 4.5. The performance of the algorithm is discussed in Section 4.6. Section 4.7 summarises and concludes the chapter.

4.2 The Local Optima Problem

Most clustering algorithms suffer from the well-known local optima problem that appears while the clustering algorithm is attempting to optimise a given metric. This problem also exists for the K-means algorithm which divides a dataset '*S*' into '*k*' clusters, then represents each cluster by the mean value of the data points within the cluster. It then attempts to minimise the sum of the Euclidean distances between data points and their closest cluster centres, *E*, (see Equation 2.1 in chapter 2).

It is known that the K-means algorithm may become trapped at local optimal solutions, depending on the choice of the initial cluster centres. To overcome this problem, the GA algorithm (Davis 1991) has been used with the K-means algorithm by introducing a GA-based clustering technique (Garai and Chaudhuri 2003; Maulik and Bandyopadhyay 2000; Murthy and Chowdhury 1996). With this algorithm, solutions (typically, cluster centres) are represented by a population of bit strings.

The search for an appropriate solution begins with a population, or collection, of initial solutions. Members of the current population are used to create the next-generation population by applying operations such as random mutation and crossover. At each step, the solutions in the current population are evaluated relative to some measure of fitness (which typically is inversely proportional to *E*), with the fittest solutions selected probabilistically as seeds for producing the next generation. The process performs a generate-and-test beam search of the solution space, in which variants of the best current solutions are most likely to be considered next.

This GA-based algorithm generates the initial population of the solution totally randomly, which can make the algorithm spend a couple of extra runs to reach a reasonably good solution and try to improve it. One other disadvantage of this algorithm is that sometimes using selection and crossover operators will tend to cause the algorithm to converge on a good but sub-optimal solution.

The main aim in this Chapter is to propose a clustering algorithm based on a new optimisation technique called the Bees Algorithm which improves the accuracy of the final clustering solution with significantly better global solutions to the clustering problem expressed as a minimum distortion (*E)*.

4.3 The Bees Algorithm

4.3.1 Motivation

The Bees Algorithm (Pham et al. 2006b), which was developed in the Manufacturing Engineering Centre in Cardiff University and proposed in 2006, is a very efficient optimisation technique. This algorithm was built on the foraging behaviour of bees.

It has many applications in wide areas, that includes: improving neural networks (Pham et al. 2006a; Pham et al. 2006c; Pham et al. 2006e; Pham and Sholedolu 2008; Pham et al. 2006f), cell formation (Pham et al. 2007a), preliminary design (Pham et al. 2007b), job scheduling (Pham et al. 2007d), feature selection (Pham et al. 2007e), engineering and manufacturing (Lee and Haj Darwish 2008; Pham et al. 2008a; Pham et al. 2008c; Pham et al. 2009a; Pham et al. 2009b; Pham et al. 2008d; Pham et al. 2008f; Pham et al. 2007h), robotics (Pham et al. 2008b; Pham et al. 2007c; Pham et al. 2006d) and solving many other optimisation problems (Pham and Ghanbarzadeh 2007; Pham et al. 2007f; Pham et al. 2008e; Pham et al. 2007g).

The Bees Algorithm is used in this research to solve the problem of becoming trapped into a local optimum in the K-means algorithm.

4.3.2 Bees in Nature

A colony of honey bees can extend itself over long distances in order to simultaneously exploit a large number of food sources (Frisch and Karl 1976; Seeley 1996). The foraging process begins in a colony by scout bees being sent to search for promising flower patches. Flower patches with large amounts of nectar or pollen that can be collected with less effort tend to be visited by more bees, whereas patches with less nectar or pollen receive fewer bees (Camazine et al. 2003).

During the harvesting season, a colony continues its exploration, keeping a percentage of the population as scout bees (Seeley 1996). When they return to the hive, those scout bees that found a patch rated above a certain quality threshold deposit their nectar or pollen and go to a "dance floor" at the entrance to the hive to perform a dance known as the "waggle dance" (Frisch and Karl 1976).

This mysterious dance is essential for colony communication, and contains three pieces of information regarding a flower patch: the direction in which it will be found, its distance from the hive and its quality rating (or fitness) (Camazine et al. 2003; Frisch and Karl 1976). This information helps the colony to send its bees to flower patches precisely. Each individual's knowledge of the location of the patch in the outside environment is gleaned solely from the waggle dance. This dance enables the colony to evaluate the relative merit of different patches according to both the quality of the food they provide and the amount of energy needed to harvest it (Camazine et al.

2003). After waggle dancing on the dance floor, the dancer (i.e. the scout bee) goes back to the flower patch with follower bees that were waiting inside the hive. A greater number of follower bees are sent to more promising patches. This allows the colony to gather more food quickly and efficiently.

While harvesting from a patch, the bees monitor its food level. This is necessary to decide upon the next waggle dance when they return to the hive (Camazine et al. 2003). If the patch is still good enough as a food source, then it will be advertised in the waggle dance and more bees will be recruited to that source.

4.3.3 The Bees Algorithm

The Bees Algorithm requires a number of parameters to be set, namely, the number of scout bees (*n*), the number of sites selected for neighbourhood search (*m*), the number of best "elite" sites out of '*m*' selected sites (*e*), the number of bees recruited for the best '*e*' sites (*nep*), the number of bees recruited for the other '*m-e*' selected sites (*nsp*) and a stopping criterion.

1. Initialise the solution population.

2. Evaluate the fitness of the population.

3. While (the stopping criterion is not met)

   // Forming a new population

   a. Select sites for neighbourhood search.

   b. Recruit bees for selected sites (more bees for the best '*e*' sites) and evaluate fitnesses.

   c. Select the fittest bee from each site.

   d. Assign remaining bees to search randomly and evaluate their fitnesses.

4. End While.

Figure 4.1 – Basic steps of the Bees Algorithm

The algorithm, see Figure 4.1, starts with an initial population of '*n*' scout bees (**step 1**). The fitnesses of the sites visited by the scout bees are evaluated in **step 2**.

In step 3.a, bees that have the highest fitnesses are designated as "selected bees" and sites visited by them are selected for neighbourhood search. Then, **in steps 3.b**, the algorithm conducts searches in the neighbourhood of the selected sites, assigning more bees to search near to the best '*e*' sites.

The bees can be chosen directly according to the fitnesses associated with the sites they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighbourhood of the best '*e*' sites which represent more promising solutions are made more detailed by recruiting more bees to follow them than other bees. **In step 3.c**, for each site only the bees with the highest fitness will be selected to form the next bee population.

In step 3.d, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. At the end of each iteration, the colony will have two parts to its new population– representatives from each selected site and other scout bees assigned to conduct random searches.

Steps 3.a, b, c and d are repeated until either the best fitness value has stabilised or the specified maximum number of iterations has been reached.

4.4 The Proposed Algorithm

As mentioned before, the proposed new clustering method exploits the search capability of the Bees Algorithm to overcome the local optimum problem of the K-means algorithm. More specifically, the task is to search for appropriate cluster centres (c_1, c_2, ..., c_k) such that the clustering metric E is minimised. The basic steps of the proposed clustering operation are essentially similar to those of the Bees Algorithm and may be considered as being shown in Figure 4.1. These steps are described in detail below.

In step 1, the algorithm starts by generating an initial population of '*n*' scout bees where each bee represents a set of '*k*' cluster centres. Thus, initially, there are '*k*' randomly selected data points.

In step 2, the fitness of the sites visited by the scout bees is evaluated. The fitness computation process consists of steps that are similar to the steps of the conventional K-means algorithm. First, the '*k*' clusters are formed based on the initial centres by allocating each data point to the cluster with the nearest centre to it.

Then, the current centres of the '*k*' clusters are replaced by the centres of the newly formed clusters. Finally, the clustering metric E for the new centres is computed (the smaller the value of E, the higher the fitness).

In step 3, the rest of the algorithm steps are performed by repeating a loop of selecting sites for neighbourhood search and assigning more bees to search near to the best *e* sites so that for each site only the bee with the highest

fitness will be selected to form the next bee population and the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions.

Each time an evaluation of bees takes place, the algorithm assigns data points to the nearest centres of clusters which are represented by the bee and then it calculates the value of *E* which represents the fitness of that bee.

4.5 Experiments

This section presents the results of testing the proposed Bees-based clustering algorithm against the K-means and GA-clustering algorithms. The algorithms were applied to three artificial datasets (Data1, Data2, and Data3) and five real datasets (Iris, Vowel, Crude Oil, Control Charts and Wood Defects). The main characteristics of these datasets are summarised in Appendix A. The clustering criterion *E* was used to evaluate the performance of the tested algorithms: the smaller the value of this metric, the better the clustering results.

Table 4.1 shows the parameter values for each algorithm used in this test. Figures 4.2a, b, c, d, e, f, g and h show the values of the clustering criterion, *E* obtained by the K-means, GA and the proposed Algorithm for five different runs for Data1, Data2, Data3, Iris, Vowel, Crude Oil, Control Charts and Wood Defects datasets respectively, where '*k*' is the number of clusters.

The algorithms were executed many times and the average, minimum and maximum values of E are given. Tables 4.2a and b summarise the results obtained for each algorithm.

All tests were conducted on a Pentium 4 2.40GHz machine with 1022MB RAM. Microsoft Visual Studio version 6 C compiler on Windows XP was used for the tests.

Algorithm	Initial Parameters	Value
K-means	Maximum number of iterations	1000
GA	Crossover probability, μ_c	0.8
	Mutation probability, μ_m	0.001
	Population size, P	100
Bees Algorithm	Number of scout bees, n	21
	Number of sites selected for neighbourhood search, m	8
	Number of best "elite" sites out of m selected sites, e	2
	Number of bees recruited for best e sites, nep	5
	Number of bees recruited for the other (m-e) selected sites, nsp	2
	Number of iterations, R	300

Table 4.1 – Initial parameters used in the crisp clustering experiments

Run	K-means	GA	Bees
1	51.013	51.013	47.536
2	64.647	51.013	47.534
3	67.167	51.013	47.644
4	51.013	51.013	47.565
5	64.725	51.013	47.531

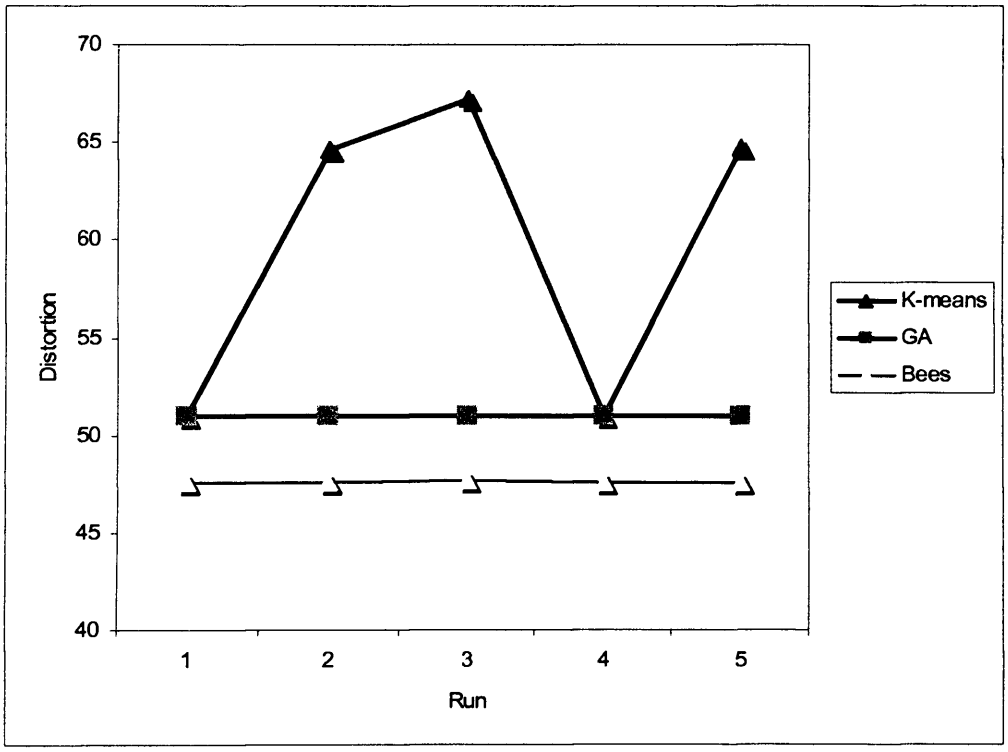


Figure 4.2a – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Data1, when *K*=3

Run	K-means	GA	Bees
1	976.24	966.35	739.56
2	976.38	966.38	749.78
3	976.38	966.35	747.04
4	976.56	966.31	744.26
5	976.38	966.35	747.24

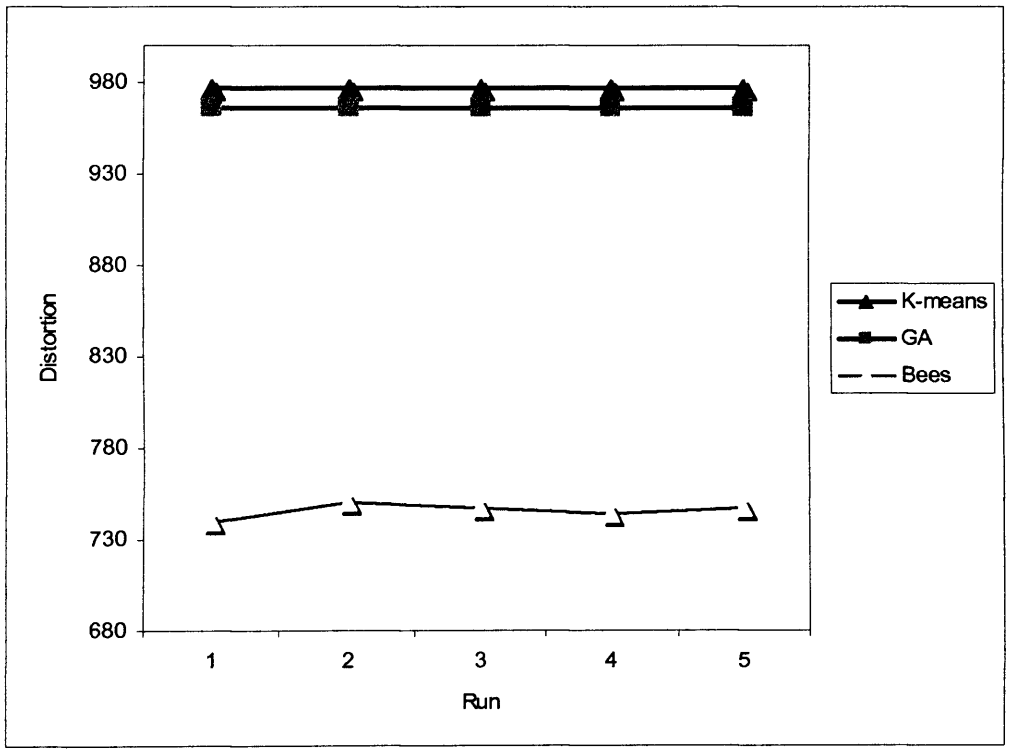


Figure 4.2b – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Data2, when *K*=9

Run	K-means	GA	Bees
1	1246.2	1246.2	864.97
2	1246.2	1246.2	864.97
3	1246.2	1246.2	863.18
4	1246.2	1246.2	863.76
5	1246.2	1246.2	862.15

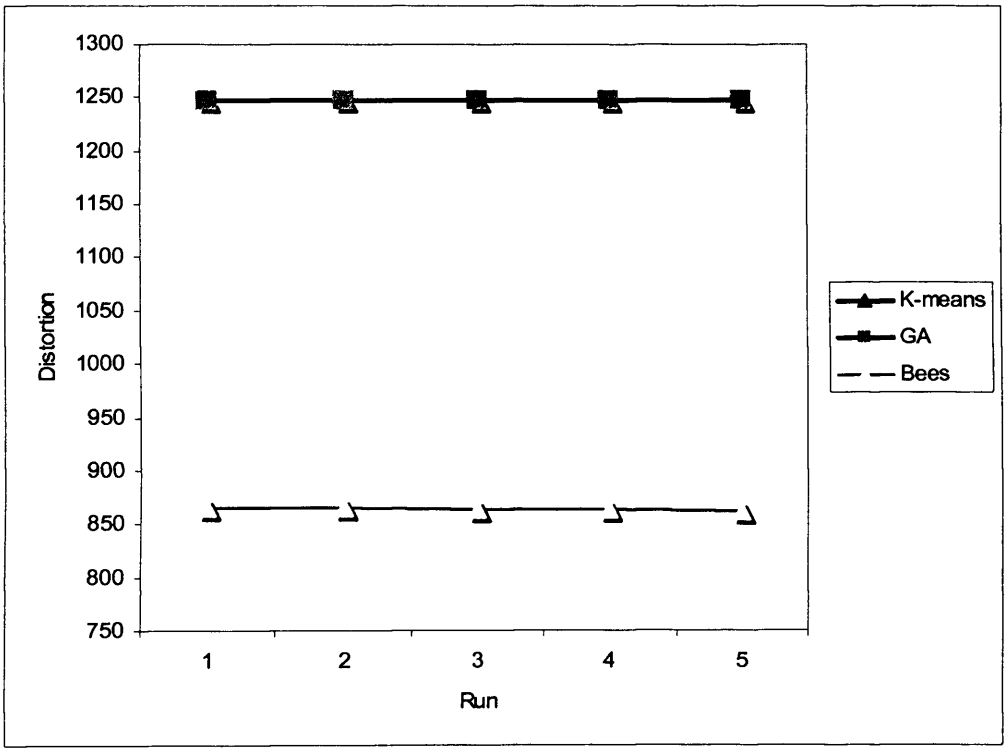


Figure 4.2c – Numerical and graphical representation of E obtained by the K-means, GA and the Bees-based Algorithms for five runs for Data3, when K=2

Run	K-means	GA	Bees
1	97.224	97.101	96.728
2	97.205	97.101	96.787
3	122.946	97.101	96.787
4	124.022	97.101	96.770
5	97.205	97.101	96.749

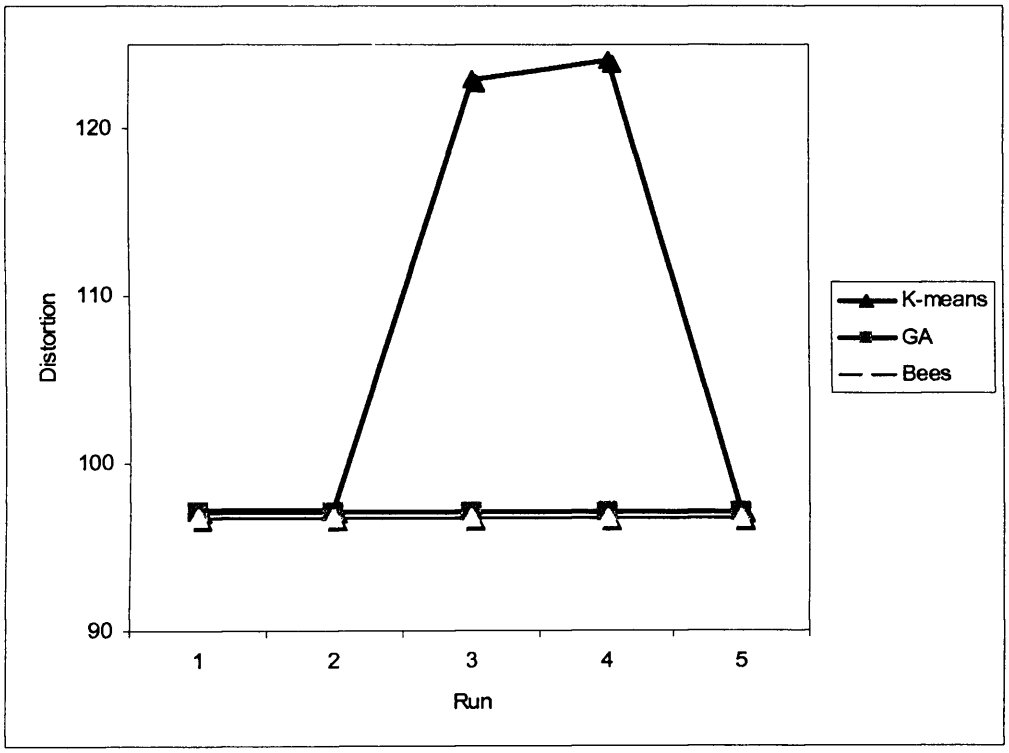


Figure 4.2d – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Iris, when *K*=3

Run	K-means	GA	Bees
1	157460	149350	149290
2	149390	149410	149110
3	161090	149350	149110
4	149370	149360	149110
5	151610	149360	149070

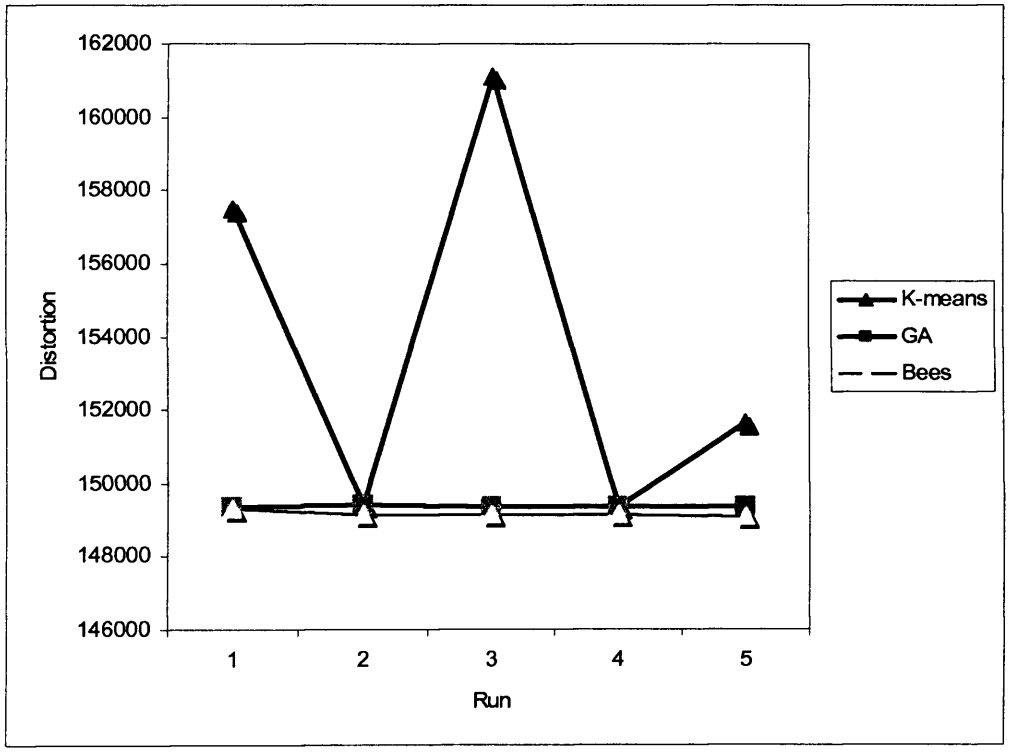


Figure 4.2e – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Vowel, when *K*=6

Run	K-means	GA	Bees
1	279.74	278.97	277.31
2	279.74	278.97	277.30
3	279.48	278.97	277.32
4	279.60	278.97	277.31
5	279.74	278.97	277.52

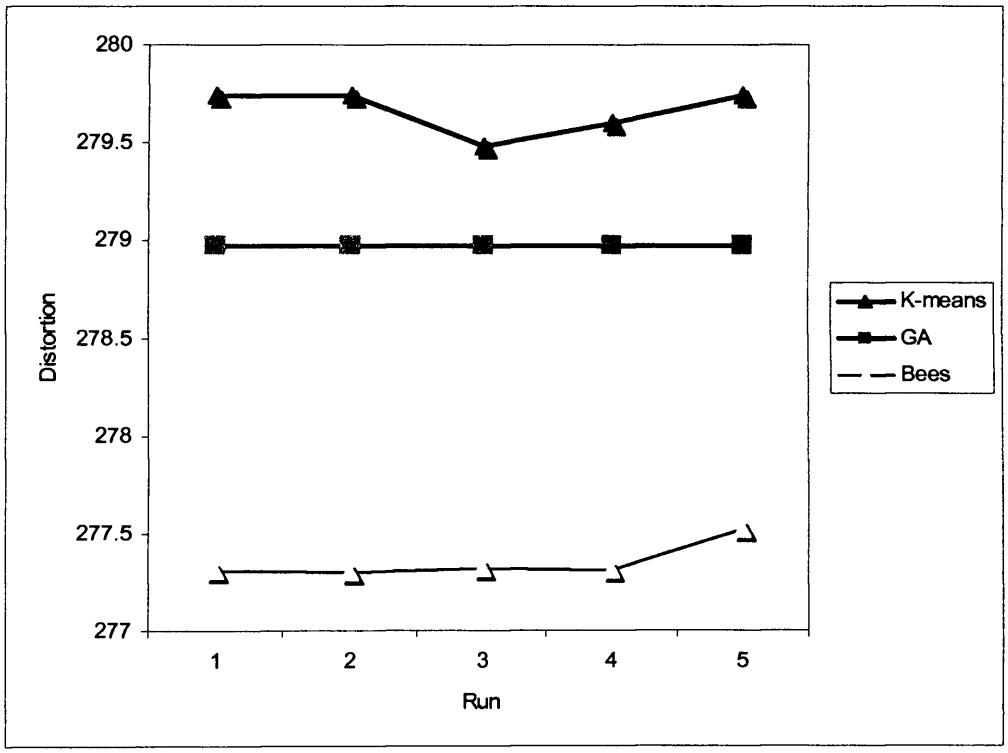


Figure 4.2f – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Crude Oil, when *K*=3

Run	K-means	GA	Bees
1	2474.5	2346.9	2318.8
2	2487.1	2316.6	2324.4
3	2528.7	2298.3	2213.6
4	2464.8	2303.3	2250.6
5	2496.2	2291.7	2318.8

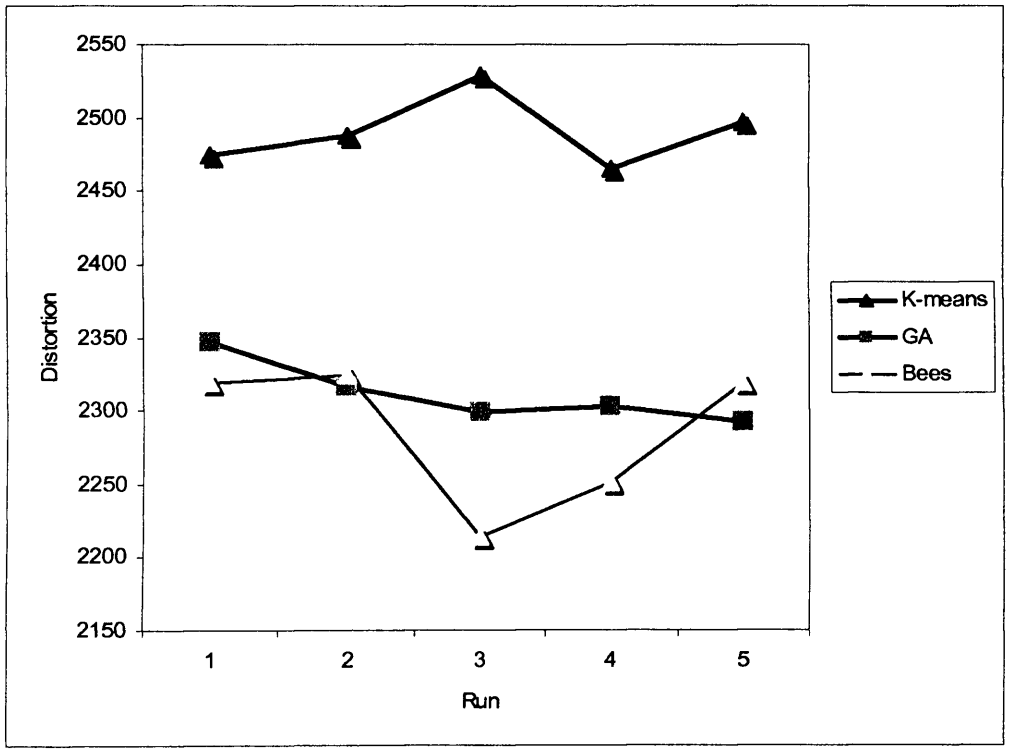


Figure 4.2g – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Control Charts, when *K*=6

Run	K-means	GA	Bees
1	204780	170390	153870
2	202250	168860	159010
3	270580	166960	168040
4	199310	166960	156330
5	263710	165740	163120

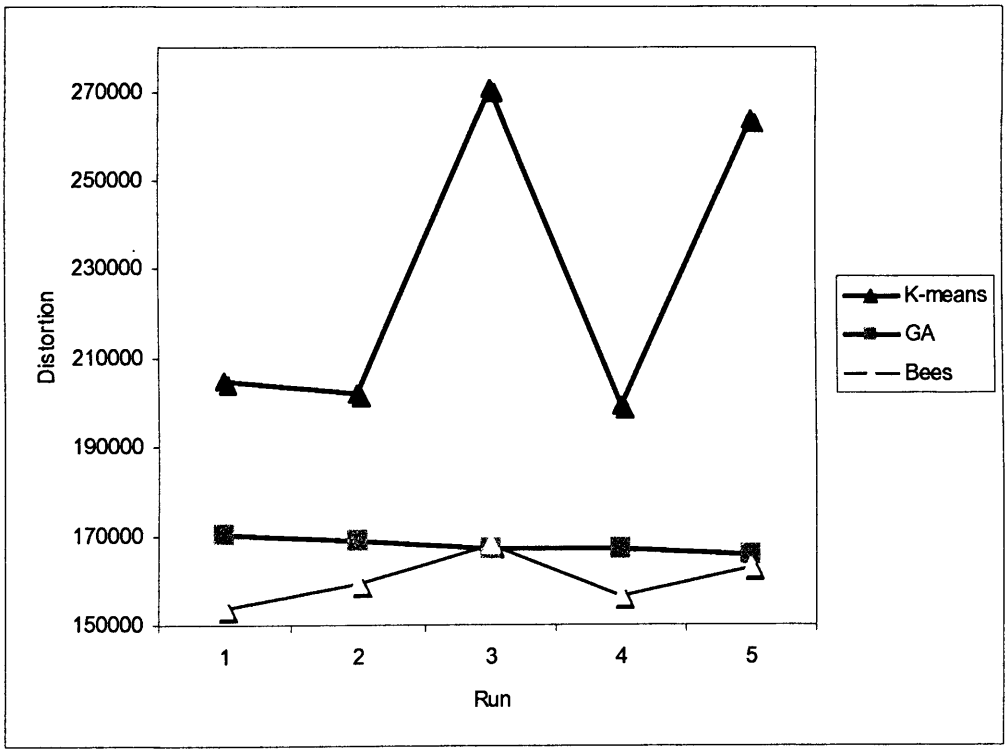


Figure 4.2h – Numerical and graphical representation of *E* obtained by the K-means, GA and the Bees-based Algorithms for five runs for Wood Defects, when *K*=13

Dataset	Algorithm	Mean	Min.	Max.
Data1	K-means	59.713	51.013	67.167
	GA	51.013	51.013	51.013
	Bees Algorithm	47.562	47.531	47.644
Data2	K-means	976.39	976.24	976.56
	GA	966.35	966.31	966.38
	Bees Algorithm	745.57	739.56	749.78
Data3	K-means	1246.2	1246.2	1246.2
	GA	1246.2	1246.2	1246.2
	Bees Algorithm	863.81	862.15	864.97

Table 4.2a – Summary of results for the values of E obtained for the three crisp clustering algorithms for artificial datasets

Dataset	Algorithm	Mean	Min.	Max.
Iris	K-means	107.72	97.205	124.02
	GA	97.101	97.101	97.101
	Bees Algorithm	96.764	96.728	96.787
Vowel	K-means	153790	149370	161090
	GA	149360	149350	149400
	Bees Algorithm	149140	149070	149290
Crude Oil	K-means	279.66	279.49	279.74
	GA	278.97	278.97	278.97
	Bees Algorithm	277.35	277.30	277.52
Control Charts	K-means	2490.3	2464.8	2528.7
	GA	2322.2	2291.70	2346.9
	Bees Algorithm	2285.3	2213.6	2324.4
Wood Defects	K-means	228130	199306	270580
	GA	167780	165740	170390
	Bees Algorithm	160074	153870	168040

Table 4.2b – Summary of results for the values of *E* obtained for the three crisp clustering algorithms for real datasets

4.6 The Algorithm Performance

The evaluation function E was calculated for three artificial and five real datasets. As can be seen in Tables 4.2a and b, the proposed new clustering method outperforms the other two algorithms in all cases. In fact, in every test the mean value of the new algorithm is less than the minimum for the K-means and GA algorithms. For example, at one extreme, for the Wood Defects dataset, the Bees-based Algorithm gave 41% and 3% better results than K-means and GA-clustering algorithms respectively, and at the other extreme for the Crude Oil dataset, the Bees-based algorithm produced a mean value for E that was 0.8% and 0.6% better than the K-means and GA-clustering algorithms, respectively.

In general, all eight test results showed that a lower value of E was found when the Bees-based Algorithm was applied to the datasets. From these results, it is inferred that the smart search technique used in the Bees Algorithm finds a better solution than the GA-based clustering and the K-means algorithms.

4.7 Summary

This chapter has presented a new crisp clustering method based on the Bees Algorithm to overcome the problem of trapping into the local optima in the K-means algorithm. The new method employs the Bees Algorithm to search for the set of cluster centres that minimises a given clustering metric. The proposed method does not become trapped at locally optimal solutions. This

is due to the ability of the Bees Algorithm to perform local and global search simultaneously.

Experimental results for eight different datasets have demonstrated that the proposed method produces better performances than those of the K-means algorithm and the GA-based clustering algorithm.

CHAPTER 5

Improvements to the C-means Algorithm based on the Bees Algorithm

5.1 Preliminaries

Traditional data clustering, also called crisp clustering, uses hard thresholds to group data points into separate groups while fuzzy clustering uses fuzzy logic to create overlapped groups of data.

In fuzzy clustering, each data object can belong to more than one cluster at the same time with a different possibility degree or membership function value. The value of the membership function of a cluster varies between 0 and 1. Conversely, the membership function in crisp clustering is based on having only one of two values; 0 and 1. The value 1 indicates that the object belongs to the cluster, otherwise it is 0.

Fuzzy clustering algorithms produce more realistic results than other clustering techniques. They have been widely used, especially in pattern recognition (Bezdek et al. 2005) and medical applications (Wang et al. 2005).

In this chapter, the Bees Algorithms, which has been already combined with the K-means algorithm to improve the crisp clustering results, is integrated with the FCM (Bezdek 1981), one of the most well-known fuzzy clustering algorithms, to solve the problem of trapping into local optima.

The rest of the chapter is organised as follows: section 5.2 reviews the FCM algorithm and its main improvements; section 5.3 describes the proposed fuzzy clustering algorithm; section 5.4 shows the experimental results obtained from testing the algorithm; the performance of the algorithm is discussed in section 5.5; finally, section 5.6 summarises the chapter.

5.2 Review

The FCM algorithm (Figure 2.16 in Chapter 2) is the most popular one in fuzzy clustering. It was introduced and developed by Dunn (Dunn 1973) and then improved by Bezdek (Bezdek et al. 2005). The algorithm is based on minimising an objective function (Equation 5.1) that indicates the sum of distances from each cluster centre to the data points in that cluster, so that the smaller the value of J, the better the clustering.

The total number of clusters is a required parameter. When performing fuzzy clustering, the FCM algorithm produces fuzzy clusters and it generates the membership degree of each data object to each cluster.

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2$$

(Equation 5.1)

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|} \right)^{\frac{2}{m-1}}}$$

(Equation 5.2)

where:

m : is the fuzziness degree (any real number greater than 1).

c_j : is the centre of the cluster j

N : is the number of data objects

x_i : is the i^{th} d-dimensional measured data object.

u_{ij} : is the degree of membership of x_i in the cluster j

||*|| : is any norm expressing the similarity between any measured data and the centre

The main problem of this iterative-based algorithm is that it can become trapped into local optima. To overcome this problem, the GA was combined with the FCM algorithm to obtain a better clustering performance in (Liu and Xie 1995). One disadvantage of this algorithm is that the usage of selection and crossover operators will sometimes tend to cause the algorithm to converge on a good but sub-optimal solution.

Another algorithm based on a combination of the FCM and the Quantum-behaved Particle Swarm Optimisation (QPSO) was proposed (Wang et al. 2007). The algorithm was used to avoid the local minimum problem of the FCM. This algorithm was applied successfully on datasets with a small number of clusters and it focused on finding circle-type or sphere-type clusters only.

A solution that benefits from the global search strategy of the evolutionary programming, called EPFCM (Evolutionary Programming based FCM), was proposed in (Donga et al. 2009). It was used to improve the FCM algorithm and to change the number of cluster centres dynamically. This algorithm generates the initial population of the solution totally randomly, which can make the algorithm spend a couple of extra runs to reach a reasonably good solution and try to improve it.

The main aim in this Chapter is to propose a new FCM algorithm based on the Bees Algorithm to overcome the problem of local optima in the FCM and to improve the accuracy of the final clustering solution.

5.3 The Proposed Algorithm

The proposed clustering method exploits the search capability of the Bees Algorithm to overcome the local optima problem of the FCM algorithm. Specifically, the task is to search for appropriate cluster centres such that the objective function J (Equation 5.1) is minimised. The basic steps of the proposed clustering operation are described in Figure 5.1.

In step 1, the algorithm starts by providing a random population of cluster centres.

In step 2, a calculation using the formula for u_{ij} (Equation 5.2) in order to fill the array U of the membership function for all data objects is performed.

A repeated generation of a new population of solutions is performed **in step 3**. This is achieved by selecting the best sites of the former population **in step 3.1** and sending more bees to these sites. **In step 3.2,** new values are assigned to the array U using the formula for u_{ij} for each bee.

In step 3.3, a selection of the fittest bees is chosen from each site. Assigning the remaining bees to search randomly and fill the array U is performed **in step 3.4**.

1. Initialise the solution population.
2. Fill the array *U* and evaluate the fitness of the population using Equation (5.1).
3. While (stopping criterion is not met)
    3.1. Select sites for neighbourhood search.
    3.2. Recruit bees for selected sites (more bees for the best e sites), fill the array *U* and evaluate the fitness.
    3.3. Select the fittest bee from each site.
    3.4. Assign remaining bees to search randomly then fill array *U* and evaluate the fitnesses.
4. End While.

Figure 5.1 – Basic steps of the proposed fuzzy clustering algorithm

5.4 Experiments

This section presents the results for the Bees-based fuzzy clustering algorithm compared to the FCM and GA-based fuzzy clustering algorithms.

Table 5.1 shows the values of the initial parameters for each algorithm used in this test, where the value of the fuzziness degree '*m*' was set to 2 in all algorithms. The algorithms were applied to the following real datasets: Iris, vowel, Crude Oil, Control Charts and Wood Defects. The main characteristics of these datasets are summarised in Appendix A.

The clustering criterion *J* (Equation 5.1) was used to evaluate the performance of the algorithms. The smaller the value of this metric, the better are the fuzzy clustering results. The algorithms were executed many times and the average, minimum and maximum values of *J* were collected, where '*k*' is the number of clusters. The results obtained from five runs for each algorithm are listed in Figures 5.3a, b, c, d, and e and summarised in Table 5.2.

Algorithm	Initial Parameters	Value
FCM	Maximum number of iterations	1000
GA	Crossover probability, μ	0.8
	Mutation probability, μ_m	0.001
	Population size, P	100
Bees Algorithm	Number of scout bees, *n*	21
	Number of sites selected for neighbourhood search, *m*	8
	Number of best "elite" sites out of m selected sites, *e*	2
	Number of bees recruited for best *e* sites, *nep*	5
	Number of bees recruited for the other (*m-e*) selected sites, *nsp*	2
	Number of iterations, *R*	300

Table 5.1 – Initial parameters used in the fuzzy clustering experiments

Run	C-means	GA	Bees
1	63.511	69.238	60.589
2	61.223	66.175	60.587
3	65.368	66.143	60.588
4	65.810	63.347	60.585
5	61.918	63.901	60.588

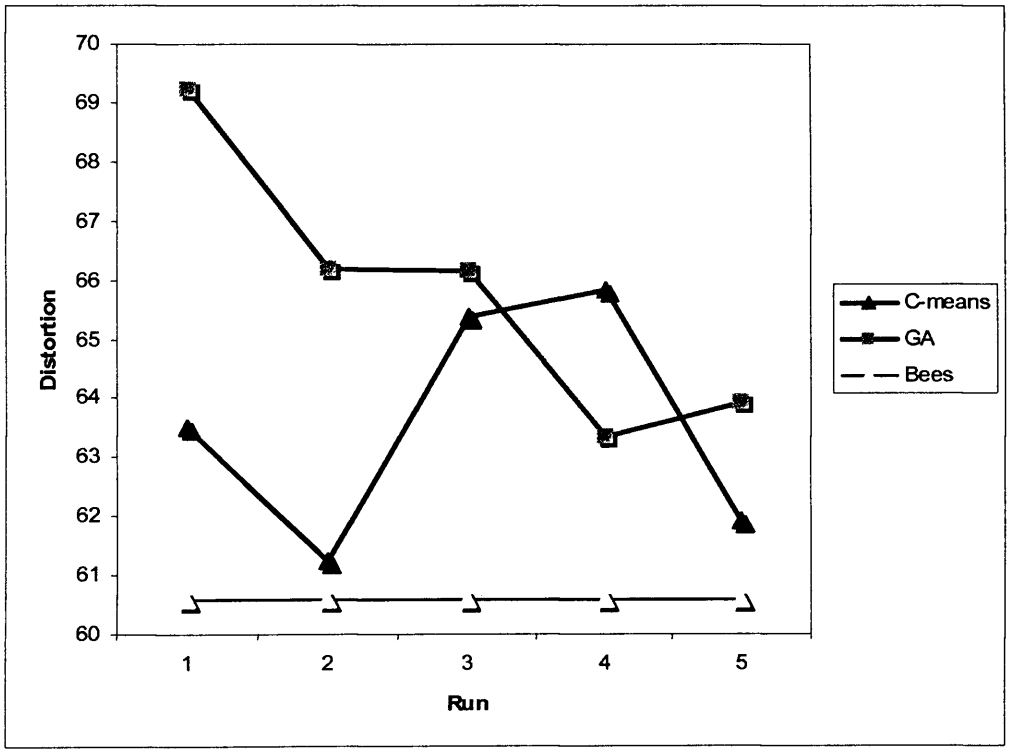


Figure 5.3a – Numerical and graphical representation of J obtained by the C-means, the GA and the Bees-based Algorithm for five runs for Iris, when K=3

Run	C-means	GA	Bees
1	18233302	17909231	17137322
2	19102759	17871130	17160744
3	18037200	17504889	17142746
4	18440670	17325517	17146654
5	18957710	17211010	17134862

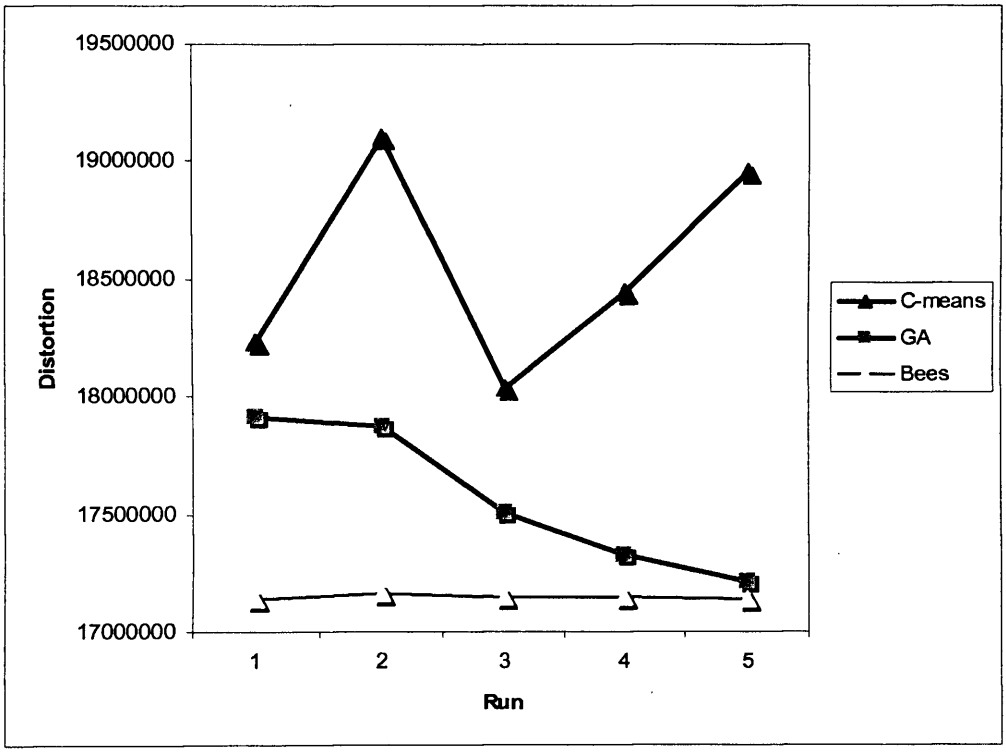


Figure 5.3b – Numerical and graphical representation of J obtained by the C-means, the GA and the Bees-based Algorithm for five runs for Vowel, when K=6

Run	C-means	GA	Bees
1	1560.6	1286.0	1236.2
2	1420.1	1238.3	1237.6
3	1242.3	1239.2	1235.8
4	1390.2	1266.0	1236.3
5	1293.7	1279.4	1235.7

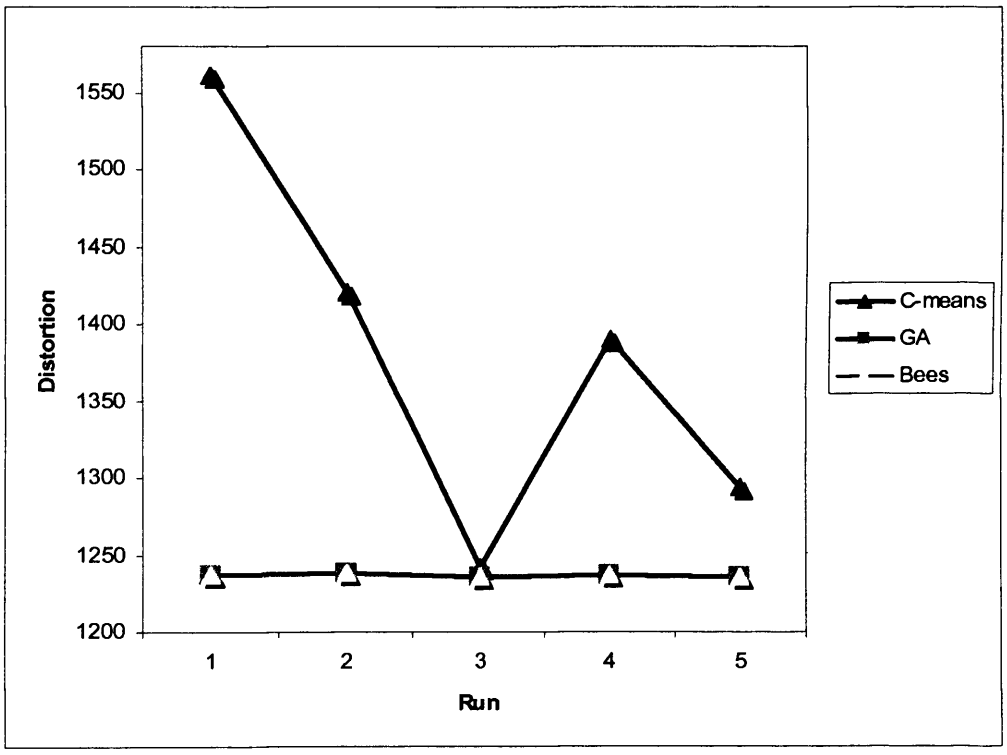


Figure 5.3c – Numerical and graphical representation of J obtained by the C-means, the GA and the Bees-based Algorithm for five runs for Crude Oil, when K=3

Run	C-means	GA	Bees
1	530.99	803.48	549.74
2	530.99	819.29	546.88
3	530.99	791.23	547.36
4	530.99	801.47	552.15
5	530.99	794.23	549.96

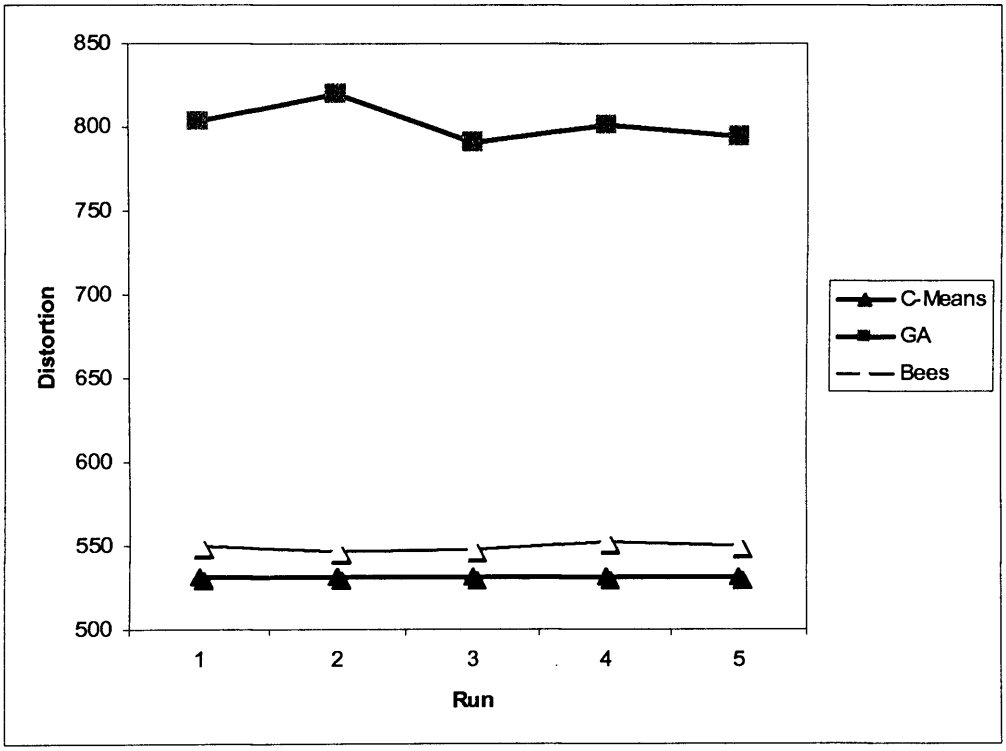


Figure 5.3d – Numerical and graphical representation of J obtained by the C-means, the GA and the Bees-based Algorithm for five runs for Control Charts, when K=6

Run	C-means	GA	Bees
1	66225010	174784	173523
2	62350148	157508	165236
3	63100483	162353	162999
4	59295094	166999	165946
5	60257719	166312	153866

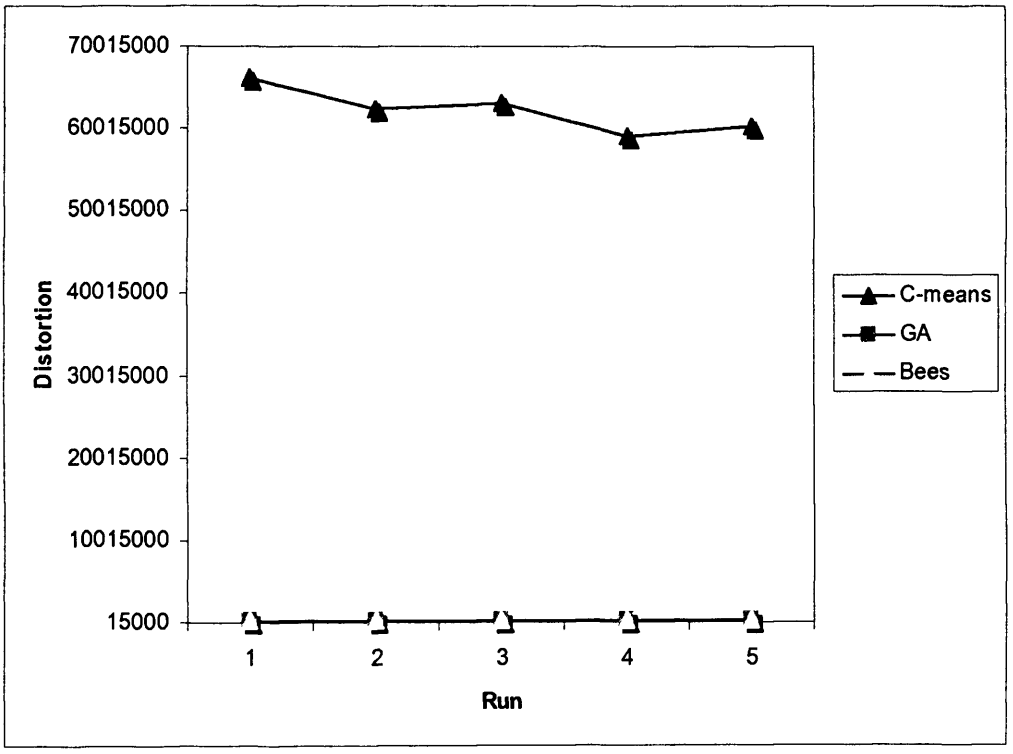


Figure 5.3e – Numerical and graphical representation of J obtained by the C-means, the GA and the Bees-based Algorithm for five runs for Wood Defects, when K=13

Dataset	Algorithm	Mean Value of *J*	Min. Value for *J*	Max. Value for *J*
Iris	C-means	63.566	61.223	65.810
	GA	65.761	63.347	69.238
	Bees Algorithm	60.587	60.585	60.589
Vowel	C-means	18554328	18037200	19102759
	GA	17564355	17211010	17909231
	Bees Algorithm	17144466	17134862	17160744
Crude Oil	C-means	1381.4	1242.3	1560.6
	GA	1261.8	1238.3	1286.0
	Bees Algorithm	1235.0	1235.7	1237.6
Control Charts	C-means	530.99	530.99	530.99
	GA	801.94	791.23	819.29
	Bees Algorithm	549.22	546.88	552.15
Wood Defects	C-means	62225690	59195094	66225010
	GA	165591	157508	174784
	Bees Algorithm	164314	153866	173523

Table 5.2 - Comparative values of *J* obtained by the C-means, the GA and the Bees-based algorithms for five real datasets

5.5 The Algorithm Performance

The results in Table 5.2 show that the GA and Bees-based algorithms outperform the traditional Fuzzy C-Means algorithm in most cases. The new proposed algorithm had a mean value for *J* which was less than the minimum for the other two algorithms for the Iris, Vowel and Crude Oil datasets. For the Control Charts the new Bees algorithm produced a mean value of *J* which was less than the minimum for GA, but was outperformed by the C-means algorithm. For the Wood Defects dataset, the mean value of *J* obtained by the Bees algorithm was less than mean values obtained by either of the other two algorithms. With the exception of the C-means algorithm with the Control Charts dataset the new Bees-based algorithm clearly outperformed the other two algorithms in the tests conducted here.

Combining the Bees Algorithm with the FCM algorithm improved the fuzzy clustering results compared to the traditional FCM algorithm in most cases. It is demonstrated that the new Bees-based algorithm produces better results than those of the GA combined with the FCM algorithm.

5.6 Summary

This chapter has presented a new fuzzy clustering method based on the Bees Algorithm to overcome the problem of trapping into the local optima in the fuzzy C-Means algorithm.

The proposed method does not become trapped at locally optimal solutions and it improved the fuzzy clustering results compared to the traditional C-means algorithm in most cases. It also produces better results than those of the GA combined with the C-Means.

CHAPTER 6

Recursive Bees Algorithm

6.1 Preliminaries

The proposed Bees-based algorithms for crisp and fuzzy clustering in Chapters 4 and 5 gave very good results. However, they retain a number of disadvantages.

One of these deficiencies is the full randomness of the local search in the selected search areas. This drawback can be overcome by improving the local search procedure of the Bees Algorithm.

In this chapter, a new version of the Bees Algorithm, named the Recursive Bees Algorithm (R-Bees), is proposed to reduce the randomness of the local search. The algorithm is applied to both crisp and fuzzy clustering.

6.2 Motivation

As mentioned above, using the Bees Algorithm for data clustering suffers from the weakness of the full randomness of the local search in the selected sites. This makes the algorithm take long time in finding the final solution. To

improve the performance of the algorithm, the recursion concept is applied to the local search procedure.

Solving a problem using recursion means that the solution depends on solutions to smaller instances of the same problem (Graham et al. 1994). In optimisation, the main problem is to find the optimum value over the whole search space. To do that using the recursion concept, the solution of the whole problem depends on finding solutions to local optima within smaller search spaces. The idea of recursion should reduce the randomness of the local search of the Bees Algorithm and its application to the fuzzy and crisp clustering.

6.3 The Proposed Algorithm

The proposed algorithm exploits the powerful search technique used by bees to improve the local search of the standard Bees Algorithm which was used for crisp and fuzzy clustering in chapters 4 and 5 respectively. More specifically, the proposed algorithm repeats the same steps of the Bees Algorithm to find the optimum solution within the local search area in the same way as the search within the whole search space. This reduces the randomness in the local search and makes it guided by the behaviour of the bees.

The pseudo code of the main steps of the proposed algorithm starts as follows (see Figure 6.1):

Step 1: Initialise the solution population randomly of '*n*' scout bees.

Step 2: Evaluate the fitness of each bee in the population and rank the fitnesses. Then bees that have the highest fitnesses are designated as "selected bees" and sites visited by them are selected for neighbourhood search.

Step 3: A recruitment loop for the bees around the selected sites, assigning more bees to search near to the best '*e*' sites, is performed until a pre-specified stopping criterion is met.

This loop starts in **step 3.1** by selecting the sites visited by the bees of the highest fitnesses for neighbourhood search. These sites are classified into elite and non-elite sites according to their fitness.

In **step 3.2**, the algorithm assigns more bees to search near the best elite sites, '*e*'. The recruitment process for these bees is performed according to the same principles of the recruitment process of the basic Bees Algorithm. The same steps (steps **1** to **4**) are applied recursively on the elite sites using the new neighbourhood search area boundaries and the specified number of bees which were already dedicated for neighbourhood search (ngh) around elite sites.

The same procedure also runs around the non-elite sites in **step 3.3** but using the set of bees which are dedicated to run the local search for non-elite sites.

1. Initialise the solution population.
2. Evaluate the fitness of the population.
3. While (the stopping criterion is not met)
   3.1. Select sites for neighbourhood search.
   3.2. For each elite site (*i*)
      3.2.1. Apply steps 1 to 4 to search around the i^{th} site using (*e*) bees and new *ngh* area as search space
   3.3. For each non-elite site (*j*)
      3.3.1. Apply steps 1 to 4 to search around the j^{th} site using (*m-e*) bees and new *ngh* area as search space
   3.4. Select the fittest bee from each site.
   3.5. Assign remaining bees to search randomly and evaluate their fitnesses.
4. End While.

Figure 6.1 – Basic steps of the Recursive Bees Algorithm

In **step 3.4**, for each site, only the position of the bee with the highest fitness is selected to recruit bees around it in the next bees' population. In **step 3.5**, the remaining bees in the population are assigned randomly in the search space scouting for new solutions.

The recursive call which happens in steps 3.2 and 3.3 ends when the algorithm reaches the allowed number of running in depth.

6.4 Experiments

To demonstrate the reduction in the randomness, the proposed recursive algorithm was used for crisp and fuzzy clustering. Each bee represent a combination of centres and the fitness computation processes is performed for each bee by assigning each data point in the dataset to the nearest centre and computing the total distortion (E in Equation 2.1 in chapter 2 for crisp clustering and J in Equation 5.1 in Chapter 5 for fuzzy clustering).

The algorithm was applied to five real datasets (Iris, Vowel, Crude Oil, Control Charts and Wood Defects - see Appendix A for details). The results were compared to those of the Bees-based clustering algorithms for crisp and fuzzy clustering.

The algorithms were run five times. The E and J criteria were used to evaluate their relative fitness. The permitted recursive depth calls in these experiments were 2 only.

Table 6.1 summarises the initial parameters used for each algorithm. Figures 6.2a, b, c, d and e show the values of E (for crisp clustering) and J (for fuzzy clustering) obtained by the Bees and the recursive Bees Algorithms for five different runs for the Iris, Vowel, Crude Oil, Control Charts and Wood Defects datasets respectively where 'k' is the number of clusters. Tables 6.2 and 6.3 summarise these results for both crisp and fuzzy clustering.

The CPU time of both algorithms under crisp and fuzzy conditions for the five datasets (Iris, Vowel, Crude Oil, Control Charts and Wood Defects) was measured. The results presented in Figures 6.3a, b, c, d, and e show the time required by the Bees and the recursive Bees Algorithms. Tables 6.4 and 6.5 summarise the times taken for crisp and fuzzy clustering respectively.

All the tests were conducted on a Pentium 4 2.40GHz machine with 1022MB RAM. Microsoft Visual Studio version 6 C compiler on Windows XP was used for all tests.

Algorithm	Initial Parameters	Value
The Bees Algorithm	Number of scout bees, *n*	21
	Number of sites selected for neighbourhood search, *m*	8
	Number of best "elite" sites out of *m* selected sites, *e*	2
	Number of bees recruited for best *e* sites, *nep*	5
	Number of bees recruited for the other (*ne=m-e*) selected sites, *nsp*	2
	Number of iterations, *R*	300
The Recursive Bees	Number of scout bees, *n*	21
	Number of sites selected for neighbourhood search, *m*	8
	Number of best "elite" sites out of *m* selected sites, *e*	2
	Number of bees recruited for best *e* sites, *nep*	n/(2*e)
	Number of bees recruited for the other (*m-e*) selected sites, *nsp*	n/(3*ne)
	Number of iterations, *R*	10

Table 6.1 – Parameters used in the clustering experiments

Run	Crisp *(E)*		Fuzzy *(J)*	
	Bees	R-Bees	Bees	R-Bees
1	96.728	97.190	60.589	60.695
2	96.787	97.218	60.587	60.695
3	96.787	97.190	60.588	60.695
4	96.770	97.190	60.585	60.695
5	96.749	97.190	60.588	60.695

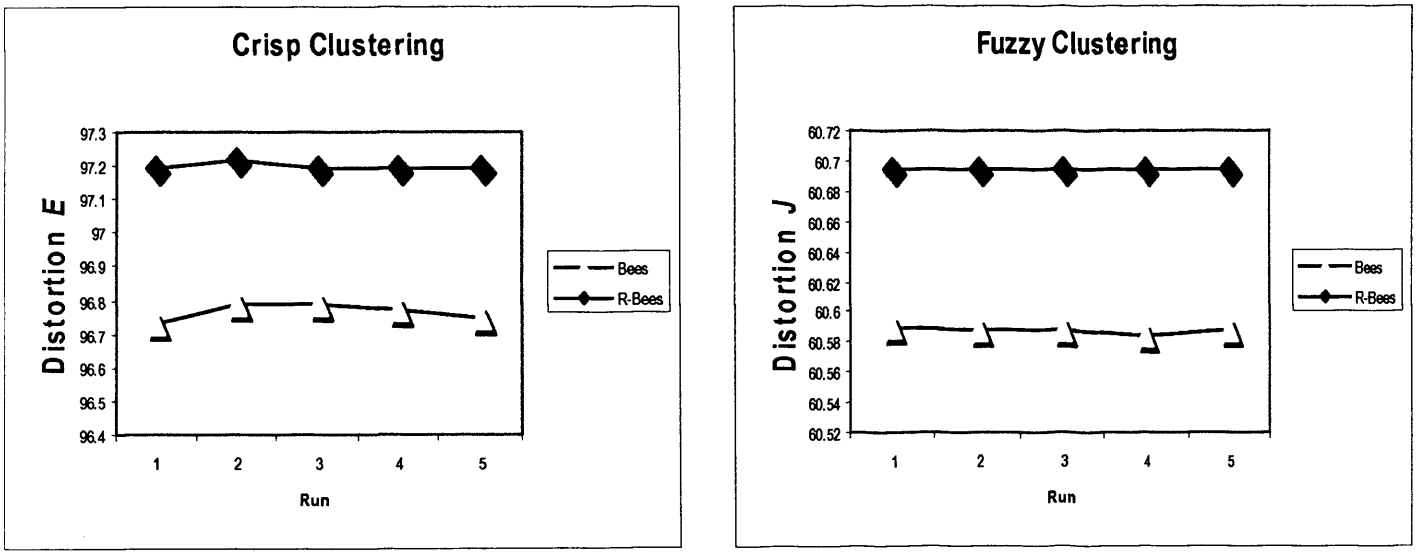
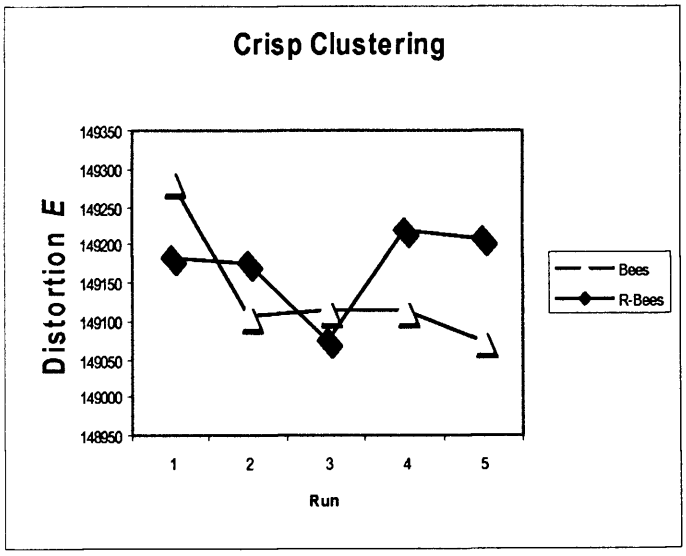


Figure 6.2a – Numerical and graphical representation of *E*, *J* obtained by the Bees-based Algorithm, and the R-Bees for five runs for Iris, when $K = 3$

Run	Crisp *(E)*		Fuzzy *(J)*	
	Bees	R-Bees	Bees	R-Bees
1	149290	149181	17137322	17138864
2	149110	149174	17160744	17149836
3	149110	149073	17142746	17138624
4	149110	149217	17146654	17144301
5	149070	149208	17134862	17134727



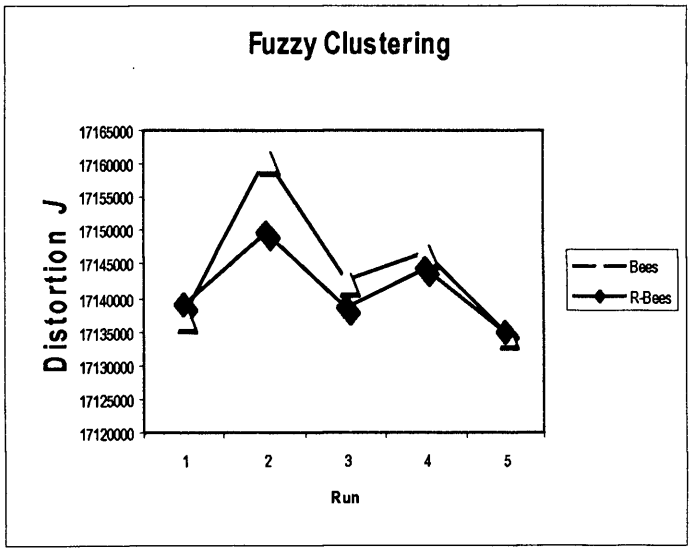
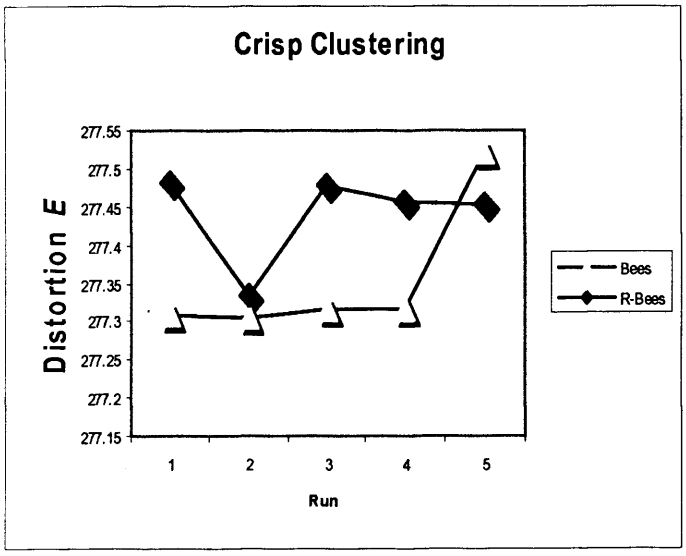


Figure 6.2b – Numerical and graphical representation of *E*, *J* obtained by the Bees-based Algorithm and the R-Bees for five runs for Vowel, when *K=6*

Run	Crisp *(E)*		Fuzzy *(J)*	
	Bees	R-Bees	Bees	R-Bees
1	277.31	277.48	1236.2	1234.9
2	277.30	277.33	1237.6	1235.2
3	277.32	277.48	1235.8	1234.9
4	277.31	277.46	1236.3	1235.2
5	277.52	277.45	1235.7	1234.7



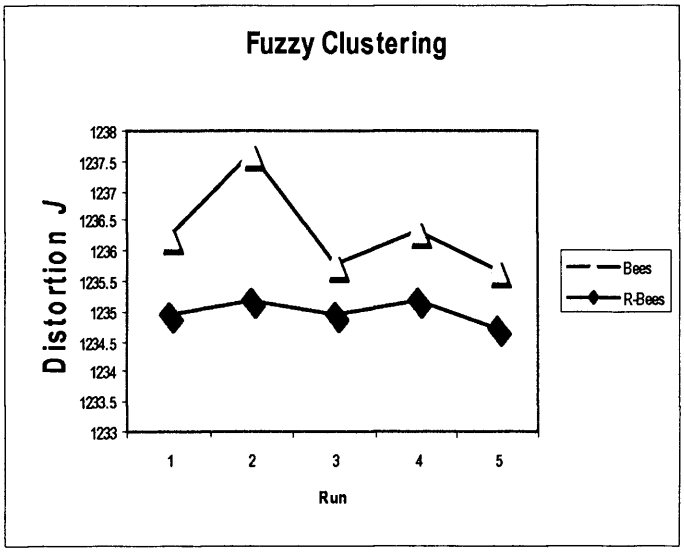


Figure 6.2c – Numerical and graphical representation of *E, J* obtained by the Bees-based Algorithm and the R-Bees for five runs for Crude Oil, when *K=3*

Run	Crisp *(E)*		Fuzzy *(J)*	
	Bees	R-Bees	Bees	R-Bees
1	2318.8	1827.4	549.74	689.84
2	2324.4	1825.6	546.88	693.67
3	2213.6	1825.5	547.36	692.61
4	2250.6	1824.1	552.15	692.81
5	2318.8	1828.7	549.96	692.61

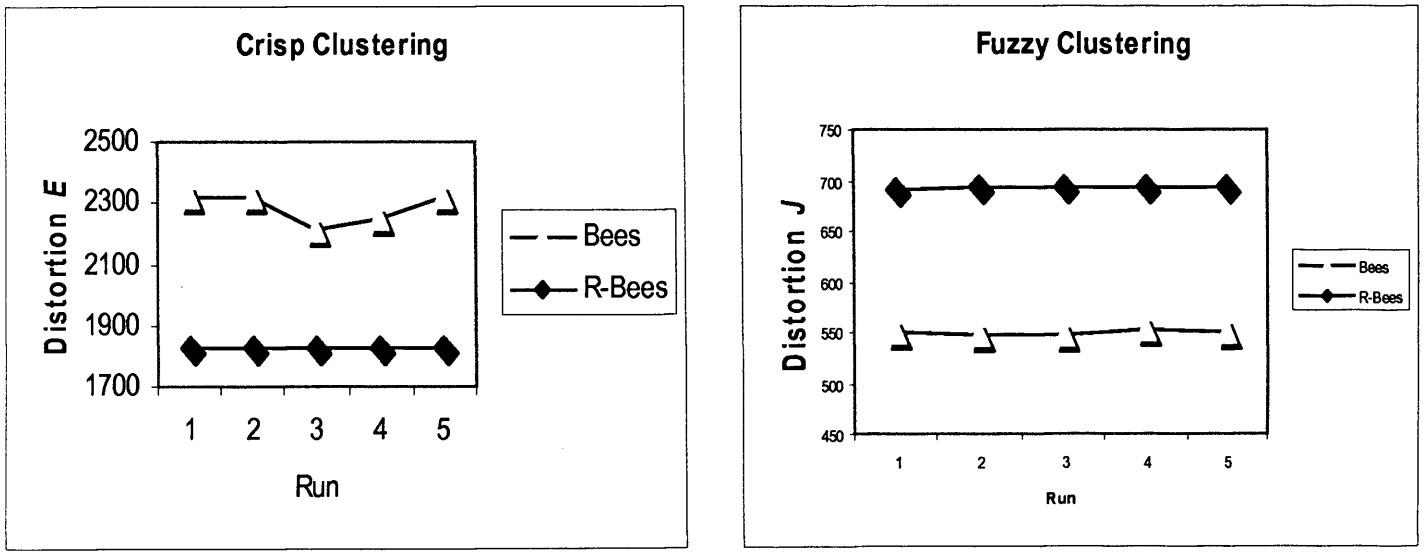
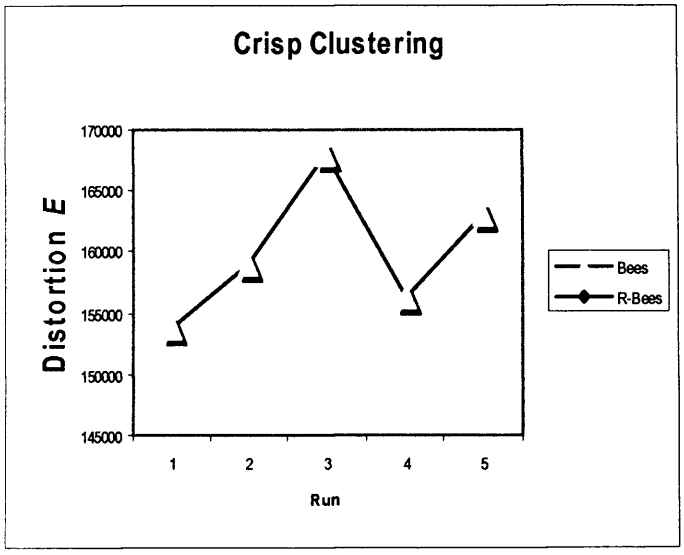


Figure 6.2d – Numerical and graphical representation of *E*, *J* obtained by the Bees-based Algorithm and the R-Bees for five runs for Control Charts, when *K=6*

Run	Crisp *(E)*		Fuzzy *(J)*	
	Bees	R-Bees	Bees	R-Bees
1	153870	158002	173523	169326
2	159010	154773	165236	161900
3	168040	158830	162999	159520
4	156330	151009	165946	160778
5	163120	167466	153866	160114



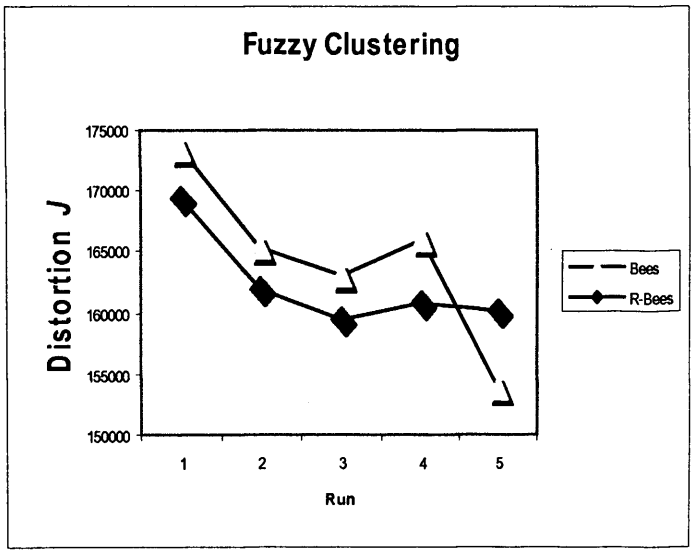


Figure 6.2e – Numerical and graphical representation of *E, J* obtained by the Bees-based Algorithm, the R-Bees for five runs for Wood Defects, when K=13

Dataset	Algorithm	Mean	Min.	Max.
Iris	Bees Algorithm	96.764	96.728	96.787
	Recursive Bees	97.195	97.190	97.218
Vowel	Bees Algorithm	149140	149070	149290
	Recursive Bees	149170	149073	149217
Crude Oil	Bees Algorithm	277.35	277.30	277.52
	Recursive Bees	277.44	277.33	277.48
Control Charts	Bees Algorithm	2285.3	2213.6	2324.4
	Recursive Bees	1826.2	1824.1	1828.7
Wood Defects	Bees Algorithm	160074	153870	168040
	Recursive Bees	158016	151009	167466

Table 6.2 – Results obtained for E for the crisp clustering algorithms

Dataset	Algorithm	Mean	Min.	Max.
Iris	Bees Algorithm	60.587	60.585	60.589
	Recursive Bees	60.695	60.695	60.695
Vowel	Bees Algorithm	17144466	17134862	17160744
	Recursive Bees	17141270	17134727	17149836
Crude Oil	Bees Algorithm	1235.0	1235.7	1237.6
	Recursive Bees	1234.9	1234.7	1235.2
Control Charts	Bees Algorithm	549.22	546.88	552.15
	Recursive Bees	692.31	689.84	693.67
Wood Defects	Bees Algorithm	164314	153866	173523
	Recursive Bees	162328	159520	169326

Table 6.3 – Results obtained for J for the fuzzy clustering algorithms

Run	Crisp		Fuzzy	
	Bees	R-Bees	Bees	R-Bees
1	22	3	192	33
2	22	3	156	31
3	22	3	142	32
4	22	3	177	32
5	22	3	149	34

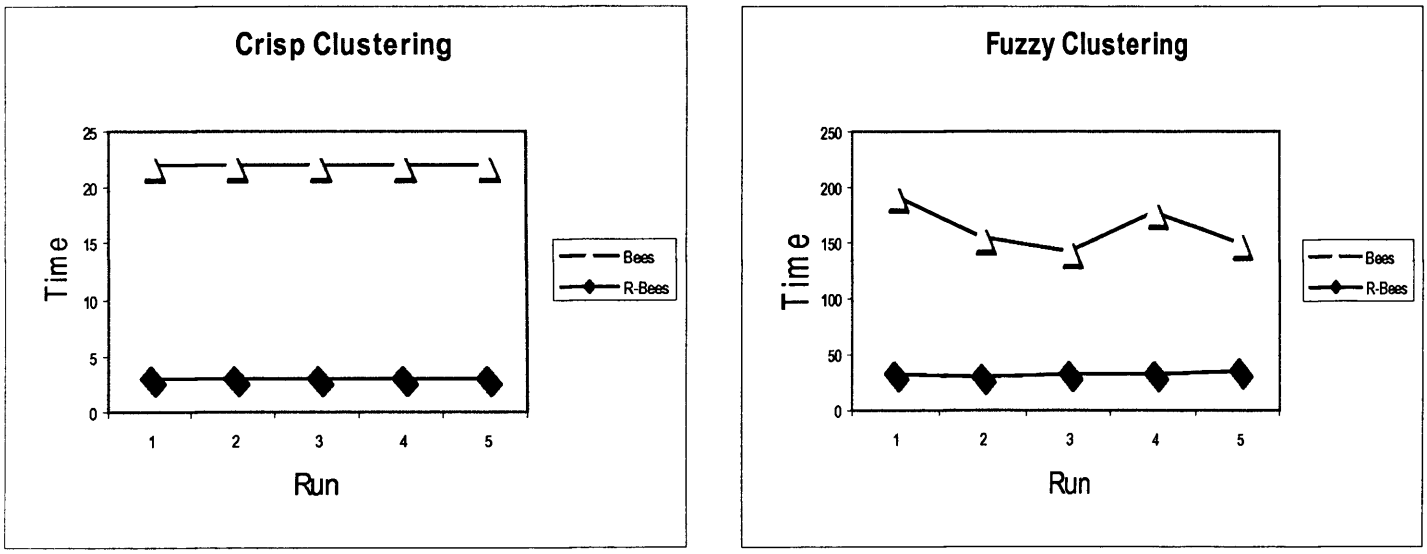


Figure 6.3a – Numerical and graphical representation of the time (seconds) required by the Bees-based Algorithm and the R-Bees for five runs for Iris, when *K=3*

Run	Crisp		Fuzzy	
	Bees	R-Bees	Bees	R-Bees
1	117	27	1786	517
2	128	27	1980	565
3	124	29	1616	494
4	147	28	1541	521
5	140	30	1811	492

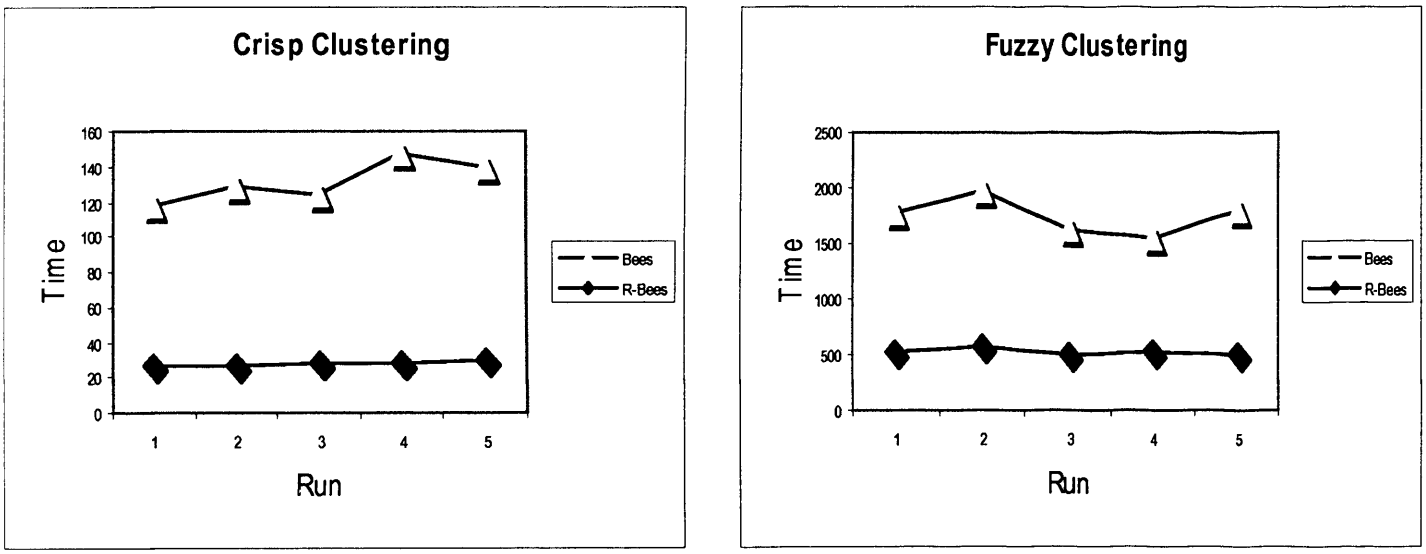
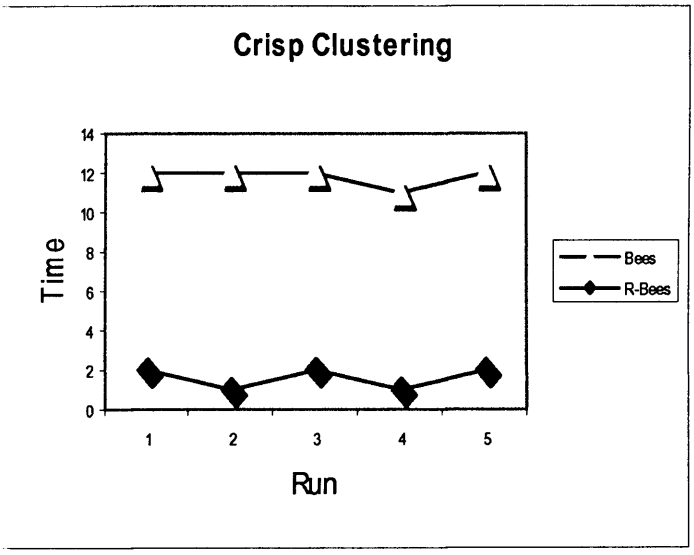


Figure 6.3b – Numerical and graphical representation of the time (seconds) required by the Bees-based Algorithm and the R-Bees for five runs for Vowel, when *K=6*

Run	Crisp		Fuzzy	
	Bees	R-Bees	Bees	R-Bees
1	12	2	41	14
2	12	1	36	14
3	12	2	34	14
4	11	1	34	14
5	12	2	42	14



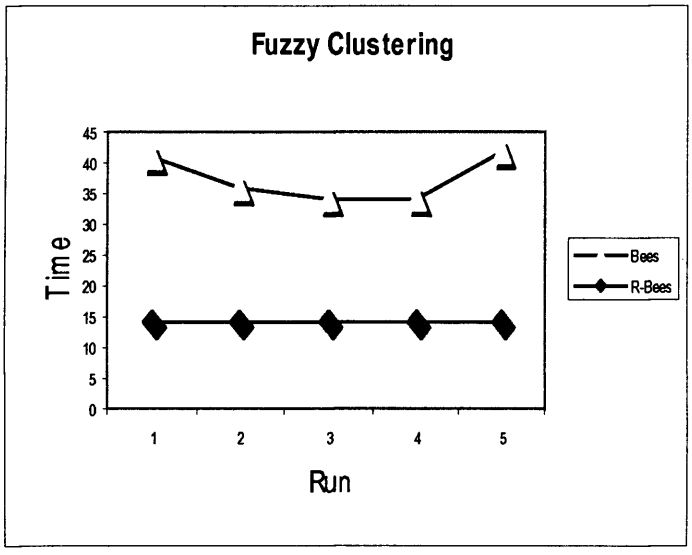


Figure 6.3c – Numerical and graphical representation of the time (seconds) required by the Bees-based Algorithm and the R-Bees for five runs for Crude Oil, when *K=3*

Run	Crisp		Fuzzy	
	Bees	R-Bees	Bees	R-Bees
1	6103	539	39906	12268
2	5902	528	37562	12778
3	6015	523	31263	13554
4	5824	572	33810	14459
5	6005	605	29488	12775

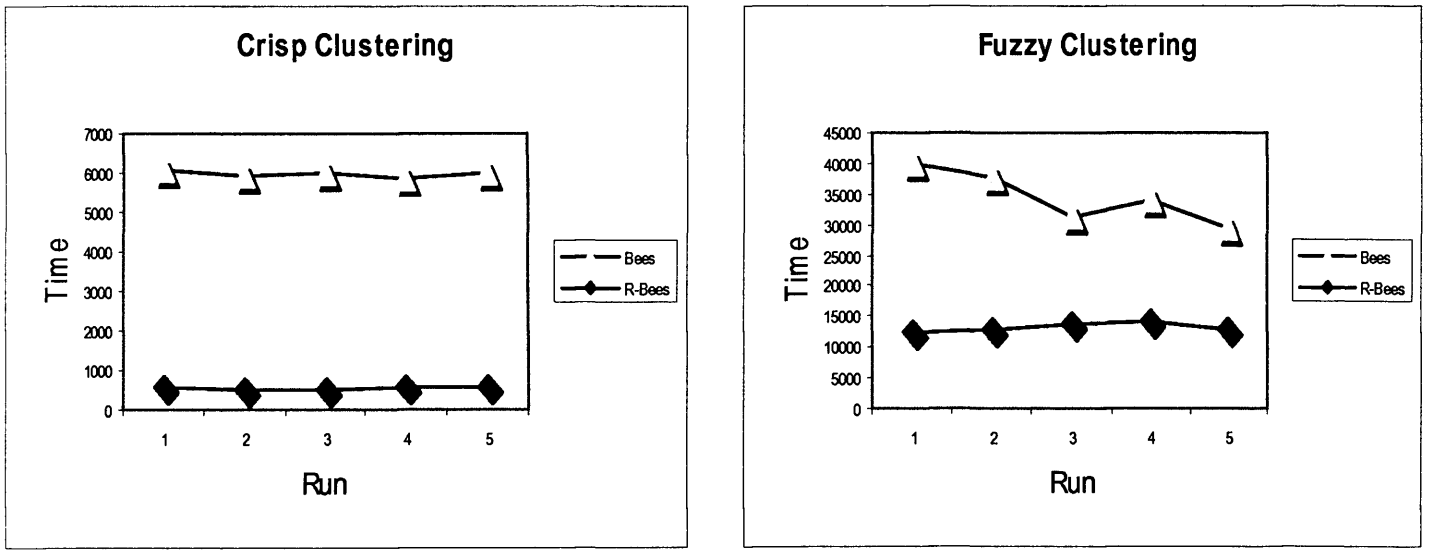
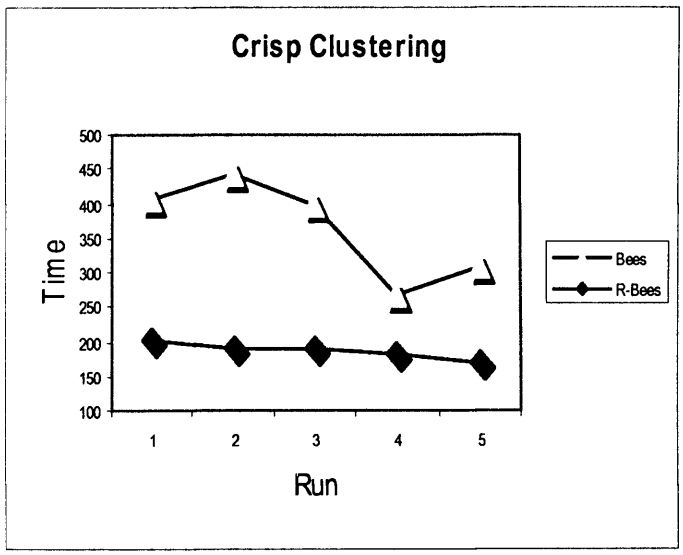


Figure 6.3d – Numerical and graphical representation of the time (seconds) required by the Bees-based Algorithm and the R-Bees for five runs for Control Charts, when K=6

Run	Crisp		Fuzzy	
	Bees	R-Bees	Bees	R-Bees
1	405	201	7482	2348
2	443	190	7441	2446
3	399	191	7853	2114
4	267	182	6855	2855
5	309	168	7012	2714



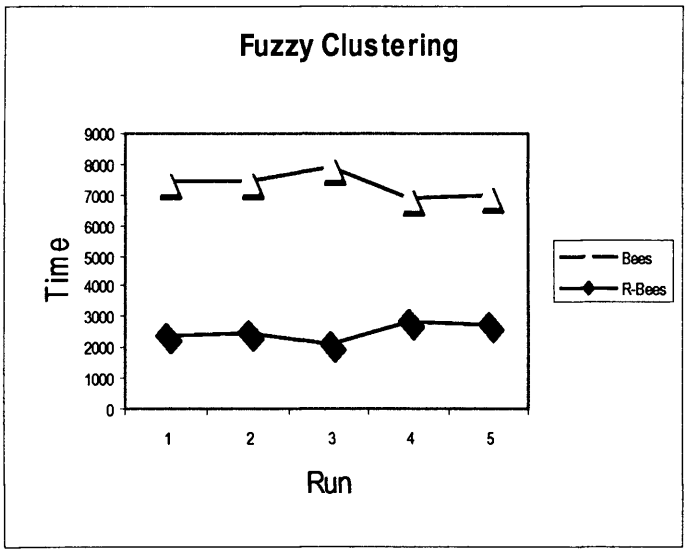


Figure 6.3e – Numerical and graphical representation of the time (seconds) required by the Bees-based Algorithm and the R-Bees for five runs for Wood Defects, when K=13

Dataset	Algorithm	Mean	Min.	Max.
Iris	Bees Algorithm	22	22	22
	Recursive Bees	3	3	3
Vowel	Bees Algorithm	131.2	117	147
	Recursive Bees	28.2	27	30
Crude Oil	Bees Algorithm	11.8	11	12
	Recursive Bees	1.6	1	2
Control Charts	Bees Algorithm	5970	5824	6103
	Recursive Bees	553.4	523	605
Wood Defects	Bees Algorithm	364.6	267	443
	Recursive Bees	186.4	168	201

Table 6.4 – Time (seconds) taken by each crisp clustering algorithm

Dataset	Algorithm	Mean	Min.	Max.
Iris	Bees Algorithm	163.2	142	192
	Recursive Bees	32.4	31	34
Vowel	Bees Algorithm	1747	1541	1980
	Recursive Bees	517.8	492	565
Crude Oil	Bees Algorithm	37.4	34	42
	Recursive Bees	14	14	14
Control Charts	Bees Algorithm	34406	29488	39906
	Recursive Bees	13167	12268	14459
Wood Defects	Bees Algorithm	7329	6855	7853
	Recursive Bees	2495	2114	2855

Table 6.5 – Time (seconds) taken by each fuzzy clustering algorithm

6.5 The Algorithm Performance

The complexity of each algorithm can be measured by calculating the number of evaluations which is the most costly operation in the algorithms. In the Bees Algorithm this can be calculated as follows:

$$\text{Complexity} = n + R * [(e * nep) + (ne * nsp)]$$

While, the complexity for the proposed recursive algorithm can be computed as follows:

$$\text{Complexity} = n + R * [(e * v1) + (ne * v2)]$$

where *v1* and *v2* are the complexity of the recursive call for the algorithm using a number of bees equal to '*nep*' for *v1* and equal to '*nsp*' for *v2*. This number is reduced from the total number of bees according to pre-specified percentage, in other words:

$$v1 = nep + R * [(e * v11) + (ne * v12)]$$

$$v2 = nsp + R * [(e * v11) + (ne * v12)]$$

where

V11 is the complexity of the second recursive call for search in elite sites.

V12 is the complexity of the second recursive call for search in other sites.

Apparently, the complexity of the proposed recursive algorithm is greater than that of the Bees Algorithm and it is going to be increased exponentially whenever there is a recursive call for the bees' steps for each local search. But, having a deeper view of the complexity, and by analysing the two algorithms, the proposed recursive algorithm (R-Bees) can get almost similar (in some cases better) results compared to those of the Bees Algorithm (See Figures 6.2a, b, c, d and e and Tables 6.2 and 6.3) using a set of parameters which are better than those of the Bees Algorithm and with significantly fewer iterations.

Although that the number of fitness calls used to run the recursive algorithm is greater than that of the Bees Algorithm, the number of iterations used in the Bees Algorithm for the tested datasets is 60 times greater than that of the proposed recursive algorithm. The number of evaluations applied by the basic Bees Algorithm is approximately three times greater than for the recursive algorithm, which means a higher evaluation number in the Bees Algorithm for all datasets.

The experiments (Figures 6.3a, b, c, d and e and tables 6.4 and 6.5) show this result clearly when comparing the time spent by the algorithms. The key feature of the proposed recursive Algorithm is that it can reach the optimum value within the local search area in better way compared to the basic Bees Algorithm.

6.6 Summary

This chapter presented a new recursive algorithm used to improve the local search procedure of the Bees Algorithm. The proposed algorithm has the advantage that it reduces the randomness and spends a shorter computing time. This is due to the ability of the recursive algorithm to perform the local search recursively.

The algorithm was applied to crisp and fuzzy clustering. Experimental results for different datasets have demonstrated that the proposed recursive method produces a better performance compared to the basic Bees-based clustering algorithm in most cases.

CHAPTER 7

Conclusion

This chapter summarises the contributions and conclusions of this research and gives suggestions for future work.

7.1 Contributions

The main contributions of this research are:

1. An improved crisp clustering algorithm based on the Kd-tree structure to overcome the disadvantages of instability of the K-means algorithm and its large number of neighbourhood search operations.
2. New clustering algorithms combining the simplicity of the K-means algorithm and the C-means algorithm with the capability of the Bees Algorithm to avoid local optima in crisp and fuzzy clustering, respectively.

3. Improvement to the standard Bees Algorithm and its application to crisp and fuzzy clustering to enhance the local search procedure and reduce its processing time.

7.2 Conclusions

In this thesis, the feasibility of using the Kd-tree and the Bees Algorithm to improve the data clustering process has been shown. The key conclusions for each topic analysed are:

- The adaptive capability of the Kd-tree-based algorithm gives better results compared to those of the K-means algorithm on its own. The improved Kd-tree structure improves the neighbourhood search and significantly decreases the time needed to allocate data points to clusters.
- The Bees Algorithm has demonstrated its effectiveness in a series of crisp and fuzzy clustering experiments using different data sets. These good results can be explained by the ability of the Bees Algorithm to perform local and global search simultaneously.
- The Recursive Bees Algorithm, which was used to find the optimum solution recursively, improves the local search procedure and gives similar results to the basic Bees Algorithm but with fewer evaluations.

7.3 Future Research Directions

Despite their efficiency, there are aspects of the proposed algorithms which can be improved.

Research is needed to reduce the number of tuneable parameters of the Bees Algorithm without affecting its overall performance. Another improvement to this algorithm can be made by applying it in parallel so that the processing time of the algorithm can be decreased.

In crisp Bees-based clustering, the algorithm can exploit the idea of using the Kd-tree as a data structure to reduce the time for assigning data points to clusters and calculating the distortion in each evaluation in the algorithm.

The recursive method presented in chapter 6 needs further investigation to avoid having a very large number of recursive calls so that the stopping criteria of the algorithm can be estimated automatically. Furthermore, the algorithm can be improved by adding new criteria that can decide according to the size of the dataset whether to apply the recursive call.

The recursive algorithm also has difficulties dealing with data of high dimensions. It might be a good idea, as well, to exploit the capability of the Kd-tree or any other suitable data structure to overcome this drawback.

Having manually to determine the number of clusters is a well-known drawback when using the K-means algorithm and all of the proposed algorithms. This can be avoided by generating different clustering solutions

with different numbers of clusters and exploiting the power of the Bees Algorithm to find the optimum number of clusters automatically.

APPENDIX A

Description of Datasets

Data1: This is a two-dimensional artificial data as described in (Maulik and Bandyopadhyay 2000). It consists of three clusters and has 76 objects.

Data2: This a two-dimensional triangular distribution artificial data (Maulik and Bandyopadhyay 2000). It consists of nine clusters and 900 objects.

Data3: This synthetic ten-dimensional data is generated using a triangular distribution (Maulik and Bandyopadhyay 2000). It consists of two clusters and 1000 objects.

Iris: This dataset is from UCI Machine Learning Repository (Newman et al. 1998). It contains four features represent the sepal length, sepal width, petal length, and the petal width. There are 3 clusters with 50 objects of each of these clusters.

Vowel: It consists of 871 objects of Indian Telegu vowel sounds (Maulik and Bandyopadhyay 2000). It has three features F1, F2, and F3, which represent

first, second and third vowel formant frequencies respectively and it has 6 clusters.

Crude Oil: This data contains 3 clusters of 56 objects with 5 features (Johnson and Wichern 2001).

Control Charts: It contains 6 clusters (normal, cyclic, increasing, decreasing, upward and downward pattern) of 1500 objects with 60 features. This dataset has been described in (Pham and Oztemel 1992).

Wood Defects: This wood data contains 13 clusters of 232 objects with 17 features (Pham et al. 2006).

Dataset Name	#Features	#Objects	#Classes
Data1	2	76	3
Data2	2	900	9
Data3	10	1000	2
Vowel	3	871	6
Iris	4	150	3
Crude Oil	5	56	3
Control Charts	60	1500	6
Wood Defects	17	232	13

Table A.1 – Datasets used in the experiments

REFERENCES

Abonyi, J. and Feil, B. 2007. *Cluster Analysis for Data Mining and System Identification*. 1 ed. Berlin: Birkhäuser Basel, p. 303.

Agrawal, R. , Gehrke, J. , Gunopulos, D. and Raghavan, P. 2005. Automatic Subspace Clustering of High Dimensional Data. *Data Mining and Knowledge Discovery* 11(1), pp. 5-33.

Al-Daoud, M. B. and Roberts, S. A. 1996. New Methods for the Initialisation of Clusters. *Pattern Recognition Letters* 17(5), pp. 451-455.

Al-Daoud, M. B. , Venkateswarlu, N. B. and Roberts, S. A. 1995. *Fast K-means Clustering Algorithms-Technical Report*. Leeds, UK: School of Computer Studies, University of Leeds.

Al-Zoubi, M. D. , Hudaib, A. and Al-Shboul, B. 2007. A Fast Fuzzy Clustering Algorithm. In: *6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*. Corfu Island, Greece. pp. 28-32.

Alata, M. , Molhim, M. and Ramini, A. 2008. Optimising of Fuzzy C-Means Clustering Algorithm Using GA. *World Academy of Science, Engineering and Technology* 39, pp. 224-229.

Aloise, D. 2009. *Exact Algorithms for Minimum Sum-of-Squares Clustering.* PhD Thesis, Université de Montréal.

Anderberg, M. R. 1973. *Cluster Analysis for Applications.* New York: Academic Press Inc., New York, p. 359

Ankerst, M. , Breunig, M. M. , Kriegel, H.-P. and Sander, J. 1999. OPTICS: Ordering Points to Identify the Clustering Structure. In: *International Conference of the Special Interest Group on Management of Data (ACM-SIGMOD-99).* Philadelphia, USA: ACM. pp. 49-60.

Ardilly, P. and Tillé, Y. 2006. *Sampling Methods: Exercises and Solutions.* 1 ed. New York: Springer, p. 382

Arnold, L. and Wollenberg, V. D. 2006. Redundancy Analysis an Alternative for Canonical Correlation Analysis *Psychometrika* 42(2), pp. 207-219.

Asuncion, A. and Newman, D. J. 2007. *UCI Machine Learning Repository.* University of California, School of Information and Computer Science. Available at: <URL: http://www.ics.uci.edu/~mlearn/MLRepository.html> [Accessed: 15 January 2008].

Au, W.-H. , Chan, K. C. C. , Wong, A. K. C. and Wang, Y. 2005. Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2(2), pp. 83-101.

Babu, G. P. and Murty, M. N. 1993. A Near-Optimal Initial Seed Value Selection in K-Means Algorithm Using a Genetic Algorithm. *Pattern Recognition Letters* 14(10), pp. 763-769

Baesens, B. , Egmont-Petersen, M. , Castelo, R. and Vanthienen, J. 2002. Learning Bayesian Network Classifiers for Credit Scoring Using Markov Chain Monte Carlo Search. In: *16th International Conference on Pattern Recognition (ICPR 02)*. Quebec City, QC, Canada: IEEE. pp. 49-52.

Ball, G. H. and Hall, D. J. 1965. *ISODATA, A Novel Method of Data Analysis and Classification-Technical Report*. Stanford, California, USA: Stanford Research Institute.

Bandyopadhyay, S. and Maulik, U. 2002. Genetic Clustering for Automatic Evolution of Clusters and Application to Image Classification. *Pattern Recognition* 35(6), pp. 1197-1208.

Bentley, J. L. 1980. Multidimensional Divide and Conquer. *Communications of the ACM* 23(4), pp. 214-229

Berry, M. J. A. and Linoff, G. 2004. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. 2 ed. New York: Wiley, p. 643

Berthouex, P. M. and Brown, L. C. 2002. *Statistics for Environmental Engineers*. 2 ed. Boca Raton, Florida: CRC Press.

Bezdek, J. C. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. 1 ed. Kluwer Academic Publishers, p. 272

Bezdek, J. C. , Keller, J. , Krisnapuram, R. and Pal, N. R. 2005. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. 1 ed. New York, USA: Springer, p. 776.

Biem, A. 2006. Minimum Classification Error Training for Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(7), pp. 1041-1051.

Bobko, P. 2001. *Correlation and Regression: Applications for Industrial Organizational Psychology and Management*. 2 ed. Thousand Oaks, California, USA: Sage Publications, Inc, p. 304.

Bottou, L. and Bengio, Y. 1995. Convergence Properties of the K-means Algorithm. *Advances in Neural Information Processing Systems* 7, pp. 585-592.

Bradley, P. S. and Fayyad, U. M. 1998. Refining Initial Points for K-means Clustering. In: *15th International Conference on Machine Learning*. Madison, Wisconsin, USA: Morgan Kaufmann Publishers Inc. pp. 91-99.

Bradley, P. S. , Fayyad, U. M. and Reina, C. 1998a. Scaling Clustering Algorithms to Large Databases. In: *4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*. New York, USA: AAAI Press. pp. 9-15.

Bradley, P. S. , Fayyad, U. M. and Reina, C. A. 1998b. *Scaling EM (Expectation Maximization) Clustering to Large Databases-Technical Report*. Redmond, Washington, USA: Microsoft Research.

Branscum, A. J. , Johnson, W. O. and Thurmond, M. C. 2007. Bayesian Beta Regression: Applications to Household Expenditure Data and Genetic Distance Between Foot-and-Mouth Disease Viruses. *Australian & New Zealand Journal of Statistics* 49(3), pp. 287-301.

Camazine, S. , Deneubourg, J.-L. , Franks, N. R. , Sneyd, J. , Theraulaz, G. and Bonabeau, E. 2003. *Self-Organization in Biological Systems*. Princeton, New Jersey, USA: Princeton University Press, p. 538.

Carpenter, G. A. and Grossberg, S. 1991. *Pattern Recognition by Self-Organizing Neural Networks*. Cambridge, Massachusetts, USA: MIT Press, p. 691.

Carpenter, G. A. , Grossberg, S. and Rosen, D. B. 1991. Fuzzy Art: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks* 4(6), pp. 759-771.

Chapman, P. , Clinton, J. , Kerber, R. , Khabaza, T. , Reinartz, T. , Shearer, C. and Wirth, R. 2000. *CRoss Industry Standard Process for Data Mining (CRISP-DM)*. USA: CRISP-DM Consortium, p. 78.

Chen, T.-S. , Tsai, T.-H. , Chen, Y.-T. , Lin, C.-C. , Chen, R.-C. , Li, S.-Y. and Chenl, H.-Y. 2005. A Combined K-means and Hierarchical Clustering Method for Improving the Clustering Efficiency of Microarray. In: *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. Hong Kong, China: IEEE. pp. 405-408.

Cheung, Y.-M. 2003. K*-Means: A New Generalized K-Means Clustering Algorithm. *Pattern Recognition Letters* 24, pp. 2883-2893.

Davis, L. 1991. *Handbook of Genetic Algorithms*. New York, USA: Van Nostrand Reinhold, p. 385.

Dellaert, F. 2002. *The Expectation Maximization Algorithm-Technical Report.* College of Computing, Georgia Institute of Technology.

Diady, E. and Simon, J. C. 1976. Clustering Analysis. In: Fu, K.S. ed. *Digital Pattern Recognition.* Heidelberg: Springer. pp. 47-94.

Dong, G. and Li, J. 1999. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In: *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* San Diego, California, USA. pp. 43-52.

Donga, H. , Donga, Y. , Zhoua, C. , Yina, G. and Houa, W. 2009. A Fuzzy Clustering Algorithm Based on Evolutionary Programming. *Expert Systems with Applications* 36(9), pp. 11792-11800

Dunn, J. C. 1973. A Fuzzy Relative of the Isodata Process and its Use in Detecting Compact Well-Separated Clusters. *Cybernetics* 3(3), pp. 32-57.

Epter, S. and M. Krishnamoorthy, M., and Zaki, M. 1999. *Clusterability Detection and Initial Seed Selection in Large Data Sets-Technical Report.* Troy, New York, USA: Department of Computer Science, Rensselaer Polytechnic Institute.

Ester, M. , Kriegel, H.-P. , Sander, J. and Xu, X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In:

2nd International Conference on Knowledge Discovery and Data Mining. Portland, Oregon, USA: AAAI Press. pp. 226-231.

Ester, M. , Kriegel, H. P. and Xu, X. W. 1995. Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. In: *4th International Symposium on Large Spatial Databases*. Portland, Maine, USA. pp. 67-82.

Estivill-Castro, V. and Yang, J. 2004. Fast and Robust General Purpose Clustering Algorithms. *Data Mining and Knowledge Discovery* 8(2), pp. 127-150.

Farnstrom, F. , Lewis, J. and Elkan, C. 2000. Scalability for Clustering Algorithms Revisited. *ACM SIGKDD Explorations Newsletter* 2(1), pp. 51-57.

Fay, R. E. 1993. Valid Inferences from Imputed Survey Data. In: *Survey Research Methods Section*. Washington, USA: American Statistical Association. pp. 41-48.

Fayyad, U. M. , Djorgovski, S. G. and Weir, N. 1996a. From Digitized Images to Online Catalogs Data Mining a Sky Survey. *AI Magazine* 12(2), pp. 53-66.

Fayyad, U. M. , Piatetsky-Shapiro, G. and Smyth, P. 1996b. From Data Mining to Knowledge Discovery in Databases. *AI Magazine* 17(3), pp. 37-54.

Feldman, R. and Sanger, J. 2006. *The Text Mining Handbook (Advanced Approaches in Analayzing Unstructured Data)*. New York, USA: Cambridge University Press, p. 422

Fisher, D. , Ling, X. , Carnes, J. R. , Reich, Y. , Fenves, J. , Chen, J. , Shiavi, R. , Biswas, G. and Weinberg, J. 1993. Applying AI Clustering to Engineering Tasks. *IEEE Expert* 8(6), pp. 51-60.

Fodor, I. K. 2002. *A Survey of Dimension Reduction Techniques-Technical Report*. Livermore, California, USA: Center for Applied Scientific Computing, Lawrence Livermore National Laboratory (LLNL).

Friedman, J. H. , Bentley, J. L. and Finkel, R. A. 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)* 2(3), pp. 209-226.

Frisch, V. and Karl 1976. *Bees: Their Vision, Chemical Senses and Language.* Ithaca, New Jersy, USA: Cornell University Press., p. 157.

Fritzke, B. 1997. The LBG-U Method for Vector Quantization - An Improvement over LBG Inspired from Neural Networks. *Neural Processing Letters* 5(1), pp. 35-45.

Garai, G. and Chaudhuri, B. B. 2003. A Novel Genetic Algorithm for Automatic Clustering. *Pattern Recognition Letters* 25(2), pp. 173-187.

Garavaglia, S. and Sharma, A. 1998. A Smart Guide to Dummy Variables: Four Applications and a Macro. In: *11th Annual Conference NorthEast SAS Users Group, Inc.* Pittsburgh, PA. pp. 750-759.

Grabmeier, J. and Rudolph, A. 2002. Techniques of Cluster Algorithms in Data Mining. *Data Mining and Knowledge Discovery* 6, pp. 303-360.

Graham, R. L. , Knuth, D. E. and Patashnik, O. 1994. *Concrete Mathematics (A Foundation for Computer Science)*. 2 ed. Addison-Wesley Professional, p. 672.

Guo, R.-c. , Ye, S.-s. , Quan, M. and Shi, H.-x. 2009. Modified Fast Fuzzy C-Means Algorithm for Image Segmentation. In: *2nd International Symposium on Electronic Commerce and Security*. Nanchang, China pp. 39-43.

Hall, L. O. , Ozyurt, I. B. and Bezdek, J. C. 1999. Clustering with a Genetically Optimized Approach. *IEEE Transactions on Evolutionary Computation* 3(2), pp. 103-112.

Hamadeh, H. K. and Afshari, C. A. eds. 2004. *Toxicogenomics: Principles And Applications.* Hoboken, New Jersey, USA: Wiley-Liss, p. 361.

Hamming, R. W. 1950. Error Detecting and Error Correcting Codes. *Bell System Technical Journal* XXVI(2), pp. 147-160.

Han, J. and Kamber, M. 2006. *Data Mining: Concepts and Techniques*. 2 ed. San Francisco, California, USA: Morgan Kaufmann, p. 770.

Han, J. , Kamber, M. and Tung, A. K. H. 2001. Spatial Clustering Methods in Data Mining: A Survey. In: Miller, H.J. and Han, J. eds. *Geographic Data Mining And Knowledge Discovery*. 1 ed. New York, USA: Taylor and Francis, pp. 188-217.

Hathaway, R. J. , Bezdek, J. C. and Hu, Y. 2000. Generalized Fuzzy C-Means Clustering Strategies Using Lp Norm Distances. *IEEE Transactions on Fuzzy Systems* 8(5), pp. 576-582.

Hinneburg, A. and Keim, D. A. 1998. An Efficient Approach to Clustering Large Multimedia Databases with Noise. In: *4th International Conference on Knowledge Discovery and Data Mining*. NewYork, USA. pp. 58-65.

Huang, C. M. and Harris, R. W. 1993. A Comparison of Several Codebook Generation Approaches. *IEEE Transactions on Image Processing* 2(1), pp. 108-112.

Huang, Z. 1998. Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery* 2(3), pp. 283-304.

Hussein, N. 2002. *A Fast Greedy K-means Algorithm.* Master Thesis, University of Amsterdam.

Jain, A. K. and Dubes, R. C. 1988. *Algorithms for Clustering Data*. New Jersey, USA: Prentice Hall, Englewood Cliffs, p. 304.

Jain, A. K. , Murty, M. N. and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Computing Surveys* 31(3), pp. 264-323.

Jain, A. K. , Ross, A. and Prabhakar, S. 2004. An Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 14(1), pp. 4-20

Jang, J.-S. R. , Sun, C.-T. and Mizutani, E. 1997. *Neuro-Fuzzy and Soft Computing*. US ed. New Jersey, USA: Prentice Hall, p. 614

Jinlan, T. , Lin, Z. , Suqin, Z. and Lu, L. 2005. Improvement and Parallelism of K-means Clustering Algorithm. *Tsinghua Science & Technology* 10(3), pp. 277-281.

Johnson, R. A. and Wichern, D. W. 2001. *Applied Multivariate Statistical Analysis*. 5 ed. Englewood Clifs, NJ: Prentice Hall, p. 767.

Johnson, S. C. 2006. Hierarchical Clustering Schemes. *Psychometrika* 32(3), pp. 241-254.

Kaiqi, Z. , Zhiping, W. and Ming, H. 2008. An Improved FCM algorithm for Color Image Segmentation. In: *3rd International Conference on Innovative Computing Information and Control*. Dalian, Liaoning China: IEEE. pp. 200-204.

Kanungo, T. , Mount, D. M. , Netanyahu, N. S. , Piatko, C. D. , Silverman, R. and Wu, A. Y. 2002. An Efficient K-Means Clustering Algorithm: Analysis and Implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(7), pp. 881-892.

Kaplan, D. 2002. Structural Equation Modeling: Foundations and Extensions. *Journal of Educational Measurement* 39(2), pp. 183-186.

Katsavounidis, I. , Kuo, J. and Zhang, Z. 1994. A New Initialization Technique for Generalized Lloyd Iteration. *IEEE Signal Processing Letters* 1(10), pp. 144-146.

Kaufman, L. and Rousseeuw, P. J. 2005. *Finding Groups in Data: An Introduction to Cluster Analysis*. 2 ed. Hoboken, New Jersey, USA: Wiley, p. 368

Kersten, P. R. , Lee, J.-S. and Ainsworth, T. L. 2005. Unsupervised Classification of Polarimetric Synthetic Aperture Radar Images Using Fuzzy

Clustering and EM Clustering. *IEEE Transactions on Geoscience and Remote Sensing* 43(3), pp. 519-527.

Khan, S. S. and Ahmad, A. 2004. Cluster Center Initialization Algorithm for Kmeans Clustering. *Pattern Recognition Letters* 25(11), pp. 1293-1302.

Kim, N. , Seo, J. B. , Lee, Y. , Lee, J. G. , Kim, S. S. and Kang, S.-H. 2008. Development of an Automatic Classification System for Differentiation of Obstructive Lung Disease Using HRCT. *Journal of Digital Imaging* 22(2), pp. 136-148.

Kirkosa, E. , Spathisb, C. and Manolopoulos, Y. 2007. Data Mining Techniques for the Detection of Fraudulent Financial Statements. *Expert Systems with Applications* 32(4), pp. 995-1003.

Klosgen, W. and Zytkow, J. M. 2002. *Handbook of Data Mining and Knowledge Discovery*. 1 ed. New York, USA: Oxford University Press, p. 1064

Kohonen, T. 2001. *Self-Organizing Maps*. 3 ed. Berlin, Germany: Springer, p. 501.

Kotsiantis, S. 2007. Credit Risk Analysis Using a Hybrid Data Mining Model. *International Journal of Intelligent Systems Technologies and Applications* 2(4), pp. 345-356.

Krause, E. F. 1986. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. New York, USA: Courier Dover Publications, p. 88

Krishna, K. and Murty, M. N. 1999. Genetic K-means Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part B* 29(3), pp. 433-439.

Kudova, P. 2007. Clustering Genetic Algorithm. In: *18th International Conference on Database and Expert Systems Applications (DEXA 2007)*. Regensburg, Germany. pp. 138-142.

Kuo, R. J. , Ho, L. M. and Hu, C. M. 2002. Integration of Self-Organizing Feature Map and K-Means Algorithm for Market Segmentation. *Computers and Operations Research* 29(11), pp. 1475-1493

Kuo, R. J. , Wang, H. S. , Hu, T.-L. and Chou, S. H. 2005. Application of Ant K-Means on Clustering Analysis. *Computers & Mathematics with Applications* 50(10-12), pp. 1709-1724.

Lee, J. Y. and Haj Darwish, A. 2008. Multi-Objective Environmental / Economic Dispatch Using the Bees Algorithm with Weighted Sum. In: *EU-Korea Conference on Science and Technology (EKC2008)*. Germany. pp. 267-274.

Lejeune, M. A. P. M. 2001. Measuring the Impact of Data Mining on Churn Management. *Internet Research: Electronic Networking Applications and Policy* 11(5), pp. 375-387.

Likas, A. , Vlassis, N. and Verbeek, J. J. 2003. The Global K-means Clustering Algorithm. *Pattern Recognition* 36, pp. 451-461.

Lin, H.-J. , Yang, F.-W. and Kao, Y.-T. 2005. An Efficient GA-Based Clustering Technique. *Tamkang Journal of Science and Engineering* 8(2), pp. 113-122.

Linde, Y. , Buzo, A. and Gray, R. M. 1980. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications* 28(1), pp. 84-95.

Liu, H. and Motoda, H. 2007. *Computational Methods of Feature Selection*. 1 ed. Boca Raton, Florida, USA: Chapman & Hall / CRC, p. 440.

Liu, J. and Xie, W. 1995. A Genetics-Based Approach to Fuzzy Clustering. In: *4th IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*. Yokohama, Japan: IEEE. pp. 2233-2240.

Lu, Y. , Lu, S. , Fotouhi, F. , Deng, Y. and Brown, S. J. 2004. Incremental Genetic K-means Algorithm and its Application in Gene Expression Data Analysis. *BMC Bioinformatics* 5(172).

MacQueen, J. B. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In: *5th Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, California: University of California Press. pp. 281-297.

Mahalanobis, P. C. 1936. On the Generalised Distance in Statistics. *Proceedings National Institute of Science* 2(1), pp. 49-55.

Marsh, H. W. 1998. Pairwise Deletion for Missing Data in Structural Equation Models: Nonpositive Definite Matrices, Parameter Estimates, Goodness of Fit, and Adjusted Sample Sizes. *Structural Equation Modeling* 5, pp. 22-36.

Matteucci, M. 2003. *A Tutorial On Clustering Algorithms.* Available at: <URL: http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html> [Accessed: 20-10-2008].

Maulik, U. and Bandyopadhyay, S. 2000. Genetic Algorithm-Based Clustering Technique. *Pattern Recognition* 33(9), pp. 1455-1465

McLachlan, G. J. and Krishnan, T. 1996. *The EM Algorithm and Extensions*. 1 ed. New York, USA: Wiley-Interscience, p. 304.

Merle, O. D. , Hansen, P. , Jaumard, B. and Mladenovi´c, N. 2000. An Interior Point Algorithm for Minimum Sum-of-Squares Clustering. *SIAM J. SCI. COMPUT.* 21(4), pp. 1485-1505.

Miyajima, C. , Tokuda, K. and Kitamura, T. 2000. Audio-Visual Speech Recognition Using Minimum Classification Errortraining. In: *IEEE Signal Processing Society Workshop*. Sydney, NSW, Australia: IEEE. pp. 3-12.

Moore, A. 1999. Very Fast EM-Based Mixture Model Clustering Using Multiresolution Kd-trees. In: *Advances in Neural Information Processing Systems II (NIPS)*. Breckenridge, Colorado: MIT Press. pp. 543-549

Moore, A. and Lee, M. S. 1998. Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Journal of Artificial Intelligence Research* 8, pp. 67-91.

Murthy, C. A. and Chowdhury, N. 1996. In Search of Optimal Clusters Using Genetic Algorithms. *Pattern Recognition Letters* 17, pp. 825-832.

Newman, D. J. , Hettich, S. , Blake, C. L. and Merz, C. J. 1998. UCI Repository of Machine Learning Databases. Irvine, CA: University of California, Department of Information and Computer Science.

Ng, R. T. and Han, J. 1994. Efficient and Effective Clustering Methods for Spatial Data Mining. In: *20th International Conference on Very Large Databases*. Santiago, Chile. pp. 145-155.

Nguyen, D.-C. 2004. *Flexible Information Management Strategies in Machine Learning and Data Mining.* PhD Thesis, Cardiff University.

Patané, G. and Russo, M. 2001. The Enhanced LBG Algorithm. *Neural Networks* 14(9), pp. 1219-1237.

Pelleg, D. and Moore, A. 1999. Accelerating Exact K-Means Algorithms with Geometric Reasoning. In: *5th ACM International Conference of the Special Interest Group on Knowledge Discovery and Data Mining (ACM-SIGKDD-99)*. San Diego, California, USA. pp. 277-281.

Pelleg, D. and Moore, A. 2000a. *Accelerating Exact K-Means Algorithms with Geometric Reasoning-Technical Report.* Pittsburgh: School of Computer Science, Carnegie Mellon University.

Pelleg, D. and Moore, A. 2000b. X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters. In: *7th International Conference on Machine Learning (ICML)*. Stanford, California, USA. pp. 727-734.

Pham, D. T. and Afify, A. A. 2007. Clustering Techniques and their Applications in Engineering. *Proceedings of the Institution of Mechanical*

Engineers, Part C: Journal of Mechanical Engineering Science 221(11), pp. 1445-1459.

Pham, D. T. , Afify, A. A. and Koç, E. 2007a. Manufacturing Cell Formation Using The Bees Algorithm. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 523-528.

Pham, D. T. , Ang, M. C. , Ng, K. W. , Otri, S. and Haj Darwish, A. 2008a. Generating Branded Product Concepts: Comparing The Bees Algorithm and an Evolutionary Algorithm. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 398-403.

Pham, D. T. , Castellani, M. and Fahmy, A. A. 2008b. Learning the Inverse Kinematics of a Robot Manipulator Using The Bees Algorithm. In: *17th IFAC World Congress COEX*. South Korea, 2008. pp. 493-498.

Pham, D. T. , Castellani, M. and Ghanbarzadeh, A. 2007b. Preliminary Design Using The Bees Algorithm. In: *8th International Conference on Laser Metrology, CMM and Machine Tool Performance*. Cardiff University, UK: Lamdamap. pp. 420-429.

Pham, D. T. , Castellani, M. , Sholedol, M. and Ghanbarzadeh, A. 2008c. The Bees Algorithm and Mechanical Design Optimisation. In: *International*

Conference on Informatics in Control, Automation and Robotics (ICINCO). Fuchal, Madeira - Portugal.

Pham, D. T. , Dimov, S. S. and Nguyen, C. D. 2004a. An Incremental K-means Algorithm. *Proceedings of the Institution of Mechanical Engineers - Part C - Journal of Mechanical Engineering Science* 218(7), pp. 783-795.

Pham, D. T. , Dimov, S. S. and Nguyen, C. D. 2004b. A Two Phase K-means Algorithm for Large Datasets. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 218(10), pp. 1269-1273.

Pham, D. T. and Ghanbarzadeh, A. 2007. Multi-Objective Optimisation Using The Bees Algorithm. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007).* Whittles, Dunbeath, Scotland. pp. 529-533.

Pham, D. T. , Ghanbarzadeh, A. , Koc, E. and Otri, S. 2006a. Application of The Bees Algorithm to the Training of Radial Basis Function Networks for Control Chart Pattern Recognition. In: *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering (ICME 06).* Ischia, Italy. pp. 711-716.

Pham, D. T. , Ghanbarzadeh, A. , Koc, E. , Otri, S. , Rahim, S. and Zaidi, M. 2006b. The Bees Algorithm – A Novel Tool for Complex Optimisation

Problems. In: *2nd International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*. Elsevier, Oxford. pp. 454-459.

Pham, D. T. , Ghanbarzadeh, A. , Otri, S. and Koç, E. 2009a. Optimal Design of Mechanical Components Using The Bees algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 223(5), pp. 1051-1056.

Pham, D. T. , Haj Darwish, A. and Eldukhri, E. E. 2009b. Optimisation of a Fuzzy Logic Controller Using The Bees Algorithm. *International Journal of Computer Aided Engineering and Technology* 1, pp. 250-264.

Pham, D. T. , Haj Darwish, A. , Eldukhri, E. E. and Otri, S. 2007c. Using The Bees Algorithm to Tune A Fuzzy Logic Controller for a Robot Gymnast. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 546-551.

Pham, D. T. , Koc, E. , Ghanbarzadeh, A. and Otri, S. 2006c. Optimisation of The Weights of Multi-Layered Perceptrons Using The Bees Algorithm. In: *5th International Symposium on Intelligent Manufacturing Systems*. Sakarya, Turkey. pp. 38-46.

Pham, D. T. , Koç, E. , Kalyoncu, M. and Tınkır, M. 2006d. A Hierarchical PID Controller Design for a Flexible Link Robot Manipulator Using The Bees

Algorithm. In: *6th International Symposium on Intelligent and Manufacturing Systems*. Sakarya, Turkey. pp. 757-765.

Pham, D. T. , Koc, E. , Lee, J. Y. and Phrueksanant, J. 2007d. Using The Bees Algorithm to Schedule Jobs for a Machine. In: *8th international Conference on Laser Metrology, CMM and Machine Tool Performance (LAMDAMAP)* Cardiff, UK: Euspen. pp. 430-439.

Pham, D. T. , Lee, J. Y. , Haj Darwish, A. and Soroka, A. J. 2008d. Multi-Objective Environmental / Economic Power Dispatch Using The Bees Algorithm With Pareto Optimality. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008).* Whittles, Dunbeath, Scotland. pp. 422-430.

Pham, D. T. , Mahmuddin, M. , Otri, S. and Al-Jabbouli, H. 2007e. Application of The Bees Algorithm to the Selection Features for Manufacturing Data. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 517-522.

Pham, D. T. , Muhamad, Z. , Mahmuddin, M. , Ghanbarzadeh, A. , Koc, E. and Otri, S. 2007f. Using The Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007).* Whittles, Dunbeath, Scotland. pp. 540-545.

Pham, D. T. , Negm, M. A. and Otri, S. 2008e. Using The Bees Algorithm to Solve a Stochastic Optimisation Problem. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 414-421.

Pham, D. T. , Otri, S. , Ghanbarzadeh, A. and Koc, E. 2006e. Application of The Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition. In: *2nd IEEE International Conference on Information and Communication Technologies: From Theory to Applications*. Damascus, Syria. pp. 1624-1629.

Pham, D. T. , Otri, S. and Haj Darwish, A. 2007g. Application of The Bees Algorithm to PCB Assembly Optimisation. In: *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*. Whittles, Dunbeath, Scotland. pp. 511-516.

Pham, D. T. and Oztemel, E. 1992. Control Chart Pattern Recognition Using Neutral Network. *Journal of Systems Engineering* 2(4), pp. 256-262.

Pham, D. T. , Pham, Q. T. , Ghanbarzadeh, A. and Castellani, M. 2008f. Dynamic Optimisation of Chemical Engineering Processes Using The Bees Algorithm. In: *17th IFAC World Congress COEX*. South Korea. pp. 6100-6105.

Pham, D. T. and Sholedolu, M. 2008. Using a Hybrid PSO-Bees Algorithm to Train Neural Networks for Wood Defect Classification. In: *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*. Whittles, Dunbeath, Scotland. pp. 385-390.

Pham, D. T. , Soroka, A. , Koc, E. , Ghanbarzadeh, A. and Otri, S. 2007h. Some Applications of The Bees Algorithm in Engineering Design and Manufacture. In: *International Conference on Manufacturing Automation (ICMA '07)*. Singapore. pp. 782-794.

Pham, D. T. , Soroka, A. J. , Ghanbarzadeh, A. , Koc, E. , Otri, S. and Packianather, M. 2006f. Optimising Neural Networks for Identification of Wood Defects Using The Bees Algorithm. In: *4th IEEE International Conference on Industrial Informatics*. Singapore: IEEE. pp. 1346-1351.

Ponniah, P. 2001. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. 1 ed. New York, USA: WileyBlackwell, p. 544.

Pyle, D. 1999. *Data Preparation for Data Mining*. San Francisco, California, USA: Morgan Kaufmann, p. 540.

Redmond, S. J. and Heneghan, C. 2007. A Method for Initialising the K-means Clustering Algorithm Using Kd-Trees. *Pattern Recognition Letters* 28(8), pp. 965-973.

Roth, P. L. 1994. Missing Data: A Conceptual Review for Applied Psychologists. *Personnel Psychology* 47(3), pp. 537-559.

Ruspini, E. H. 1969. A New Approach to Clustering. *Information and Control* 15(1), pp. 22-32.

Sahami, M. , Dumais, S. , Heckerman, D. and Horvitz, E. 1998. *A Bayesian Approach to Filtering Junk E-Mail-Technical Report*. Madison, Wisconsin: AAAI

Seeley, T. D. 1996. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Cambridge, MA, USA: Harvard University Press, p. 318.

Sheikh, R. H. , Raghuwanshi, M. M. and Jaiswal, A. N. 2008. Genetic Algorithm Based Clustering: A Survey. In: *1st International Conference on Emerging Trends in Engineering and Technology (ICETET 08)*. pp. 314-319.

Sheikholeslami, G. , Chatterjee, S. and Zhang, A. 1998. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In: *24th International Conference on Very Large Databases (VLDB 98)*. NewYork, USA. pp. 428-439.

Smith, S. D. G. , Escobedo, R. , Anderson, M. and Caudell, T. P. 1997. A Deployed Engineering Design Retrieval System Using Neural Networks. *IEEE Transactions on Neural Networks* 8(4), pp. 847-851.

Song, W. and Park, S. C. 2006. Genetic Algorithm-Based Text Clustering Technique: Automatic Evolution of Clusters with High Efficiency. In: *7th International Conference on Web-Age Information Management Workshops*. IEEE Computer Society. pp. 17-25.

Szilágyi1, L. , Szilágyi, S. M. and Benyó, Z. 2007. A Modified FCM Algorithm for Fast Segmentation of Brain MR Images. In: Kacprzyk, J. ed. *Analysis and Design of Intelligent Systems Using Soft Computing Techniques*. Vol. 41. Heidelberg: Springer Berlin, pp. 119-127.

Teng, C. M. 2001. A Comparison of Noise Handling Techniques. In: *FLAIRS-01*. Key West, Florida: American Association for Artificial Intelligence. pp. 269-273.

Thiesson, B. , Meck, B. , Chickering, C. and Heckerman, D. 1997. *Learning Mixtures of Bayesian Networks-Technical Report*. Redmond, WA: Microsoft

Tou, J. T. and Gonzalez, R. C. 1977. *Pattern Recognition Principles*. 2 ed. Reading, MA, USA: Addison-Wesley, p. 377.

Waarts, E. , Carree, M. A. and Wierenga, B. 1991. Full-information Maximum Likelihood Estimation of Brand Positioning Maps Using Supermarkt Scanning Data. *Marketing Research* 28(4), pp. 483-490.

Wang, H. , Yang, S. , Xu, W. and Sun, J. 2007. Scalability of Hybrid Fuzzy C-Means Algorithm Based on Quantum-Behaved PSO. In: *4th International Conference on Fuzzy Systems and Knowledge Discovery* Haikou, Hainan, China pp. 261-265.

Wang, S. , Zhou, M. and Geng, G. 2005. Application of fuzzy cluster analysis for medical image data mining. In: *Mechatronics and Automation, 2005 IEEE International Conference*. pp. 631- 636.

Wang, W. , Yang, J. and Muntz, R. 1997. STING : A Statistical Information Grid Approach To Spatial Data Mining. In: *23rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. pp. 186-195.

Xu, Y. 2005. Hybrid Clustering with Application to Web Mining. In: *International Conference on Active Media Technology (AMT 2005)*. Takamatsu, Kagawa, Japan. pp. 574-578.

Yang, S. Z. and Luo, S. W. 2005. A Novel Algorithm for Initializing Clustering Centers. In: *4th International Conference on Machine Learning and Cybernetics*. Guangzhou, China. pp. 5579-5583.

Yeh, T. , Lee, J. J. and Darrell, T. 2008. Scalable Classifiers for Internet Vision Tasks. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. Anchorage, Alaska, USA: IEEE. pp. 1-8.

Zamir, O. , Etzioni, O. , Madani, O. and Karp, R. M. 1997. Fast and Intuitive Clustering of Web Documents. In: *The 3rd International Conference on Knowledge Discovery and Data Mining*. Newport Beach, California, USA: AAAI Press. pp. 287–290.

Zhang, B. 2001. Generalized k-harmonic means – dynamic weighting of data in unsupervised learning. In: *1st SIAM International Conference on Data mining (SDM-01)*. Chicago, Illinois, USA. pp. 1-13.

Zhijie, X. , Laisheng, W. , Jiancheng, L. and Jianqin, Z. 2005. A Modified Clustering Algorithm for Data Mining. In: *IEEE Geoscience and Remote Sensing Symposium*. Seoul, South Korea: IEEE. pp. 741-744.