ENSEMBLE CLUSTERING VIA HEURISTIC OPTIMISATION

A thesis submitted for the degree of Doctor of Philosophy

by

Jian Li

Department of Information Systems and Computing
Brunel University, West London, UK

June 2010

Abstract

Traditional clustering algorithms have different criteria and biases, and there is no single algorithm that can be the best solution for a wide range of data sets. This problem often presents a significant obstacle to analysts in revealing meaningful information buried among the huge amount of data. Ensemble Clustering has been proposed as a way to avoid the biases and improve the accuracy of clustering. The difficulty in developing Ensemble Clustering methods is to combine external information (provided by input clusterings) with internal information (i.e. characteristics of given data) effectively to improve the accuracy of clustering.

The work presented in this thesis focuses on enhancing the clustering accuracy of Ensemble Clustering by employing heuristic optimisation techniques to achieve a robust combination of relevant information during the consensus clustering stage. Two novel heuristic optimisation-based Ensemble Clustering methods, Multi-Optimisation Consensus Clustering (MOCC) and K-Ants Consensus Clustering (KACC), are developed and introduced in this thesis. These methods utilise two heuristic optimisation algorithms (Simulated Annealing and Ant Colony Optimisation) for their Ensemble Clustering frameworks, and have been proved to outperform other methods in the area. The extensive experimental results, together with a detailed analysis, will be presented in this thesis.

Acknowledgements

I would like to take this opportunity to thank the people who gave me great help while completing my PhD. This thesis could not be completed without their support.

In the first place, I would like to thank my supervisors, Professor Xiaohui Liu and Dr. Stephen Swift, for their massive support and excellent supervision throughout my research. Your impressive advice and encouragement gave me much successful experience and confidence.

I would also like to thank Dr. Famili for his constructive and helpful comments on early drafts of Chapter 4. Thanks to Professor Sherry Chen and all other members of CIDA (Centre of Intelligent Data Analysis) for their assistance and support.

Last but not least, I am grateful to my parents, parents in law and my wife. Without their encouragement and support, I would not be able to get through the challenges during my research.

List of Publications

The following papers have been (or will be) published as a result of the research carried out during this thesis.

1. J. Li, X. Liu, S. Swift and Y. Shi, "K-Ants Consensus Clustering based on Weighted Description Length," *IEEE Transactions on Knowledge and Data Engineering*, 2010 (under review).

2. J. Li, S. Swift and X. Liu, "The Effect of Cooling Functions on Ensemble Clustering Using Simulated Annealing," *International Journal of Intelligent Data Analysis*, vol. 14, no. 6, 2010.

3. J. Li, S. Swift and X. Liu, "Multi-Optimisation Consensus Clustering," *Lecture Notes in Computer Science*, Springer Berlin, vol. 5772, pp. 345-356, 2009.

Table of Contents

List of Figures

List of Tables

1

Chapter 1: Introduction

Cluster analysis has been an indispensable technique for the pre-processing step of analysing high-dimensional data, where the number of dimensions could be a few dozen, hundreds or thousands. The principle of cluster analysis is to assign given objects into different groups based on their characteristics [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Berkhin, 2002; Xu and Wunsch, 2005; Pandey *et al.*, 2007]. Within a group, the objects are expected to have the same or very similar characteristics; between different groups, the characteristics of objects are expected to be much more different from each other.

In the last few decades, many individual clustering methods have been developed and used for different applications across many research areas including bioinformatics, pattern recognition, machine learning, data mining, and image processing [Jain and Dubes, 1988; Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Berkhin, 2002; Xu and Wunsch, 2005; Duda *et al.*, 2001; Hastie *et al.*, 2001].

The clustering analysis of gene-expression data provides a good example of contributions offered by clustering techniques. Based on gene-expression data, clustering techniques are used to partition genes into different clusters so that we can analyse the relationships between genes and predict the functions of unknown genes [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Jiang *et al.*, 2004; Pandey *et al.*, 2007; Causton *et al.*, 2007]. In other words, within a cluster, it is tended to that the genes have similar functions or play similar roles during the genetic process [Everitt, 1993; Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Webb, 1999; Causton *et al.*, 2007]. Clustering gene-expression data can help analysts to discover potential information about genes among huge amounts of data.

Individual clustering methods have made significant contributions to data analysis. However, it is important to note that different clustering methods have different clustering criteria. For a given data set, different algorithms may produce very different clustering results [Swift *et al.*, 2004; Hu and Yoo, 2004; Pandey *et al.*, 2007; Causton *et al.*, 2007]. For example, Swift *et al.* used four different individual clustering algorithms to cluster a B-cell lymphoma data set [Jenner *et al.*, 2003] in the literature [Swift *et al.*, 2004], where the results of the four algorithms are very different from each other.

In addition, some clustering algorithms are very sensitive to their initialisations. For a given data set, they may generate different results after different runs [Tseng and Kao, 2005; Hu *et al.*, 2006; Viswanath and Jayasurya, 2006; Lv *et al.*, 2006]. For instance, the K-means [Lloyd, 1982] algorithm is a well known partitional algorithm. It is capable of dealing with large data sets with a low computational complexity. However it is very sensitive to its initialisation so that the results of different runs of K-means may differ from each other.

Due to the above limitations of individual clustering methods, it is often hard for analysts to decide which algorithm is most suitable for a specific data set. Thus it is also often difficult for analysts to judge which clustering results reveal true structures of data [Hu and Yoo, 2004; Pandey *et al.*, 2007; Causton *et al.*, 2007; Tseng and Kao; 2005, Hu *et al.*, 2006; Lv *et al.*, 2006; Swift *et al.*, 2007]. Consequently, following the extensive research experience in classification methods, a new research topic, Ensemble Clustering, has emerged in recent years [Asur *et al.*, 2007; Viswanath and Jayasurya, 2006; Hu and Yoo, 2004; Swift *et al.*, 2004; Hu *et al.*, 2006; Lv *et al.*, 2006].

Existing Ensemble Clustering methods can be roughly classified into the following categories: (a) graph partitioning-based methods [Yu *et al.*, 2007; Karypis *et al.*, 1999]; (b) heuristic optimisation-based methods [Swift *et al.*, 2004; Tumer and Agogino, 2008; Yang and Kamel, 2003]; (c) re-sampling-based methods [Topchy *et al.*, 2003; Topchy *et al.*, 2004; Topchy *et al.*, 2005]; (d) soft clustering-based methods [Punera and Ghosh, 2008; Asur *et al.*, 2007]; (e) others [Fred and Jain, 2005; Hu *et al.*, 2006; Lv *et al.*, 2006]. According to surveys of existing Ensemble Clustering methods, it has been demonstrated that Ensemble Clustering methods are sufficiently robust to produce better clustering results than individual clustering methods [Kuncheva, 2006; Goder and Filkov, 2008; Topchy *et al.*, 2004].

The principle of Ensemble Clustering is to combine results produced by a set of individual clustering algorithms to generate more accurate clustering results. Therefore Ensemble Clustering methods usually contain two main steps: the generation of input clusterings, and the combination of input clusterings (also known as the consensus clustering step) [Yu *et al.*, 2007; Azimi *et al.*, 2007]. A common difficulty presented in the Ensemble Clustering process is performing the combination efficiently when input clusterings are very different from each other (and even some of them represent noise in the consensus step) [Fern and Brodley, 2004; Li, 2004; Vega-Pons *et al.*, 2008; Goder and Filkov, 2008]. It is necessary to develop more robust Ensemble Clustering methods.

1.1 Motivation

We claim to employ heuristic optimisation techniques to combine overall clustering information to improve the clustering accuracy of Ensemble Clustering for analysing high-dimensional data. In this thesis, we define high-dimensional data as the data with more than three dimensions. The number of dimensions is defined as the number of attributes of instances. The dimensionalities (i.e. the number of attributes) of most data sets used for our experiments are larger than 10 (the minimal dimensionality is 4, and the maximal dimensionality is 600).

We prefer using heuristic optimisation techniques for Ensemble Clustering because heuristic optimisation methods have three significant advantages. The first is that heuristic optimisation is capable of integrating overall information by different objective functions. The second is that the definition of objective functions is flexible. Analysts can define different objective functions based on different purposes, and even can define multiple objective functions for one single optimisation. Finally, heuristic optimisation methods have a feedback regulation mechanism, which can automatically supervise the progress of optimisation. During the optimisation process, the quality of each candidate solution is evaluated by the objective function(s) and other criteria. The candidate solution will be accepted or discarded based on the evaluation. After that, this information will be fed back to the solution generator to regulate the generation of the next candidate solution. In this way, the solution generator can enable candidate solutions to move closer to the optimum solution.

Although heuristic optimisation techniques have many advantages, the existing Heuristic Optimisation-Based Ensemble Clustering (HOBEC) methods have two main limitations:

- ✧ **The clustering combination relies solely on input clusterings.** Most existing heuristic optimisation-based Ensemble Clustering algorithms achieve the clustering combination only based on input clusterings. Hence the clustering accuracy of these algorithms will be influenced significantly when input clusterings are noisy or differ significantly from each other. Consensus Clustering (CC) proposed by Swift *et al.* [2004] is one of the widely used heuristic optimisation-based Ensemble Clustering algorithms. The Simulated Annealing (SA) [Kirkpatrick *et al.*, 1983, Granville *et al.*, 1994] algorithm is employed by CC. The principle of the CC algorithm is to use SA to seek optimal solutions that maximise the value of the objective function. Since the formation of the objective function is only based on input clusterings, it is obvious that the accuracy of final results depends heavily on the quality of input clusterings [Li *et al.* 2009].

- ✧ **The computational cost of optimisation is high.** Heuristic optimisation methods have been applied to seek possible optimal solutions in the state space, where an exhaustive search is impossible or difficult to be achieved. For a given problem, the state space is defined as the aggregation of all possible solutions. Based on the state space, heuristic methods are implemented. In theory, global optimisation only can be achieved by an infinite iterative procedure; in practice, we can only have a finite time to run the heuristic optimisation [Granville *et al.*, 1994; Eiben *et al.*, 1994]. However, it is impossible to guarantee that the global optimum can be reached within a finite time; this is known as a Nondeterministic Polynomial problem (NP-problem) [Garey and Johnson, 1979]. In order to find the global optimum, traditional heuristic optimisation methods require a long time to implement the optimisation especially for analysing large data sets. To overcome this difficulty, lots of efforts have been made to balance time cost against quality of results [Goldberg, 1989; Colorni *et al.*, 1991; Granville *et al.*, 1994; Eiben *et al.*, 1994]. In this thesis, we do not claim to guarantee generating global optimal results, but claim to offer results with a high enough degree of accuracy within the application context.

In order to overcome these problems, we have developed two novel heuristic optimisation-based Ensemble Clustering methods, which can achieve sufficiently robust Ensemble Clustering via heuristic optimisation. The concept of robustness in computer science is defined as the ability of a software/hardware system withstanding errors, faults and variations during its operating procedure [Dictionary.com, 2010]. In this thesis, the robustness of our novel Ensemble Clustering methods is defined to be the ability of coping with noisy input clusterings (i.e. input clusterings are very different from each other).

1.2 Framework

Two Ensemble Clustering methods are proposed in this thesis. The basic framework of our Ensemble Clustering methods is illustrated by Fig. 1.

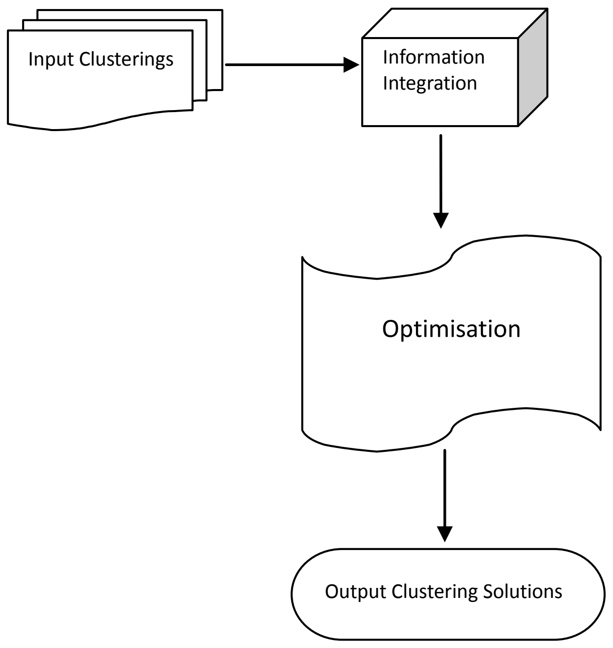


Fig. 1: The basic framework of our Ensemble Clustering methods

In order to analyse a given data set, firstly Input Clusterings will be generated. Input Clusterings contains a set of clustering solutions, which are generated by a set of individual clustering algorithms selected by analysts. Each solution is displayed by a one-dimensional vector, where each element indicates an instance of the given data set. Within the vector, the elements will have the same number if they belong to the same cluster, otherwise they will have different numbers.

The Information Integration component combines the clustering information provided by Input Clusterings. It transforms the external information into tractable information for the Optimisation component. In other words, the Information Integration component is the bridge between Input Clusterings and the Optimisation component.

The Optimisation component is where Heuristic Optimisation techniques are used. This is the key component of the basic framework, so that the performance of the framework depends mainly on the Optimisation component. Based on characteristics of given data and the information delivered from the

Information Integration component, the Optimisation component generates final results by seeking optimal clustering solutions.

Based on the basic framework, we have developed two Ensemble Clustering methods: Multi-Optimisation Consensus Clustering (MOCC) and K-Ants Consensus Clustering (KACC). MOCC utilises a Simulated Annealing-based multiple Optimisation component to integrate internal and external clustering information to enhance the accuracy of clustering; KACC adopts a three-phase Optimisation component to integrate overall clustering information to provide more accurate clustering. These two methods have been compared with some well-known Ensemble Clustering methods; the evaluation of results has demonstrated that both MOCC and KACC are capable of providing better clustering results than those methods in general.

1.3 Main Contributions

- **This thesis presents successful applications of employing Heuristic Optimisation techniques for Ensemble Clustering.** We have successfully integrated two well known heuristic methods, Simulated Annealing and the Ant Colony Optimisation theory, into our Ensemble Clustering frameworks. Many promising results have been generated by our methods.

- **The development of the Multi-Optimisation Consensus Clustering (MOCC) method.** We developed an advanced consensus clustering algorithm called MOCC, which combines agreement fitness evaluation of input clusterings with internal clustering separation criterion to enhance the clustering accuracy. MOCC generated very promising results, and is a good example for achieving Ensemble Clustering by integrating multiple appropriate clustering evaluation techniques into a multi-optimisation framework.

- **Effect analysis of cooling functions for Ensemble Clustering using SA.** The performance of SA depends heavily on configurations of the Cooling

Schedule (i.e. the cooling function). It is the most significant limitation of SA. In this thesis, a comprehensive analysis is given to demonstrate effect of different cooling functions on the performance of Ensemble Clustering. CC and MOCC are used as representatives of Ensemble Clustering methods during the analysis. This analysis offers insights into behaviours of cooling functions in the context of Ensemble Clustering.

- **The development of the K-Ants Consensus Clustering (KACC) method.** KACC combines internal and external clustering information by multiple optimisations with a low computational cost to enhance the accuracy of clustering. We developed the Attribute Weighted Description Length (AWDL) criterion to express internal information (i.e. characteristics of given data) for Ensemble Clustering, and utilise an Agreement Matrix to combine and express external information (provided by input clusterings). A K-Ants Multiple Optimisation Framework is developed to achieve three optimisation phases: the Agreement Based MDL Optimisation (ABMDLO), the Equal Probability MDL Optimisation (EPMDLO), and the Equal Probability Agreement Fitness Optimisation (EPAFO). These three optimisation phases achieve the combination of overall clustering information to provide more accurate clustering results.

- **Our methods have overcome the limitation of the clustering combination that is only based on input clusterings.** Since the clustering combination of existing HOBEC methods relies solely on input clusterings, the clustering accuracy of these methods is affected by noise and biases of input clusterings. Our novel HOBEC methods overcome the problem by combining all relevant clustering information (internal and external information) to achieve a robust clustering combination. Extensive experimental results are illustrated in this thesis and demonstrate that our novel methods have significant advantages in reducing effects of noise and biases of input clusterings.

- **Our method KACC performs its clustering optimisation with a much lower computational cost than existing HOBEC methods.** Expensive computational costs of existing HOBEC methods limit the clustering efficiency of these methods. Our novel method KACC solves this difficulty by carefully restricting the search space of its heuristic optimisation. In other words, KACC aims to provide a sufficient clustering accuracy in the context of applications instead of being pertinacious to search for global optima. The restriction of the search space is achieved by a well constructed initial clustering solution. In this way, MOCC can not only provide high accurate clustering results but also have a good clustering efficiency.

1.4 Thesis Organisation

The rest of this thesis is organised as follows:

Chapter 2 introduces background knowledge of the techniques and methods that relate to our work. Firstly, a review of individual clustering algorithms and Ensemble Clustering techniques is given. Specially, a related existing Ensemble Clustering method called Consensus Clustering (CC), which is the foundation of our research work, is presented in detail. A deep understanding of CC will be a good preparation for understanding the novel Ensemble Clustering methods proposed in this thesis. Secondly, several heuristic optimisation methods employed in our work will be introduced. Finally, a few related clustering evaluation techniques are described in detail.

Chapter 3 describes a novel Ensemble Clustering method called Multi-Optimisation Consensus Clustering (MOCC), where its multi-optimisation section is based on a heuristic algorithm Simulated Annealing (SA). The multi-optimisation framework and an advanced objective function are firstly introduced in this chapter. After that, results of experiments are discussed for performance comparison between CC and MOCC.

Chapter 4 investigates the effect of cooling functions for CC and MOCC using SA. Since SA is employed by both CC and MOCC for their optimisation sections, it is important to understand the performance of SA. One of the key components of SA is the cooling function. Therefore this chapter gives an in-depth study about cooling functions of SA, and explores how cooling functions affect the performances of CC and MOCC.

Chapter 5 introduces another novel Ensemble Clustering method named K-Ants Consensus Clustering (KACC). KACC contains a K-Ants Multiple Optimisation (KAMO) framework. The construction of KAMO is based on the basic principle of applying the Ant Colony Optimisation theory for clustering. After the description of the KAMO framework, results of experiments are discussed to give performance comparison between KACC and some other Ensemble Clustering methods.

Chapter 6 gives performance comparison between CC, MOCC and KACC. The comparison is conducted and discussed from two aspects: the accuracy of clustering, and the efficiency of clustering. This chapter gives the insights into different performances of the three Ensemble Clustering methods.

Chapter 7 draws a summary of the whole thesis. It consists of conclusions about what have been achieved through the research work, and an outline of possible further research directions.

1.5 Summary

This chapter has described the motivation of our research; illustrated the basic framework of our Ensemble Clustering methods; listed the main contributions of the work; outlined the whole structure of this thesis. It gives a general view of our research.

In the next chapter, we will detail the related background knowledge of our research to provide a good preparation for readers to have a thorough understanding of our novel Ensemble Clustering methods and the research carried out during this thesis.

2

Chapter 2: Background

This chapter is organised as follows: firstly, an overview of Individual Clustering methods will be introduced, followed by an overview of Ensemble Clustering methods. After that, related heuristic optimisation methods and the clustering validation indexes will be presented in detail.

2.1 Individual Clustering

Nowadays, it is important to understand the huge amount of high-dimensional data being accumulated. Cluster analysis has become a key technology for analysing these data. Many clustering algorithms have been developed in the last a few decades. Jain *et al.* [1999] and Everitt *et al.* [2001] roughly sorted existing individual clustering algorithms into two categories: hierarchical clustering and partitional clustering.

The principle of hierarchical clustering is to construct a hierarchical tree to characterise distances between different objects across a data set. A set of clusters of the data set are obtained by cutting the tree at a particular level [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005, Pandey *et al.*, 2007]. There are two ways of building the hierarchical tree [Johnson, 1967; Xu and Wunsch, 2005]. One is called Division; the other is called Agglomeration. Division allocates all objects into one cluster at the beginning, and then splits the cluster step by step until each object becomes an individual cluster [Johnson, 1967; Xu and Wunsch, 2005]. In contrast, Agglomeration treats each object as a cluster at the beginning, and then merges the two closest clusters into one cluster at each step until all objects are merged into one cluster [Johnson, 1967; Xu and Wunsch, 2005]. Hierarchical clustering can provide information for visualisation of the clustering structure to help analysts establish potential clusters. However this approach has a very high time complexity when dealing with large data sets [Johnson, 1967; Xu and Wunsch, 2005]. The typical hierarchical clustering methods are HC (Hierarchical Clustering) [Johnson, 1967; D'andrade, 1978; Murtagh, 1983], BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [Zhang *et al.*, 1996], and CURE (Clustering Using Representatives) [Guha *et al.*, 1998].

In partitional clustering, given objects are divided into a predefined number of clusters without a hierarchical structure [McQueen, 1967; Xu and Wunsch, 2005]. A “good” result of partitional clustering is expected to have a high homogeneity within each cluster and a high separation between clusters [Xu and Wunsch, 2005]. The typical partitional clustering methods are K-means [McQueen, 1967], and PAM (Partitioning Around Medoids) [Kaufman and Rousseeuw, 1990].

Another common taxonomy of clustering algorithms was introduced by Berkhin [2002], Xu and Wunsch [2005]. This taxonomy divides clustering algorithms into five categories. The first two categories are still hierarchical clustering and partitional clustering. The other three are density-based clustering, grid-based clustering, and model-based clustering.

Density-based clustering is to cluster objects based on the connectivity and density of objects in a data space [Berkhin, 2002; Xu and Wunsch, 2005]. This kind of clustering methods uses a density threshold to determine whether objects within a region can be formed into one cluster. If the density of a set of objects exceeds the threshold, the objects will be assigned into one cluster; otherwise, they will not be assigned into the same cluster. Density-based clustering has the ability of detecting arbitrary shaped clusters and dealing with high-dimensional data [Berkhin, 2002; Xu and Wunsch, 2005]. In addition, density-based clustering is sufficiently robust to deal with noise and outliers in data. Typical density-based clustering algorithms are DBSCAN [Ester *et al.*, 1996], and OPTICS [Ankerst *et al.*, 1999].

The process of grid-based clustering is based on a multiple-level granularity structure [Berkhin, 2002; Xu and Wunsch, 2005]. Grid-based clustering divides data space uniformly into a number of grids. Clustering is performed on each grid. Since the time complexity only relates to the number of grids, grid-based clustering has a good speed for dealing with large data sets [Berkhin, 2002; Xu and Wunsch, 2005]. There are some typical grid-based clustering methods such as STING (a STatistical INformation Grid approach) [Wang *et al.*, 1997], and WaveCluster [Sheikholeslami *et al.*, 1998].

Model-based clustering methods achieve clustering based on the assumption that clusters of given data can be determined by a series of probability distributions [Berkhin, 2002; Xu and Wunsch, 2005]. Model-based clustering hypothesises a model (such as a density distribution function) for each cluster, and then uses these models to find appropriate objects for corresponding clusters [Kohonen, 1981; Fisher, 1987]. Statistical model-based clustering and neural network model-based clustering are the most widely research topics. Model-based clustering has the ability of detecting arbitrary shaped clusters and better stabilities of clustering [Xu and Wunsch, 2005], but it is not capable of dealing with large data sets, and is sensitive to initialisation [Berkhin, 2002; Xu and Wunsch, 2005]. Typical model-based clustering methods are Self Organising Map (SOM) [Kohonen, 1981], Autoclass [Cheeseman and Stutz, 1996], and Cobweb [Fisher, 1987].

In the last a few years, a new taxonomy of clustering methods has emerged. According to the flexibility of clustering results, this taxonomy classifies individual clustering methods into two types: hard clustering methods, and fuzzy clustering methods [Berkhin, 2002; Xu and Wunsch, 2005]. The hard clustering means that each object is permanently assigned into one and only one cluster. Clustering algorithms such as K-means, HC, BIRCH, SOM and PAM all belong to hard clustering. In contrast, fuzzy clustering assigns each object into more than one cluster by a membership function [Xu and Wunsch, 2005], where the memberships are presented in the form of probabilities. A typical fuzzy clustering method is the Fuzzy C-Means (FCM) algorithm [Dunn, 1973; Bezdek, 1981].

For more details of the different taxonomies of clustering methods, readers are referred to references [Jain *et al.*, 1999; Everitt *et al.*, 2001; Berkhin, 2002; Xu and Wunsch, 2005].

Before further describing individual clustering algorithms, it is necessary to understand the concept of Distance Metric. The way of measuring similarity or distances between two objects is defined as Distance Metric. For any clustering approach, Distance Metric is a foundational component of clustering analysis. There are many different distance metrics for measuring the similarity between objects. Using different distance metrics may generate different shapes of clusters. Some common distance metrics are listed in Table 1 [Quarteroni and Fausto, 2006; Parr and Schucany, 1980; Xu, 2005].

In Table 1, X_r is a one dimensional vector, and denotes the object *r* in the data set *X* (with *n* objects); *m* indicates the number of attributes for each object (i.e. the number of elements of X_r); D is a diagonal matrix, where the diagonal elements are variances of attributes across all *n* objects; *V* is a sample covariance matrix; X_{rj} stands for the *j*th attribute of the object *r*; R_{rj} is the Spearman rank for the *j*th element of the vector X_r; $\overline{X_r}$ is the mean of the vector X_r. $N(X_{rj} \cup X_{sj})$ (where *j*=1, …, *m*) represents the number of elements in the union of vectors X_r and X_s. $N(X_{rj} \cap X_{sj})$ (where *j*=1, …, *m*) signifies the size of the intersection of vectors X_r and X_s. For example, we suppose X_r=[1,2,3,3], X_s=[2,3,5,6], then $[X_{rj} \cup X_{sj}]$=[1,2,3,5,6] and $[X_{rj} \cap X_{sj}]$=[2,3], where *j*=1, 2, 3, 4. Therefore, $N(X_{rj} \cup X_{sj}) = 5$, $N(X_{rj} \cap X_{sj}) = 2$. More details of these distance metrics can be found in the literature [Xu, 2005].

Table 1 The common distance metrics

Distance Metric	Distance Function
Euclidean distance	$d_{rs}^2 = (X_r - X_s)(X_r - X_s)^T$
standardised Euclidean distance	$d_{rs}^2 = (X_r - X_s)D^{-1}(X_r - X_s)^T$
Mahalanobis distance	$d_{rs}^2 = (X_r - X_s)V^{-1}(X_r - X_s)^T$
City-block distance	$d_{rs} = \sum_{j=1}^{m} \lvert X_{rj} - X_{sj} \rvert$
Minkowski distance	$d_{rs} = \left(\sum_{j=1}^{m} \lvert X_{rj} - X_{sj} \rvert^p \right)^{\frac{1}{p}}$
Spearman correlation	$d_{rs} = 1 - \frac{6\sum_{j=1}^{m}(R_{rj} - R_{sj})^2}{m(m^2-1)}$
Pearson correlation	$d_{rs} = \frac{1}{2}\left[1 - \frac{\sum_{j=1}^{m}(X_{rj} - \overline{X_r})(X_{sj} - \overline{X_s})}{\sqrt{\sum_{j=1}^{m}(X_{rj} - \overline{X_r})^2 \sum_{j=1}^{m}(X_{sj} - \overline{X_s})^2}}\right]$
Jaccard distance	$d_{rs} = \frac{N(X_{rj} \cup X_{sj}) - N(X_{rj} \cap X_{sj})}{N(X_{rj} \cup X_{sj})}$, where *j*= 1, …, m
Cosine distance	$d_{rs} = 1 - \frac{X_r X_s^T}{(X_r^T X_r)^{\frac{1}{2}} (X_s^T X_s)^{\frac{1}{2}}}$
Correlation distance	$d_{rs} = 1 - \frac{(X_r - \overline{\overline{X_r}})(X_s - \overline{\overline{X_s}})^T}{[(X_r - \overline{\overline{X_r}})(X_r - \overline{\overline{X_r}})^T]^{\frac{1}{2}} [(X_s - \overline{\overline{X_s}})(X_s - \overline{\overline{X_s}})^T]^{\frac{1}{2}}}$ where $\overline{\overline{X_r}} = \frac{1}{m}\sum_j X_{rj}$ and $\overline{\overline{X_s}} = \frac{1}{m}\sum_j X_{sj}$

In this thesis, some individual clustering algorithms have been chosen to generate clustering solutions as inputs for Ensemble Clustering methods. A brief description of these individual clustering algorithms is as follows:

2.1.1 Hierarchical Clustering (HC)

HC [Johnson, 1967; D'andrade, 1978; Murtagh, 1983] is a typical hierarchical clustering method. According to the process of constructing its hierarchical tree (also called dendrogram), HC has two different forms: the agglomerative HC and the divisive HC [Everitt, 1993; Jain *et al.*, 1999; Pandey *et al.*, 2007]. The key steps of the agglomerative HC algorithm are outlined as follows (the divisive HC algorithm has a reversed procedure to build the hierarchical tree) [Johnson, 1967; D'andrade, 1978]:

1) Treat each object as a cluster at the beginning;
2) Calculate distances between each pair of clusters based on a distance metric;
3) Find the closest two clusters and merge them into one cluster;
4) Repeat the steps 2 and 3 until all objects are assigned into one cluster.
5) Output a hierarchical tree.

There are several different Linkage Metrics (listed in Table 2) for defining the linkage between two clusters. Based on different Linkage Metrics, HC may generate different hierarchical trees.

Table 2 Six different Linkage Metrics for defining the distance between two clusters

Linkage Criterion	**Description**
Single linkage	Use the distance between two closest objects (which belong to two different clusters) to indicate the distance between the corresponding two clusters
Complete linkage	Use the distance between two farthest objects (which belong to two different clusters) to indicate the distance between the corresponding two clusters
Average linkage	Use the average distance between the objects in cluster A and the objects in cluster B to indicate the distance between the clusters A and B
Weighted linkage	It is similar to Average Linkage but use the weighted average distance to indicate the distance between two clusters
Centroid linkage	The distance between two clusters is defined as the Euclidean distance between the centroids of the two clusters
Median linkage	The distance between two clusters is defined by the Euclidean distance between the weighted centroids of the two clusters. The weighted centroids is the mean of two related previous weighted centroids,

The main advantages of HC are as follows [Everitt, 1993; Jain *et al.*, 1999; Webb, 1999]:

- It can provide very informative visualisation for predicting clustering structures.

The main limitations and biases of HC are as follows [Everitt, 1993; Jain *et al.*, 1999; Webb, 1999]:

- Using different Distance Metrics to define distances between objects may bias results of HC.
- Using different Linkage Metrics to define distances between clusters may bias results of HC.

- It is often difficult to appraise which Distance Metric (or Linkage Metric) is the most appropriate for analysing a given high-dimensional data set.
- Its time complexity is $O(n^2)$ so that HC is not capable of dealing with large data sets (where n is the number of objects in a given data set).
- Suppose HC has constructed a hierarchical tree for a given data set. If we need to add new objects into the data set, HC cannot predict the cluster assignment for the new objects without constructing a new hierarchical tree.
- It is sensitive to outliers and noise in data.

2.1.2 K-means Clustering

The K-means [McQueen, 1967] Clustering algorithm is a well-known partitioning algorithm. The key steps of the K-means clustering algorithm are outlined as follows [Jain *et al.*, 1999; Pandey *et al.*, 2007]:

1) Randomly select k objects to be centres of k clusters;
2) Assign the rest objects into their closest cluster centres;
3) Re-calculate the cluster centre (which is defined as the mean of all objects in the cluster) for each of the k clusters;
4) Re-allocate the objects into k clusters according to the new cluster centres;
5) Repeat steps 3 and 4 until there is no more change in each cluster or terminative conditions have been reached.

There are several ways of selecting objects for the initial cluster centres (or centroids). In this thesis, we use the following two approaches ("Sample" and "Uniform") to initialise cluster centres for the K-means algorithm [Quarteroni and Fausto, 2006;].

- *Sample* means Random Sampling, i.e. randomly selecting k objects from a given data set to be initial cluster centres.
- *Uniform* is defined as Uniform Sampling, i.e. selecting k objects uniformly from the range of a given data set to be initial cluster centres.

The main advantages of the K-means clustering algorithm are as follows [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Pandey *et al.*, 2007]:

- It is good for analysing large data sets.
- It has a good (low) time complexity.
- It is easy to be implemented.

The main limitations and biases of the K-means clustering algorithm are as follows [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005; Pandey *et al.*, 2007]:

- Using different Distance Metrics to define distances between objects may bias results of K-means.
- Each run of K-means may generate a different clustering result.
- It does not scale well for data where sizes of clusters vary significantly.
- It is sensitive to initialisation (i.e. the selection of *k* initial cluster centres). Hence an inappropriate initialisation can bias results of K-means.
- It is sensitive to noise and outliers in data.
- It cannot deal with non-convex shaped clusters.
- It is sensitive to the order of input data.
- It is not capable of dealing with categorical data (i.e. the data with categorical attributes).

2.1.3 Partitioning Around Medoids (PAM)

PAM was proposed by Kaufman and Rousseeuw [1987]. It is a K-medoids clustering method. The "medoid" means an object of a cluster, where the object (medoid) has the minimal average dissimilarity to all other objects within the cluster [Kaufman and Rousseeuw, 1987]. The dissimilarity between two objects can be defined by any distance metric such as the Euclidean distance. Comparing to the K-means algorithm, K-medoids algorithms are more robust to handle outliers and noise in data [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005].

The key steps of PAM are outlined as follows [Kaufman and Rousseeuw, 1987]:

1) Randomly chooses *k* objects as medoids to represent *k* clusters from a given data set;
2) Assign each object (non-medoid) to its nearest medoid;
3) Use an objective function, which is defined as the sum of the dissimilarities between all objects and their corresponding nearest medoids, to evaluate the quality of clustering. The best clustering solution is expected to minimise the objective function.
4) Calculate a swapping cost for each pair of non-medoid and medoid. Swapping mans using a non-medoid to replace a medoid. If the replacement can decrease the value of the objective function, the swap will be confirmed; otherwise, the medoid will not be replaced by the non-medoid.
5) Repeat steps 2, 3 and 4 until medoids have no more changes.

The main advantages of PAM are as follows [Kaufman and Rousseeuw, 1987; Jain *et al.*, 1999]:

- It can be applied to the data with categorical attributes.
- It is more robust than K-means to handle noise and outliers in data.

The main limitations and biases of PAM are as follows [Kaufman and Rousseeuw, 1987; Jain *et al.*, 1999]:

- Results could be biased by using different Distance Metrics.
- PAM is based on the assumption that each cluster can be well represented by its "medoid". It may bias results for the data sets that the assumption cannot be applied.
- The time complexity of PAM is $O(k(n\text{-}k)^2)$ so that it is not efficient for dealing with large data sets.
- It also needs to set the initial number of clusters *k*.

2.1.4 Cluster LARger Application (CLARA)

CLARA is also a K-medoids clustering method. It was proposed by Kaufman and Rousseeuw [1990]. CLARA is a multiple samples-based clustering method. CLARA employs PAM for clustering each of samples (i.e. subsets of original data). If we compare CLARA with PAM, an advantage of CLARA is that it can deal with much larger data sets than PAM [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005]. A weakness of CLARA is that the accuracy of clustering depends on the quality and size of samples [Xu and Wunsch, 2005].

The key steps of CLARA are outlined as follows [Kaufman and Rousseeuw, 1990]:

1) Randomly select a sample set from original data;
2) Use the clustering algorithm PAM to partition objects of the sample set into *k* clusters;
3) Assign each object into its nearest cluster for the rest objects of the original data;
4) Repeat the above three steps for a predefined number of iterations, and then choose a clustering solution, which has the shortest average distance, to be the final result. The average distance is defined as the mean distance between objects and their corresponding medoids.

The main advantages of CLARA are as follows [Kaufman and Rousseeuw, 1990; Xu and Wunsch, 2005]:

- It has a good efficiency for dealing with very large data sets.

The main limitations and biases of CLARA are as follows [Kaufman and Rousseeuw, 1990; Xu and Wunsch, 2005]:

- Sampling biases can affect the accuracy of clustering of CLARA.
- Results also could be biased by using different Distance Metrics.
- It also needs prior knowledge to define the number of clusters.

2.1.5 Affinity Propagation [Frey and Dueck, 2007]

Affinity Propagation (AP) [Frey and Dueck, 2007] extends the K-centres (K-means) clustering algorithm. AP does not require prior knowledge of the number of clusters. It uses predefined similarities between pair-wise objects to be inputs. The similarity between two objects indicates how suitable one object is to be the exemplar for the other object. A larger similarity denotes a higher probability of the object being the exemplar. Commonly, the similarities are initialised to be the same for all pairs of objects [Frey and Dueck, 2007].

AP treats each object as a node of a network, and conveys messages along edges between objects to seek appropriate objects to be central exemplars of corresponding clusters [Frey and Dueck, 2007]. AP has two kinds of messages to be transferred between objects. One is the *responsibility* message; the other is the *availability* message [Frey and Dueck, 2007]. For a pair of objects (i, j), the *responsibility* message is passed from i to j as cumulated evidence to reflect the suitability of i asking j to be its exemplar; in contrast, the *availability* message is sent from j to i to reveal how appropriate j becomes the exemplar for i. The messages are updated by minimising an energy function. If an object is assigned to an exemplar, the object will decrease its *availability* and increase its *responsibility*, whereas the exemplar will increase its *availability* and decrease its *responsibility*.

The key steps of Affinity Propagation are outlined as follows [Frey and Dueck, 2007]:

1) Set similarities for all pairs of objects;
2) Set availabilities and responsibilities of objects to be zeros;
3) For each iteration
    a) update all responsibilities based on the availabilities and similarities;
    b) update all availabilities based on the updated responsibilities;
    c) Combine responsibilities and availabilities to determine exemplars;
4) Terminate the algorithm if exemplars have no more changes for 10 iterations;
5) Generate the final clustering result by assigning each object into its closest exemplar.

The main advantages of AP are as follows [Frey and Dueck, 2007; Leone, Sumedha and Weight, 2007; Zhang, Furtlehner and Sebag, 2008]:

- AP does not require piror knowledge of the number of clusters.
- It can be applied to general types of data sets.

- AP is very efficient for dealing with large data sets.

The main limitations and biases of AP are as follows [Frey and Dueck, 2007; Leone, Sumedha and Weight, 2007; Zhang, Furtlehner and Sebag, 2008]:

- The principle of AP is based on the assumption that a cluster can be well represented by an exemplar. It leads to that AP has biased results for the data that has irregularly shaped clusters.
- Since noise can influence the choice of exemplars, AP is not robust to deal with noisy data.

2.1.6 Mean Shift Clustering [Fukunaga and Hostetler, 1975]

The Mean Shift Clustering algorithm was originally proposed by Fukunaga and Hostetler [1975]. Mean Shift Clustering is a non-parametric mode clustering method [Georgescu *et al.*, 2003]. It clusters objects in a data space based on a gradient function and an empirical probability density function [Georgescu *et al.*, 2003]. The gradient function is used as an objective function to detect modes for the density function. The zeros of the gradient function locate modes of the density function. The gradient function consists of a multivariate kernel density estimate section and a mean shift vector. A gradient ascent procedure is executed for each object on a local estimated density, and continued until convergence conditions are reached. During the gradient ascent procedure, stationary objects are treated as distribution modes of clusters, and related objects will be treated as members of the corresponding clusters. [Cheng, 1995; Comaniciu & Meer, 2002]

The key steps of Mean Shift Clustering are outlined as follows [Comaniciu & Meer, 2002; Georgescu *et al.*, 2003; Wang *et al.*, 2004]:

1) A mean shift procedure is run for each object;
    a) Computing the mean shift vector for an object;
    b) Updating a corresponding density estimation window based on the mean shift vector;

   c) Repeating the above two steps until the mean shift procedure is converged to find stationary objects for the density function;
2) These stationary objects are pruned by retaining only local maxima (these local maxima are treated as distribution modes of clusters);
3) Objects associated with the same mode are assigned into the same cluster.

The main advantages of Mean Shift Clustering are as follows [Cheng, 1995; Comaniciu & Meer, 2002; Georgescu *et al.*, 2003]:

- It is capable of dealing with arbitrary shaped clusters.
- It does not require prior knowledge of the number of clusters as an input.
- It is capable of dealing with noisy data.

The main limitations and biases of Mean Shift Clustering are as follows [Cheng, 1995; Comaniciu & Meer, 2002; Georgescu *et al.*, 2003]:

- Mean Shift Clustering only uses radially symmetric kernels for its multivariate kernel density estimate section. This restriction can bias the estimation of anisotropically shaped clusters.
- The constant kernel bandwidth of Mean Shift Clustering may limit the robustness of tracking objects.
- Mean Shift Clustering is sensitive to the selection of kernels.
- It can only be used to analyse data in Euclidean spaces. In other words, results of Mean Shift Clustering will be biased for analysing data in non-Euclidean spaces.
- Values of some parameters can only be specified empirically.

2.1.7 Fuzzy C-means (FCM)

The FCM algorithm is the most widely used fuzzy clustering algorithm [Bezdek, 1981; Jain *et al.*, 1999; Pandey *et al.*, 2007]. It was developed by Dunn [1973] and improved by Bezdek [1981]. The most important difference between hard clustering and fuzzy clustering is that fuzzy clustering does not assign each object into a cluster permanently. Fuzzy clustering uses a concept "degree" to express how objects belong to clusters. For each object, the sum of degrees (also called fuzzy coefficients) is 1 [Dunn, 1973; Bezdek, 1981].

The objective function of FCM is named Membership Function [Dunn, 1973; Bezdek, 1981]. A "good" clustering solution of FCM is expected to minimise the objective function. The Membership Function is defined as equations (2.1), (2.2) and (2.3) [Dunn, 1973; Bezdek, 1981]:

$$F_m = \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}^{m}\left\|x_i - C_j\right\|^2, \qquad (1 \le m < \infty) \tag{2.1}$$

$$u_{ij} = \frac{1}{\sum_{q=1}^{k}\left(\frac{\left\|x_i - C_j\right\|}{\left\|x_i - C_q\right\|}\right)^{\frac{2}{m-1}}} \tag{2.2}$$

$$C_j = \frac{\sum_{i=1}^{n} u_{ij}^{m} x_i}{\sum_{i=1}^{n} u_{ij}^{m}} \tag{2.3}$$

where *n* is the number of objects in a given data set; *k* is the number of clusters; m can be any real number that is greater than or equal to 1; u_{ij} is the fuzzy coefficient that indicates the degree of the object x_i belonging to the cluster *j*; C_j is the centre of the cluster *j*; $\left\|x_i - C_j\right\|$ denotes the distance between the object x_i and the cluster centre C_j.

The key steps of FCM are outlined as follows [Dunn, 1973; Bezdek, 1981]:

1) Assign fuzzy coefficients into each object according to an estimated number of clusters *k*;
2) Calculate cluster centres based on the fuzzy coefficients;
3) Re-calculate fuzzy coefficients based on the new cluster centres;
4) Evaluate the variance of fuzzy coefficients (and comparing with the one of previous fuzzy coefficients);
5) Compare the variance with a predefined sensitivity threshold;
6) Repeat steps 2, 3, 4 and 5 until the variance of fuzzy coefficients is less than the sensitivity threshold.

The main advantages of FCM are as follows [Jain *et al.*, 1999; Pandey *et al.*, 2007]:

- FCM is robust and flexible for dealing with uncertain and vague data.

The main limitations and biases of FCM are as follows [Jain *et al.*, 1999; Pandey *et al.*, 2007]:

- FCM is not capable of analysing data with the shapes such as the lip and skin.
- Using different Distance Metrics may bias results of FCM.
- It is not capable of dealing with noise and outliers in data.
- FCM requires prior knowledge of the number of clusters.
- Results of FCM may be stuck within local minima.

2.2 Ensemble Clustering

From the introduction of individual clustering algorithms in the previous section, it is clear that different individual algorithms have different biases and criteria. There is no single 'best' clustering algorithm for a wide range of data sets [Everitt, 1993; Jain *et al.*, 1999; Swift *et al.*, 2004; Hu and Yoo, 2004]. Different clustering algorithms may generate very different results for a given data set. For a given clustering problem, it should be considered that which algorithm is the most efficient. Sometimes, however, it is uncertain that which algorithm is the most suitable for a given problem. It is increasingly difficult for analysts to choose appropriate clustering algorithms for given problems. Therefore, Ensemble Clustering has emerged to avoid the biases of individual clustering algorithms. The purpose of Ensemble Clustering is to enhance the accuracy of clustering by combining a set of clustering solutions generated by individual clustering algorithms.

Ensemble Clustering syncretises results of individual clustering algorithms to generate more accurate clustering solutions. It usually contains two main steps: the generation of input clusterings, and the combination of input clusterings (also called the consensus clustering step) [Yu *et al.*, 2007; Azimi *et al.*, 2007]. Input clusterings are usually obtained from one of the following four sources [Topchy *et al.*, 2004; Kuncheva, 2006; Azimi *et al.*, 2007; Goder and Filkov, 2008]: (1) using different individual clustering algorithms to generate different clusterings; (2) using multiple times of sub-feature (subsequent) clustering for a data set based on feature extraction techniques; (3) running a single clustering algorithm for a number of times with different initialisations; (4) running a number of times of subset clustering for a data set. Many Ensemble Clustering methods have been proposed in the last decade. The way of combining input clusterings is a main feature that distinguishes different Ensemble Clustering methods. Existing Ensemble Clustering methods can be roughly sorted into the following categories:

(a) **Graph partitioning-based methods** [Yu *et al.*, 2007; Hu and Yoo, 2004].
For example, Hu and Yoo [2004] developed the Cluster Ensemble Algorithm (CEA), which is a typical graph-based Ensemble Clustering method. CEA uses a weighted graph to represent relationships between objects. Firstly, a set of individual clustering algorithms are used to generate input clusterings. Secondly, each input clustering is used to construct a distance matrix. Thirdly, these distance matrices are integrated to form a master distance matrix. Afterwards, a weighted graph is built based on the master distance matrix. Finally, a graph partitioning algorithm, METIS [Kayypis and Kumar, 1998], is applied to partition the weighted graph to obtain final clustering results. Hu and Yoo [2004] demonstrate that CEA generated significantly better clustering results than any of input clustering algorithms.

(b) **Heuristic optimisation-based methods** [Yang and Kamel, 2003; Swift *et al.*, 2004; Yang and Kamel 2006; Yang *et al.*, 2006].
This kind of methods employs heuristic algorithms to optimise the combination of input clusterings. The Consensus Clustering (CC) algorithm designed by Swift *et al.* [2004] is a typical heuristic optimisation-based Ensemble Clustering method. They employed a heuristic algorithm, Simulated Annealing, to combine input clusterings. Swift *et al.* [2004] claimed that CC generated reasonably better results than input clustering algorithms. Yang and Kamel [2003] proposed a method called ESIC

(Ensemble of Swarm Intelligence Clustering) based on the Ant Colony heuristic optimisation theory. ESIC uses three ant colonies (with different moving speeds) to produce three clustering solutions as inputs, and combines these inputs through a hypergraph model. Final results of ESIC are generated by implementing the ant colony heuristic algorithm again on the hypergraph model.

(c) **Re-sampling-based methods** [Topchy *et al.*, 2003; Topchy *et al.*, 2004; Topchy *et al.*, 2005].

These methods use an individual clustering algorithm to cluster sample sets, which are subsets randomly selected from an original data set, to generate a set of clustering solutions as inputs. Then the consensus clustering will be achieved by combining these input solutions. Bidgoli *et al.* [2004] present a resample-based Ensemble Clustering method that has two types of sampling: sub-sampling (sampling without replacement) and bootstrap (sampling with replacement). They have shown that integrating multiple clusterings of small size subsamples can generate meaningful partitions for an entire data set with a reduced computational cost.

(d) **Soft clustering-based methods** [Punera and Ghosh, 2008; Asur *et al.*, 2007; Punera and Ghosh, 2007].

Soft Ensemble Clustering methods combine results generated by a set of individual soft clustering algorithms. They use posterior membership probability distributions to represent clustering properties for each object, and utilise a distance metric to measure distances between probability distributions for objects in a soft ensemble [Punera and Ghosh, 2007]. Soft Ensemble Clustering has significant advantages for analysing vertically partitioned data [Punera and Ghosh, 2007]. Avogadri and Valentini [2008] proposed a fuzzy Ensemble Clustering method that employs the fuzzy k-means algorithm to generate a set of input clusteirngs in the form of a membership matrix. Each element of the membership matrix presents the membership of an object belonging to a cluster. Consensus clustering is implemented on the membership matrix.

(e) **Others** [Viswanath and Jayasurya, 2006; Hu *et al.*, 2006; Lv *et al.*, 2006].
Lv *et al.* [2006] proposed a multiple Ensemble Clustering method based on Core Groups. A core group is a set of objects that are always clustered together by whichever clustering algorithm is used to cluster a given data set [Lv *et al.* 2006]. Lv *et al.* use three agglomerative hierarchical algorithms to construct initial core groups; and these core groups are further refined by another hierarchical algorithm. A final clustering result is generated by the K-means algorithm based on the core groups. Hu *et al.* [2006] integrated text mining techniques with Ensemble Clustering for microarray gene identification clustering. Viswanath and Jayasurya [2006] developed a method called Ensemble of Leaders Clustering (ELC) to implement a fast Ensemble Clustering. Leaders Clustering assigns each object to its nearest leader. A group of leaders represents final clusters.

Surveys of existing Ensemble Clustering methods have demonstrated that Ensemble Clustering methods are sufficiently robust to produce better clustering results than individual clustering methods [Kuncheva, 2006; Goder and Filkov, 2008; Topchy *et al.*, 2004]. However, a common difficulty presented in the existing Ensemble Clustering methods is performing the consensus clustering step efficiently when input clusterings differ from each other and even some of them represent noise during the procedure of clustering combination [Fern and Brodley, 2004; Li, 2004; Vega-Pons *et al.*, 2008; Goder and Filkov, 2008]. It is necessary to develop more robust Ensemble Clustering methods to improve the accuracy of clustering.

In this thesis, we claim to use heuristic optimisation techniques to combine internal and external clustering information to improve the accuracy of clustering for Ensemble Clustering. Therefore our work is focused on developing Heuristic Optimisation-Based Ensemble Clustering (HOBEC) methods. We treat input clusterings as external clustering information, and characteristics of given data as internal clustering information.

In Chapter 1.1, we have presented advantages of employing heuristic optimisation techniques for Ensemble Clustering, and two difficulties with existing HOBEC methods. This thesis describes how we overcome the two difficulties by our novel Ensemble Clustering methods. The foundation of our research work is an existing HOBEC method called Consensus Clustering (CC) [Swift *et al.*, 2004]. A good understanding of CC will be helpful for understanding novel methods proposed in this thesis. Hence a detailed introduction of the CC algorithm is given in the following subsection.

2.2.1 Consensus Clustering (CC)

CC is a heuristic optimisation-based Ensemble Clustering method that was developed by Swift *et al.* [2004]. This method fuses results of other individual clustering algorithms to generate more reliable clustering results for given data sets [Swift *et al.*, 2004]. Hirsch *et al.* [2007] compared CC with different Ensemble Clustering methods included in the well-known CLUster Ensembles (CLUE) package, which was developed by Hornik [2005]. Results show that CC achieved comparable or even better performance than those ensemble methods.

CC has three key components: an Agreement Matrix, an Agreement Fitness Function (AFF), and a Simulated Annealing Optimisation (SAO) section. The Agreement Matrix is used to combine clustering information of input clustering solutions, which are generated by a set of individual clustering algorithms selected by analysts. The AFF is utilised by Simulated Annealing (as an only evaluation criterion) for optimisation searches in the SAO section. Swift *et al.* [2004] adopt the Agreement Matrix technique to build a State Space for the SAO section, and then use Simulated Annealing to seek optimal clustering solutions within the State Space. An optimal clustering solution is expected to maximise the value of the objective function AFF. Final clustering results will be generated by the SAO section.

- *Agreement Matrix*

The Agreement Matrix is built based on an input clustering matrix. Each row of the input clustering matrix is a clustering solution, which is generated by one individual clustering algorithm selected by analysts, in the form of a one-dimensional arrangement (vector). For a given data set, the length of the arrangement is equal to the total number of instances in the data set. Within the arrangement, each element indicates an instance of the given data set, and is labelled by a number. For instances assigned into the same cluster, the corresponding elements within the arrangement have the same label number; whereas elements will have different label numbers if the corresponding instances are assigned into different clusters.

The Agreement Matrix is a square symmetric matrix with zeros along the leading diagonal. Each element of the Agreement Matrix indicates how many input clustering algorithms agree that the two corresponding instances are assigned into one cluster. If we use a variable *A* to indicate the Agreement Matrix, equation (2.4) gives the definition of the Agreement Matrix as follows:

$$A = \begin{bmatrix} 0 & a_{1,2} & & \cdots & & a_{1,n-1} & a_{1,n} \\ a_{2,1} & 0 & & & & a_{2,n-1} & a_{2,n} \\ & & & & a_{i,j} & & \\ \vdots & & & \ddots & & \vdots & \\ & & a_{j,i} & & & & \\ a_{n-1,1} & a_{n-1,2} & & \cdots & & 0 & a_{n-1,n} \\ a_{n,1} & a_{n,2} & & & & a_{n,n-1} & 0 \end{bmatrix} \tag{2.4}$$

It is shown that *A* is a $n \times n$ square symmetric matrix, where *n* means the total number of instances in a given data set. Each element $a_{i,j}$ denotes the number of input clustering algorithms that agree the instances *i* and *j* are assigned into the same cluster.

- *Agreement Fitness Function*

The Agreement Fitness Function (AFF) is defined as the objective function of Simulated Annealing for the SAO section. The value of AFF denotes agreement of input clustering algorithms for a clustering solution. The calculation of AFF can be simply achieved by two steps. The first step is to calculate sub-agreement fitness for each cluster; the second step is to sum up all sub-agreements fitness to obtain the value of AFF. The definition of AFF is shown by equations (2.5) - (2.6) [Swift *et al.* 2004].

$$f(C) = \sum_{i=1}^{m} F(C_i) \tag{2.5}$$

$$F(C_i) = \begin{cases} \sum_{k=1}^{S_i-1} \sum_{q=k+1}^{S_i} \left(A_{C_{ik},C_{iq}} - \beta\right), & S_i > 1 \\ 0, & otherwise \end{cases} \tag{2.6}$$

Equation (2.5) shows the sum of all sub-agreements fitness, where *$F(C_i)$* is sub-agreement fitness of the cluster *i* in the clustering arrangement *C*. The total number of clusters in *C* is *m*. Equation (2.6) defines how to calculate sub-agreement fitness of a cluster. The variable S_i stands for the total number of instances in the cluster *i*. If a cluster only has one instance (i.e. S_i = 1), the sub-agreement fitness of this cluster will be zero. *β* is the agreement threshold, which can reward or punish agreement of a cluster. C_{ik} means the *k*th instance in the cluster *i*; the variable $A_{C_{ik},C_{iq}}$ denotes the agreement value (in the Agreement Matrix *A*) for the *k*th and *q*th instances in the cluster *i*.

In order to further understand $A_{C_{ik},C_{iq}}$, We use an example to explain how to obtain the value of $A_{C_{ik},C_{iq}}$. Suppose a data set consists of six instances {1, 2, 3, 4, 5, 6} that are assigned into two clusters C_1 and C_2, where C_1 is {1, 3} and C_2 is {2, 4, 5, 6}. We want to know the value of $A_{C_{21},C_{23}}$. According to the definition of C_{ik}, we get C_{21}=2 and C_{23}=5. Then $A_{C_{21},C_{23}}$ can be rewritten as $A_{2,5}$. Therefore, we can obtain the agreement value of $A_{C_{21},C_{23}}$ from the Agreement Matrix *A* according to the equation $A_{C_{21},C_{23}} = A_{2,5} = a_{2,5}$ and the equation (2.4).

Swift *et al.* [2004] defined the Agreement Threshold as equation (2.7):

$$\beta = \frac{\mathrm{Max}(A) + \mathrm{Min}(A)}{2} \tag{2.7}$$

where Max(A) and Min(A) indicate the maximum agreement value and the minimum agreement value respectively in the Agreement Matrix A. It is clear that β is defined as the mean agreement value, which rewards instance-pairs that have agreement values above the mean value, and penalises instance-pairs that have agreement values below it. Therefore, if instance-pairs (such as $A_{C_{ik}C_{iq}}$) in each cluster have very high agreement values for a clustering solution, the final value of AFF will be high; otherwise AFF will have a low agreement value or even a negative agreement value (i.e. most of input clustering algorithms disagree with the clustering solution).

The CC algorithm aims to seek an optimal clustering solution that has the maximum agreement fitness value. In other words, the value of AFF is expected to be as large as possible.

- *Simulated Annealing-based Optimisation*

The Simulated Annealing (SA) algorithm will be introduced in Section 2.3.1 of Chapter 2. CC employs SA for its optimisation section to optimise combination of clustering information for input clusterings. SA-based optimisation is achieved by searching for the best clustering solution (which has the maximal agreement fitness value of the objective function AFF) within a State Space restricted by the Agreement Matrix. For a given data set, the State Space consists of all possible clustering solutions.

SA-based optimisation starts from a random clustering solution selected from the State Space. SA treats the random solution as a current solution, and the agreement fitness value of the current solution is calculated according to equations (2.5) – (2.7). After that, SA uses a Solution Generator to generate candidate clustering solutions. In the CC algorithm, the Solution Generator produces candidate solutions by three different actions, which are moving an instance from one cluster to another, splitting a cluster into two clusters, and merging two clusters into one cluster. These actions have an equal probability to be chosen during the optimisation process.

Once a candidate solution is generated, the corresponding agreement fitness value will be calculated to be compared with the one of the current solution. If the candidate solution has a higher agreement fitness value than the current solution, the current solution will be updated (replaced) by the candidate solution; otherwise the candidate solution will be discarded. And then SA continues to generate new candidate solutions. The above process will be continued until the value of AFF is converged (or a certain number of iterations have been reached).

- *The Whole Framework of CC*

Fig. 2 displays the key steps of the CC algorithm as follows:

```
Input: Input Clustering Matrix, InpMat; Upper-limit of the Number of Clusters, $K_{max}$; Number of
        Iterations, $N_C$; Initial Temperature, $T_0$.

(1) Generate a Agreement Matrix, A, according to the Input Clustering Matrix InpMat;
(2) Calculate the Agreement Threshold, $\beta$, according to Equation (2.7);
(3) Generate a random clustering solution to be the current solution Z, and calculate its
    agreement fitness value $z$ according to Equations (2.5) and (2.6);
(4) Based on the current solution Z, Simulate Annealing starts to seek the optimal clustering
    solution that maximises the value of the Agreement Fitness Function.
    (i)     $T = T_0$
    (ii)    For $i$ = 1 to $N_C$ do
    (iii)       Generate a candidate solution $Z'$ by the Solution Generator, and calculate its
                agreement fitness value $z'$ (the number of clusters of $Z'$ cannot exceed $K_{max}$)
    (iv)        IF  $z' > z$, Then
    (v)             $Z$ will be updated by $Z'$
    (vi)        Otherwise
    (vii)           The acceptance of $Z'$ will be based on an probability $p = e^{-\Delta f}$, where
                    $\Delta f = [z - z']/T$.
    (viii)          IF $p > p'$, Then       (where $p'$ is an random probability within [0, 1])
    (ix)                $Z$ will be updated by $Z'$
    (x)             Otherwise
    (xi)                $Z'$ will be discarded
    (xii)           End IF
    (xiii)      End IF
    (xiv)       IF the value of AFF is converged, Then, Exist FOR LOOP, End IF
    (xv)        Use a predefined cooling function to update $T$
    (xvi)   End For
Output: An optimal clustering solution
```

Fig. 2: The Consensus Clustering algorithm.

It is clear that CC has four main steps. Firstly, CC builds an Agreement Matrix based on Input Clustering Matrix *InpMat*. Secondly, based on the Agreement Matrix, equation (2.7) is used to calculate the Agreement Threshold β. After that, CC generates an initial Current Solution, and calculates its agreement fitness value based on β and the Agreement Fitness Function. Finally, SA starts from the Current Solution to seek an optimal solution that maximise the value of AFF.

Within the four steps, the final step (Step (4)) is the most complicated. It is an iteration procedure. The number of iterations is set to be N_C. In each iteration, the Solution Generator produces a candidate solution, and the corresponding agreement fitness value is calculated based on AFF. Then, the candidate solution will be evaluated by comparing its agreement fitness value with the one of the Current Solution. If the candidate solution has a higher agreement value, it will be accepted to replace the Current Solution (i.e. the Current Solution is updated); otherwise a probability p will be used to decide whether accept the candidate solution or not (the definition of p is shown in Step (vii)). If p is high enough, the candidate solution will still be accepted, whereas it will be discarded if p is not. The whole iteration procedure is continued until the value of AFF is converged or the predefined number of iterations N_C has been reached. Finally, the latest Current Solution will be treated as an optimal clustering solution to be the final result of CC.

From the framework shown in Fig. 2, we can see that the Agreement Fitness Function and the Simulated Annealing Optimisation section are the most important components of CC. Therefore, two related situations need to be noticed. One is that, within the Agreement Fitness Function, the Agreement Threshold β is the key parameter. The definition of β can directly affect selections of candidate solutions during the SA-based optimisation. The other is that, in the Simulated Annealing Optimisation section, the Agreement Fitness Function is the only evaluation criterion. Hence results of CC may be enslaved by the accuracy of input clusterings.

2.3 Heuristic Optimisation

In computer science, optimisation is to find the best solution for a programming problem from a defined solution space [Elster, 1993]. Optimisation algorithms are mainly sorted into two classes: classical optimisation algorithms, and heuristic optimisation algorithms [Elster, 1993; Yang, 2008].

Classical optimisation algorithms have been playing a very important role for solving optimisation problems since the concept "linear programming" was proposed as the first term of optimisation by Dantzig [1947]. Classical optimisation algorithms have three characteristics [Dantzig, 1947; Elster, 1993; Yang, 2008]: 1) they use enumeration or differential calculus to seek the best solution; 2) the best solution is assumed to be existing and unique; 3) the convergence of algorithms is guaranteed to find the best solution for the first-order conditions. Classical optimisation algorithms can be divided into two categories: linear optimisation (or linear programming) algorithms and nonlinear optimisation (or nonlinear programming) algorithms [Dantzig, 1947; Elster, 1993]. Linear optimisation is to utilise a linear objective function to study convex programming problems [Dantzig, 1947; Elster, 1993]. Constraints of linear optimisation are defined by linear equalities or inequalities [Dantzig, 1947; Elster, 1993]. Nonlinear optimisation is to (but not limited to) analyse convex programming problems by a nonlinear objective function and constraints [Dantzig, 1947; Elster, 1993].

Although classical optimisation algorithms have significant contributions for solving optimisation problems, it is not practical or possible to apply them for problems that have large solution spaces [Elster, 1993; Yang, 2008]. Heuristic optimisation algorithms have solved the difficulty. For clustering analysis, analysts often need to deal with large data sets with high dimensionalities. In this case, heuristic optimisation algorithms can only be applied.

Heuristic optimisation combines heuristic information obtained from itself to further guide its search directions during the optimisation process. An important advantage of heuristic optimisation algorithms is that heuristic optimisation algorithms can obtain an optimal solution with a reasonable lower time cost than classical optimisation algorithms [Elster, 1993; Yang, 2008].

Heuristic optimisation includes simple heuristic optimisation algorithms, and meta-heuristic optimisation algorithms [Elster, 1993; Yang, 2008]. Greedy algorithms and local search algorithms belong to simple heuristic optimisation. These heuristic algorithms have two limitations [Elster, 1993; Yang, 2008]: 1) they cannot explore Solution Space systematically to find global optima; 2) since their results are often stuck in local maxima, the quality of results may not be satisfied. Meta-heuristic optimisation means advanced heuristic optimisation, which is mostly applied for solving combinational optimisation problems for which other optimisation techniques cannot be implemented [Elster, 1993; Yang, 2008]. There are many types of meta-heuristic optimisation algorithms including discontinuous methods, guided search methods, single agent methods, and multi-agent or population-based methods [Elster, 1993; Yang, 2008].

In this thesis, we employ two meta-heuristic optimisation algorithms (we simply call them Heuristic Algorithms) for our Ensemble Clustering frameworks. These algorithms are Simulated Annealing, and Ant Colony Optimisation.

2.3.1 Simulated Annealing

Simulated Annealing (SA) [Kirkpatrick *et al.*, 1983; Granville *et al.*, 1994] is a single agent meta-heuristic search algorithm which simulates the process of metal cooling into a minimal energy state. The most significant advantage of SA is its capability of seeking global optima for nonlinear problems. In general, SA has three main components: an objective function, a solution generator, and a cooling schedule [Snedecor and Cochran, 1980; Kirkpatrick *et al.*, 1983]. Firstly, for a given problem, an objective function must be designed. It must be guaranteed that the domain (i.e. the state space) of the objective function contains all possible solutions for the problem. Secondly a solution generator needs to be developed for coming up with candidate solutions during the search process. Finally a cooling schedule needs to be carefully set up in order to achieve the optimisation search.

The purpose of SA is to find a solution that optimises the value of the objective function. SA achieves this goal by decreasing a probability of accepting worse solutions during the search process. SA starts with a random solution, and then uses the solution generator to create candidate solutions that are close to previous solutions in the state space. If a candidate solution is worse, it will be discarded by a random probability based on a cooling schedule; otherwise, it will be accepted directly. The search will be continued from the accepted new solution. Fig. 3 shows the key steps of the SA algorithm. For further details, please refer to references [Snedecor and Cochran, 1980; Kirkpatrick *et al.*, 1983; Granville *et al.*, 1994].

```
Parameters:   Number of iterations, $Ite$; Initial Temperature, $T_0$; Final Temperature, $T_n$; Cooling
                 Function, $Fc$; Objective Function, $F$.
(1) Generate a random solution $S$ for the Objective Function $F$
(2) $T_i = T_0$
(3) For $i = 1$ to $Ite$ do
(4)       Generate another random solution $S'$ that close to the previous solution $S$
(5)       If   $S'$ is better than $S$, then
(6)             $S'$ will be accepted directly to replace $S$
(7)        End If
(8)        If   $S'$ is worse than $S$, then
(9)              The acceptance of $S'$ will depend on a random probability $p$; if $S'$ is not accepted, $S$
                   will be remained.
(10)      End If
(11)      If   $T_i > T_n$ ,   then   $T_i = Fc(i, T_0)$,   End If
(12) End For
Output:   An optimal clustering solution
Note:   In line (9), $p = e^{-\Delta f}$, where $\Delta f = [F(S) - F(S')]/T_i$.
```

Fig. 3: The Simulated Annealing algorithm

SA has two main disadvantages [Kirkpatrick *et al.*, 1983; Granville *et al.*, 1994]. One is that SA is sensitive to settings of its parameters such as the number of iterations, initial temperature, definition of the objective function, and the cooling function. The other is that its convergence rate is very low. This limitation relates to the cooling schedule. One may change the cooling schedule to enable SA to converge in a very short time period, but SA may converge too early to reach a global optimum. We have to balance time costs against the quality of results. In practice, performance of SA much depends on experience of users [Kirkpatrick *et al.*, 1983]. It is necessary to understand performance of these parameters to make an appropriate configuration.

2.3.2 Ant Colony Optimisation

Ant Colony Optimisation (ACO) is a typical discontinuous meta-heuristic optimisation method. It was firstly proposed by Colorni, Dorigo and Maniezzo [1991]. The paper named "Ant System: Optimization by a colony of cooperating agents", which was written by Dorigo *et al.* [1996] and published in 1996, is a landmark of developing the ACO theory. After that, a systematic account of the ACO theory was given by Bonabeau *et al.* [2000].

The basic ACO mechanism has the following characteristics [Dorigo *et al.*, 1996; Mullen *et al.*, 2009]. Firstly, ants communicate with each other by synthetic pheromones. Each ant makes decisions according to pheromones (left by other ants), and then leaves new pheromones about its decisions for other ants. Secondly, although each ant makes decisions independently according to its surrounding environment, the colony of ants exhibits orderly activities as a whole. Finally, ACO is an intelligent multi-agent system. Each agent (ant) explores resources for the system; meanwhile, the whole system exploits the obtained resources to provide further guidance for explorations of ants. This recurrence will be continued until an objective is achieved.

We take the TSP problem as an example to describe the key steps of the stand ACO algorithm as follows [Colorni *et al.*, 1991; Deneubourg *et al.*, 1991; Dorigo *et al.*, 1996; Lumer and Faieta, 1994]:

1) Initialisation: the current number of iterations $N = 0$, the maximal number of iterations N_{max}, the number of ants m, data of n cities, initialise the vector diagram Q;
2) $N= N+1$;
3) Set the index k of Tabu List to be 1 for an ant;
4) Based on a probability calculated by a state transition equation, the ant

selects a city *j* to move on (where *j* is not in the Tabu List);

5) Update the Tabu List for the ant by adding the city *j* to the list;
6) If $k<n$, then $k= k+1$ and go to step 4, otherwise, go to step 7;
7) Update information of each path (edge) of the vector diagram *Q*, and clear the Tabu List;
8) Repeat steps from 2 to 7 until $N \geqslant N_{max}$;
9) Output an optimal solution.

For different applications, analysts developed different ACO-based algorithms. Since the work described in this thesis relates to clustering analysis, the corresponding ACO-based method is the Ant Colony Clustering algorithm. This section only gives a brief description of the stand ACO algorithm. The details of the Ant Colony Clustering algorithm will be described in Chapter 5.2.4.

For more details of the ACO theory, please refer to relevant literatures, [Bonabeau *et al.*, 2000; Colorni *et al.*, 1991; Deneubourg *et al.*, 1991; Dorigo *et al.*, 1996; Lumer and Faieta, 1994] and [Mullen *et al.*, 2009], listed at the end of this thesis.

2.4 Clustering Validity

Clustering validity is a measure to indicate how a partitioning generated by a clustering method fits a given data set [Halkidi, Batistakis and Vazirgiannis, 2001]. The validity of clustering results is evaluated by clustering validation indices. Clustering validation indices are basically sorted into two types: external validation indices, and internal validation indices [26]. External validation indices (such as Weighted-Kappa [Viera and Garrett, 2005] and Adjusted Rand [Hubert and Arabie, 1985]) evaluate the accuracy of clustering results based on prior knowledge (e.g. true clusters of given data). They measure similarities between generated clustering results and prior knowledge. Internal validation indices (such as Silhouette Width [Rousseeuw, 1987], Homogeneity and Separation [Shamir and Sharan, 2002]) rely on interior characteristics of given data instead of prior knowledge. Which type of validation indexes we should use depends on what information we have. If we know true clusters of a given data set, external validation indices will be used for evaluating accuracies of clustering results. For analysing an untested data set, we do not know its true clusters so that we can only use internal validation indices for evaluation.

For each data set used in this thesis, we have known clusters (i.e. true clusters). The simplest way of evaluating accuracies of results is to compare the results with the known clusters to see how close they are. The closer they are, the more accurate the results are. Therefore, in this thesis, we employ a well-known external validation index, Weighted-Kappa (WK) [Viera and Garrett, 2005], to evaluate clustering results for the accuracy comparison. The concept of accuracy, in the domain of science, is defined as the degree of conformity between a measured quantity and its true value [Taylor, 1999]. We use WK to measure the degree of agreement between the true clusters and a clustering result to indicate the clustering accuracy of the clustering result. This validation index was also employed by Hirsch *et al.* [2007] and Swift *et al.* [2004].

We use Weighted-Kappa instead of a huge range of cluster validation indices for a number of reasons. Firstly we have the known clusters (i.e. the true clusters) for each of experimental data sets. Hence the "best" clustering method should be able to produce results that exactly match the known clusters. Secondly, internal validation indices have notable limitations. For example, the well-known Silhouette [Rousseeuw, 1987] index aims to reflect separation and compactness of clusters. A larger Silhouette value is expected to indicate a better clustering result [Rousseeuw, 1987]. In fact, however, this illation is not always tenable because the largest Silhouette value is always obtained only when each instance becomes an individual cluster. Finally it is worth noting that outcomes of the Weighted-Kappa index are identical to results produced by the well-known Adjusted Rand index [Hubert and Arabie, 1985].

Weighted-Kappa constructs a (2x2) contingency table (as shown in Table 3) to measure agreement (i.e. similarity) between two clustering arrangements [Viera and Garrett, 2005]. For a given data set, all unique pairs of objects will be counted in this table. The total number of unique pairs for n objects is defined as equation (2.8):

$$N = \frac{n(n-1)}{2} \tag{2.8}$$

Table 3: The WK contingency table

	Cluster Arrangement 2	
Cluster Arrangement 1	$\boldsymbol{U_{SS}}$ (The total number of unique pairs, which two arrangements all agree the objects of each pair are in the same cluster)	$\boldsymbol{U_{SD}}$ (The total number of unique pairs, which Arrangement 1 agrees the objects of each pair are in the same cluster, but Arrangement 2 does not)
	$\boldsymbol{U_{DS}}$ (The total number of unique pairs, which Arrangement 2 agrees the objects of each pair are in the same cluster, but Arrangement 1 does not)	$\boldsymbol{U_{DD}}$ (The total number of unique pairs, which two arrangements all disagree the objects of each pair are in the same cluster)

The way of calculating WK is shown by equations (2.9), (2.10) and (2.11):

$$WK = \frac{P_{ob} - P_{ex}}{1 - P_{ex}} \tag{2.9}$$

$$P_{ob} = \frac{1}{N}(U_{SS} + U_{DD}) \tag{2.10}$$

$$P_{ex} = \frac{1}{N^2}[(U_{SS} + U_{SD})(U_{SS} + U_{DS}) + (U_{DS} + U_{DD})(U_{SD} + U_{DD})] \tag{2.11}$$

where P_{ob} means the observed agreement, and P_{ex} means the expected agreement [Sims, 1980]. N is the total number of unique instance-pairs. U_{SS} denotes the number of unique instance-pairs which the Cluster Arrangements 1 and 2 all agree that two instances of each pair are assigned into the same cluster, whereas U_{DD} stands for the number of unique instance-pairs which the Cluster Arrangements 1 and 2 all disagree two instances of each pair are assigned into the same cluster. U_{SD} means the number of unique instance-pairs which the Cluster Arrangement 1 agrees that two instances of each pair should be in the same cluster but the Cluster Arrangement 2 disagrees. U_{DS} counts unique instance-pairs that have an opposite situation of U_{SD}.

The value of WK is between 1.0 (which means two clustering arrangements are exactly the same) and -1.0 (which means the two clustering arrangements are totally different). A WK score of zero stands for the expected level of agreement between two totally random clustering arrangements. Suggested interpretations of WK scores are listed in Table 4 [Viera and Garrett, 2005].

Table 4: The range of the WK score

Weighted Kappa	Agreement
$-1.0 \le WK \le 0.0$	Very Poor
$0.0 < WK \le 0.2$	Poor
$0.2 < WK \le 0.4$	Fair
$0.4 < WK \le 0.6$	Moderate
$0.6 < WK \le 0.8$	Good
$0.8 < WK \le 1.0$	Very Good

2.5 Summary

In this chapter, firstly we introduced the background of individual clustering methods and Ensemble Clustering methods. Two conclusions can be summarised as follows: 1) different individual clustering methods have different drawbacks and biases; 2) Ensemble Clustering is an efficient way to reduce effects of biases of individual clustering methods. Secondly, an existing Ensemble Clustering method, Consensus Clustering (CC), was introduced as the foundation of understanding our novel Ensemble Clustering methods. Finally, two heuristic optimisation methods (SA and ACO) and a clustering validation index (WK) were introduced in detail. SA and ACO are used to be integrated with our novel Ensemble Clustering methods to achieve the clustering information combination and heuristic search, and WK will be used to evaluate the quality of clustering results generated by our methods.

In the next chapter (Chapter 3), our novel method Multi-Optimisation Consensus Clustering (MOCC) will be introduced in detail. MOCC employs the heuristic optimisation method Simulated Annealing (SA) to achieve its multi-optimisation heuristic search.

3

Chapter 3: Multi-Optimisation Consensus Clustering

3.1 Introduction

Ensemble Clustering has been suggested to tackle biases of individual clustering methods [Strehl and Ghosh, 2003], [Topchy *et al.*, 2005]. It aims to combine results of a number of clustering algorithms to improve the clustering accuracy [Jain *et al.*, 1999; Berkhin, 2002; Xu and Wunsch, 2005], [Lv *et al.*, 2006], [Strehl and Ghosh, 2003], [Berkhin, 2002], [Topchy *et al.*, 2005]. However, to develop an effective Ensemble Clustering method is still a challenge. For example, the Consensus Clustering (CC) algorithm, developed by Swift *et al.* [Swift *et al.*, 2004], is an existing heuristic optimisation-based Ensemble Clustering method. Hirsch *et al.* [Hirsch *et al.*, 2007] compared CC with six different Ensemble Clustering methods developed by Hornik [Hornik, 2005], and results show that CC achieved comparable or even better performance than those six ensemble methods. Nevertheless, CC seeks for optimal clustering solutions by only maximising agreement fitness, which is calculated by an objective function. This objective function combines information of input clustering arrangements, and depends fully on the input clusterings, so that final results would be bad if most of the input clusterings had low accuracies. It is a common difficulty among existing heuristic optimisation-based Ensemble Clustering methods. In order to overcome this difficulty, an advanced Ensemble Clustering method is introduced in this chapter.

Over the past several years, extensive studies have been conducted in the area of multi-objective optimisation and clustering methods. Based on experience of developing multi-objective optimisation methods, we present an advanced consensus clustering algorithm called Multi-Optimisation Consensus Clustering (MOCC), which evaluates not only the agreement fitness but also internal clustering characteristics of data sets to enhance the clustering accuracy. First of all, MOCC exploits an optimised agreement separation criterion to produce a Weighted Agreement Matrix (WAM). And then a Probability Based Solution Generator (PBSG) is developed to create candidate solutions. Based on the WAM and PBSG, finally, a Multi-Optimisation structure is adopted to produce final results. We demonstrated MOCC on fifteen different data sets and compared it with the original CC algorithm. Results of our experiments reveal that MOCC clearly performs better than CC.

This chapter is organised as follows: Section 3.2 describes related works; Section 3.3 details the framework of the MOCC algorithm; all experimental data sets are presented in Section 3.4, followed by experimental results and discussion in Section 3.5; finally in Section 3.6, we draw some conclusions.

3.2 Related Work

3.2.1 Separation Score Evaluation Criterion

The Homogeneity and Separation criterion, which was proposed by Shamir and Sharan, is one of the widely-applied cluster validation indices [Chen *et al.*, 2002]. Homogeneity and Separation are relative to each other. Homogeneity is defined as the average distance between each pattern and the centre of the cluster the pattern belongs to. It reflects the inside compactness of clusters. Separation is defined as the weighted average distance between cluster centres. It reflects the outside distance between clusters. Increasing the Separation score or decreasing the Homogeneity score suggests an improvement in the clustering results [Chen *et al.*, 2002]. According to above characteristics of Homogeneity and Separation and the cost of computation, we adopt Separation Score to evaluate the intrinsic clustering characteristics for the optimisation search in MOCC. Equation (3.1) displays the way of calculating the Separation Score *S*.

$$S = \frac{\sum_{i \neq j} N_{CL_i} N_{CL_j} D\left(Cent_i, Cent_j\right)}{\sum_{i \neq j} N_{CL_i} N_{CL_j}} \tag{3.1}$$

N_{CL_i} indicates the number of instances in the *i*th cluster. The $D(Cent_i, Cent_j)$ is the distance function that calculates the distance between two cluster centres $Cent_i$ and $Cent_j$. Euclidean Distance is adopted in the MOCC algorithm.

3.2.2 Correlation Index

Maximising the value of the objective function to seek more accurate clustering results (i.e. the results have higher WK scores) is the basic principle of CC and MOCC. Therefore the correlation of the function value and the WK score is meaningful for CC and MOCC (the correlation definitions of CC and MOCC are described in Section 3.5.2, and they are different from each other). In this chapter, we employ the Pearson Product-Moment Correlation Coefficient (PMCC) [Snedecor and Cochran, 1980] to evaluate the correlation. PMCC denotes the degree of the linear correlation between two variables. The range of its values is from -1 to +1; where -1 means these two variables have totally opposite linear correlation, +1 indicates the two variables have the best positive linear correlation [Snedecor and Cochran, 1980]. For CC and MOCC, we expect the PMCC value to be as close to +1 as possible.

3.3 The Multi-Optimisation Consensus Clustering algorithm (MOCC)

The MOCC algorithm consists of four key components, which are the weighted agreement matrix, the fitness function, the probability based solution generator, and the multi-optimisation section. Each of these components is detailed in the following subsections.

3.3.1 The Weighted Agreement Matrix and the Agreement Fitness Function

The weighted agreement matrix is generated by the agreement threshold β and the agreement matrix A. The β is used to evaluate the agreement of clustering each pair of instances. It rewards (or penalizes) a clustering that has an agreement value above (or below) the threshold. MOCC redefines the agreement threshold by equation (3.2), where the optimised agreement separation is 0.6 (the original value is 0.5). The details of the experiments and analysis of how 0.6 was derived are described in Section 3.5.1. *Max*(*A*) and *Min*(*A*) indicate the maximal agreement value and the minimal agreement value respectively in the agreement matrix *A*.

$$\beta = [Max(A) - Min(A)] \times 0.6 + Min(A) \tag{3.2}$$

$$A' = A - \beta \tag{3.3}$$

Equation (3.3) shows the definition of the weighted agreement matrix A'. It means that the A' is given by subtracting β from each element of *A*. The leading diagonal elements of A' will never be used, and we set them to be zeros.

The agreement fitness function is shown as equation (3.5), where $f'(C_i)$ is defined by equation (3.4).

$$f'(C_i) = \begin{cases} \sum_{k=1}^{S_i-1} \sum_{q=k+1}^{S_i} A'_{C_{ik}C_{iq}}, & S_i > 1 \\ 0, & otherwise \end{cases} \tag{3.4}$$

$$f(C) = \sum_{i=1}^{m} f'(C_i) \tag{3.5}$$

The function $f'(C_i)$ calculates the agreement fitness of the *i*th cluster of the clustering arrangement *C*. The variable *m* is the number of clusters in *C*. Each $A'_{C_{ik}C_{iq}}$ indicates the corresponding weighted agreement value, which related to the instances *k* and *q* in the *i*th cluster C_i, in the weighted agreement matrix A'. The variable S_i denotes the size of the *i*th cluster (i.e. the number of instances in the *i*th cluster).

3.3.2 The Probability based Solution Generator

In the original CC algorithm, the solution generator has three different operators, which are Move (moving an instance form one cluster to another cluster), Merge (merging two random clusters to be one cluster), and Split (splitting one cluster to be two clusters) [Swift *et al.*, 2004]. Each of these three operators is randomly chosen to generate a clustering solution (i.e. each operator can be chosen with an equal probability).

The proposed MOCC algorithm also adopts these three operators for the solution generator but with different probabilities. The purpose of the Move operator is to find better solutions by estimating the neighbours of current solutions. The Merge and Split operators are used for avoiding results to be stuck at local maximums. The total probability of these three operators is 1. We set the operation probability of Move to be 80%, Merge to be 10% and Split to be 10%.

The Simulated Annealing algorithm in CC starts from a random clustering solution to seek the optimal solution. For analysing high-dimensional data sets, it might cost much time to get convergence if the random solutions were very bad. Therefore we adopt a well-known clustering algorithm, K-means, to generate initial solutions for SA in MOCC. The number of clusters for K-means can be roughly inferred from the input clustering matrix. In general, the adoption of K-means aims to keep the quality of initial solutions at a relative good level.

3.3.3 The Multi-optimisation Section

MOCC implements SA to generate more accurate clustering results by a multi-optimisation framework. This framework integrates the Agreement Fitness Evaluation (AFE) with the Clustering Quality Evaluation (CQE) to evaluate solutions during the optimisation search of SA. The purpose of the AFE is to seek the solutions that have the maximal agreement fitness. The CQE uses Separation Score (SS) as a criterion to evaluate the clustering quality. With the same number of clusters, the value of SS is expected to be as high as possible. During the optimisation search of SA, candidate solutions will be accepted if they are eligible for both criteria otherwise will be discarded by a probability based on the annealing temperature.

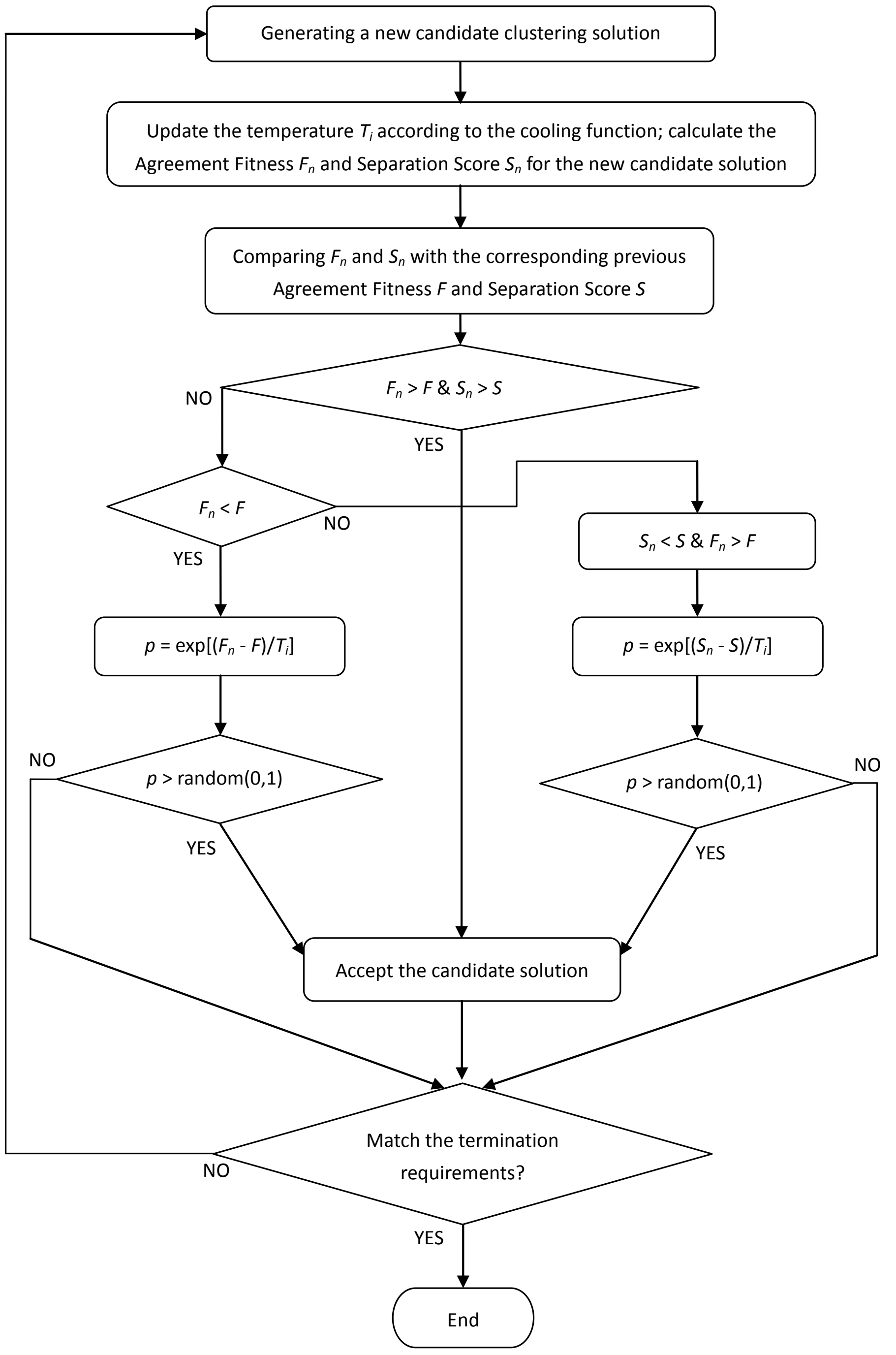

Generating a new candidate clustering solution
Update the temperature T_i according to the cooling function; calculate the Agreement Fitness F_n and Separation Score S_n for the new candidate solution
Comparing F_n and S_n with the corresponding previous Agreement Fitness F and Separation Score S
$F_n > F$ & $S_n > S$
NO
YES
$F_n < F$
NO
YES
$S_n < S$ & $F_n > F$
p = exp[(F_n - F)/T_i]
p = exp[(S_n - S)/T_i]
NO
p > random(0,1)
YES
p > random(0,1)
NO
YES
Accept the candidate solution
Match the termination requirements?
NO
YES
End

Fig. 4: The multi-optimisation section of MOCC.

Fig. 4 illustrates the multi-optimisation section of MOCC. First of all, a new candidate solution is generated by the solution generator. Secondly, the annealing temperature T_i is updated by the cooling function, and then the agreement fitness F_n and separation score S_n of the new candidate solution are calculated respectively. Thirdly, we compare F_n with the previous agreement fitness F, and compare S_n with the previous separation score S. Finally, the candidate solution will be accepted or discarded according to the result of the comparison. The above process will be repeated until the termination conditions are matched. Three different operations are available for dealing with new candidate solutions. If $F_n>F$ and $S_n>S$, the new candidate solution will be accepted directly. If $F_n<F$, the acceptance probability p of the new candidate solution will be calculated according to the left side equation $p=\exp[(F_n-F)/T_i]$ (where T_i denotes the annealing temperature). If $F_n>F$ and $S_n<S$, p will be calculated according to the right side equation $p=\exp[(S_n-S)/T_i]$.

```
Input:  The maximal number of clusters, $K$; A matrix of results of a set of input methods, $Matr$;
        Original Data set Matrix, $X$; Initial Temperature, $T_0$; Number of Iterations, $N_M$; Cooling
        Rate, $cool$.
(1)  Construct the Agreement Matrix, $A$
(2)  Setup the agreement threshold $\beta$ according to Equation (3.2)
(3)  Generate the weighted agreement matrix  $A'$
(4)  Use the K-means algorithm to generate the initial clustering arrangement $C$ for SA
(5)  Calculate the Separation score, $S$, according to Equation (3.1)
(6)  Calculate the Agreement Fitness function (according to Equation (3.5)) to obtain $f$.
(7)  $T_i = T_0$
(8)  For  $i = 1$ to $N_M$
(9)       Use Solution Generator to produce a new arrangement $C_{new}$
(10)      Re-calculate Equation (3.5) to obtain a value $f_{new}$;  and re-calculate Equation (3.1) to
          obtain a value  $S_{new}$
(11)      If  $f_{new} < f$  or  $S_{new} < S$
(12)          Calculate probability $p$:   {If   $f_{new} < f$,    $p = \exp[(f_{new} - f)/T_i]$;
(13)                                        Else,        $p = \exp[(S_{new} - S)/T_i]$.
(14)                                        End If }
(15)          If  $p >$ random(0,1)
(16)              Accept the new clustering arrangement.
(17)          End If
(18)      Else
(19)          Accept the new clustering arrangement.
(20)      End If
(21)      $T_i = cool \times T_i$
(22) End For
Output:  An optimised clustering arrangement.
```

Fig. 5: THE MOCC ALGORITHM

The key steps of the MOCC algorithm are described in Fig. 5. First of all, a weighted agreement matrix A' is generated based on the Agreement Matrix A and the agreement threshold β (Step 1-3). And then an initial clustering arrangement C is produced for SA (Step 4). After that, the agreement fitness f

and the separation score *S* are calculated for *C* according to equations (3.5) and (3.1) respectively (Step 5-6). Finally, SA uses Probability Based Solution Generator to obtain new candidate arrangements, and implements the multi-optimisation framework to seek an optimal clustering arrangement (Step 7-22). The multi-optimisation framework is described from Step 9 to Step 20.

3.4 Experimental Data Sets

We evaluate our method, MOCC, against 15 different data sets. Since we not only compare the performance between MOCC and CC but also observe the performance of MOCC and CC between different data sets, the sizes of these data sets have been made to be approximately the same. We chose about 100 instances from each of the original data sets. The value of 100 was chosen as it is not too trivially small, but small enough to allow the large number of experiments run in this chapter to complete in a feasible amount of time. For some of the data sets, the instances were randomly chosen from the corresponding original data sets. Some of the data sets were simply formed by the first 100 instances in the original data sets. Because of the stochastic nature of the MOCC and CC methods, we run these methods 10 times, and look at the average performance.

Ten of these data sets, which are Ecoli, Glass, Iris, Lung Cancer, Poker Hand, Soybean, WDBC, Wine, Yeast_2 and Zoo, were downloaded from the UCI Machine Learning Repository database. The information of these experimental data sets is presented in Table 5. The details of the original data sets can be found from the UCI website: (http://archive.ics.uci.edu/ml/datasets.html).

The other five data sets are ASC, Malaria, Normal, VAR and Yeast. The ASC (Amersham Score Card) was introduced and used in the literature [Swift *et al.*, 2004]. It is a set of multiply repeated control element spots, which using the Amersham Score Card to probe on the Human Gene clone set arrays of Human Genome Mapping Project [Swift *et al.*, 2004]. This data set has 108 genes/probe elements, which are clustered into 15 clusters. The Malaria microarray data set

[Bozdech *et al.*, 2003] has fourteen clusters where the genes have the same function within each cluster and different functions between clusters. We choose the first one hundred instances for our experiments. VAR (Vector Auto-Regressive) [Lütkepohl, 2007] is a synthetic data set which was obtained from a vector autoregressive process [Sims, 1980]. The total number of clusters is 60. In each cluster, the number of instances varies from 1 to 60. In this chapter, we choose the first one hundred instances of the VAR data set for our experiments. The Normal data set has the same cluster structure as the VAR data set. The difference is that Normal is generated from multivariate normal distribution. The Yeast data set was used in the literature [Yeung and Ruzzo, 2001]. We choose the first twenty instances from each cluster in the original data set to form the experimental data set. The information of these five data sets is also presented in Table 5.

Table 5: Fifteen different data sets

Data Sets	Instances Chosen from the Original Data Sets	Number of Instances	Number of Attributes	Number of Clusters
ASC	1 - 108	108	23	15
Ecoli	CP(1-12); IM(1-12); PP(1-12); and all instances of the rest clusters	100	7	8
Glass	1-16; 71-86; 147-214	100	9	6
Iris	26 - 125	100	4	3
Lung Cancer	1-32; deleted the attributes with “?”	32	54	3
Malaria	1 - 100	100	48	12
Normal	1 - 100	100	20	14
Poker Hand	1 - 100	100	10	8
Soybean	1-30; 71-140	100	35	7
VAR	1 - 100	100	30	14
WDBC	51 - 150	100	30	2
Wine	51 - 150	100	13	3
Yeast	1-20; 68-97; 203-222; 278-297; 330-349	100	17	5
Yeast_2	1 - 100	100	8	8
Zoo	1 - 101	101	16	7

3.5 Results and Discussion

3.5.1 Optimising Agreement Separation for the Agreement Threshold β

From the definition of the agreement threshold β in equation (3.2), it is clear that the *Max*(*A*) and *Min*(*A*) are constants for a specific agreement matrix. The β is therefore only affected by the agreement separation. In order to seek a comparative optimal agreement separation, we use fifteen data sets to test CC on a number of different agreement separations. The numerical range of the agreement separation is between 0 and 1 (the values 0 and 1 are not included because they have no meaning for β), consequently we analyse a series of agreement separations that varies from 0.1 to 0.9 in steps of 0.1.

The nine agreement separations were tested on fifteen data sets (described in Section 3.4). We use WK to indicate the clustering accuracy, and the results are illustrated in Fig. 6. The solid line with two arrows indicates the range between maximal and minimal WK scores for each agreement separation. The broken line links the mean WK scores for the nine agreement separations. We can see that the agreement separation 0.6 has the highest mean WK score. In addition, PMCC is used to indicate the Fitness-WK correlation for these separations. The mean Fitness-WK correlation values (linked by a broken line) are displayed in Fig. 7. It is clear that the agreement separation 0.4 has the highest mean Fitness-WK correlation. By comparing Fig. 6 and Fig. 7, we note that the separation 0.4 has a much lower mean WK score than 0.6, 0.7 and 0.8, but the mean Fitness-WK correlation of the separation 0.6 is only slightly lower than the one of 0.4. Therefore, 0.6 seems to be a good agreement separation.

In order to validate the inference further, we analyse the distribution of WK scores for each agreement separation. From the definition of WK, it is clear that the agreement of two clustering arrangements is almost perfect when the WK score is greater than 0.8 and very poor when the WK score is less than 0.2 [Viera and Garrett, 2005]. Therefore we count and compare the numbers of WK scores, which are between 0.2 and 0.8, for each separation. Based on Fig. 6, the numbers of WK scores between 0.2 and 0.8 for different separations are listed in

Table 6. The corresponding mean WK scores (which are different from those in Fig. 6) are illustrated in Fig. 8. It is apparent that the agreement separation 0.6 still has the highest mean WK score, which shows that 0.6 is an ideal separation.

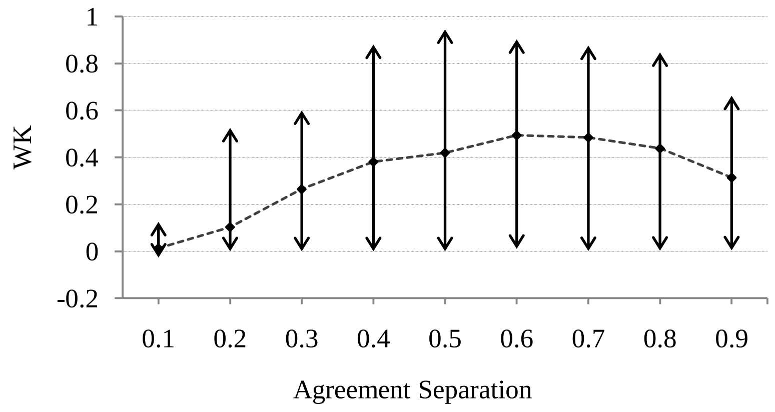


Fig. 6: The WK comparison of nine different agreement separations tested on fifteen data sets.

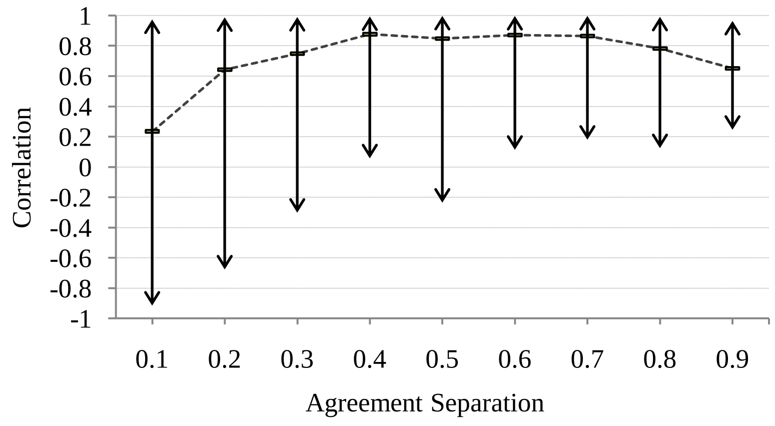


Fig. 7: The Fitness-WK correlation comparison of nine different agreement separations tested on fifteen data sets.

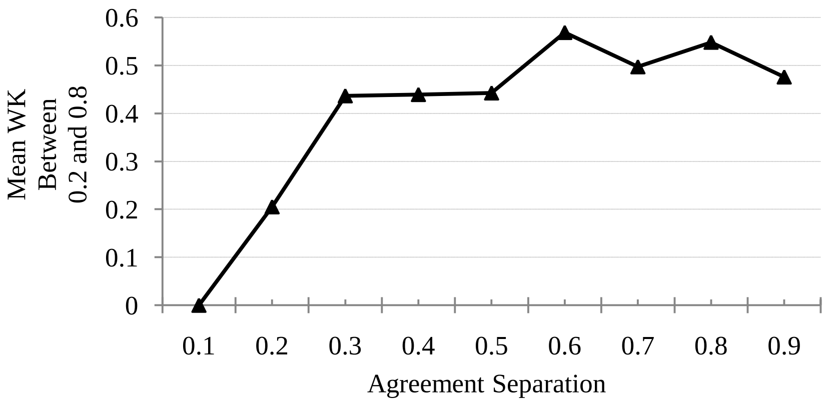


Fig. 8: The comparison of mean WK scores that corresponding to Table 6.

3.5.2 Results Comparison between CC and MOCC

CC was compared with a number of Ensemble Clustering methods implemented within the Clue [Hornik, 2005] R package in the literature [Hirsch *et al.*, 2007]. Therefore, within this chapter, we will compare our novel method, MOCC, against CC only, to simplify the number of experiments being evaluated. The experiments were achieved by testing both CC and MOCC on each of the fifteen data sets respectively. Eight individual clustering algorithms were chosen to generate input clustering matrixes for CC and MOCC. The eight clustering algorithms are as follows: PAM (Correlation); PAM (Euclidean); Affinity Propagation (Correlation); Affinity Propagation (Euclidean); Hierarchical (Average, Correlation); Hierarchical (Average, Euclidean); Hierarchical (Complete, Correlation); Hierarchical (Complete, Euclidean). The Number of Clusters, which is one of the parameters of these algorithms, is set according to Table 5. For the CC and MOCC algorithms, we do not need to set the number of clusters as an input parameter. We analyse the experimental results and compare the performance between MOCC and CC in two aspects.

Table 6: Number of WK scores between 0.2 and 0.8 for each of the nine separations

Agreement Separation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Number of WK Scores Between 0.2 and 0.8	0	4	7	9	8	9	7	10	9

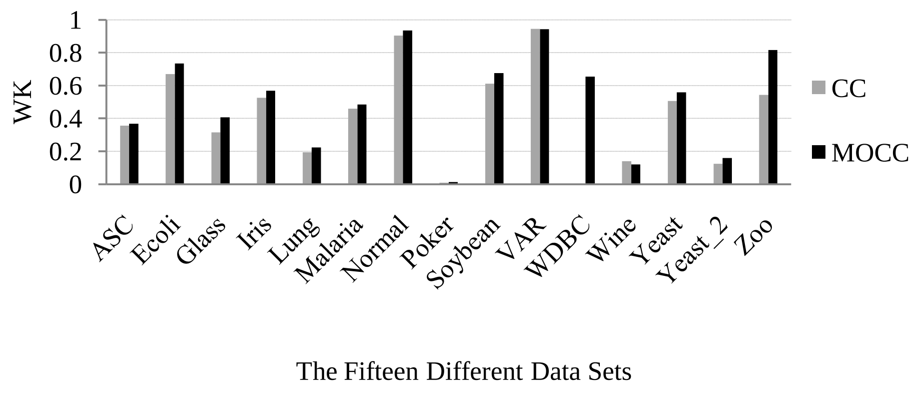


Fig. 9: The WK scores comparison of the results between CC and MOCC tested on fifteen different data sets.

Firstly, we analyse the accuracy of the clustering results. We still use WK to evaluate the clustering accuracy. Fig. 9 displays the WK score comparison between CC and MOCC. In general, we can say MOCC generated better results than CC (especially for the WDBC and Zoo). We take WDBC as an example, the WK score of CC is zero (it means the result has no agreement related to the true clustering arrangement), but MOCC has a score that is over 0.6 (it means the result has a good agreement with the true clustering arrangement). The reason may be the WDBC has a very small number of clusters (2 clusters) but with a very high number of attributes (30 attributes). For the Poker Hand data set, it is interesting to observe that both MOCC and CC have very low WK scores, which means this data set is not suitable for clustering analysis. In order to highlight the overall difference of WK scores between CC and MOCC, we calculated the mean values of WK, which displayed in Fig. 11 (a). It is clear that MOCC has a better mean WK score than CC.

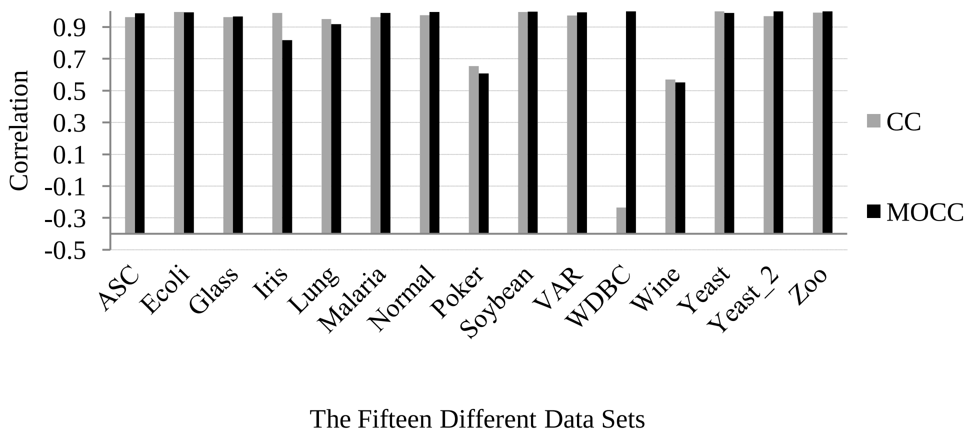


Fig. 10: The correlation comparison of the results between CC and MOCC tested on fifteen different data sets.

Secondly, we use PMCC to evaluate the correlation between the WK score and the value of the objective function. The correlation is expected to be as high as possible so that the algorithm can generate more accurate (higher WK scores) results when maximising the value of the objective function. For CC, the objective function value is the agreement fitness value. For MOCC, we define the objective function value to be the sum of the agreement fitness and the separation score. Fig. 10 illustrates the correlation comparison between CC and MOCC. It is clear that MOCC and CC have similar correlation scores for most of the data sets. However, when we compare Fig. 10 with Fig. 9, it is interesting to see that MOCC has better WK scores than CC if it has higher correlation scores, and MOCC still has better WK scores even its correlation scores are lower than CC for some data sets such as Iris, and Lung Cancer. It means MOCC performed better than CC. For the WDBC data set, Fig. 10 explains why MOCC has a much higher WK score than CC. We also calculated the mean correlation values of CC and MOCC. In Fig. 11 (b), it is apparent that the mean correlation value of MOCC is higher than CC's.

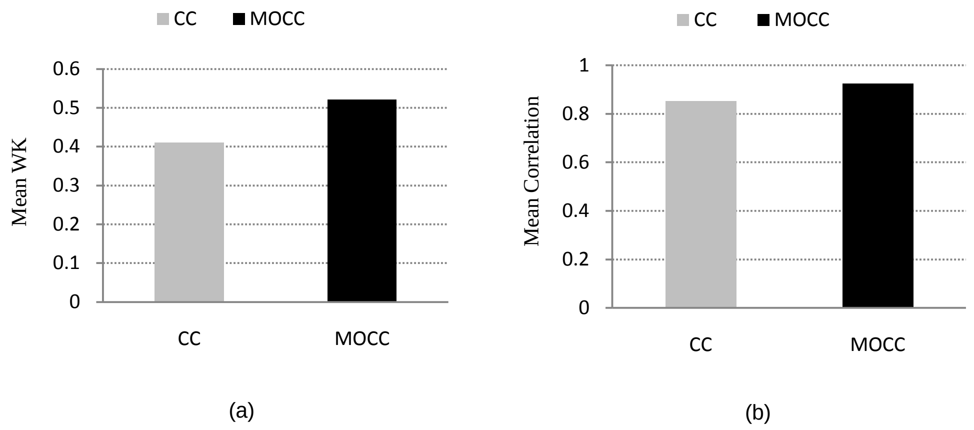


Fig. 11: (a) The comparison between CC and MOCC by the mean WK of the fifteen different data sets; (b) The comparison between CC and MOCC by the mean correlation of the fifteen different data sets.

Following the above results and discussion, it is clear that MOCC is more robust than CC in general. In addition, the most important difference between the two algorithms is that MOCC is not only seeking the solutions with high Agreement Fitness, but also impelling the results as close to the true clustering arrangement as possible by the multi-optimisation framework.

3.6 Summary

We have presented a novel Consensus Clustering framework called MOCC, which uses an optimised Agreement Separation and a Multi-Optimisation structure to enhance the Ensemble Clustering accuracy. Within the Multi-Optimisation structure, the Separation Score (SS) index is combined with the Agreement Fitness Evaluation (AFE) to ameliorate the performance of optimisation, but the combination is not limited to the SS index. The results show that MOCC has better stability of clustering and clearly performs better than original CC.

This chapter has displayed promising results of combining other appropriate clustering validity indexes with AFE to improve the accuracy of clustering. However, the experiments designed in this chapter have two limitations. One is that the data sets used for our experiments are ideal data sets (i.e. there are no noise and outliers in the data). The other is that the sizes of these data sets are almost the same (i.e. around 100 instances), and most of the data sets are subsets of the original data sets. The reason of only choosing subsets is that we need to analyse how MOCC performs for different types of data sets that have the same size. However, using subsets of original data sets may bias results. According to the above two limitations, in future work, we will further test MOCC from the following two aspects: 1) testing MOCC on a larger rang of data sets (using original data sets instead of subsets); 2) testing MOCC on data sets that have noise and outliers.

In addition, from our experiments, we have noted that the computational cost of MOCC is significantly high (the details of analysing computational costs will be described in Chapter 6.3.2). It suggests that MOCC is not capable of analysing large data sets. In order to reduce the computational cost as well as to improve the accuracy of clustering, we have developed another novel Ensemble Clustering method called KACC, which will be analysed in Chapter 5.

4

Chapter 4: The Effect of Cooling Functions for CC and MOCC

4.1 Introduction

Simulated Annealing is an efficient heuristic global optimisation method for combinational problems. The most important feature of SA is that it can escape from local optima to find global optimal solutions [Granville *et al.*, 1994]. This feature distinguishes SA from some traditional search methods such as Hill Climbing [Russell and Norvig, 2003]. Therefore SA has been applied for extensive research areas such as image processing, control system design, machine learning, and data mining [Buckham and Lambert, 1999, Laarhoven and Aarts, 1987, Wang and Li, 2004].

In our previous work, two Ensemble Clustering (EC) methods have been developed to use Simulated Annealing to achieve a global optimal clustering combination. One is the Consensus Clustering (CC) [Swift *et al.*, 2004] algorithm that employs SA to seek optimal solutions that maximise an agreement fitness function. The other is the Multi-Optimisation Consensus Clustering (MOCC) [Li *et al.*, 2009] algorithm (described in Chapter 3) which implements SA to perform a multi-objective optimisation for the clustering combination. MOCC has been compared with CC in the literature [Li *et al.*, 2009], which demonstrates that MOCC achieved better performance than CC on average.

Although many successful applications (such as CC and MOCC) have appeared in the literature due to the simplicity and capability of global convergence of SA, there are some considerable limitations of SA affecting its performance. In theory, global optimisation only can be achieved by an infinite iteration procedure of SA [Kirkpatrick *et al.*, 1983, Granville *et al.*, 1994]. In practice, the only way we can implement SA is to utilise a finite iteration procedure. Therefore numerous efforts have been made to simulate the asymptotic convergence behaviour of SA in order to obtain comparable optimal results within a finite iteration procedure, which we refer to as the Cooling Schedule. In this way, the performance of SA depends heavily on the configuration of the Cooling Schedule. This is the most important limitation of SA. The core component of Cooling Schedule is the cooling function. In order to choose or design a suitable cooling function for a specific problem, it is vital to understand the behaviour of cooling functions.

A variety of cooling functions have been proposed in the literature [Andresen and Gordon, 1993, Hoffmann and Salamon, 1989, Nourani and Andresen, 1998]. Atiqullah [Atiqullah, 2004] proposed a Simple Cooling Schedule, and claimed that it could be more efficient. Hajek [Hajek, 1988] stated some conditions for a logarithmic cooling function to obtain a stable convergence. However, to date, there is no comprehensive analysis that has been reported for analysing a wide range of cooling functions. Furthermore, no attempt has been made in the literature to analyse the effect that cooling functions have on Ensemble Clustering. Therefore this chapter aims to bridge this gap, and offers insights into the behaviour of cooling functions in the context of Ensemble Clustering.

We do not attempt to make the performance comparison between different Ensemble Clustering methods. Instead we shall choose two representative EC methods, CC and MOCC, for our study of cooling functions. Comprehensive experiments have been performed by using these two EC methods to test each of the cooling functions on thirteen different data sets. In this way, the chapter presents the findings for those who are interested in Ensemble Clustering techniques as well as who want to obtain a deep understanding of the behaviour of the cooling functions.

The rest of this chapter is outlined as follows: Section 4.2 gives a brief introduction of ten key cooling functions; Section 4.3 describes designed experiments and corresponding data sets used; results and discussion are presented in Section 4.4, followed by some conclusions drawn in Section 4.5.

4.2 Cooling functions of SA

The cooling function is the core component of the cooling schedule. Some widely used cooling schedules have been studied in the literature. For example, Nourani and Andresen [Nourani and Andresen, 1998] compared five cooling schedules to find which schedule has the minimal entropy production. In fact, the essence of the comparison between different cooling schedules is the comparison between different cooling functions. The most widely used cooling functions are the linear function, the logarithmic function, and the simple exponential function. Luke has listed some other existing cooling functions in the literature [Luke, 2002]. Since some of these cooling functions are very similar to each other, we finally selected ten representative cooling functions (which are listed in Table 7) to study in this chapter.

Table 7: The ten cooling functions

$T(t) = \dfrac{T_0}{log(1+t)}$	(4.1)
$T(t) = T_0 - t\dfrac{T_0 - T_N}{N}$	(4.2)
$T(t) = T_0 - t^A;\ where\ \ A = \frac{ln(T_0 - T_N)}{ln(N)}$	(4.3)
$T(t) = \dfrac{1}{2}(T_0 - T_N)\left(1 + cos\left(\frac{t\pi}{N}\right)\right) + T_N$	(4.4)
$T(t) = T_0\left(\dfrac{T_N}{T_0}\right)^{t/N}$	(4.5)
$T(t) = \dfrac{T_0 - T_N}{1 + e^{3(t-N/2)}} + T_N$	(4.6)
$T(t) = T_0 e^{-Bt^2};\ where\ \ B = \left(\frac{1}{N^2}\right) ln\left(\frac{T_0}{T_N}\right)$	(4.7)
$T(t) = T_0 A^{-[\frac{t}{fN}]^B};$ where $B = ln(\frac{ln(T_0/T_N)}{ln(A)})/ln(\frac{1}{f}),\quad A{=}2, f{=}1/3$	(4.8)
$T(t) = \dfrac{1}{2}(T_0 - T_N)\left(1 - tanh\left(\frac{10t}{N} - 5\right)\right) + T_N$	(4.9)
$T(t) = \dfrac{(T_0 - T_N)}{cosh(\frac{10t}{N})} + T_N$	(4.10)

Note: T_0 is the initial temperature; T_N is the final temperature; N is the number of search iterations for temperature cooling from T_0 to T_N.

We have divided these cooling functions in Table 7 into three groups: a hyperbolic function group, an exponential function group, and a mixed group. The hyperbolic function group contains equations (4.9) and (4.10), where Equation (4.9) is a Hyperbolic Tangent function and Equation (4.10) is a Hyperbolic Cosine function. The exponential function group consists of equations (4.5), (4.6), (4.7), and (4.8). The rest equations (4.1), (4.2), (4.3) and (4.4) are assigned into the mixed group, where Equation (4.1) is a logarithmic cooling function, Equation (4.2) is a linear cooling function, Equation (4.3) is a power cooling function, and Equation (4.4) is a cosine cooling function. Fig. 12 illustrates the curves of the ten cooling functions.

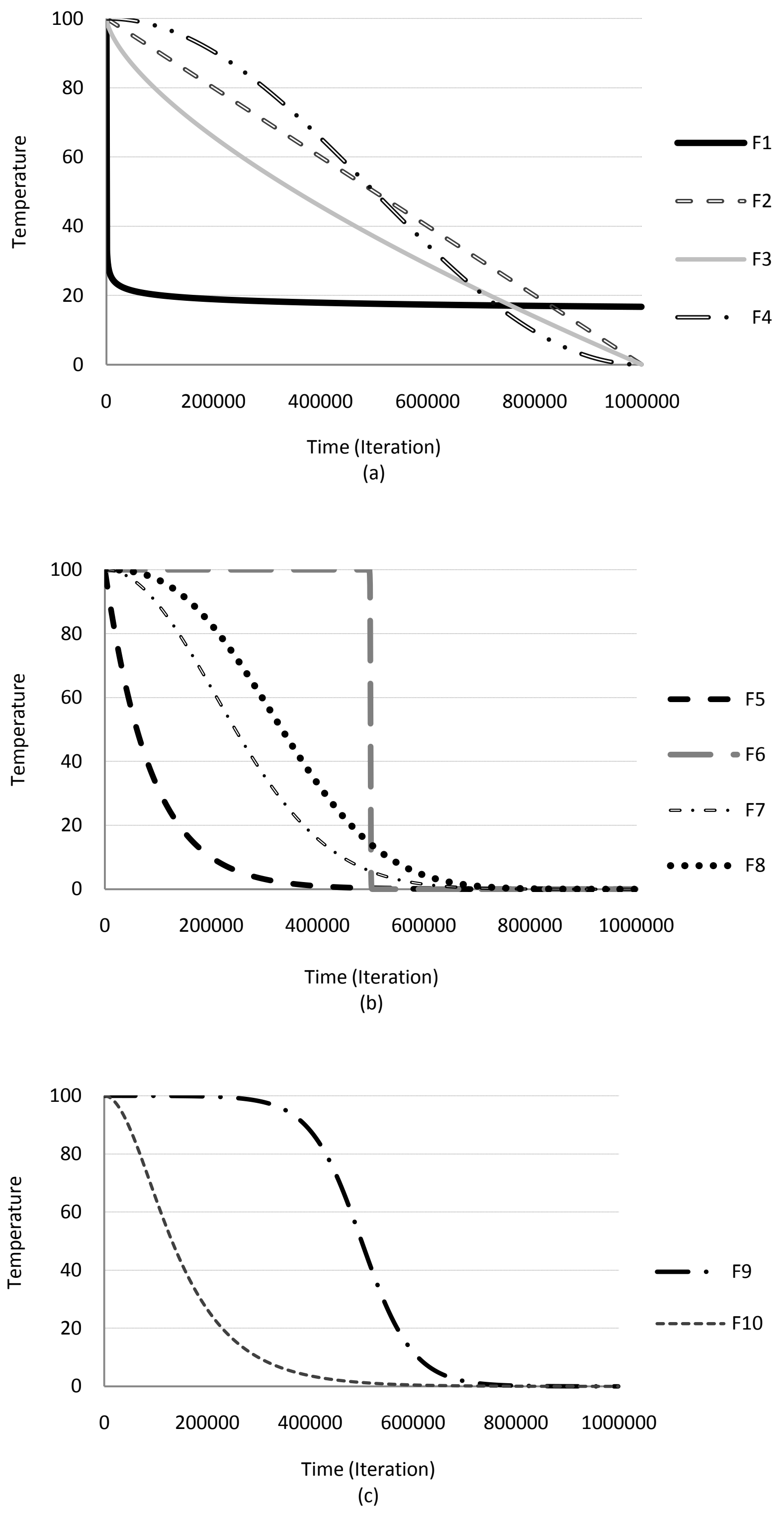


Fig. 12: The curves of the ten different cooling functions that corresponding to Table 7.

4.3 Data sets and experiments

4.3.1 Data sets

We selected thirteen different data sets and divided them into four groups, which are the small, medium, and large data sets groups, and the variant dimensionality data sets group.

The small data sets group has three data sets: ASC (Amersham Score Card), E-coli, and Zoo. The number of instances in each data set is between 100 and 400. The ASC data set was generated from a set of multiply repeated control element spots for probing on the human gene clone set arrays [Swift *et al.*, 2004]. It contains 108 genes that are sorted into 15 clusters. E-coli is a recorded set of Protein Localisation Sites data, and was created by Kenta Nakai in 1996 [Asuncion and Newman, 2007]. There are a total of 336 instances sorted into 8 clusters in the E-coli data set. The Zoo data set contains 101 instances (animals) with 16 Boolean-valued attributes plus one class attribute [Asuncion and Newman, 2007]. It is a very small data set with 7 clusters but has been used in many research papers.

The medium group has two data sets, which are Normal and Yeast. The Normal data set is a synthetic data set that was generated from a multivariate normal distribution [Swift *et al.*, 2004]. It has a total of 1830 instances with 20 attributes. The total number of clusters is 60, and each cluster was generated from a multivariate normal distribution with a different covariance and mean [Swift *et al.*, 2004]. The Yeast data set was used in the literature [Yeung and Ruzzo, 2001]. It has 1484 instances with 8 attributes. This data set was generated for the analysis of predicting the localisation sites of proteins [Asuncion and Newman, 2007].

The large group also contains two data sets, which are ISOLET (Isolated Letter Speech Recognition) and Letter (Letter Image Recognition Data). ISOLET was created by Fanty and Cole, and used in a paper entitled Spoken Letter Recognition [Fanty and Cole, 1991]. The total number of instances is 7797. The total number of attributes of each instance is 617 in the original data set but we only used the first 10 attributes. There are two reasons. One is that the computation cost would be extremely high if the whole 617 attributes were involved for experiments. Another is that the groups, 1, 2 and 3, are used to study the performance of cooling functions for analysing different sizes of data sets. So we need each of these data sets to have a comparable dimensionality (i.e. a comparable number of attributes). The Letter data set was created by Slate in 1991 [Asuncion and Newman, 2007]. The original data set has a total of 20,000 instances with 16 attributes. In order to have a size comparable with the ISOLET data set, we only use the first 7,000 instances.

The above three groups of data sets have a relative low dimensionality which is less than 30. In order to analyse how cooling functions perform for Ensemble Clustering methods on data sets with different dimensionalities, we created the variant dimensionality data sets group. This group has six data sets created from the VAR (Vector Auto-Regressive) data set and the original ISOLET data set. The six data sets have the same number of instances but with different dimensionalities (i.e. the numbers of attributes), which are 30, 150, and 600 or 617. The VAR data set is a synthetic data set that has the same cluster structure as the Normal data set. It was generated from a vector autoregressive process, and has a total of 60 clusters [Sims, 1980, Lütkepohl, 2007]. The number of instances of these clusters varies from 1 to 60.

The four groups of data sets have been listed in Table 8. The data sets, ASC, E-coli, Zoo, Yeast, ISOLET and Letter, were obtained from the UCI Machine Learning Repository website. More information about these data sets can be found on the UCI website (http://archive.ics.uci.edu/ml/datasets.html).

Table 8: The experimental data sets

	Data Sets	**Number of Instances**	**Instances Chosen from the Original Data Set**	**Dimensionality (Number of Attributes)**	**Number of Clusters**
Group 1	ASC	108	all	23	15
	E-coli	336	all	7	8
	Zoo	101	all	16	7
Group 2	Normal	1830	all	20	60
	Yeast	1484	all	8	10
Group 3	ISOLET	7797	all	10	26
	Letter	7000	1 -7000	16	26
Group 4	VAR	900	1 - 900	30	42
	VAR	900	1 - 900	150	42
	VAR	900	1 - 900	600	42
	ISOLET	900	1 - 900	30	26
	ISOLET	900	1 - 900	150	26
	ISOLET	900	1 - 900	617	26

Note: In the original data sets, we define that the rows of the data matrices as the Instances, and the columns of the data matrices as the attributes of the Instances. The dimensionality means the number of attributes of an instance. For a given data set, all instances have the same dimensionality.

This chapter focuses on analysing behaviours of cooling functions when MOCC and CC perform for different sizes and dimensionalities of data sets. Based on this purpose, we only choose subsets for some data sets. In Table 8, the data sets Letter, VAR and ISOLET are subsets of their original data. It is worth to note that, although choosing subsets may bias clustering results, it does not affect the analysis of behaviours of cooling functions.

4.3.2 Experiments

We designed two groups of experiments to analyse the effect of cooling functions on Ensemble Clustering. The first group of experiments is that CC and MOCC employ each of the ten cooling functions to analyse different sizes of data sets. In this group, the first three groups of data sets (i.e. the small, medium and large data sets groups) have been used to test these ten cooling functions. The second group of experiments is that CC and MOCC employ the ten different cooling functions to analyse the data sets with different dimensionalities. The variant dimensionality data sets group has been used for the second group of experiments. In each of these two groups of experiments, we analyse the performance of different cooling functions from three aspects as follows: the convergence rate of cooling functions, the final value of the objective function, and the accuracy of results.

In order to analyse the behaviour of different cooling functions, we recorded the traces of the objective function values (which are sampled every 2000 iterations) for all experiments achieved by CC and MOCC. In addition, we configured three input parameters of SA for CC and MOCC. These three parameters are the initial temperature T_0, the final temperature T_n , and the number of iterations *Ite*. The settings of T_0 and T_n are fixed for all experiments (where T_0= 100, T_n =0.001). The setting of *Ite* may vary according to different clustering problems. We set the default value of *Ite* to be 1,000,000.

Ensemble Clustering combines the results of input clustering methods to generate a better clustering. In order to analyse how cooling functions perform when the number of input methods is changed, we designed two groups of input clustering methods. One group has 8 input clustering methods. The other group has 16 input methods. These two groups of input methods have been used in both two groups of experiments. Table 9 and Table 10 list the two groups of input clustering methods.

For the clustering methods listed in Table 9 and Table 10, the basic reference information has been given as follows. The Hierarchical and K-means clustering algorithms are the most widely used traditional clustering algorithms. Hierarchical clustering was firstly proposed by Ward in 1963 [Ward, 1963], and the standard K-means algorithm was suggested by Lloyd [Lloyd, 1982]. Mean Shift Clustering (MSC) was proposed by Funkunaga and Hosteler, and can be found in the literature [Fukunaga and Hostetler, 1975]. The original Affinity Propagation (AP) algorithm was proposed by Frey and Dueck [Frey and Dueck, 2007], and the improved version (i.e. the Adaptive AP algorithm) was developed by Wang *et al.* [Wang *et al.*, 2007]. The PAM (Partitioning Around

Medoids) was firstly proposed by Kaufman and Rousseeuw in the literature [Kaufman and Rousseeuw, 1987], and after that they proposed an advanced version CLARA (Clustering Large Applications) in the literature [Kaufman and Rousseeuw, 1990]. The FCM (Fuzzy C-Means) algorithm was proposed by Bezdek *et al.* [1999]. For the details of these clustering algorithms, please refer to the corresponding references.

Table 9: The eight input clustering methods

Method	Distance Matrix	Linkage	Initial Cluster Centre
Hierarchical	Euclidean	Single	N/A
Hierarchical	Spearman	Complete	N/A
Hierarchical	Correlation	Average	N/A
Hierarchical	Jaccard	Weighted	N/A
K-means	Squared Euclidean	N/A	Sample
K-means	Correlation	N/A	Uniform
K-means	Cosine	N/A	Sample
K-means	Cosine	N/A	Uniform

Table 10: The sixteen input clustering methods

Method	Distance Matrix	Linkage	Initial Cluster Centre
Hierarchical	Euclidean	Single	N/A
Hierarchical	Spearman	Complete	N/A
Hierarchical	Correlation	Average	N/A
Hierarchical	Jaccard	Weighted	N/A
K-means	Squared Euclidean	N/A	Sample
K-means	Correlation	N/A	Uniform
K-means	Cosine	N/A	Sample
K-means	Cosine	N/A	Uniform
Mean Shift Clustering	N/A	N/A	N/A
Affinity Propagation (Original)	Euclidean	N/A	N/A
Affinity Propagation (Original)	Pearson	N/A	N/A
Affinity Propagation (Adaptive)	Euclidean	N/A	N/A
Affinity Propagation (Adaptive)	Pearson	N/A	N/A
PAM	Euclidean	N/A	N/A
FCM	N/A	N/A	N/A
CLARA	Euclidean	N/A	N/A

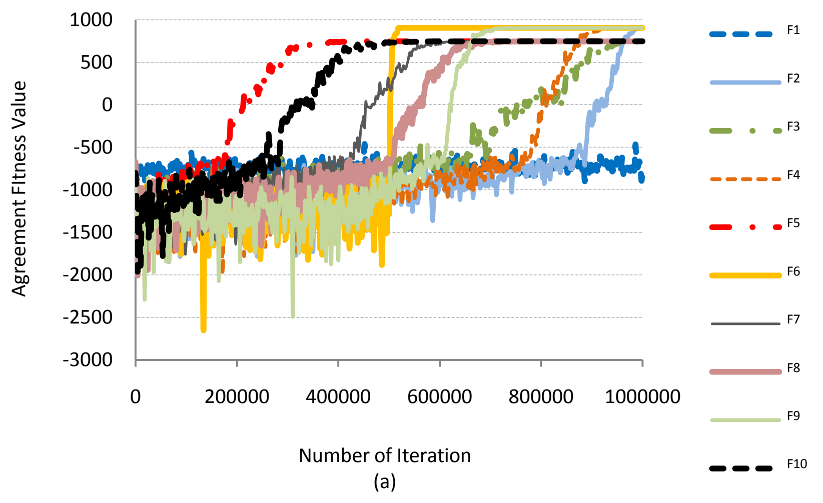
4.4 Results and discussion

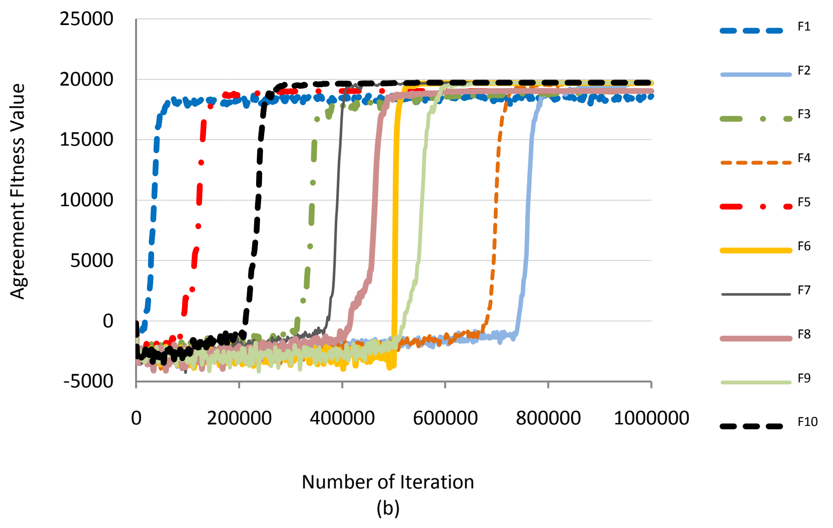
We discuss the results based on the two groups of experiments mentioned in Section 4.3.2. One is that CC and MOCC employ different cooling functions to be tested on the data sets with different sizes, and the other is that CC and MOCC use different cooling functions to be tested on the data sets have the same size but with different dimensionalities. For each group of experiments, we discuss the results from three aspects, which are the convergence rate of cooling functions, the final values of objective functions, and the accuracy of results. For the values of objective functions, in MOCC, it should be the sum of the agreement fitness value and the separation score (because MOCC adopts two evaluation criteria). In order to simplify the description, we use Evaluation Score to denote the sum for MOCC.

4.4.1 The first group of experiments: tested on data sets with different sizes

In this group, each of the ten cooling functions was tested by CC and MOCC on the first three groups of data sets (i.e. the small, medium, and large data sets groups). First of all, we analyse the convergence rate of the ten cooling functions. We need to answer the following questions. Have they converged after the defined number of iterations? How are the convergence speeds of the cooling functions?

The small data sets group was firstly used to test the cooling functions. The number of iterations is default, which is set to be 1,000,000 as described in Section 4.3.2. The sample series traces (sampled during the search process) have been illustrated in Fig. 13 and Fig. 14. Fig. 13 shows the traces generated by CC, and Fig. 14 shows the traces generated by MOCC. We notice that different cooling functions have different convergence speeds during the search process. The cooling function F5 has the fastest convergence speed, and F1 has the lowest speed which comes along with a concussive sample trace during the whole search process. In Fig. 13 (b) and (c), although the traces of F1 rose very fast at the beginning, they had not converged after 1,000,000 iterations. It means that the results generated by using F1 as the cooling function are not valid.





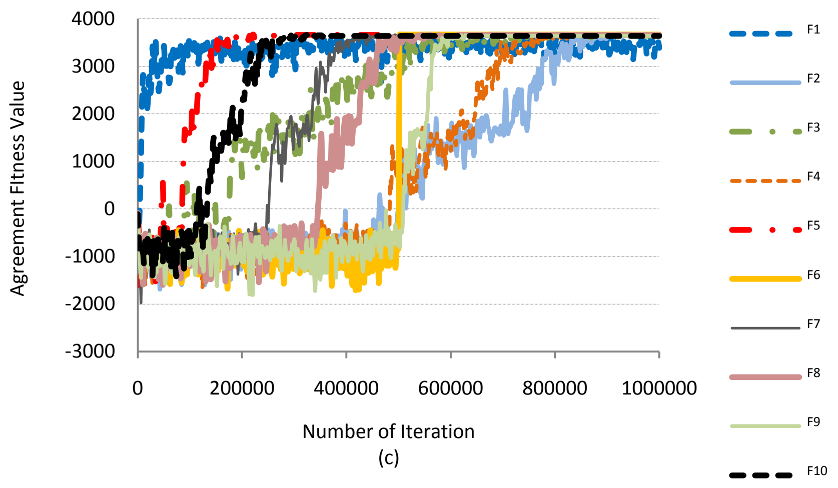
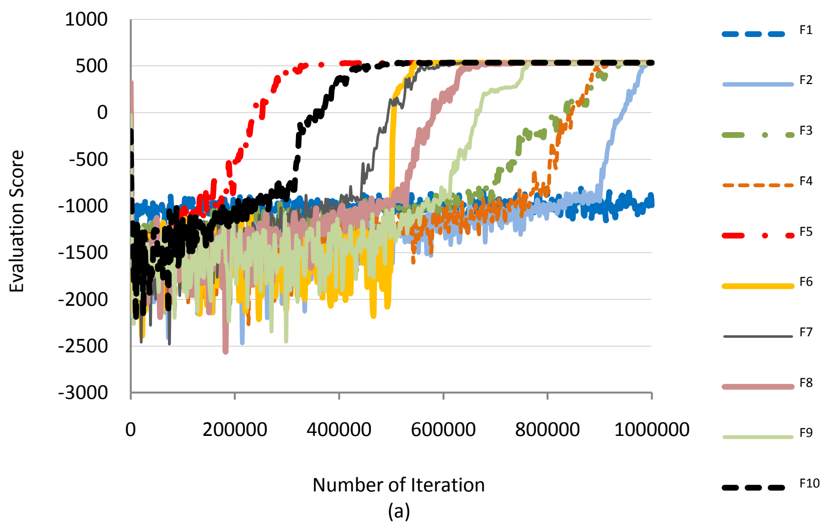
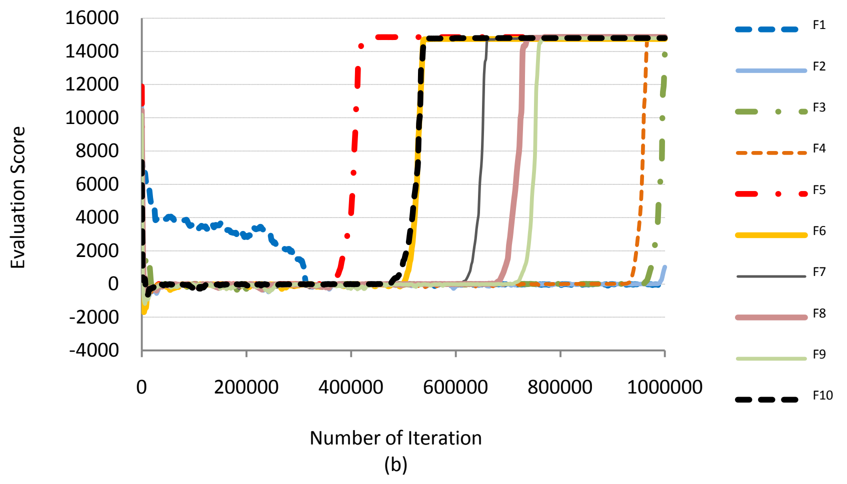


Fig. 13: (a) The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on ASC by CC; (b) tested on E-coli by CC; (c) tested on Zoo by CC.





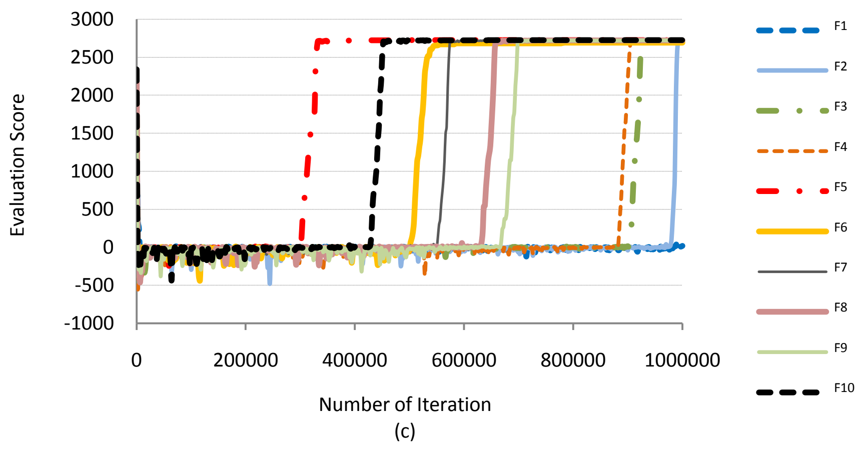


Fig. 14: The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on ASC by MOCC; (b) tested on E-coli by MOCC; (c) tested on Zoo by MOCC.

If we compare Fig. 14 with Fig. 13, we can see that the convergence traces of the cooling functions are shown in a correlative order. If we ignore F3 and F6, other cooling functions can be listed in an order as follows: F5, F10, F7, F8, F9, F4, F2 and F1. We pick up the curves of these eight cooling functions from Fig. 12, and display them again in Fig. 15. It can be seen that, when the temperature is lower than 20, these cooling functions close to the abscissa in an order that is exactly the same as the one shown in Fig. 13 and Fig. 14. This situation suggests that the convergence speeds of cooling functions could be predicted by the speeds of the cooling functions' curves closing to the abscissa.

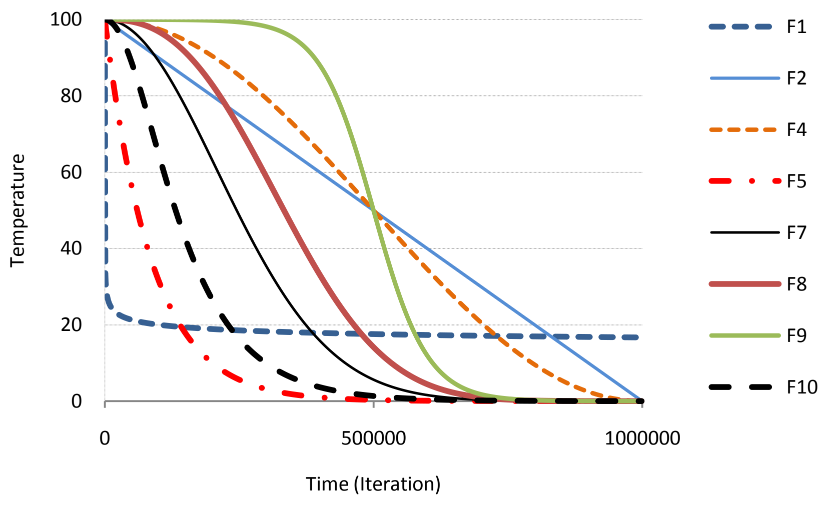


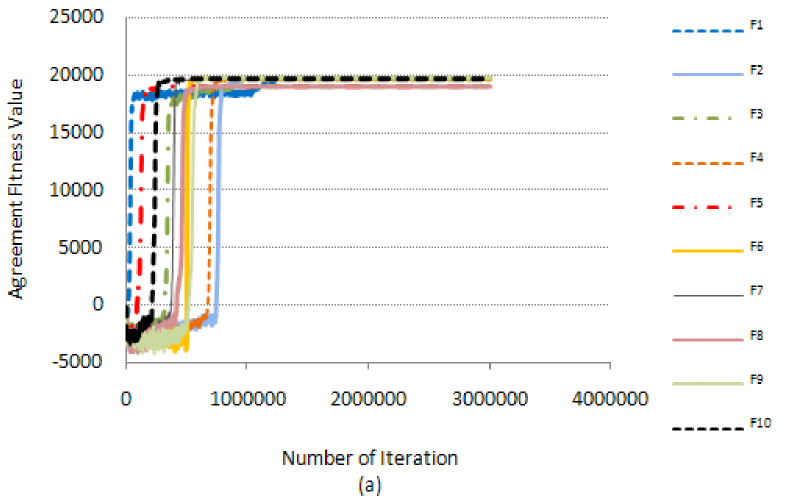
Fig. 15: The curves of eight cooling functions: F1, F2, F4, F5, F7, F8, F9, and F10.

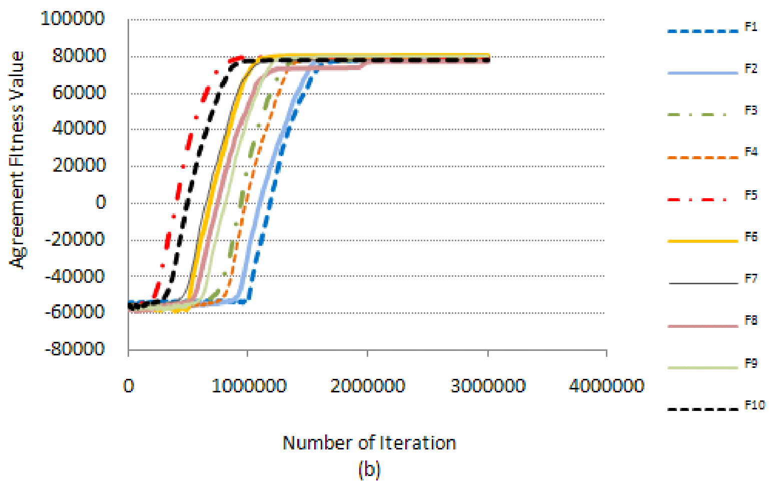
There is a question: could we predict the convergence speeds of cooling functions F3 and F6 by the same way? We use F3 as a representative to study and answer this question. In Fig. 12 (a), the curve of F3 crossed the one of F4 when the temperature decreased to 20, and then closed to the abscissa. In Fig. 13, F3 has a slower convergence speed than F4 for ASC, and a little faster convergence speed than F4 for E-coli and Zoo. In other words, the convergence speed of F3 varies against the speed of F4 for different data sets. By comparing Fig. 12 (b) with Fig. 13 and Fig. 14, it can be seen that F6 has the same situation as F3. Therefore, we may have a summary that, if the curve of a cooling function *M* crosses another one *Q* when the temperature decreases to or is below a very low value (such as 20), the convergence speed of *M* may vary against the one of *Q* for analysing different data sets.

From the above discussion of Fig. 13 and Fig. 14, it is important to note that F1 had never converged in any of these tests. It means that the number of iterations is not big enough for F1 to gain convergence. Therefore, for a certain number of iterations, the results of Ensemble Clustering may be invalid if the EC methods employ a very slow cooling function. For a new cooling function, its convergence speed may be predicted by comparing its algebraic curve with the one of a known cooling function.

In order to obtain valid results for analysing the small data sets, we retested the cooling functions on the small data sets with 3,000,000 iterations. Based on the experience of analysing small data sets, we changed the number of iterations to be 3,000,000 and 6,000,000 for medium and large data sets respectively. In order to compare the performances of cooling functions for analysing different sizes of data sets, we chose one data set from each of the three data set groups, and illustrated their results in Fig. 16 and Fig. 17. Fig. 16 (a) and (b) show the results of CC using ten cooling functions tested on one small data set E-coli and one medium data set Normal respectively. All cooling functions had converged after 1,500,000 iterations in Fig. 16 (a), but after 2,300,000 iterations in Fig. 16 (b). Fig. 16 (c) shows the results of CC being tested using ten cooling functions on one large data set ISOLET with 6,000,000 iterations. It is clear that the cooling functions had not converged until after 5,000,000 iterations.

Fig. 17 shows the results generated by MOCC being tested using the cooling functions on the same three data sets. It also appears that the convergence speeds of cooling functions get slower when the size of data sets becomes bigger. Moreover, in Fig. 16 (c) and Fig. 17 (c), we notice that the final convergent values of F5 and F10 are clearly smaller than others for analysing large data sets. It suggests that, for large data sets, F5 and F10 may converge too early to obtain global optima. The results of CC and MOCC being tested using the cooling functions on other data sets (i.e. ASC, Zoo, Yeast, and Letter) have the similar findings, so we do not list those figures here. It is clear that, if we set the number of iterations to be 1,000,000 for medium and large data sets, all results of the ten cooling functions would be invalid.





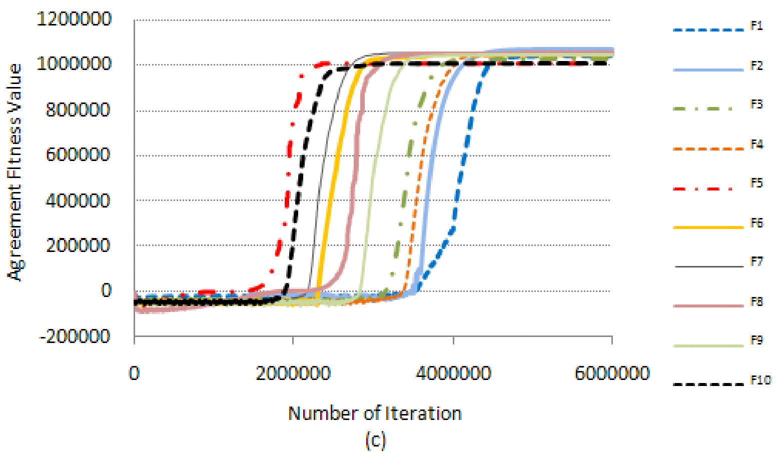


Fig. 16: The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on E-coli by CC; (b) tested on Normal by CC; (c) tested on ISOLET by CC.

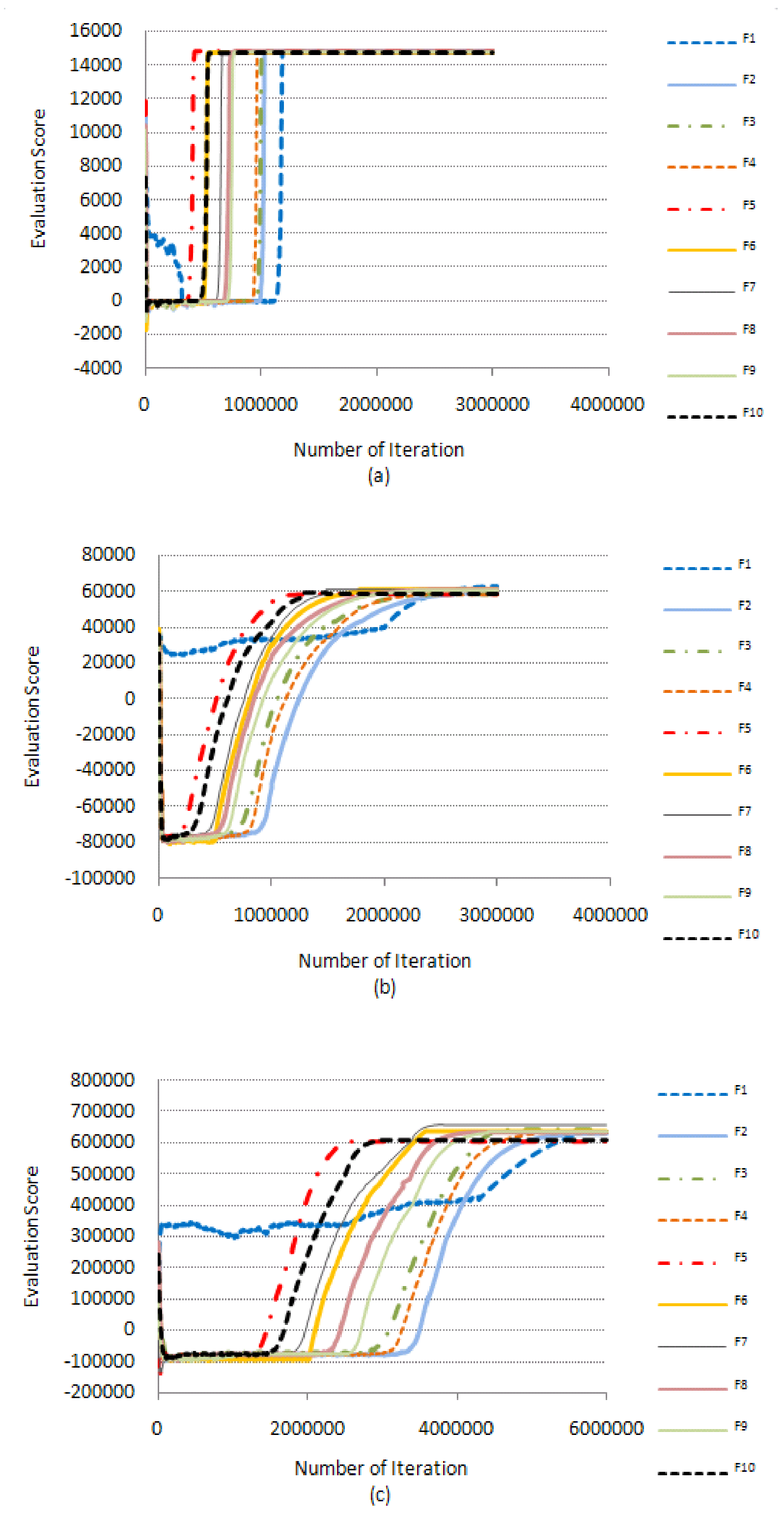
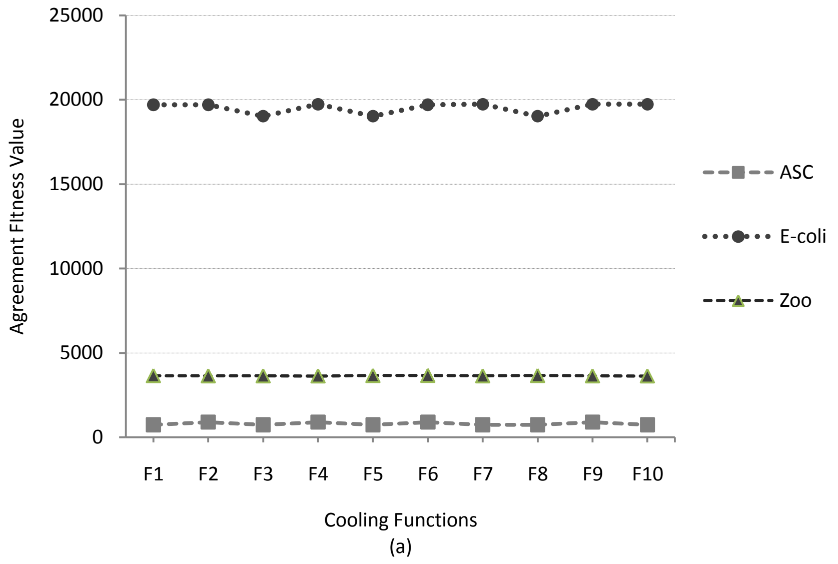


Fig. 17: The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on E-coli by MOCC; (b) tested on Normal by MOCC; (c) tested on ISOLET by MOCC.

By comparing these figures, it becomes clear that, when the size of data sets becomes bigger, the cooling functions require more time to get convergence. In other words, when Ensemble Clustering methods are used to analyse large data sets, an appropriate number of iterations should be set up carefully even for the same cooling function. In addition, the results illustrated in Fig. 16 and Fig. 17 also show that the convergence speeds of cooling functions are related to the speeds of the cooling functions' curves closing to the abscissa, which is similar to those shown in Fig. 13 and Fig. 14.

Secondly, we analyse the final values of objective functions for the ten cooling functions. For the small data sets, we plot the final convergent results in Fig. 18 according to the retested experiments. Fig. 18 (a) displays the results generated by CC, and Fig. 18 (b) shows the results generated by MOCC. It seems that, for analysing small data set, the final convergent values obtained by using different cooling functions have no big difference between each other. Fig. 19 shows the final convergent results for the medium data sets. It has the same situation as in Fig. 18. Fig. 20 shows the final values of objective functions for the large data sets. We found that F5 and F10 have lower values than other cooling functions in both results of CC and MOCC. This situation is the same as those shown in Fig. 16 (c) and Fig. 17 (c).



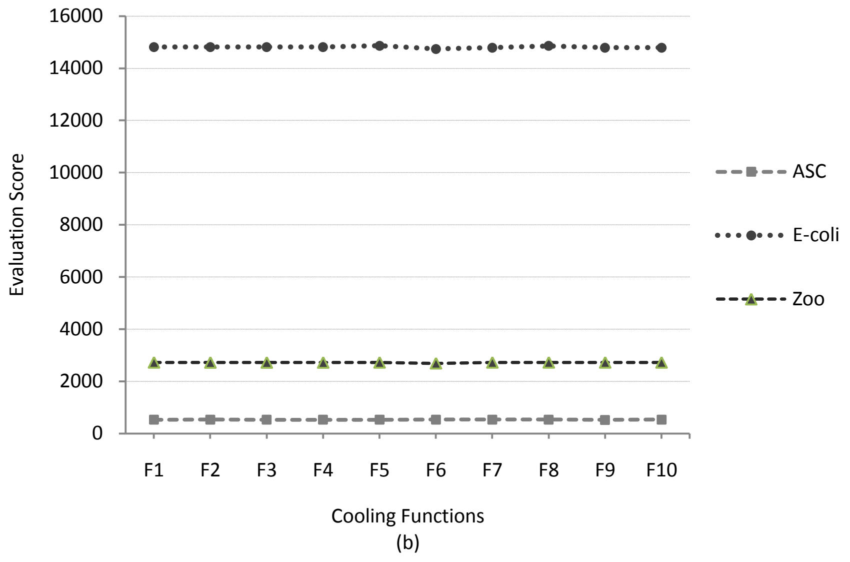
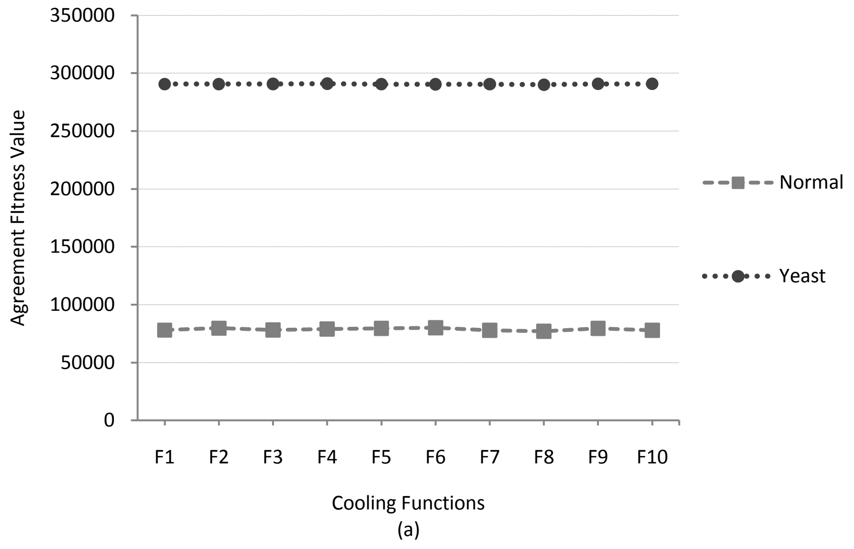


Fig. 18: The comparison of final values of the objective functions between the ten cooling functions (with 8 input clustering methods): (a) tested on the small data sets ASC, E-coli, and Zoo by CC; (b) tested on the same data sets by MOCC.



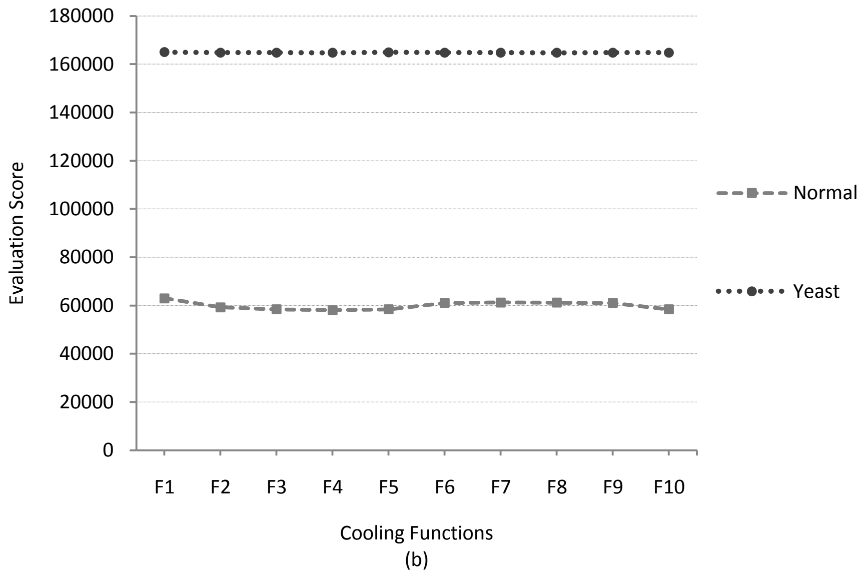
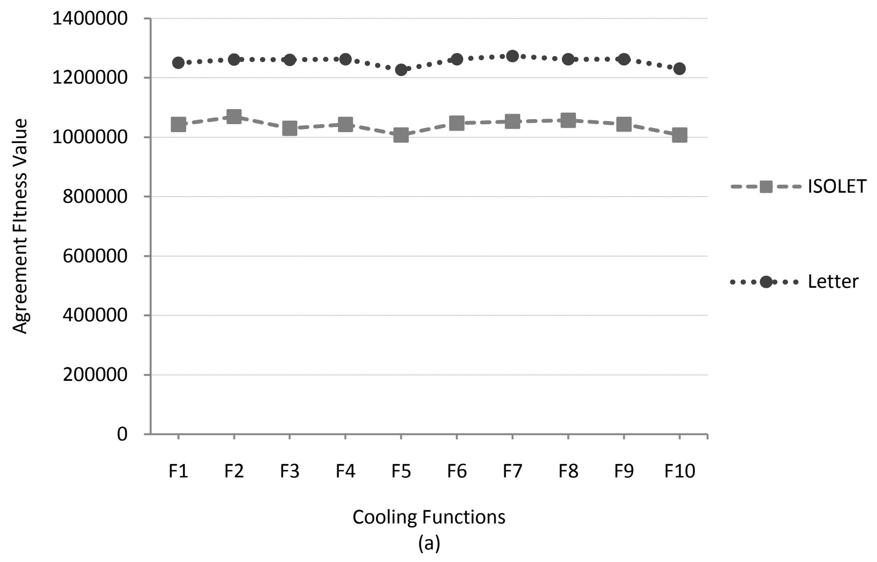


Fig. 19: The comparison of final values of the objective functions between the ten cooling functions (with 8 input clustering methods): (a) tested on the medium data sets Normal and Yeast by CC; (b) tested on the same data sets by MOCC.



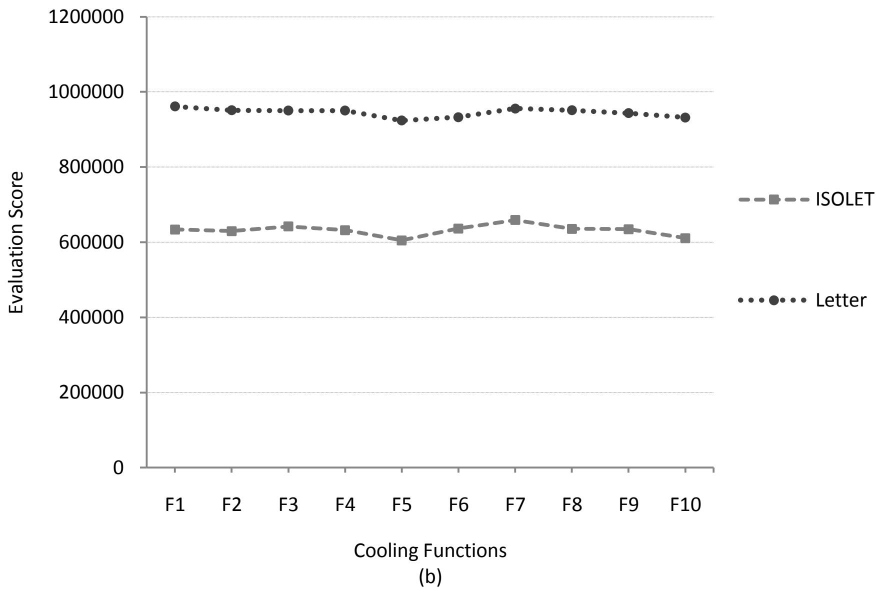


Fig. 20: The comparison of final values of the objective functions between the ten cooling functions (with 8 input clustering methods): (a) tested on the large data sets ISOLET and Letter by CC; (b) tested on the same data sets by MOCC.

Based on the above discussion, we may have the following findings. For small data sets, once the final values of objective functions are convergent, there will be no much difference between these values in general. However, for analysing large data sets, F5 and F10 may not be able to obtain global optima as well as other cooling functions.

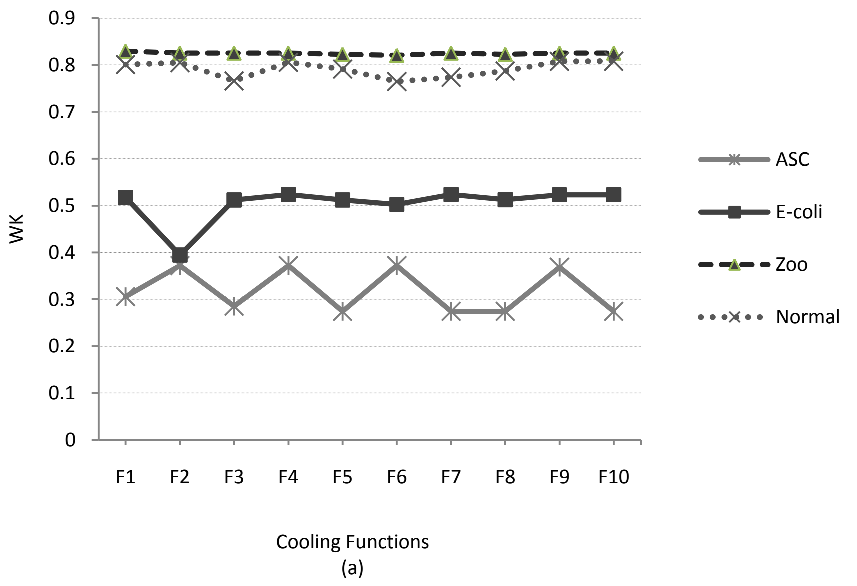
From Fig. 13 to Fig. 20, all illustrated results are generated by using 8 input clustering methods (we call them R8). For the results generated by using 16 input clustering methods, we call them R16. For the convergence rate of cooling functions and the final values of objective functions, the findings obtained from R16 are the same as those from R8. So we do not describe them again for R16.

Thirdly, we analyse the accuracy of the results. Fig. 21 shows the WK scores for all results generated by CC using 8 input clustering methods. By comparing Fig. 21 with Fig. 18, Fig. 19 and Fig. 20, we can see that, although ten cooling functions have similar final convergent values for small and medium data sets, the accuracies of these results differ from each other. For large data sets, the accuracy of the results also cannot be predicted by the objective function values shown in Fig. 20 (a). The similar findings have been obtained from the results generated by MOCC. It means the accuracy of these results depends on the nature of CC and MOCC. In other words, if cooling functions generate the same (or comparable) convergent results for a data set, the accuracy of these results will only depend on the nature of the Ensemble Clustering method.

One question is how the accuracy of results will change if more single clustering algorithms are chosen to be the input methods for the Ensemble Clustering method. We discuss this issue by comparing R8 with R16.

In this chapter, we plot the best case and the worst case of the comparison in Fig. 22 and Fig. 23 respectively. Fig. 22 shows the best case where CC and MOCC are tested using the ten cooling functions on E-coli. In this case, R16 generated by CC and MOCC are better than R8. Fig. 23 displays the worst case where CC and MOCC are tested using the ten cooling functions on Yeast, and shows that

R16 are worse than R8.



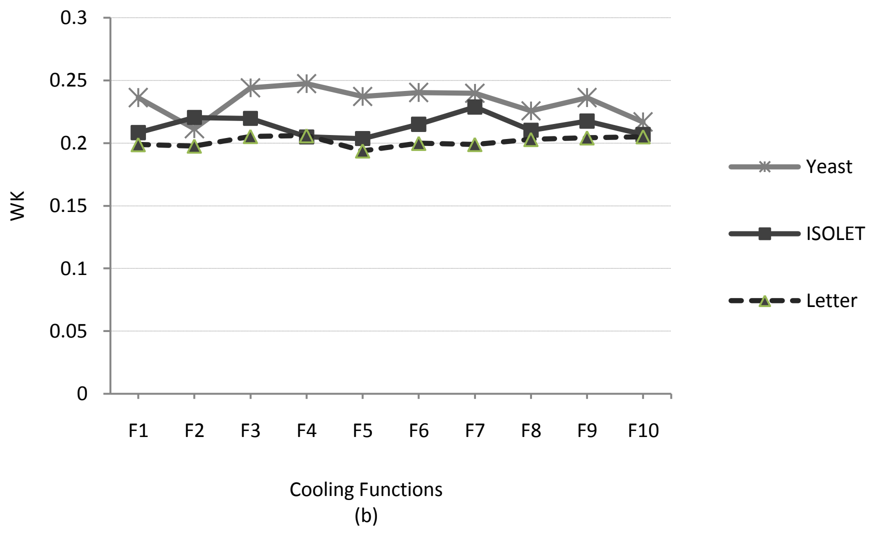
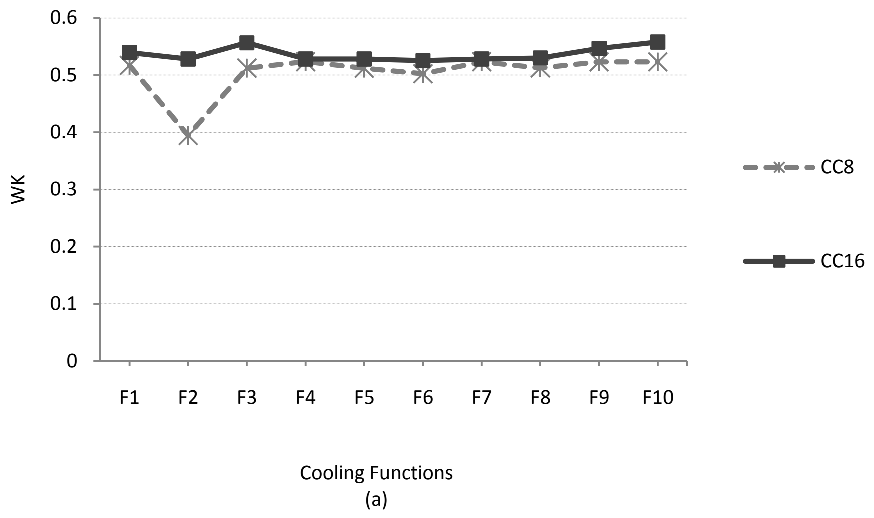


Fig. 21: The comparison of WK scores for the results generated by CC (with 8 input clustering methods): (a) CC being tested using ten cooling functions on ASC, E-coli, Zoo and Normal; (b) CC being tested using ten cooling functions on Yeast, ISOLET and Letter.



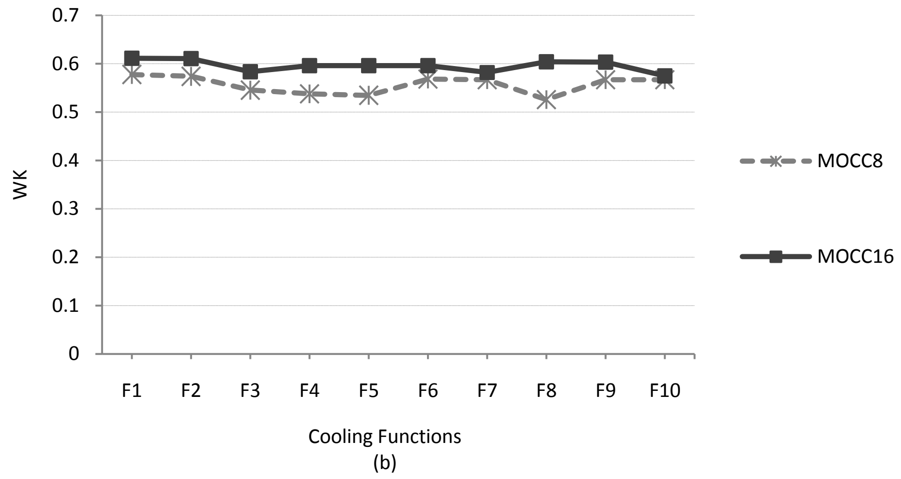
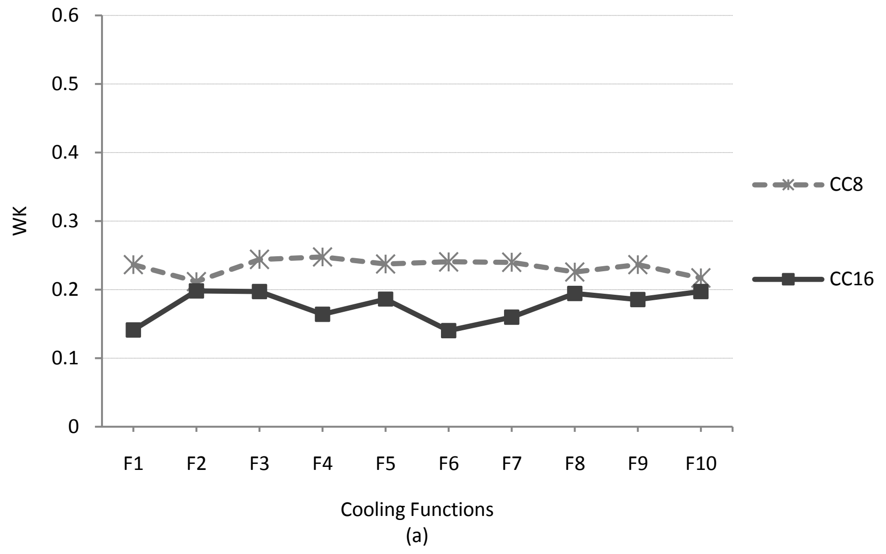


Fig. 22: The best case of comparing the accuracy of the results between using different numbers (8 and 16) of input clustering methods: (a) the WK scores of the results generated by CC analysing E-coli; (b) the WK scores of the results generated by MOCC analysing E-coli.



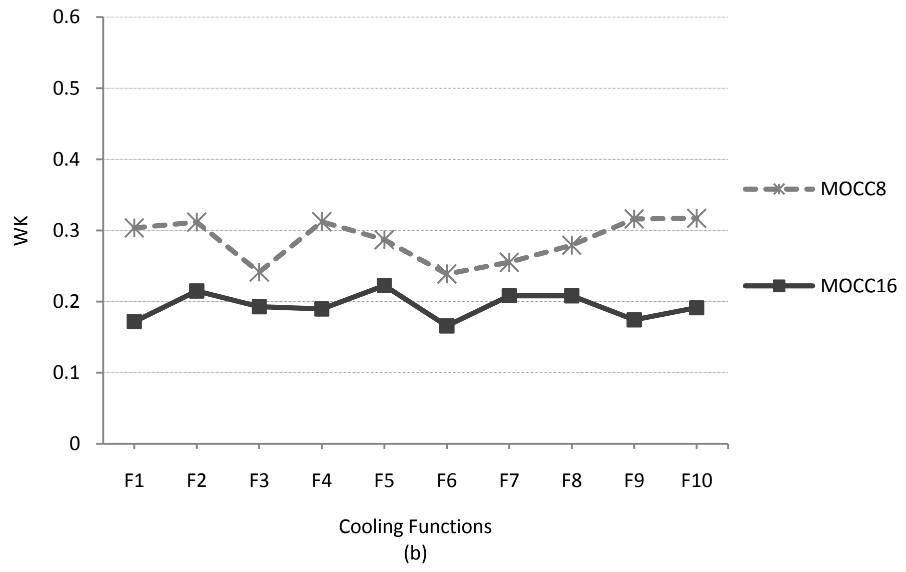
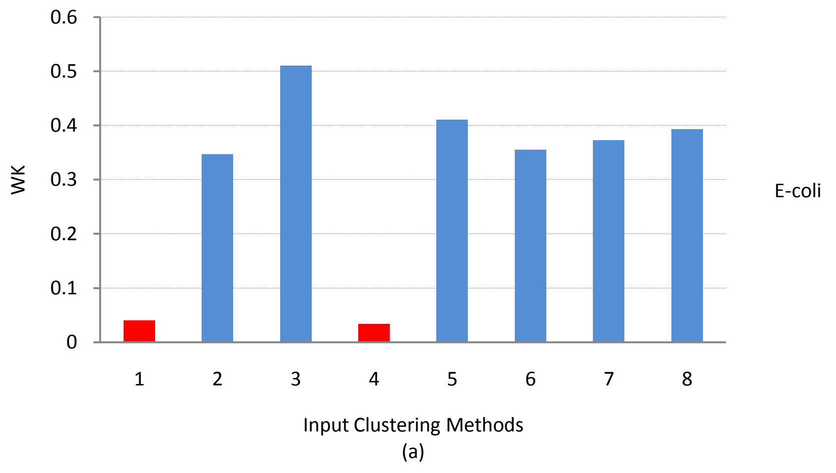


Fig. 23: The worst case of comparing the accuracy of the results between using different numbers (8 and 16) of input clustering methods: (a) the WK scores of the results generated by CC analysing Yeast; (b) the WK scores of the results generated by MOCC analysing Yeast.

In order to find out the reason behind, we check the quality of the input clusters generated by the input clustering methods. Fig. 24 illustrates the clustering accuracy of the two input methods groups (i.e. the 8 input methods group and the 16 input methods group) for E-coli corresponding to Fig. 22. The red bars in

Fig. 24 denote the WK scores that are much lower than the mean of the group. In other words, the red bars mean that the corresponding methods are weak (or not appropriate) for the E-coli data set. It is clear that there are two weak clustering methods in the 8 input methods group (i.e. 25% of the input methods are weak), and four weak clustering methods in the 16 input methods group (i.e. there is also 25% of these input methods are weak). Fig. 25 illustrates the clustering accuracy of the input clusters for the Yeast data set corresponding to Fig. 23. There are three weak input methods in the 8 methods group (i.e. over 37% of these methods are weak), and eight weak methods in the 16 methods group (i.e. 50% of these methods are weak).



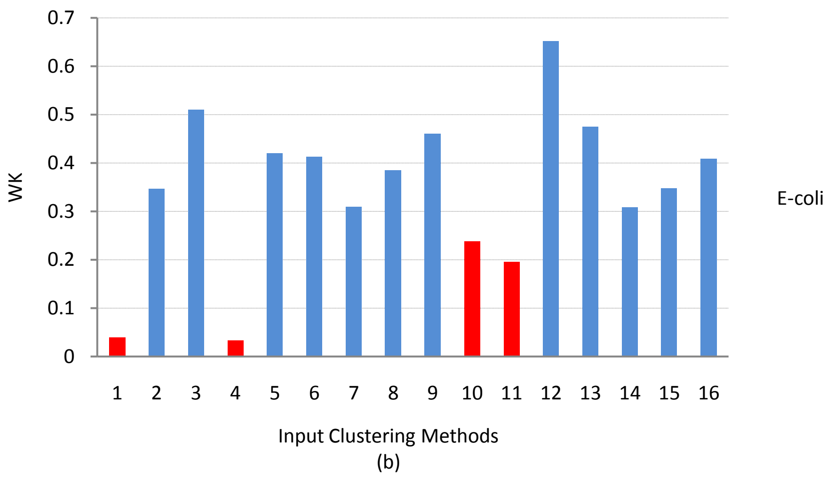
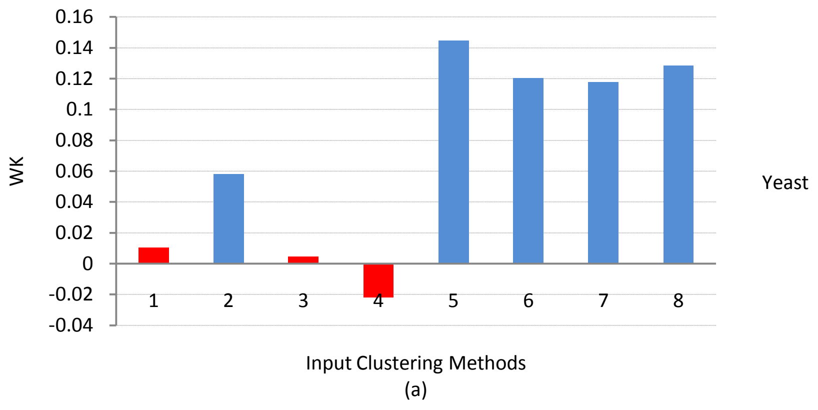


Fig. 24: The clustering accuracy comparison between input methods of the two input clustering methods groups for E-coli corresponding to Fig. 22: (a) the 8 input methods group; (b) the 16 input methods group.

For the E-coli data set, the added input methods increased the mean accuracy without raising the percentage of weak methods. For the Yeast data set, the added input methods not only increased the percentage of weak methods, but also decreased the mean accuracy of the group. By comparing Fig. 25 with Fig. 24, we can observe that, for any of the cooling functions, using more input methods for Ensemble Clustering can only improve the accuracy of results when most of these input methods are appropriate (or most of them generate comparable or better results than the mean) for the given problem.



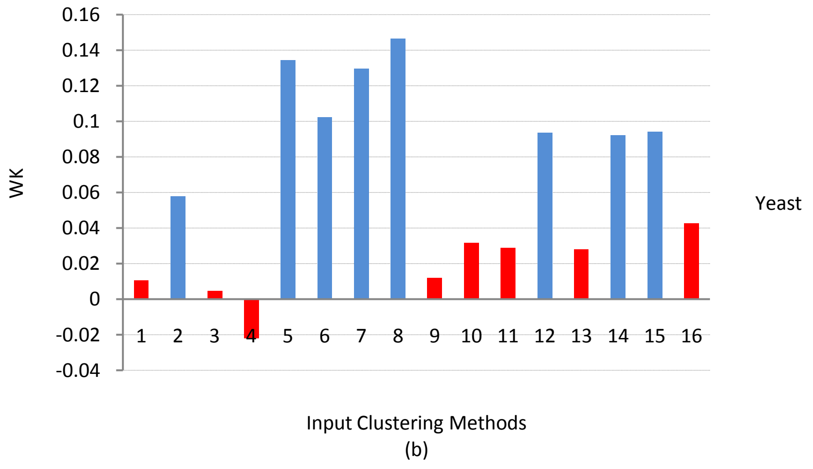
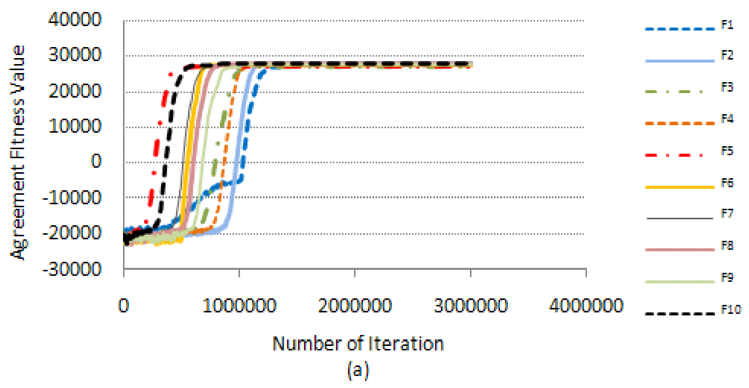


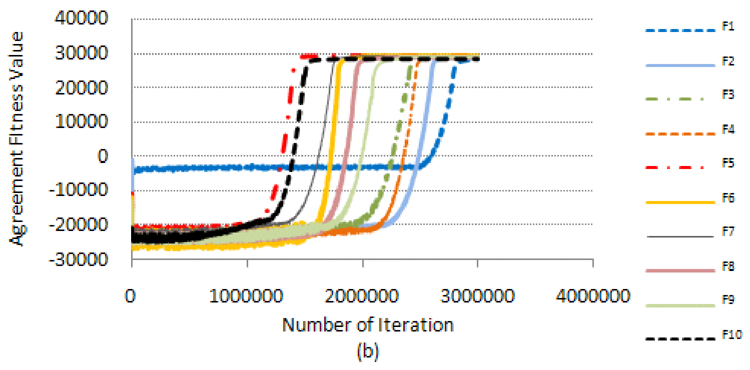
Fig. 25: The clustering accuracy comparison between input methods of the two input clustering methods groups for Yeast corresponding to Fig. 23: (a) the 8 input methods group; (b) the 16 input methods group.

4.4.2 The second group of experiments: tested on data sets with different dimensionalities

In order to analyse the effect of different cooling functions on Ensemble Clustering for analysing data sets with different dimensionalities, we discuss the relevant results in this section. The experiments were tested on the fourth group of data sets, i.e. the variant dimensionality data sets group. We still analyse the results from three aspects (i.e. the convergence rate, final values of objective functions, and the accuracy of results) as discussed in the first experiment group.

First of all, we analyse the convergence rate of the ten cooling functions. Fig. 26 and Fig. 27 show the sample traces generated by CC being tested using ten cooling functions on three different dimensional VAR (900 instances) data sets and ISOLET (900 instances) data sets respectively. After comparing these sample traces with the figures from Fig. 13 to Fig. 17, we have the same findings as those discussed in the first experiment group for the convergence speeds of cooling functions. The finding is that convergence speeds of cooling functions are relevant to the speeds of the cooling functions' curves closing to the abscissa.





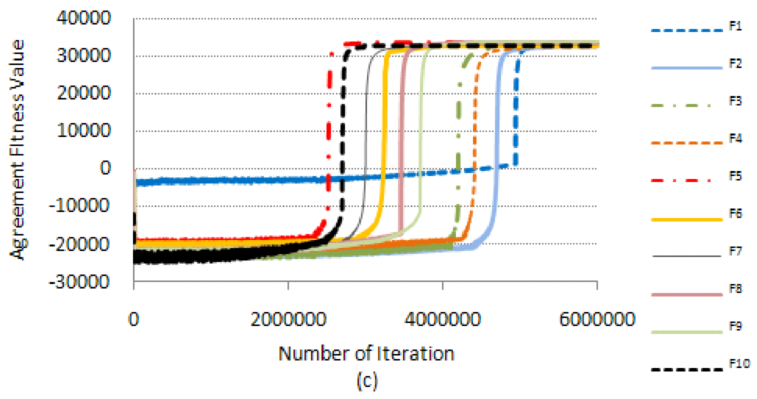
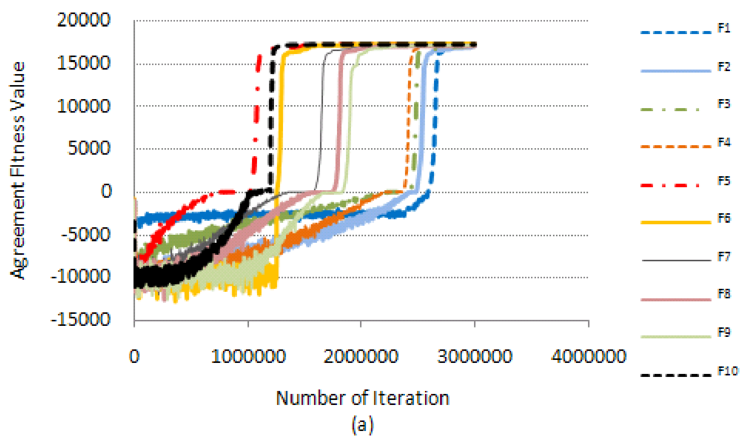
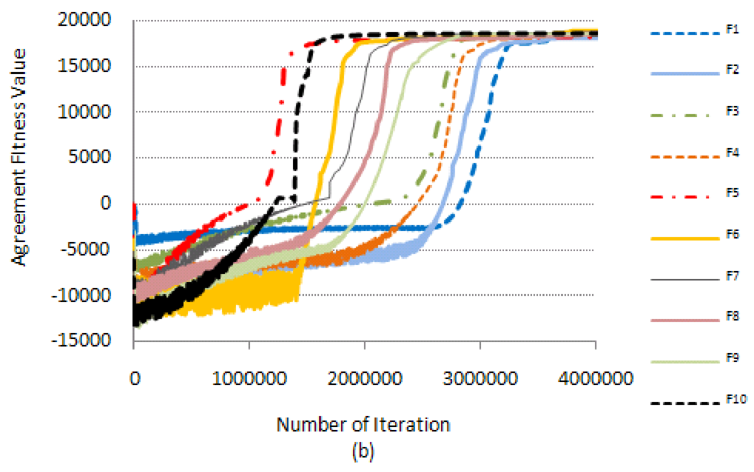


Fig. 26: The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on VAR (900 instances with 30 dimensionalities) by CC; (b) tested on VAR (900 instances with 150 dimensionalities) by CC; (c) tested on VAR (900 instances with 600 dimensionalities) by CC.





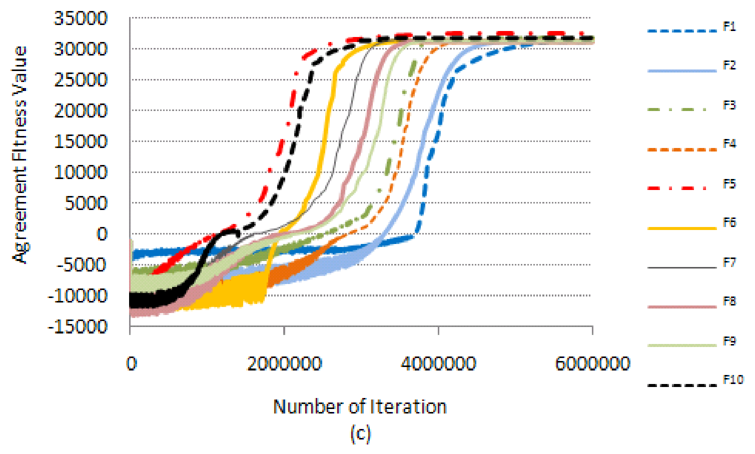
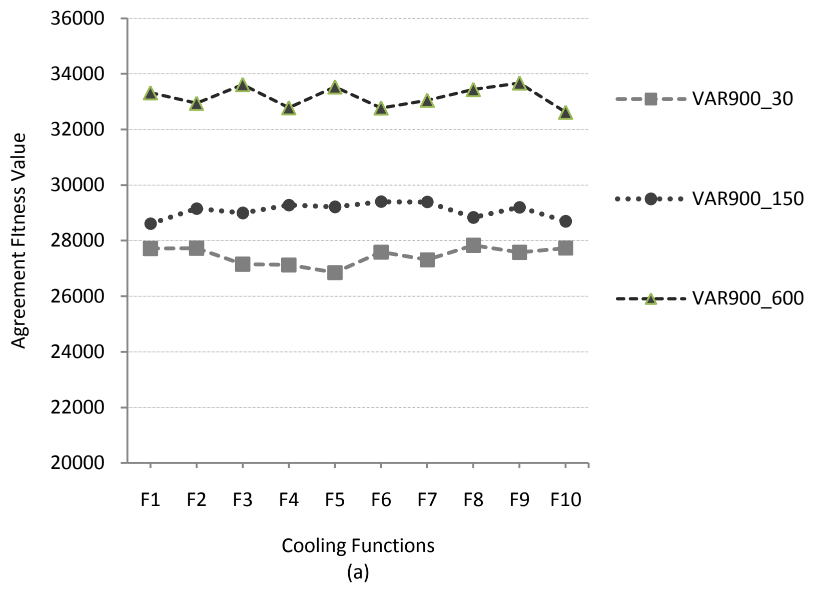


Fig. 27: The comparison of the convergence performance between ten cooling functions (with 8 input clustering methods): (a) tested on ISOLET (900 instances with 30 dimensionalities) by CC; (b) tested on ISOLET (900 instances with 150 dimensionalities) by CC; (c) tested on ISOLET (900 instances with 617 dimensionalities) by CC.

In addition, we found that the cooling functions have the slowest convergence speeds for the highest dimensional data sets, while have the fastest convergence speeds for the smallest dimensional data sets. In addition, if we compare Fig. 26 with Fig. 27, it can be seen that the convergence speeds of cooling functions have different delays for VAR900 and ISOLET900 when their dimensionality becomes higher. Additionally, for different data sets with the same dimensionality, we found that one cooling function may have different convergence speeds. The same findings can be obtained from the results generated by MOCC.

The above phenomena suggests that, for the same clustering problem, the higher the dimensionality of data sets is, the slower the convergence speeds of cooling functions will be; for different clustering problems, the delay of convergence speeds may be different when the dimensionality of data sets becomes higher. Therefore, when using Ensemble Clustering methods to analyse high-dimensional problems, it is important to consider the delay of convergence speeds of cooling functions.

Secondly, we analyse the final values of objective functions for the cooling functions. Fig. 28 shows the final convergent results generated by CC. In Fig. 28 (a), the highest dimensional data set is VAR900_600. For this data set, the final convergent agreement fitness values are the highest. For the lowest dimensional data set VAR900_30, the results have the lowest convergent agreement fitness values. Fig. 28 (b) also shows that the highest dimensional data set ISOLET900_617 has the highest convergent values, and the lowest dimensional data set ISOLET900_30 has the lowest convergent values. The reason is that, in this data set group, the small and medium dimensional data sets were generated from the highest dimensional data sets.



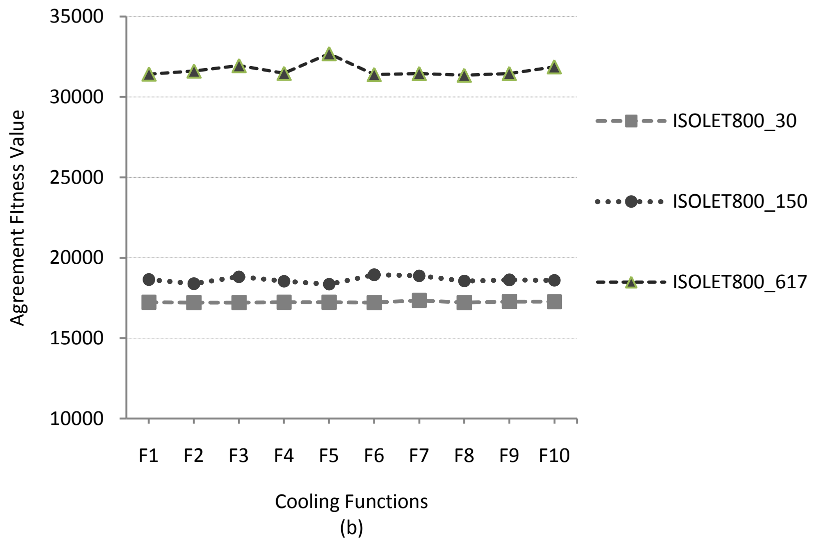
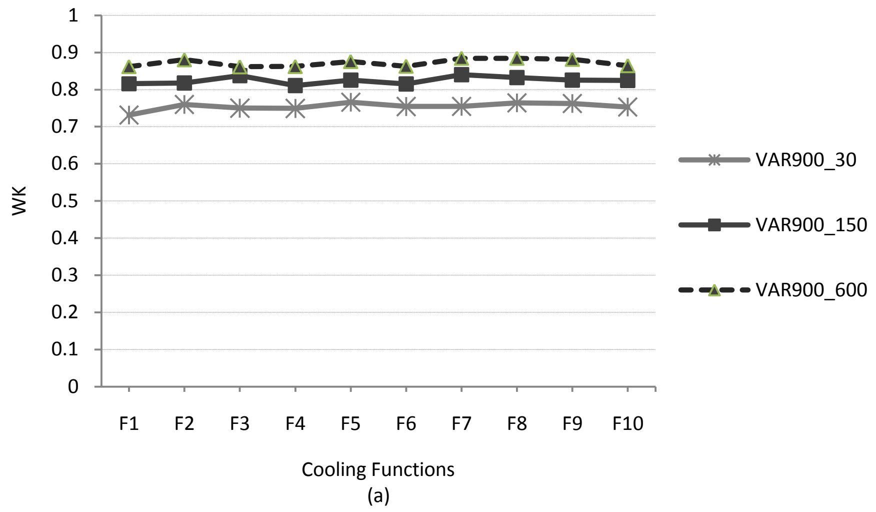


Fig. 28: The comparison of final values of the objective functions between the ten cooling functions (with 8 input clustering methods): (a) tested on the three different dimensional VAR900 (i.e. 900 instances) data sets by CC; (b) tested on the three different dimensional ISOLET900 (i.e. 900 instances) data sets by CC.

In the small and medium dimensional data sets, each instance is a subset of the corresponding instance in the high-dimensional data sets. However, the known clusters we have only correspond to the high-dimensional data sets. Therefore the clustering results of small and medium dimensional data sets may be much different from the known clusters. We can observe that, the more attributes we choose, the more accurate the input clusters (generated by input methods) tend to be, so that the final results generated by Ensemble Clustering methods will be more accurate. The above characteristics also appeared in the results generated by MOCC.

From Fig. 26 to Fig. 28, all illustrated results are generated by using 8 input clustering methods (i.e. R8). For the convergence rate of cooling functions and the comparison of objective functions' values, R16 has demonstrated the same phenomena as above, so we do not include those figures of R16 in this section.

Finally, we discuss the accuracy of results. Fig. 29 shows the WK scores of the results generated by CC. In Fig. 29 (a), the highest dimensional data set has the most accurate results, and the lowest dimensional data set has the worst accuracy. This situation is also shown in Fig. 29 (b). It is clear that Fig. 29 provides further evidence for the message mentioned in the last two paragraphs.



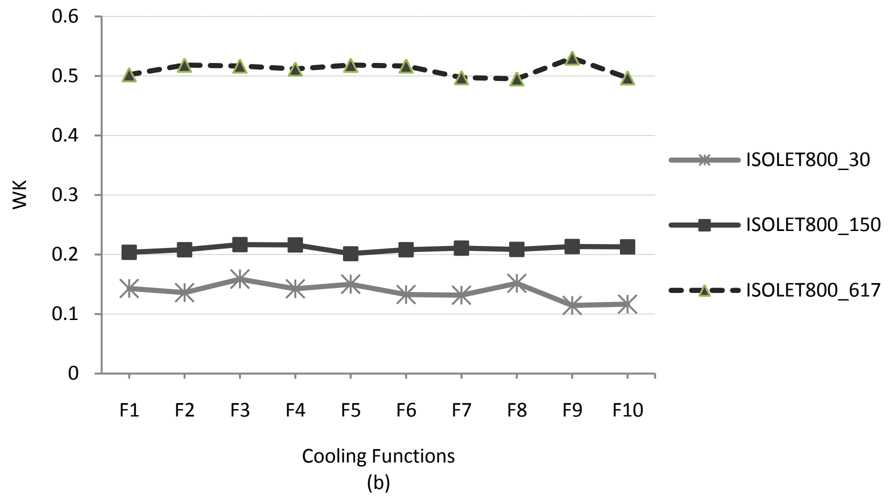
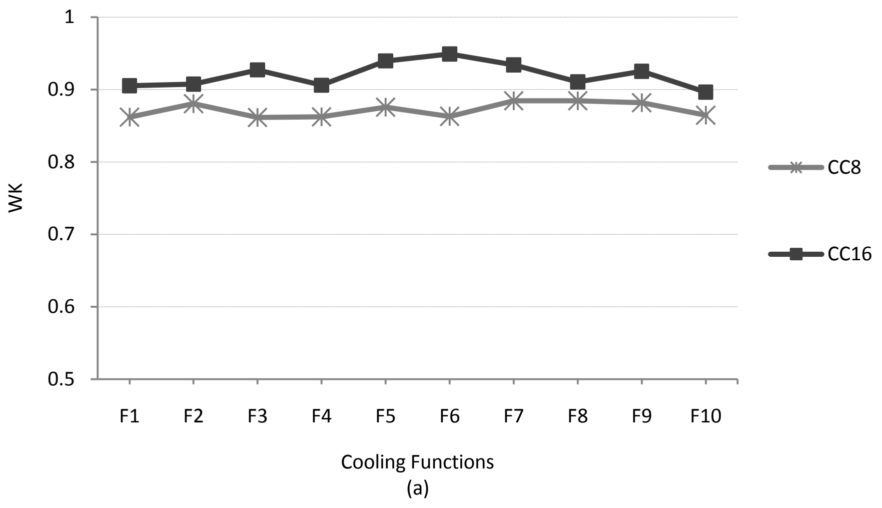


Fig. 29: The comparison of WK scores for the results generated by CC (with 8 input clustering methods): (a) CC being tested using ten cooling functions on the three different dimensional VAR900 data sets; (b) CC being tested using ten cooling functions on the three different dimensional ISOLET900 data sets.

In this section, we also discuss the effect of the number of input methods on the accuracy of results. As the discussion of the first experiment group, we choose the best case and the worst case for our discussion. Fig. 30 (a) illustrates the best case where CC is tested using the ten cooling functions on the VAR900_600 data set. The results R16 (generated by CC with 16 input methods) are clearly better than those results R8 (generated by CC with 8 input methods). Fig. 30 (b) illustrates the worst case where CC is tested using the ten cooling functions on the ISOLET900_30 data set. Although it is the worst case, the results R16 are still better than R8. From the discussion of the first experiment group, we could hypothesise that most of the input methods are appropriate for these two data sets (i.e. VAR900_600 and ISOLET900_30), or most of them generated results that have comparable or better accuracy than the mean.



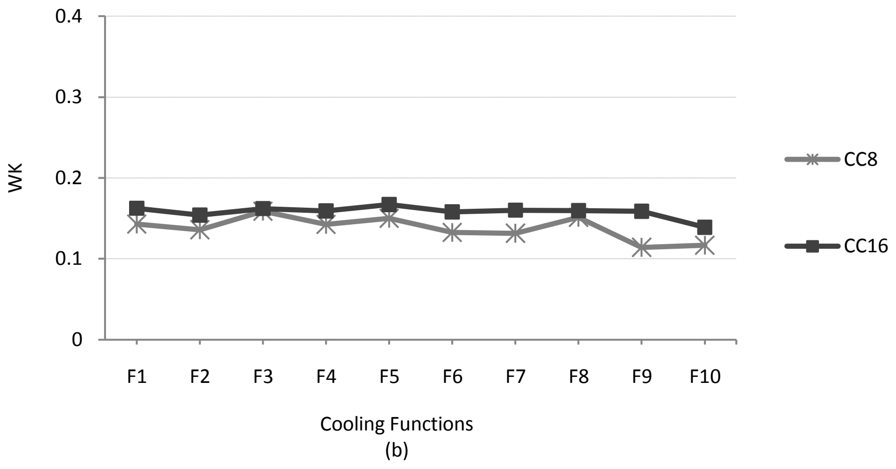
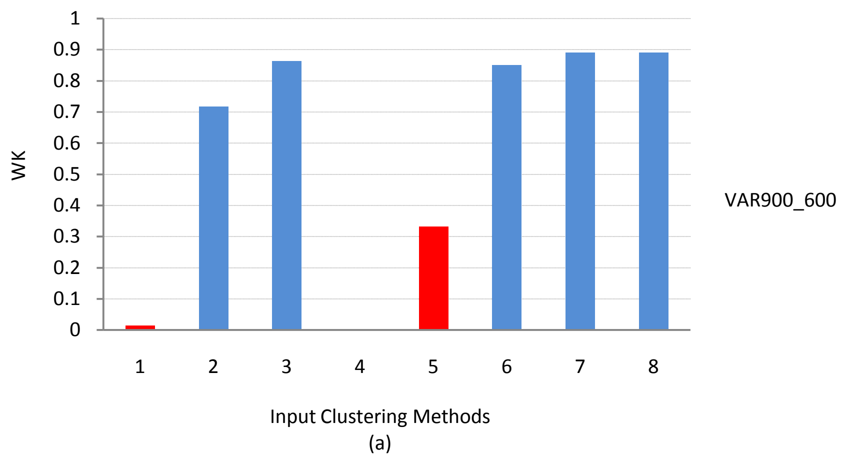


Fig. 30: The accuracy comparison of the results between using different numbers (8 and 16) of input clustering methods for analysing different dimensional data sets: (a) the best case that the results generated by CC analysing VAR900_600; (b) the worst case that the results generated by CC analysing ISOLET900_30.

In order to see if the hypothesis is valid, we analyse the quality of the 8 input methods group and the 16 input methods group for these two data sets. The results are shown in Fig. 31 and Fig. 32 respectively. Fig. 31 shows that three of the 8 input methods and six of the 16 input methods are weak for the VAR900_600 data set. In other words, there are about 63% input methods of each group that are appropriate for VAR900_600. Fig. 32 shows that two of the 8 input methods and four of the 16 input methods generated very bad results (which are much worse than the mean) for the ISOLET900_30 data set. In other words, 75% input methods of each group generated good results that are comparable or better than the mean for ISOLET900_30. By analysing the above two figures, it is clear that our hypothesis is valid. Therefore, for analysing different dimensional data sets, Ensemble Clustering methods using more input methods may improve the clustering accuracy if most of the input methods are appropriate for the data sets.



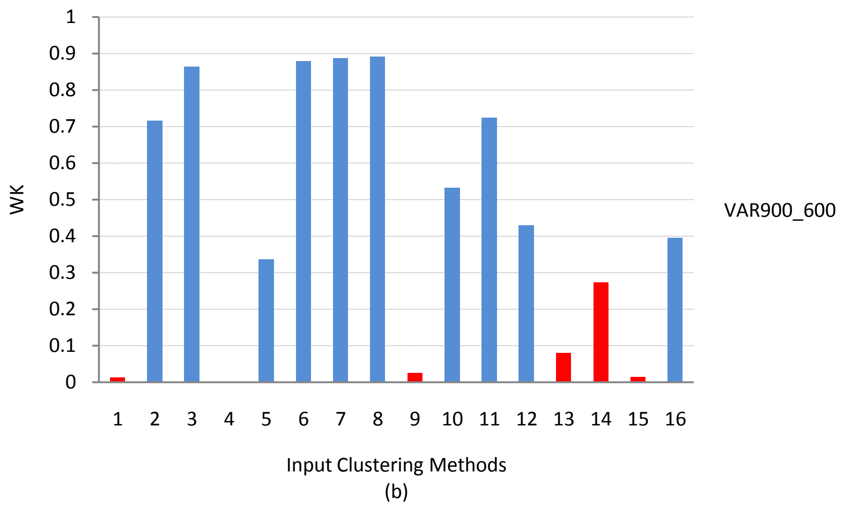
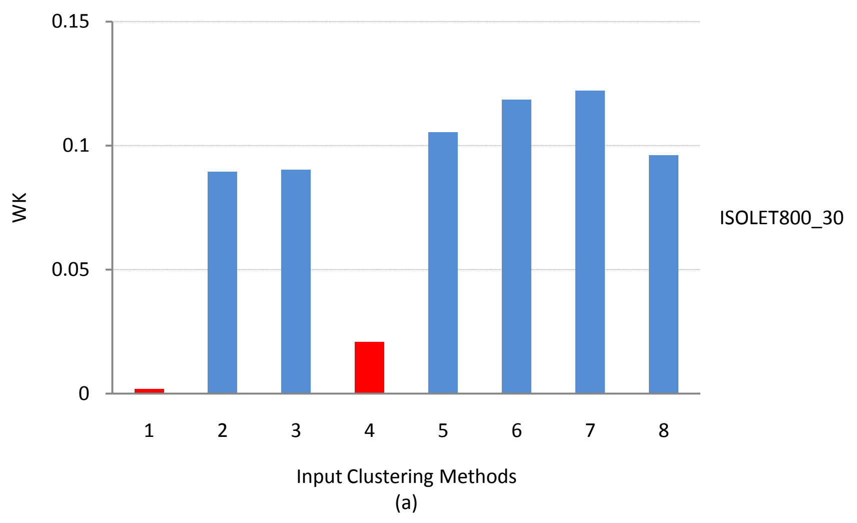


Fig. 31: The clustering accuracy comparison between input methods of the two input clustering methods groups for VAR900_600 corresponding to Fig. 30(a): (a) the 8 input methods group; (b) the 16 input methods group.



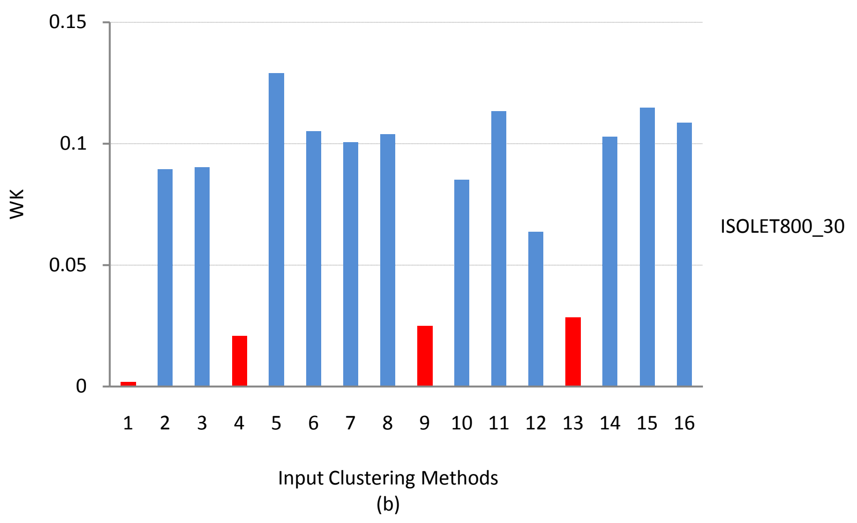


Fig. 32: The clustering accuracy comparison between input methods of the two input clustering methods groups for ISOLET900_30 corresponding to Fig. 30(b): (a) the 8 input methods group; (b) the 16 input methods group.

4.4.3 Conclusions

Based on the above discussion, we can summarise our findings as follows:

Firstly, different cooling functions have different convergence speeds. An unsuitable cooing function can cause the Ensemble Clustering methods generating invalid results. In this chapter, the exponential cooling function F5 has the fastest convergence speed, and the logarithmic cooling function F1 has the slowest convergence speed.

Secondly, the convergence speeds of cooling functions could be forecasted by observing the speeds of the cooling functions' curves closing to the abscissa. This information would be very useful when employing a new cooling function for an Ensemble Clustering method.

Thirdly, for analysing small data sets, it is good to use some fast cooling functions to save time. However, when using EC methods to analyse large data sets, the fast cooling functions may converge too quickly to reach global optima.

Fourthly, the bigger the size of the data set is, the slower the convergence of cooling functions will be. In other words, if EC methods are used to analyse large data sets, the number of iterations should be set to be big enough to allow the cooling function to gain convergence.

Fifthly, for the same clustering problem, the bigger the dimensionality of data sets is, the slower the convergence of cooling functions will be. When using EC methods to analysing high-dimensional problems, it is important to consider whether the convergence speed of cooling functions will be delayed.

Sixthly, for different data sets with the same size and dimensionality, the convergence speed of a cooling function may be different. It depends on the nature of these data sets.

Seventhly, for a specific clustering problem, if all cooling functions generated the same (or comparable) convergent results, the accuracy of these results will only depend on the nature of the Ensemble Clustering method.

Finally, for analysing a specific data set, Ensemble Clustering methods using more input methods can improve the accuracy of results when most of the input methods are appropriate for the data set (or most of them generate comparable or better results than the mean).

4.5 Summary

This chapter has presented many important insights into behaviours of cooling functions in the context of SA-based Ensemble Clustering. Ten cooling functions have been selected and studied in this chapter, and two Ensemble Clustering methods (CC and MOCC) have been used as representatives to test the cooling functions on thirteen different data sets. These insights can be very useful in assisting with the choice of cooling functions for different clustering problems. For example, F5 and F10 can be an ideal choice for saving time when analysing small data sets, but may not be good for analysing large data sets (because they may converge too early to obtain global optima). In general, for choosing a suitable cooling function or adjusting settings of a cooling function, we need to combine overall information to make an informed decision.

5

Chapter 5: K-Ants Consensus Clustering

In Chapter 3, we have introduced a novel heuristic optimisation-based Ensemble Clustering method called Multi-Optimisation Consensus Clustering (MOCC). MOCC employs Simulated Annealing to optimise the combination of internal and external clustering information. The results of MOCC have demonstrated that MOCC performed more stably and generated better results than CC. However, based on our experiments, we have observed that MOCC has a very expensive time cost. In order to have a good clustering efficiency as well as the clustering accuracy, this chapter presents another novel heuristic optimisation-based Ensemble Clustering method, which can overcome the weakness of MOCC.

5.1 Introduction

To develop an efficient and robust optimisation framework for Ensemble Clustering is still a challenge [Fern and Brodley, 2004; Li *et al.*, 2009; Li *et al.*, 2004; Swift *et al.*, 2004]. First of all, developing an efficient objective function is not an easy task. Secondly, for heuristic optimisation-based methods, computational costs are very high especially for analysing large data sets. Last but not least, it is difficult to guarantee that global optima can be reached after an optimisation search. Among these difficulties, how to design an efficient objective function is the key. If we treat input clusterings as external information, and characteristics of given data as internal information, we can comprehend the key problem as designing an objective function that can efficiently combine internal information with external information for Ensemble Clustering.

A promising method for describing internal information of given data is called Minimum Description Length (MDL), which was firstly introduced by Rissanen [1983; 1978]. A key advantage of the MDL principle is that it is sufficiently robust for describing a variety of data [Graunwald *et al.*, 2004; Rissanen, 1983; Rissanen, 1996; Rissanen, 2001; Witten and Frank, 2005]. Graunwald *et al.* [2004] stated that MDL is a versatile method, whose performance is comparable to other well-known inductive inference methods such as Bayesian Statistics [Kontkanen *et al.*, 2006; Lee, 1997].

For solving optimisation problems, we take notice of the Ant Colony Optimisation (ACO) techniques [Colorni *et al.*, 1991; Dorigo and Caro, 1999; Dorigo *et al.*, 1999; Dorigo *et al.*, 2006; Yang and Kamel, 2003]. Many improved versions of ACO have been employed for data clustering analysis. For example, Jiang and Chen [2007] introduced an ant colony algorithm for General Clustering; Zhao [2007] developed an ant colony clustering algorithm that combines the global pheromone updating with heuristic information to construct solutions. These literatures have revealed promising results for the applications of ACO [Jiang *et al.*, 2007; Zhao, 2007]. The most remarkable advantage of ACO, for data clustering, is that each ant is an intelligent agent and has the ability of judging candidate objects with purposes [Colorni *et al.*, 1991; Yang and Kamel, 2003]. When an ant finds an object, it can evaluate the object to decide whether to pick it up or not. This characteristic can enable ACO to have more efficient performance, and speed up the convergence of the optimisation.

Based on the ideas and principles of MDL and ACO, we developed a novel Ensemble Clustering method called K-Ants Consensus Clustering (KACC) with the following advantages.

1) The generation of the initial clustering solution is based on the agreement of input clusterings. KACC combines external information (i.e. input clusterings) by an Agreement Matrix, which has been presented in our previous work [Swift *et al.*, 2004]. Based on the Matrix, we construct a set of agreement lists. And then the initial clustering solution will be generated based on the agreement lists.

2) Based on the idea of MDL [Graunwald *et al.*, 2004; Rissanen, 1983], we developed Attributed Weighted Description Length (AWDL) as the criterion to combine internal information (i.e. characteristics of given data) for Ensemble Clustering. For each attribute of instances, we utilise a variance ratio as weight to score the contribution of the attribute.

3) We developed a K-Ants Multiple Optimisation (KAMO) framework based on ant colony clustering theory without introducing parameters. Existing Ant Colony Algorithms (ACA) have too many parameters so that the results are very sensitive to the initialisation. It is often difficult for analysts to configure these parameters properly. Our Ensemble Clustering method solves this difficulty by the KAMO framework, which does not introduce any extra parameters.

4) Our method has achieved the clustering combination across external and internal information. For Ensemble Clustering, we treat the input clusteirngs (generated by input clustering algorithms) as external clustering information, and treat the characteristics of given data as internal information. The existing ACA-based Ensemble Clustering methods only (or mainly) based on the external information provided by input methods. Therefore the clustering accuracy of these methods relies heavily on the accuracy of input clusterings. Our method, KACC, combines not only the external information but also the internal information to achieve the clustering optimisation, which can avoid biases contained by input clusterings.

5) The heuristic optimisations of KACC have very good efficiencies. In order to seek global optima, computational costs of traditional heuristic optimisation methods are often very high especially for analysing large data sets. It is known as the NP-problem. In our method, we only try to guarantee the results with high enough degree of accuracies in the application context instead of trying to find global optima. We construct a well initial solution to restrict the search space. Based on the reasonable restricted search space, KACC has very low computational costs.

6) A validation vector, which consists of five clustering validation indices, is developed and used to evaluate the clustering solutions generated by the K-Ants Multiple Optimisation Framework. The final clustering result of KACC will be generated by the validation vector.

In order to evaluate the performance of our method KACC, we compare KACC with several other Ensemble Clustering methods. Yang and Kamel [2003] proposed a method called Ensemble of Swarm Intelligence Clustering (ESIC), which is one of the few papers about clustering ensembles based on Ant Colony Optimisation techniques [Azimi *et al.*, 2009]. We use ESIC as a representative to be compared with KACC. Moreover, seven Ensemble Clustering algorithms of the well-known R [R Development Core Team, 2005] package CLUE (CLUster Ensemble), which was developed by Hornik [2005], are also used for the performance comparison in this chapter. There are totally ten data sets that have been used for the experiments. The experimental results reveal that KACC generated much better results than above Ensemble Clustering methods.

The rest of this chapter is organised as follows: Section 5.2 describes the work related to our Ensemble Clustering method. Section 5.3 introduces the details of the KACC algorithm. Section 5.4 describes designed experiments and the data sets used. Section 5.5 discusses the results generated from the experiments. Finally in Section 5.6, we draw some conclusions.

5.2 Related Works

5.2.1 The Agreement Matrix

The Agreement Matrix (*A*) was firstly presented by Swift *et al.* [2004], where *A* is used to describe the clustering agreement of a set of input clustering algorithms for a give data set. the formation of Agreement Matrix is based on an input clustering arrangement matrix *Z*. Each row of *Z* is an Input Clustering Arrangement generated by an Input Algorithm. Suppose we use eight individual clustering algorithms to generate eight clustering arrangements (i.e. one algorithm generates one arrangement) for the given data set, these individual algorithms are called Input Algorithms, and these clustering arrangements are called Input Clustering Arrangements (or Input Clusterings). The definition of the Agreement Matrix has been detailed in Chapter 2.2.1.

5.2.2 Minimum Description Length (MDL)

Our Attribute Weighted Description Length (AWDL) criterion is developed based on the basic principle of MDL. MDL is an inductive inference method, and was developed for solving the model selection problem. MDL was firstly introduced by Rissanen [1983; 1978], who claimed that the more the data can be compressed, the more the data have been learned by us [Graunwald *et al.*, 2004]. The principle of MDL is to detect the regularities in the given data and use the regularities to compress the description of the data [Graunwald *et al.*, 2004].

Nowadays, MDL has been improved and applied in many research areas such as data clustering. Kontkanen *et al.* [2006] proposed a MDL framework, which is a model-based method, for data clustering. The behind idea is that the more the description of the clusters can be compressed, the better the similarity of the instances within a cluster is. The total description length of all clusters is the global criterion that measures the dependence between clusters [Kontkanen *et al.*, 2006]. Witten and Frank [2005] claimed that MDL is sufficiently robust for describing various types of data.

The following example simply describes the principle of using MDL for data clustering [Kontkanen *et al.*, 2006; Witten and Frank, 2005]:

Suppose we need to partition a data set (with *n* instances) into *k* clusters. We use C_i indicates the centre of the cluster *i* (where *i*= 1, 2, …, *k*). Based on the MDL principle, the data set can be described by the following two parts: 1) the cluster centres C_1 …C_k; 2) for each instance, recording the difference between the instance and the cluster centre it belongs to. In theory, the shortest description length of the data set indicates the best cluster partitioning.

Since we use AWDL instead of the existing MDL formulisations, we only introduce the basic principle of MDL in this chapter. For more detailed descriptions of MDL, please refer to the references [Graunwald *et al.*, 2004; Kontkanen *et al.*, 2006; Rissanen, 1983; Rissanen, 1996; Rissanen, 1978; Rissanen, 2001] and [Witten and Frank, 2005].

It is worth to note that, in this chapter, we focus on showing how the results are generated by KACC, which utilises the AWDL criterion. We know that the modern formulisation of MDL is called Normalized Maximum Likelihood (NML) that was proposed by Rissanen [1996; 2001]. In future work, we will employ NML instead of AWDL for our KACC so that we can compare the performance of AWDL with the one of NML. By analysing the differences between them, we will refine AWDL or integrate their advantages to further improve the performance of KACC.

5.2.3 Ant Colony techniques for Ensemble Clustering

For different applications, various ant colony algorithms have been developed based on the Ant Colony Optimisation (ACO) theory during the past two decades [Bonabeau *et al.*, 2000; Mullen *et al.*, 2009]. One of the valued efforts is the development of ant colony clustering algorithms. The insights about ant colony clustering algorithms came from observing wild ants sorting their eggs, larvae and cocoons into different piles without direct supervisions in their nest [Deneubourg *et al.*, 1991; Lumer and Faieta, 1994]. Moreover, the above successful experiences have started to benefit the development of Ensemble Clustering.

To our best knowledge, there is a limited amount of work that employs Ant Colony Algorithms (ACA) for Ensemble Clustering. The first ACA-based Ensemble Clustering method was proposed by Yang and Kamel [2003]. The method is called ESIC (Ensemble of Swarm Intelligence Clustering) [Yang and Kamel, 2003], which uses three ant colonies (with different moving speeds) to produce three clustering solutions as inputs, and then combines these inputs through a hypergraph model. The final result of ESIC is generated by implementing an ant colony algorithm again based on the hypergraph model.

Based on ESIC, Yang *et al.* further improved their method and introduced their works in the literature [Yang and Kamel, 2006] and [Yang *et al.*, 2006]. In the literature [Yang *et al.*, 2006], Yang *et al.* added a clustering validity index for ant colonies to evaluate the clustering performance and find the best number of clusters; moreover, they employed the Adaptive Resonance Theory (ART) [Carpenter and Grossberg, 2003; Carpenter and Grossberg, 1987; Carpenter and Grossberg, 1990; Grossberg, 1987] to combine the clusterings produced by the ant colonies to generate final results. In the literature [Yang and Kamel, 2006], Yang and Kamel presented an Aggregated Multi-Ant Colonies (AMAC) algorithm, which introduces a queen ant agent to integrate with the implementation of several parallel ant colonies to generate clusters. The combination of these clusters is achieved by a similarity matrix calculated by a hypergraph model [Yang and Kamel, 2006].

The above three Ensemble Clustering methods proposed by Yang *et al.* have two main drawbacks. One is that their ant colony algorithms have too many input parameters that need to be initialised. The propriety of initialisation depends much on subjective experiences and judgements, so that the results of their ant colony algorithms are very sensitive to the initialisation. The other is that these methods combine clustering information only based on their input clusterings. Therefore the clustering accuracy of these methods could be significantly affected by their input clusterings.

Wei [2009] proposed an ACA-based Ensemble Clustering method, which is very similar to the ESIC framework introduced by Yang and Kamel. Azimi *et al.* [2009] suggested a new ACA-based Ensemble Clustering method, which introduces a new ant colony algorithm, where a co-association matrix is used for simplifying and reducing input parameters. However this method still has the difficulty in which the clustering accuracy could be significantly affected by input clusterings.

Recently, Gu *et al.* [2009] proposed an improved ant colony algorithm for Ensemble Clustering. Strictly speaking, this Ensemble Clustering method is not a real ACA-based framework since its consensus clustering part is just the NMF (Nonnegative Matrix Factorization) algorithm [Lee and Seung, 1999], which has no relations with ACA. Moreover, NMF produces final results only based on input clusterings, and the proposed ant colony algorithm has so many parameters.

5.2.4 The Ensemble Clustering method called ESIC

In this chapter, we use ESIC (Ensemble of Swarm Intelligence Clustering), proposed by Yang and Kamel [2003], as a representative of existing ACA-based Ensemble Clustering to compare with our method KACC. ESIC consists of three key components: Ant Colony Clustering, Hypergraph Representation, and Clustering Ensembles. The following subsections give detailed descriptions of these components.

- *Ant Colony Clustering*

The Ant Colony Clustering module contains three ant colony algorithms with different moving speeds of ants, which are the constant speed, the random speed, and the decreasing random speed. These ant colony algorithms are used to generate three clustering solutions to form into the input clusterings. The only difference between the three ant colony algorithms is the moving speed of ants. First we present the details of the Constant Speed Ant Colony Algorithm. For other two algorithms, we only present the differences of the definition for the ant moving speed.

The Constant Speed Ant Colony Algorithm (CSACA) randomly maps all instances onto a plane, where each instance is represented by a coordinate value on the plane. The plane is divided into a number of square cells with the same size. Suppose that there are a number of ants on the plane. Each ant randomly chooses an instance *i* (not-selected by other ants), and measures the average similarity between the instance *i* and other instances within the cell that the instance *i* belongs to.

The average similarity of the instance *i* in its cell is defined as equation (5.1)

$$f(i) = max\left\{0, \frac{1}{s^2}\sum_{j\in N_s}\left[1 - \frac{\mathrm{d}(i,j)}{\alpha(1+(v-1)/v_{max})}\right]\right\}, \tag{5.1}$$

where s is the side length of the cell, and N_s is the neighbours of i within the cell; d(i, j) denotes the distance between the instances i and j on the plane; v is the moving speed of ants, and v_{max} is the maximal speed ants could have. The value of v can be simply set to be a number of cells. The variable α is the similarity coefficient. Under the same conditions, a larger α could enable the ant colony algorithm to have a quicker convergence with fewer clusters; the other way round, a smaller α could cause the algorithm to have a slower convergence with more clusters.

After measuring the average similarity of the instance i, the probability of the ant picking up *i* is calculated by equation (5.2):

$$P_p = 1 - \Gamma(f(i)) \tag{5.2}$$

$$P_d = 1 - P_p = \Gamma(f(i)) \tag{5.3}$$

where

$$\Gamma(x) = \frac{1}{1+e^{-cx}} \tag{5.4}$$

Equation (5.4) is a natural exponential expression, where the variable c is a slope coefficient. Under the same conditions, increasing c can speed up the convergence of the algorithm. The probability of the ant dropping off the instance i is defined by equation (5.3).

Once the ant picks up the instance i, it will randomly move to another cell on the plane with the moving speed v. We assume that the ant moved to the cell w. The ant will measure the average similarity again for i within the cell w and the probability of dropping off the instance i. If the probability P_d is high enough so that the ant drops i within the cell w, it will randomly choose another instance (unselected by other ants) to start again. Each ant will repeat the above activities until the whole system is converged (i.e. there are no more instances that need to be moved by ants) or after a certain number of iterations. The final clustering solution is obtained from square cells of the plane. All instances within a square cell are assigned into one cluster; in contrast, instances in different cells are assigned into different clusters. The number of non-empty cells on the plane indicates the total number of clusters.

For the Random Speed Ant Colony Algorithm (RSACA), before each movement of ants, the moving speed v in equation (5.1) is randomly chosen from the interval $[1, v_{max}]$. For the Decreasing Random Speed Ant Colony Algorithm (DRSACA), the moving speed v is set to be a large value within the interval $[1, v_{max}]$ at the beginning, and then the value of v will be decreased randomly as time goes on.

- *Hypergraph Representation*

For this part, Yang and Kamel [2003] employed a hypergraph model proposed by Strehl and Ghosh [2003] to combine the results generated from the Ant Colony Clustering module. The hypergraph model transforms input clustering arrangements into a hypergraph, where each vertex indicates an instance and each hyperedge represents the similarity between two vertices. In this way, the consensus clustering can be achieved by partitioning the hypergraph. The hypergraph model is constructed by two steps. The first step is to construct the Hypergraph Adjacency Matrix (HAM), and the second step is to build the

Symmetric Similarity Matrix (SSM).

The rows of HAM denote the vertices (i.e. instances) of the hypergraph, the columns of HAM indicate the edges of the hypergraph. Suppose that a data set Q has n instances, and the Ant Colony Clustering module generated three clustering solutions $R1$, $R2$ and $R3$ for Q. The corresponding numbers of clusters for the three solutions are k_1, k_2 and k_3. For the data set Q, HAM will have n rows and (k_1+ k_2+ k_3) columns. Each column of HAM stands for one cluster. For each column, the members of the cluster are set to be 1; others are set to be 0.

After constructing HAM, the Symmetric Similarity Matrix is built to combine the results of the three ant colony algorithms. Suppose we get a Hypergraph Adjacency Matrix H, the matrix SSM can be represented as equation (5.5):

$$S = HH^T \tag{5.5}$$

where H^T is the transpose of H. Each row or column of S indicates a vertex of the hypergraph. Each element (except those along the leading diagonal) signifies the weight of the corresponding hyperedge.

- *Clustering Ensembles*

The final clustering result is generated by the Clustering Ensembles module. Based on the Symmetric Similarity Matrix, an ant colony algorithm (e.g. CSACA) is used again to cluster the vertices of the hypergraph to obtain final results. Equations (5.1) - (5.4) are still used to calculate the average similarity, picking up and dropping off probabilities, but the distance definition is different. In equation (5.1), each instance is represented by a coordinate value on the plane, whereas in the Clustering Ensembles module, the vertices of the hypergraph are regarded as instances. The distance between two vertices is defined as equation (5.6)

$$d(i,j) = \frac{D(i,j)}{g(i) + g(j)} \tag{5.6}$$

where

$$D(i,j) = [g(i) \cup g(j)] - [g(i) \cap g(j)] \qquad (5.7)$$

$g(i)$ indicates an aggregation of vertices that have edges connected to vertex i. $D(i, j)$ means the aggregation of the vertices that belong to $g(i)$ or $g(j)$ (but not both).

5.2.5 Seven algorithms of the CLUE package

The CLUE (CLUster Ensemble) package developed by Hornik [2005] is an extension package for R, which is a well known programming language developed by R Development Core Team [2005]. CLUE provides the *cl_consensus*() function to implement different consensus clustering algorithms [Hornik, 2005].

In this chapter, we choose seven consensus clustering algorithms provided by the *cl_consensus*() function to compare with our method KACC. The seven algorithms are outlined as follows (cited from [Hornik, 2005]):

1) "SE" – a fixed-point algorithm for obtaining soft least squares Euclidean consensus partitions;
2) "GV1" – the fixed-point algorithm for the "first model" proposed by Godon and Vichi [2001];
3) "DWH" – an extension of the greedy algorithm proposed by Dimitriadou, Weingessel and Hornik [2002];
4) "HE" – a fixed-point algorithm for obtaining hard least squares Euclidean consensus partitions;
5) "SM" – a fixed-point algorithm for obtaining soft median Manhattan consensus partitions;
6) "GV3" – a SUMT algorithm for the "third model" proposed by Godon and Vichi [2001];
7) "soft" – a SUMT method based on soft partitions.

For the details of these algorithms, please refer to the references [Gordon and Vichi, 2001; Hornik, 2005] and [Dimitriadou *et al.*, 2002].

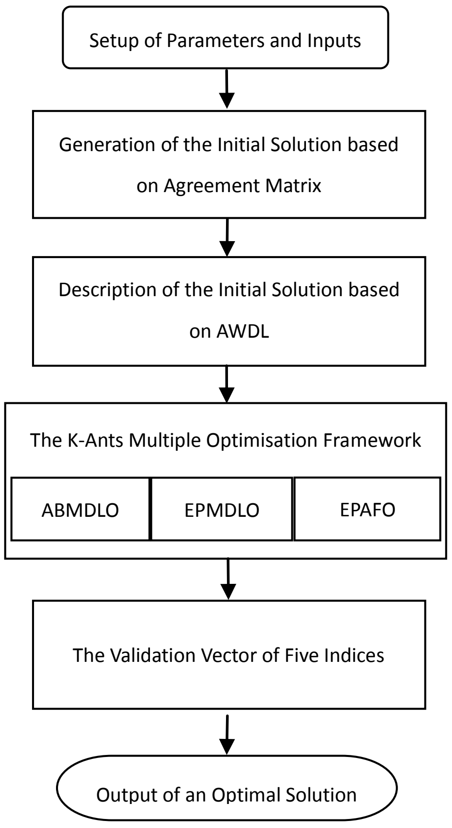

Setup of Parameters and Inputs
Generation of the Initial Solution based on Agreement Matrix
Description of the Initial Solution based on AWDL
The K-Ants Multiple Optimisation Framework
ABMDLO
EPMDLO
EPAFO
The Validation Vector of Five Indices
Output of an Optimal Solution

Fig. 33: The flow chart of the K-Ants Consensus Clustering algorithm.

5.3 The K-Ants Consensus Clustering algorithm

Fig. 33 shows the flow chart of the K-Ants Consensus Clustering (KACC) algorithm. It is clear that KACC consists of four modules, which are the Generation of the Initial Solution, the Description of the Initial Solution based on AWDL, the K-Ants Multiple Optimisation Framework, and the Validation Vector module. The KACC algorithm is detailed in the following subsections.

5.3.1 Generation of the Initial Solution

- *Construction of Agreement Lists*

The construction of the agreement lists is based on an Agreement Matrix that has been introduced in Section 2.2.1. In order to construct an Agreement Matrix, the KACC algorithm uses a clustering arrangement matrix *Z* as one of its inputs. The *Z* is formed by the clustering results generated by a set of selected clustering methods (these methods are called input clustering algorithms). Each row of *Z* is one clustering result. Based on the *Z*, KACC creates an Agreement Matrix. Each element of the Agreement Matrix (except the elements along the leading diagonal) corresponds to a pair of instances. The value of the element indicates how many input clustering methods agree that the corresponding two instances are assigned into the same cluster.

After creating the Agreement Matrix, we can start to construct the agreement lists. For the values in the Agreement Matrix, we assume that a pair of instances with a high agreement value should have a high probability of assigning this pair of instances into one cluster. Therefore, the unique instance-pairs, which have the same agreement value, should have the same probability of assigning instances of each pair into the same cluster. Based on this assumption, we assign unique instance-pairs into different groups according to their agreement values in the Agreement Matrix. In other words, each group is an agreement list that only contains the unique instance-pairs that have the same agreement value. Once the agreement lists are constructed, the next step of KACC is to generate the initial combinational clustering solution C_0.

- *The Initial Combinational Solution C_0*

The agreement list with the highest agreement value will have the highest priority of being processed for generating the initial solution. Other agreement lists will be processed in descending order according to their agreement values. In addition, within these agreement lists, there is a special list which contains unique instance-pairs that have the zero agreement. It means that, for any of these instance-pairs, all input clustering methods agree that the corresponding two instances should not be assigned into the same cluster. In order to highlight this agreement list, we name it Full Disagreement List (FDL). The FDL list will be treated as a condition list. In other words, during the whole process of generating C_0, any instance-pair of FDL cannot appear in any cluster.

The procedure of generating C_0 starts from the agreement list with the highest priority. For each instance-pair, four different situations need to be considered. Firstly, if the corresponding two instances have not been assigned into any of existing clusters, these two instances will become a new cluster. Secondly, if these two instances have been assigned into one cluster, no operation will be executed. Thirdly, if these two instances exist in two different existing clusters, these two clusters will be merged into one cluster. Finally, if no more than one of these two instances has been assigned into an existing cluster, the remaining instance will be assigned into the same cluster. For the third and fourth situations, it is important to note that, after assigning an instance into an existing cluster or merging two existing clusters, any instance-pair of the FDL list must not appear in the updated cluster, otherwise, the operation will be undone.

The KACC algorithm uses an estimated number of clusters k as one of the inputs. It means the initial clustering solution C_0 should have k clusters. Therefore, the process of generating C_0 will be continued until all instances have been assigned into k clusters based on the agreement lists.

5.3.2 Description of C_0 based on AWDL

The original data set M is one of the inputs of KACC. Rows of M correspond to instances, and columns of M correspond to attributes. Before using AWDL to describe C_0, we need to standardise the M to be X. For each attribute, the range of the values will be standardised into [0, 1]. The standardisation equation is shown as equation (5.8):

$$X_{ij} = \frac{M_{ij} - \min\limits_{1 \le i \le n}(M_{ij})}{\max\limits_{1 \le i \le n}(M_{ij}) - \min\limits_{1 \le i \le n}(M_{ij})} \tag{5.8}$$

where X_{ij} and M_{ij} indicate the jth attribute of the ith instance in X and M respectively; n denotes the total number of instances; $\max_{1 \le i \le n}(M_{ij})$ denotes the maximal value of the jth attribute for all instances, and $\min_{1 \le i \le n}(M_{ij})$ corresponds to the minimal value. The purpose of standardising the original data set is to prepare for calculating the Attribute Weighted Coefficient (AWC).

- *The Attribute Weighted Coefficient*

Clustering aims to assign instances into different groups according to the differences between these instances. Variance measures the degree of variation for values of a variable. The higher the variance is, the larger the variety of the values is. A high variance means a high difference between values. Different attributes often provide different contributions of describing differences between instances. The bigger the range of the variation of an attribute is, the bigger contribution the attribute provides. In order to describe the contributions provided by different attributes, we adopt the variance to weight each attribute.

The way of calculating the weight for each attribute is shown by the following equations:

$$D_j = \frac{\sum_{i=1}^{n}(X_{ij} - \mu_j)^2}{n} \tag{5.9}$$

$$\mu_j = \frac{\sum_{i=1}^{n} X_{ij}}{n} \tag{5.10}$$

$$w_j = \frac{D_j}{\max(D_1, D_2, \dots, D_a)} \tag{5.11}$$

where D_j indicates the variance of the jth attribute for instances in the data set X, n indicates the total number of instances, X_{ij} denotes the value of the jth attribute of the ith instance, and μ_j denotes the mean value for the jth attribute over the n instances. The variable w_j indicates the weight of the jth attribute. The total number of attributes is a.

- *The Attributed Weighted Description Length criterion*

The Attributed Weighted Description Length (AWDL) criterion L consists of three parts, which are the sum of mean values S_m, the sum of weighted absolute deviations S_d, and the attribute weight vector W.

$$L = \{S_m, S_d, W\} \tag{5.12}$$

AWDL is defined by equation (5.12), where W is a vector (as defined by equation (5.13)) that consists of weights of all attributes.

$$W = [w_1, w_2, \cdots, w_a] \tag{5.13}$$

According to equations (5.9)-(5.11), we know that the attribute weight vector is a constant vector. In other words, the attribute weight vector will not be changed whatever the clustering result is.

Therefore, we simplify the description of AWDL as equations (5.14)-(5.16):

$$L' = S_m + S_d \tag{5.14}$$

$$\begin{aligned} S_m &= \sum_{p=1}^{k} (w_1 m_{p1} + w_2 m_{p2} + \cdots + w_a m_{pa}) \\ &= \sum_{p=1}^{k} \sum_{j=1}^{a} w_j m_{pj} \end{aligned} \tag{5.15}$$

$$\begin{aligned} S_d = \sum_{p=1}^{k} \sum_{q=1}^{T} \big(w_1 |X_{q1} - m_{p1}| + w_2 |X_{q2} - m_{p2}| + \cdots \\ + w_a |X_{qa} - m_{pa}|\big) \\ = \sum_{p=1}^{k} \sum_{q=1}^{T} \sum_{j=1}^{a} w_j |X_{qj} - m_{pj}| \end{aligned} \tag{5.16}$$

where S_m signifies the sum of mean values of all clusters; m_{pj} denotes the mean value of the jth attribute in the pth cluster, where the value of j varies from 1 to a; k is the total number of clusters. In equation (5.16), we use the absolute deviation to measure the difference between the mean and the value of each instance for each cluster. S_d stands for the sum of weighted absolute deviations for all instances, and X_{qj} denotes the value of the jth attribute of the qth instance in the pth cluster, where the value of q varies from 1 to T. The variable T indicates the number of instances in the pth cluster. According to equations (5.14)-(5.16), the description of the initial clustering solution C_0 can be simplified to be L'_0.

5.3.3 The K-Ants Multiple Optimisation Framework

The K-Ants Multiple Optimisation (KAMO) framework contains three independent optimisation phases: the Agreement Based MDL Optimisation (ABMDLO), the Equal Probability MDL Optimisation (EPMDLO), and the Equal Probability Agreement Fitness Optimisation (EPAFO). The EPMDLO phase optimises the combination of internal information of the given data. The EPAFO phase optimises the combination of external information provided by input clusterings. And the ABMDLO phase optimises the integration of internal and external information. Each optimisation phase starts from the initial clustering solution C_0. K-Ants means the total number of ants used for each optimisation phase is K which is equal to the estimated number of clusters k. Each ant is in charge of one cluster. We also endow these ants with intelligence. Each ant can remember all members of its cluster, and the information can be automatically updated if the members of its cluster have been changed.

Candidate solutions are generated by each ant moving instances from one cluster to another cluster. The AWDL criterion is used to evaluate these solutions to decide whether accept them or not. The activity of an ant moving an instance consists of two actions: picking up, and dropping off. The KAMO framework has two different definitions for the picking up probability. One definition uses the equal probability, another uses the agreement-based probability.

- *The Agreement Based MDL Optimisation*

Agreement Based MDL Optimisation (ABMDLO) is based on both agreements of input clusterings and interior characteristics of data sets. The agreements of input clusterings are used to evaluate the probability of an ant picking up an instance; and the interior characteristics of data sets are presented by the L' which is expected to be as small as possible. The objective function has been given as equations (5.14)-(5.16).

The probability of picking up an instance by an ant is defined by equations (5.17) - (5.20),

$$v_1 = \frac{N_q \max(A_{iq})}{N_i \max(A_i)} \tag{5.17}$$

$$v_2 = \frac{\max(A_{iq})}{r} \tag{5.18}$$

$$v = \begin{cases} 0 & , \max\left(A_{iq}\right) < \max(A_i) \\ \min(v_1, v_2) & , \max\left(A_{iq}\right) = \max(A_i) \end{cases} \tag{5.19}$$

$$V_p = 1 - v \tag{5.20}$$

where v_1 is named Compound Agreement Ratio, and v_2 is the Simple Agreement Ratio. The variable A denotes the Agreement Matrix. A_{iq} indicates the agreement values of instance-pairs within the cluster q, and each of these instance-pairs must include the instance i. max(A_{iq}) means the maximum value within these agreement values. N_q denotes the number of the maximum agreement values within A_{iq}. The variable A_i stands for the agreement values of all instance-pairs that contain the instance i, and max(A_i) denotes the maximum agreement value within A_i. N_i signifies the number of maximum values within A_i.

In equation (5.18), the variable r indicates the number of input clusterings. The picking up probability V_p is finally defined by equation (5.20), where v is the un-picking up probability. For an instance i, if the instance-pair (i, j) has the maximal agreement value max(A_i), but j does not stay with the instance i in the same cluster, v will be 0. It means the instance i will be definitely picked up by the ant. If i and j stay in the same cluster, the value of v will be the smaller one of v_1 and v_2.

Once an ant picks up an instance from its cluster, it will go through other clusters to see whether the movement of the instance can reduce L'. If the movement does make L' smaller, the ant will drop the instance into that cluster and go back to its own cluster to pick up another instance. Meanwhile, the information of instance members for these two clusters will be updated in the memory of the corresponding two ants. If the movement does not make L' smaller, the ant will put the instance back and mark it as *picked*. And then the ant will try to pick up another *un-picked* instance to repeat the above actions. If the picking up probability of an instance is very low so that the ant does not pick it up, the instance will be also marked as *picked*. If all instances of a cluster are marked as *picked*, the ant will clear all marks and start again. All ants take actions in turn. The whole process will be continued until the value of L' is converged or the predefined number of iterations is reached.

The probability of dropping off an instance by an ant is defined by equation (5.21),

$$V_d = \begin{cases} 1, & L'_{new} < L'_{old} \\ 0, & otherwise \end{cases} \tag{5.21}$$

where L'_{old} indicates the value of AWDL before an ant moving an instance, and L'_{new} stands for the value of AWDL after a movement of the ant. The definition of V_d is applied for all optimisation phases: ABMDLO, EPMDLO, and EPAFO.

- *The Equal Probability MDL Optimisation*

Equal Probability MDL Optimisation (EPMDLO) starts from the Initial Solution C_0 to seek the optimal clustering solutions by minimising L'. The objective function is also defined by the equations (5.14)-(5.16). Each ant randomly picks up instances within its cluster. The whole process of EPMDLO is similar to the process of ABMDLO. The only difference is that, the instances within a cluster have the equal probability to be picked up. The quality of the results of EPMDLO is based on the linear correlation between the value of L' and the accuracy of candidate solutions. We expect that the linear correlation could be the minimal value -1. In other words, the ideal situation is that, when the value of L' goes down, the accuracy of candidate solutions should be going up.

- *The Equal Probability Agreement Fitness Optimisation*

The whole process of Equal Probability Agreement Fitness Optimisation (EPAFO) is the same as the process of EPMDLO but with a different objective function. The objective function of EPAFO is an agreement fitness function. EPAFO aims to seek the solutions that maximise the value of the agreement fitness function. According to our previous works in [Li *et al.*, 2009; Swift *et al.*, 2004], the agreement fitness function is defined as equations (5.22)-(5.25):

$$\beta = [\mathrm{Max}(A) - \mathrm{Min}(A)] \times 0.6 + \mathrm{Min}(A) \tag{5.22}$$

$$A' = A - \beta \tag{5.23}$$

$$f(C_i) = \begin{cases} \sum_{k=1}^{S_i-1} \sum_{q=k+1}^{S_i} A'_{C_{ik}C_{iq}}, & S_i > 1 \\ 0, & otherwise \end{cases} \tag{5.24}$$

$$f(C) = \sum_{i=1}^{m} f(C_i) \tag{5.25}$$

β is the agreement threshold, which rewards clusters with agreement values above β, and penalises clusters with agreement values below it; Max(A) and Min(A) are the maximum agreement value and the minimum agreement value respectively in the agreement matrix A. A' is the weighted agreement matrix, which is given by subtracting β from each element of A. The leading diagonal elements of A' are set to be zeros. $f(C_i)$ is the agreement fitness for the ith cluster of the clustering arrangement C. The variable $A'_{C_{ik}C_{iq}}$ indicates the corresponding weighted agreement value, which is related to the instances k and q in the ith cluster C_i, in A'. The variable S_i denotes the size of the ith cluster (i.e. the number of instances in the ith cluster). $f(C)$ indicates the total agreement fitness of the clustering arrangement C, where the number of clusters in C is m.

- *Relations of the three optimisation phases*

These three optimisation phases are complementary to each other. The key characteristic of EPMDLO is that it mainly focuses on describing interior characteristics of data sets, so that it can abate the effect of biases of input clusterings. EPAFO combines external information (i.e. input clusterings) to seek an optimal solution with the maximal agreement. ABMDLO integrates the comments (agreements) of input clusterings with the description of interior characteristics of data sets to seek the optimal solution. In this way, the three optimisation phases can abate the biases of both internal and external information.

Based on the results of EPMDLO, ABMDLO and EPAFO, the final result is generated by selecting the best solution from these results. A validation vector, which combines five validation indices, is developed to evaluate the quality of these results.

5.3.4 Validation Vector of Five Indices

The clustering validation indices are basically classified into two types: the internal validation indices, and the external validation indices [Halkidi, 2001]. External validation indices evaluate clustering results based on prior knowledge (e.g. the true clusters of the given data set), and measure how close the generated clustering results and the prior knowledge are. Internal validation indices do not rely on prior knowledge. They evaluate clustering results based on the given data set itself. During the process of clustering a given data set, internal validation indices can only be used to evaluate the quality of candidate clustering solutions.

For the KACC algorithm, we developed a validation vector to combine five internal validation indices to evaluate the quality of clustering solutions. The five validation indices are Silhouette Width [Rousseeuw, 1987], Homogeneity [Shamir and Sharan, 2002], Separation [Shamir and Sharan, 2002], Davies-Bouldin Index [Davies and Bouldin, 1979], and C-Index [Hubert and Schultz, 1976]. The following five paragraphs give a brief description of these indices.

Silhouette Width [Rousseeuw, 1987] was firstly proposed by Rousseeuw in 1987. It is calculated as equation (5.26):

$$SW = \frac{1}{n}\sum_{i=1}^{n}\frac{R(i) - r(i)}{\max\{R(i), r(i)\}} \tag{5.26}$$

where *n* is the total number of instances, *R*(*i*) denotes the average distance between instance *i* and the instances in the nearest cluster (which is the closest neighbour to the cluster instance *i* belongs to), and *r*(*i*) indicates the average distance between instance *i* and other instances in the same cluster. The value of SW is expected to be maximised.

Homogeneity [Shamir and Sharan, 2002] was suggested by Shamir and Sharan. Its definition is as follows:

$$H = \frac{1}{n}\sum_{i=1}^{n} d(n_i, Ctr_{n_i}) \tag{5.27}$$

where *n* is the number of instances. $d(n_i, Ctr_{ni})$ indicates the distance between the instance n_i and the cluster centre Ctr_{n_i} that n_i belongs to. A smaller *H* infers a better clustering result.

Separation [Shamir and Sharan, 2002] was also proposed by Shamir and Sharan. It is defined as equation (5.28):

$$Sep = \frac{\sum N_i N_j d(c_i, c_j)}{\sum N_i N_j} \quad , (i \neq j) \tag{5.28}$$

where c_i and c_j denote the cluster centres of the *i*th and *j*th clusters respectively. N_i is the number of instances in the *i*th cluster, and N_j is the number of instances in the *j*th cluster. A larger *Sep* indicates a better clustering result.

Davies-Bouldin Index [Davies and Bouldin, 1979] is defined as the average maximal ratio of intra-cluster mean distances and inter-cluster distances across all clusters. The definition is shown as equation (5.29):

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max\left(\frac{\theta_i + \theta_j}{d(c_i, c_j)}\right) \quad , \begin{matrix}(i \neq j)\\(j = 1, \dots, k)\end{matrix} \tag{5.29}$$

where k is the total number of clusters. $d(c_i, c_j)$ is the distance between the centre of cluster i and the centre of cluster j. θ_i is the average distance between the cluster centre c_i and all instances in cluster i; and θ_j is the average distance between c_j and all instances in cluster j. The smaller the value of DB is, the better the clustering result is.

C-Index [Hubert and Schultz, 1976] was proposed by Hubert. The definition is shown as equations (5.30) - (5.31):

$$CI = \frac{Q - Q_{P_{min}}}{Q_{P_{max}} - Q_{P_{min}}} \tag{5.30}$$

where

$$Q = \sum_{i=1}^{k} \sum_{a,b=1}^{N_i} d(n_{ia}, n_{ib}) \quad , (a \neq b). \tag{5.31}$$

In equation (5.30), Q is the sum of distances between unique instance-pairs within each cluster, and p stands for the number of unique instance-pairs in Q. $Q_{P_{min}}$is the sum of p smallest distances between all unique instance-pairs, and $Q_{P_{max}}$is the sum of p largest distances between all unique instance-pairs. In equation (5.31), k is the total number of clusters. N_i is the number of instances in the cluster i. The variable n_{ia} denotes the *a*th instance in the cluster i, and n_{ib} denotes the *b*th instance in the same cluster i. $d(n_{ia}, n_{ib})$ indicates the distance between n_{ia} and n_{ib}. The value of CI is expected to be as small as possible within the interval [0, 1].

For more details of these validation indices, readers are referred to the respective literature [Davies and Bouldin, 1979; Hubert and Schultz, 1976; Rousseeuw, 1987; Shamir and Sharan, 2002] given in the Reference Section.

Decisions of the above five validation indices are mapped into a validation vector. The vector is one dimension and contains five elements. Each element corresponds to the decision of one validation index. The validation vector is defined by equations (5.32) - (5.33):

$$I = [x_1,\ x_2,\ x_3,\ x_4,\ x_5] \tag{5.32}$$

where

$$x_i = \begin{cases} 0 \\ 1 \\ 2 \end{cases} \quad ,(i = 1, 2, \dots, 5) \tag{5.33}$$

If a validation index *i* proves the result of EPMDLO is better than those of ABMDLO and EPAFO, the corresponding x_i will be 0. If index *i* proves the result of ABMDLO is the best, the x_i will be 1; and the x_i will be 2 if EPAFO has been proved to have the best result. The final clustering result will be determined by comparing the number of 0 with the number of 1, and with the number of 2 in the validation vector. If a result has the most votes, it will be the winner and be selected as the final result.

5.3.5 The whole framework of KACC

In order to outline the whole framework of KACC, we describe its key steps in Fig. 34.

Input parameters: the clustering arrangement matrix *Z*, the estimated number of clusters *k*, the original data set *M*, the maximal number of iterations N_K.

1. Build the Agreement Matrix *A* based on the input clustering arrangement matrix *Z*.
2. Assign unique instance pairs into different agreement lists based on their agreement values in *A* (sorted from large agreement values to small ones).
3. Generate the initial clustering solution C_0 by assigning all instances into *k* clusters based on the sorted Agreement Lists. The process starts from the agreement list with the maximal agreement value, followed by the list with the secondary agreement value, and so on.
4. Standardise the original data set *M* to be *X*, where the values of each attribute are scaled into [0, 1].
5. Calculate the weighted coefficient for each attribute of *X* based on variances of attributes.
6. Describe the initial clustering solution C_0 to obtain the initial description length L_0 (can be simplified to be L'_0.) based on the AWDL criterion.
7. Use the novel K-Ants algorithm to start to optimise the initial clustering solution C_0.
    a) Perform the Agreement Based MDL Optimisation (ABMDLO). This process is continued until the value of L' is converged or the predefined number of iterations N_K is reached.
    b) Perform the Equal Probability MDL Optimisation (EPMDLO). This process is continued until the value of L' is converged or the number of iterations N_K is reached.
    c) Perform the Equal Probability Agreement Fitness Optimisation (EPAFO). This process is continued until the value of L' is converged or the number of iterations N_K is reached.
8. Use a validation vector to evaluate the results of ABMDLO, EPMDLO and EPAFO, and select the best one to be the final result.

Output: an optimised clustering solution.

Fig. 34: The key steps of K-Ants Consensus Clustering algorithm

5.4 Data Sets and Experiments

5.4.1 Data sets

Two groups of data sets have been used for our experiments. The information of these data sets is listed in Table 11. For each data set, the information includes the total number of instances, the number of attributes, and the true number of clusters. Group 1 consists of four data sets: Image (Image Segmentation), Iris, Wine, and Zoo. This group of data sets are used for comparing the performance between our KACC algorithm and the Ensemble Clustering method ESIC (Ensemble of Swarm Intelligence Clustering) proposed in the literature [Yang and Kamel, 2003]. Group 2 contains six data sets, which are SCCTS (Synthetic Control Chart Time Series), E-coli, Malaria, Normal, VAR (Vector Auto-Regressive), and WDBC (Wisconsin Diagnostic Breast Cancer). The sizes of the data sets in Group 2 are comparative bigger than those in Group 1, and most data sets in Group 2 have clear higher dimensionalities.

Table 11: The data sets for experiments

	Data Sets	**Number of Instances**	**Dimensionality (Number of Attributes)**	**Number of Clusters**
Group 1	Image	210	19	7
	Iris	150	4	3
	Wine	178	13	3
	Zoo	101	16	7
Group 2	SCCTS	600	60	6
	E-coli	336	7	8
	WDBC	569	30	2
	Malaria	530	48	14
	Normal	1830	20	60
	VAR	900	600	42

Note: We define the rows of each data set as instances, and the columns of each data set as attributes of instances. For a given data set, all instances have the same dimensionality.

All data sets in Group 1 and the three data sets, SCCTS, E-coli and WDBC, in Group 2 were obtained from the UCI Machine Learning Repository website [Asuncion and Newman, 2007]. Since this website provides detailed information for each of these data sets, readers are referred to [Asuncion and Newman, 2007] for more information. The other three data sets are VAR, Normal, and Malaria.

VAR [Lütkepohl, 2007] is a synthetic data set that was generated from a vector autoregressive process [Sims, 1980]. In the original VAR data set, there are 60 clusters. The number of instances in clusters varies from 1 to 60. In other words, the first cluster of VAR has one instance; the second cluster of VAR has two instances; and so on. Therefore a subset (which consists of the first *m* clusters of VAR, where *m*<60) has the same capability of testing clustering algorithms as the whole VAR data set without biasing results. In order to complete our experiments in a feasible amount of time, we only choose the first 42 clusters of the original VAR data set. The Normal data set was used in the literature [Swift *et al.*, 2004]. Normal has a similar cluster structure to the VAR data set, but the main difference is that each cluster of Normal was generated from a multivariate normal distribution with different covariance and means [Swift *et al.*, 2004]. Normal has 60 clusters. The number of instances of each cluster varies from 1 to 60. Malaria [Bozdech *et al.*, 2003] is a microarray data set which has known true clusters: within each cluster, genes should have the same function; between clusters, different genes have different functions.

5.4.2 Experiments Setup

In order to validate the performance of KACC, we designed two groups of experiments. The first group is designed to compare the performance between KACC and ESIC. The Group 1 data sets are used for this group of experiments. For each data set, we run KACC for 20 times and look at the average performance. Since we have the true clusters of each data set, we use external validation indices to evaluate the accuracy of clustering results. Yang and Kamel employed an external index, F-measure [Rijsbergen, 1979] (to be described in Section 5.4.3), to evaluate the performance of ESIC in the literature [Yang and Kamel, 2003]. In order to compare KACC with ESIC, we also use F-measure to evaluate the performance of KACC.

In order to further compare KACC with other consensus clustering methods, we designed the second group of experiments. The Group 2 data sets are used for these experiments. Seven consensus clustering algorithms in the CLUE package (described in Section 5.2.5) are tested to compare with KACC. We analyse the average performance of 20 runs for each algorithm tested on each data set. For evaluating the accuracy of these results, we not only use F-measure but also utilise another external validation index, Weighted-Kappa [Viera, and Garrett, 2005] (described in Section 2.4.1), to verify the results.

In addition, for each group of experiments, the results of KACC are also compared with the Input Clusterings (generated by input clustering algorithms) to see how much the clustering accuracy has been improved by our KACC algorithm. In this chapter, we use the most common and simplest clustering algorithms, HC [Johnson, 1967; D'andrade, 1978; Murtagh, 1983] and K-means [Mcqueen, 1967], to be the input algorithms for KACC. These input clustering algorithms are listed in Table 12. For a given data set, each algorithm generates an input clustering, and then the total eight input clusterings form into the Input Clustering Matrix.

Besides, it is worth to note that the value of the maximal number of iterations N_K is based on given data. In this chapter, we set N_K to be 30, which is appropriate and large enough for our experiments.

Table 12: A set of input clustering algorithms

Algorithm	**Parameters**		
	Distance Matrix	**Linkage**	**Initial Cluster Centre**
Hierarchical (H)	Euclidean	Single	N/A
Hierarchical (HSC)	Spearman	Complete	N/A
Hierarchical (HCA)	Correlation	Average	N/A
Hierarchical (HJW)	Jaccard	Weighted	N/A
K-means (K)	Squared Euclidean	N/A	Sample
K-means (KCorU)	Correlation	N/A	Uniform
K-means (KCS)	Cosine	N/A	Sample
K-means (KCosU)	Cosine	N/A	Uniform

5.4.3 Two External Validation Indices

For each data set used in this chapter, we have the known clusters (i.e. the true clusters). The simplest way of evaluating the accuracy of results is to compare the results with the known clusters to see how close they are. Therefore, in this chapter, we employ two external validation indices, F-measure and Weighted-Kappa, to evaluate the clustering results for the performance comparison.

- *F-measure Index*

F-measure was firstly derived by Van Rijsbergen in 1979 [Rijsbergen, 1979]. It is a comprehensive unary measure index, and contains two key conceptions: Precision and Recall. Precision indicates the fraction of a true cluster that is taken up by a cluster of an obtained clustering result for a given data set [Rijsbergen, 1979]. Recall means the fraction of a generated cluster that is included by a true cluster for a given data set [Rijsbergen, 1979]. The definition of Precision is given by equation (5.34)

$$P\,r(c_i, t_j) = \frac{N_{ij}}{N_j}, \tag{5.34}$$

where c is the generated clustering result of a given data set, and t is the true clusters of the data set. The variable c_i indicates the ith cluster of c; and t_j is the jth cluster of t. N_{ij} is the number of instances in the intersection of c_i and t_j, and N_j means the number of instances in the true cluster t_j. Equation (5.35) defines Recall as

$$R\,e(c_i, t_j) = \frac{N_{ij}}{N_i}, \tag{5.35}$$

where N_i denotes the number of instances in the cluster c_i.

For a given data set, the F-measure between the jth true cluster and the generated ith cluster is calculated as equation (5.36). The F-measure between the *j*th true cluster and the whole clustering result *c* is defined by equation (5.37), where *l* denotes any cluster of the clustering result *c*, and k_c is the total number of clusters of *c*.

$$F(c_i, t_j) = \frac{2 \times Pr(c_i, t_j) \times Re(c_i, t_j)}{Pr(c_i, t_j) + Re(c_i, t_j)} \tag{5.36}$$

$$F(t_j) = \max_{1 \le l \le k_c} \left(F(c_l, t_j) \right) \tag{5.37}$$

To a generated clustering result *c*, its final F-measure is defined as the weighted average F-measure based on the true clusters *t*. The calculation is shown as equation (5.38):

$$F = \frac{\sum_{j=1}^{K_t} N_j F(t_j)}{\sum_{j=1}^{K_t} N_j} \tag{5.38}$$

where K_t is the number of clusters in *t*. For more information about F-measure, readers are referred to [Rijsbergen, 1979].

- *Weighted-Kappa Index*

The Weighted-Kappa (WK) [Viera, and Garrett, 2005] index has been introduced in Chapter 2.4.1.

5.5 Results and Discussion

This section illustrates and discusses results generated from the experiments mentioned in Section 5.4.2. The results shown by figures in Section 5.5 are the averages of 20 runs.

5.5.1 Comparison between KACC and ESIC

Before comparing the performance between KACC and ESIC, we first look at how much the accuracy has been improved by KACC based on the input clusterings.

Fig. 35 displays the F-measure score comparison between results of KACC and the mean accuracy of eight input clusterings for the Group 1 data sets. The black bars indicate the F-measure scores of the results of KACC, and the grey bars denote the average F-measure scores of the eight input clusterings. It is clear that the results of KACC are significantly better than the mean accuracies of input clusterings for these four data sets. In order to see the differences between each input clustering and the results of KACC, the accuracies of all input clusterings are illustrated by Fig. 36. We can see that, for data sets Image Segmentation, Wine and Zoo, KACC clearly generated better results than all of the input clusterings. For the data set Iris, the result of KACC is better than most of the input clusterings, and comparable to the best one among them. Based on Fig. 35 and Fig. 36, it is evident that KACC performed better than the input clustering algorithms for the Group 1 data sets.

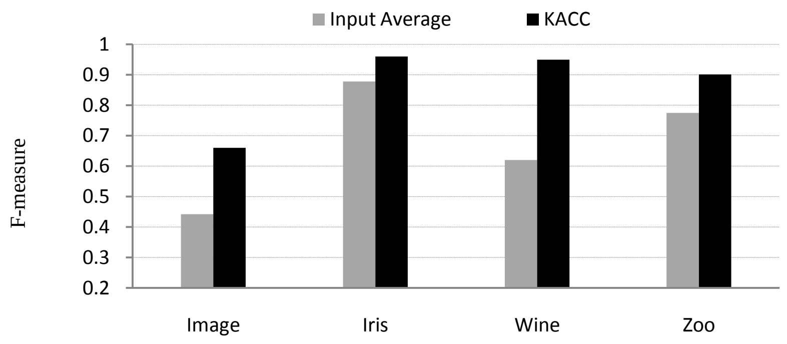
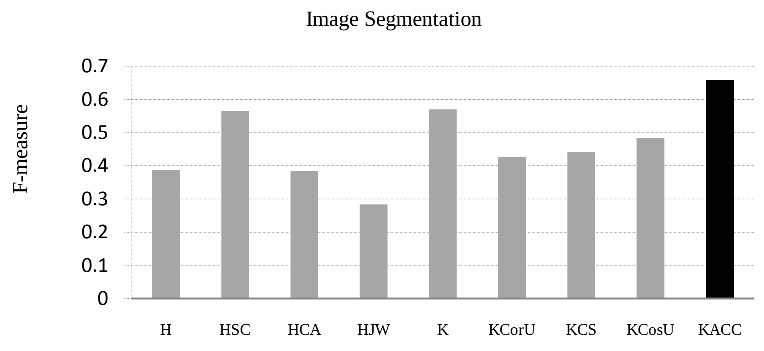
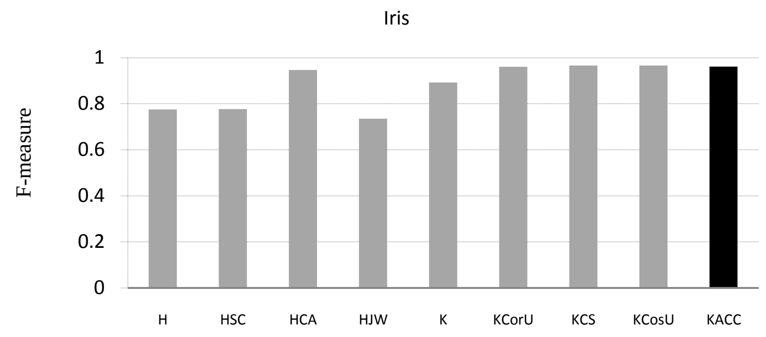


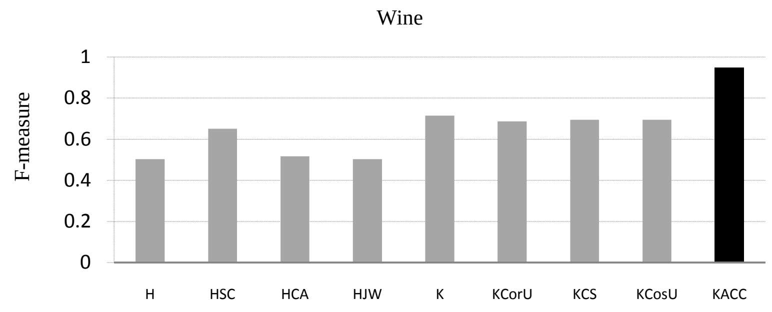
Fig. 35: The F-measure score comparison between the results of KACC and the average accuracy of the input clusterings for Group 1 data sets.



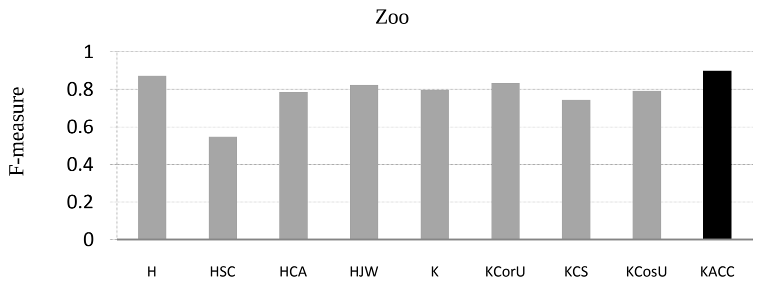
(a)



(b)



(c)



(d)

Fig. 36: The F-measure score comparison between the results of KACC and the eight input clusterings for the Group 1 four data sets.

The comparison of F-measure scores between KACC and ESIC is shown by Fig. 37. Since Yang and Kamel listed the highest, lowest and average F-measure scores of the results generated by ESIC in the literature [Yang and Kamel, 2003], we illustrate all these F-measure scores in Fig. 37. It is apparent that KACC clearly has better F-measure scores than ESIC especially for the Image and Wine data sets. Therefore, we can say that KACC performed significantly better than ESIC for these four data sets.

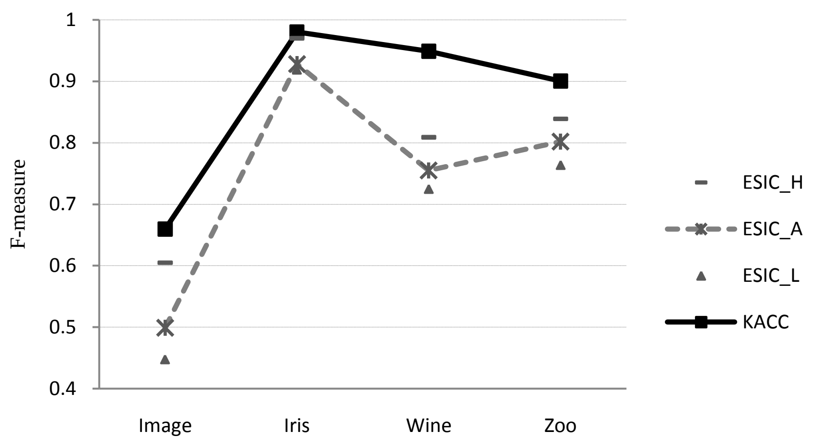
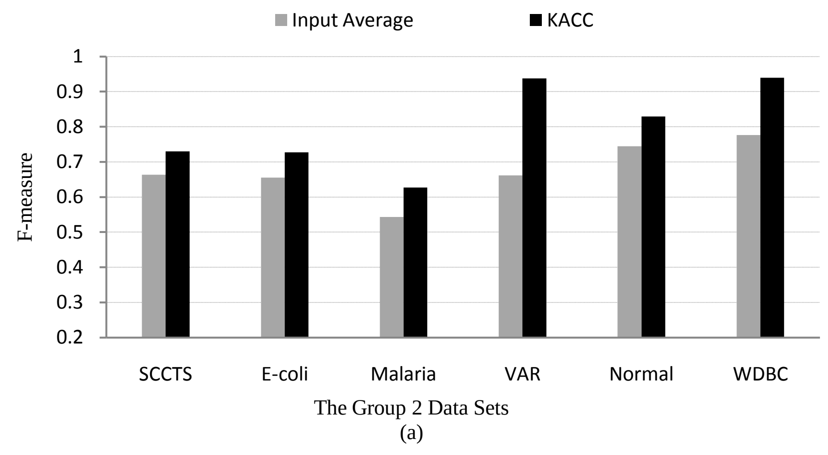


Fig. 37: The comparison of F-measure scores between KACC and ESIC for the four data sets of Group 1. (ESIC_H denotes the highest F-measure scores of results of ESIC; ESIC_L means the lowest scores; and ESIC_A is the average score of each data set.)

5.5.2 Comparison between KACC and Algorithms in CLUE

We also first compare the results of KACC with the input clusterings for the data sets of Group 2. We use both F-measure and WK to evaluate the accuracy of results. Fig. 38 illustrates the comparison between the average of the input clusterings and the results of KACC. Both F-measure scores and WK scores demonstrate that the results generated by KACC are much more accurate than the average of the input clusterings. In other words, the two validation indices all demonstrate that KACC has improved the clustering accuracy. Another important point is that the improvements of the accuracy illustrated by Fig. 38 (a) (the F-measure comparison) and (b) (the WK comparison) are quite similar to each other.



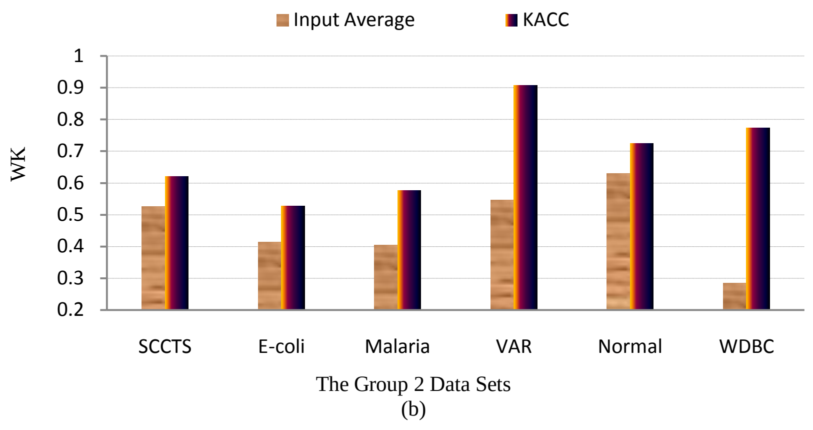
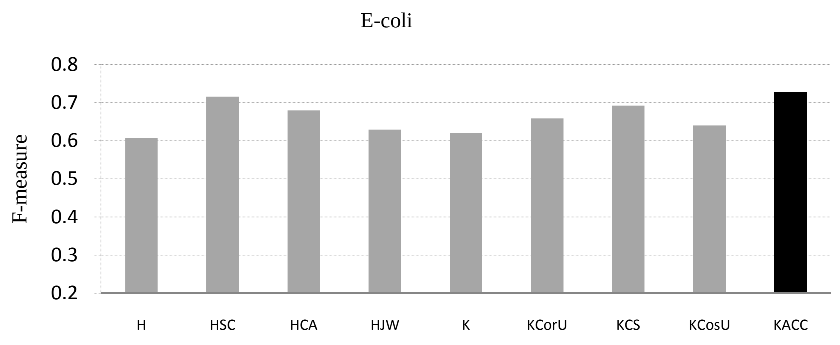
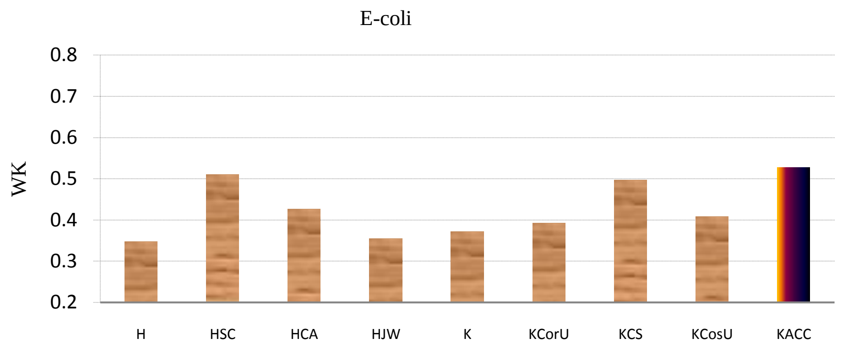


Fig. 38: The comparison between the results of KACC and the average of the input clusterings for the data sets of Group 2. (a) F-measure scores. (b) WK scores.

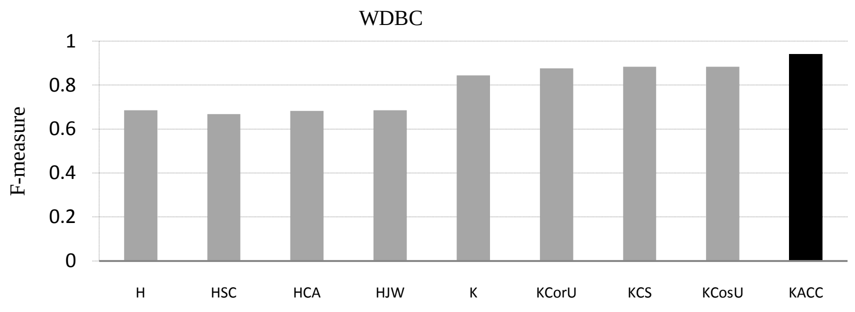
From Fig. 38, readers may notice that the WK scores are much lower than the F-measure scores in general (especially for the E-coli and WDBC data sets). Fig. 39 displays the evaluation of the eight input clusterings for the E-coli and WDBC data sets. In Fig. 39 (a) and (b), it is obvious that, for the same input clusterings, the WK scores are much lower than the F-measure scores. The obove situation also appears in Fig. 39 (c) and (d). Hence Fig. 39 interprets why the corresponding average WK scores of the input clusterings are much lower than the corresponding average F-measure scores in Fig. 38. In addition, Fig. 39 illustrates that the results of KACC are better than all input clusterings for E-coli and WDBC. It has been further verified that KACC has improved the accuracy of clustering by combining input clusterings.



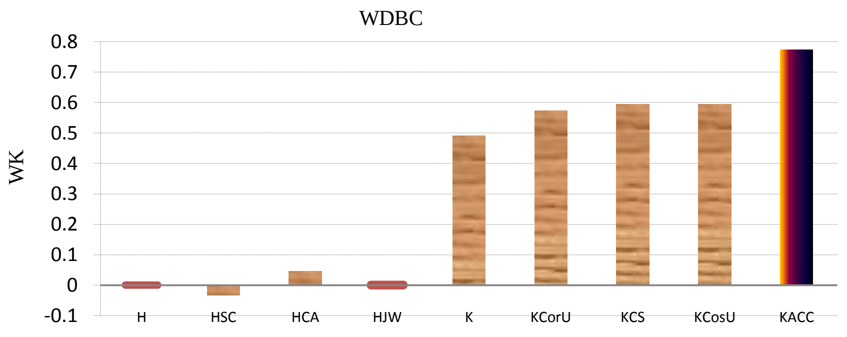
(a)



(b)



(c)



(d)

Fig. 39: The accuracy comparison between the results of KACC and the eight input clusterings for the E-coli and WDBC data sets. (a) F-measure scores for E-coli. (b) WK scores for E-coli. (c) F-measure scores for WDBC. (d) WK scores for WDBC.

Fig. 40 presents the accuracy comparison between KACC and the Seven Consensus Clustering Algorithms (SCCA) of the CLUE package. In Fig. 40, the left side pictures show the F-measure scores for the six data sets of Group 2, and the right side pictures display the corresponding WK scores.

For the SCCTS data set, the F-measure scores show that KACC generates better results than SCCA, while the corresponding WK scores of KACC and SCCA are quite similar to each other. For the Normal data set, although KACC only has a WK score comparable to the best one of SCCA, KACC has a good F-measure score that is better than all those of SCCA. Therefore, for data sets SCCTS and Normal, we can say that KACC generated results comparable to those of SCCA. For other data sets, both F-measure and WK scores demonstrate that the results of KACC are clearly better than those of SCCA. Specially, KACC performed extremely better than SCCA for E-coli, Malaria, and WDBC.

In addition, for the WDBC data set, the results generated by SCCA vary significantly. If we compare Fig. 39 (c) and (d) with Fig. 40 (f) and (vi), it is evident that SCCA is very sensitive to the accuracy of input clusterings for WDBC. It suggests that KACC has a better stability than SCCA.

Based on the above discussion, we can say that KACC clearly generated better results than the average of input clusterings, and performed comparably or better than the Seven Consensus Clustering Algorithms of the CLUE package.

5.5.3 Contributions of ABMDLO, EPMDLO and EPAFO

We take three data sets as examples to describe the contributions of ABMDLO, EPMDLO and EPAFO. From the results discussed in Sections 5.5.1 and 5.5.2, it is clear that F-measure scores are always higher than WK scores. In other words, the F-measure score will be good if the WK score is good.[1] Therefore we only illustrate the traces of WK scores for analysing the three optimisation phases.

[1] *We have analysed the correlation between F-measure and WK, and found that they have a very high correlation. Since this correlation analysis is out of the scope of this thesis, we do not describe it here.*

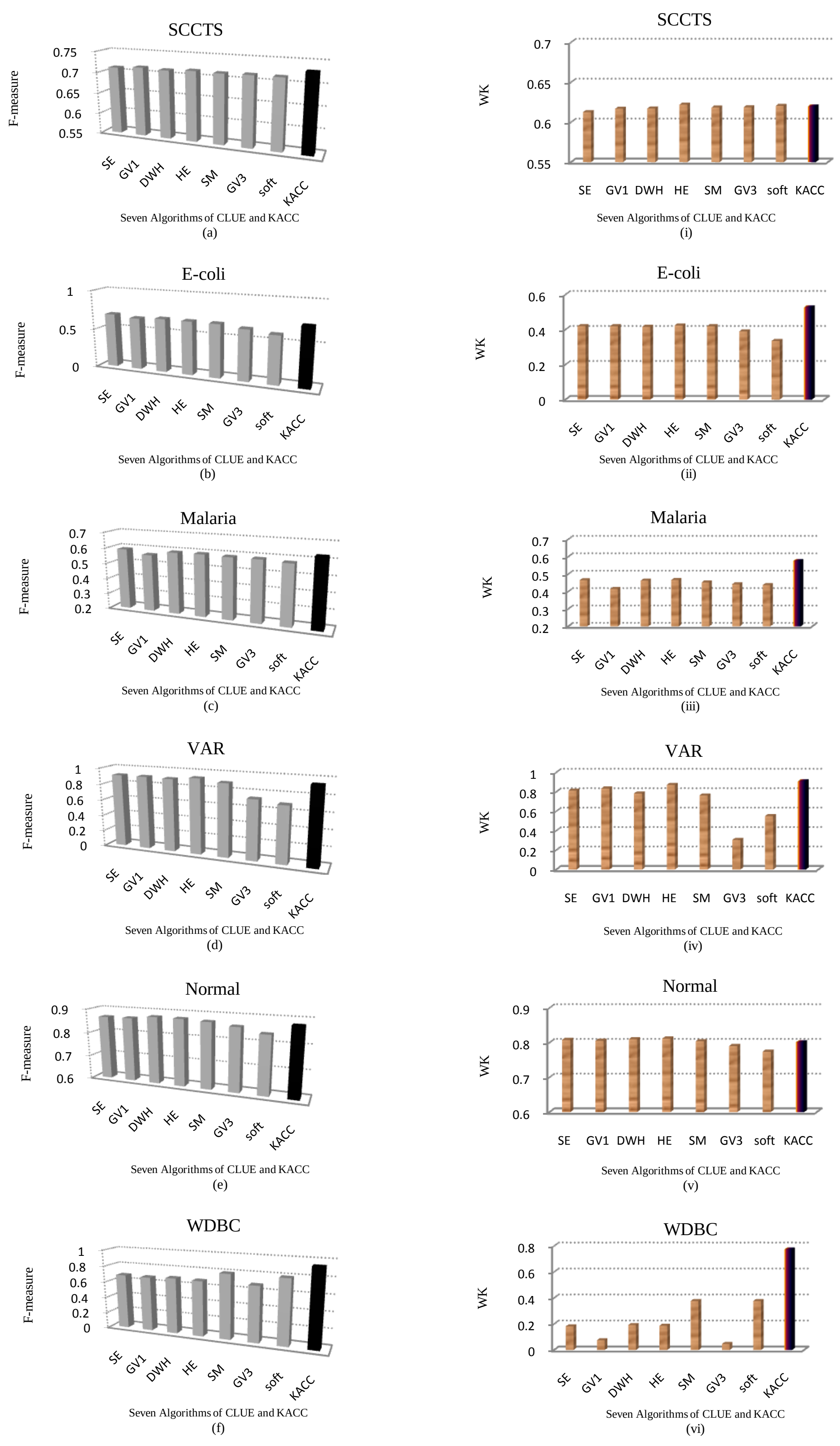


Fig. 40: The comparison between KACC and the seven consensus clustering algorithms of CLUE for the six data sets of Group 2 via the validation of F-measure and WK. The left side pictures (from (a) to (f)) display the F-measure scores. The right side pictures (from (i) to (vi)) display the WK scores.

First of all, we take the E-coli data set as an example to illustrate the contribution of ABMDLO in Fig. 41. Pictures (a1) and (a2) show the traces of the AWDL value and the corresponding WK scores respectively for the EPMDLO phase; pictures (b1) and (b2) display the traces for the ABMDLO phase; and pictures (c1) and (c2) illustrate the traces of Agreement Fitness values and WK scores respectively for the EPAFO phase.

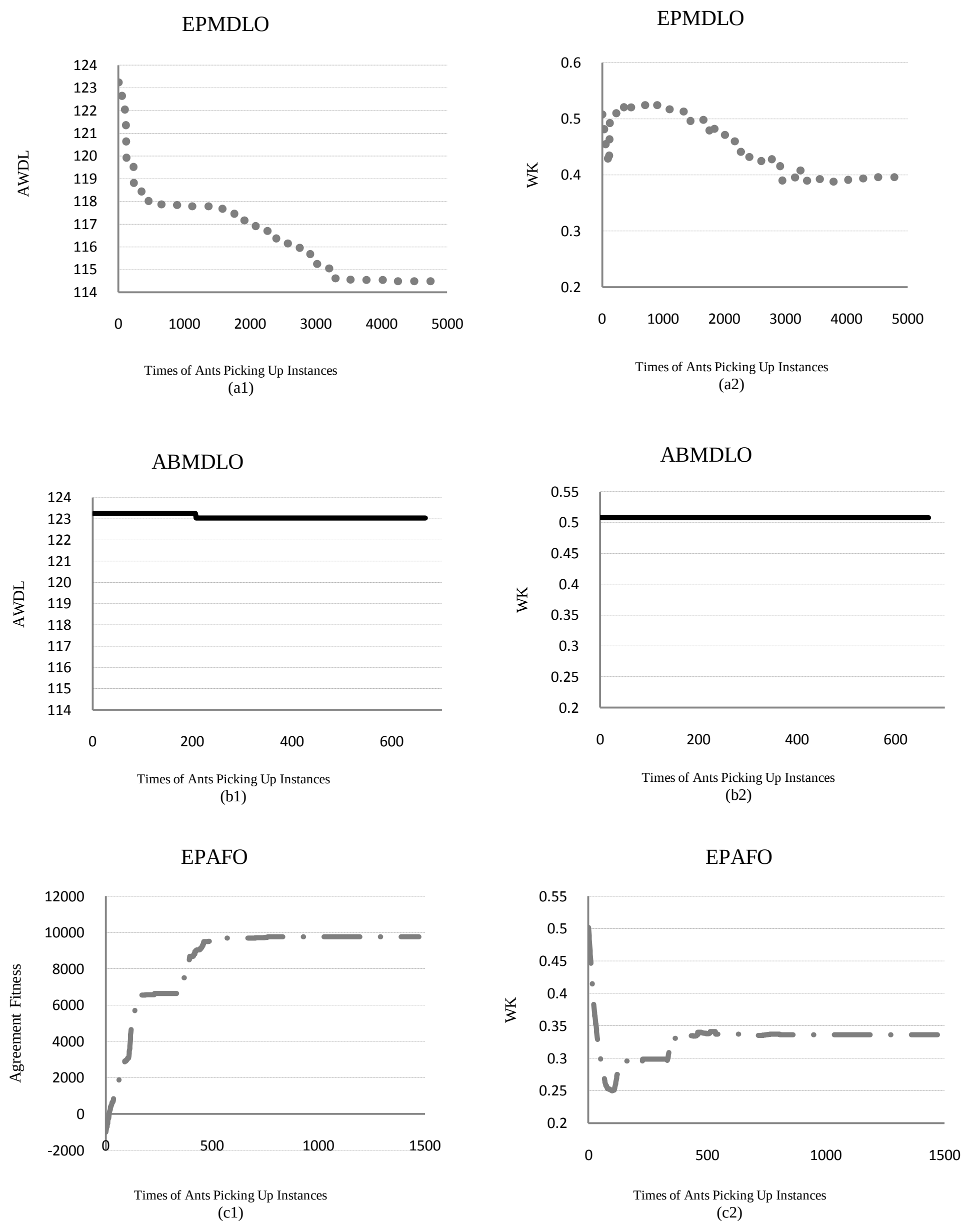


Fig. 41: Take the E-coli data set as an example for illustrating the contribution of ABMDLO.

In Fig. 41 (a1) and (a2), during the process of EPMDLO, the AWDL value becomes lower and lower, while the WK score does not increase correspondingly. Picture (a2) shows that, after a rapid increase at the beginning of EPMDLO, the WK score started to decrease significantly when the AWDL value goes down. It means that minimising the AWDL value does not obtain better clustering solutions for the E-coli data set. For the ABMDLO process, Fig. 41 (b1) and (b2) clearly show that the WK score is kept at a reasonable high level when it is compared with Fig. 41 (a2) and (c2).

Now we analyse the traces of Agreement Fitness values and WK scores for the EPAFO process. The Agreement Fitness value is expected to be maximised so that we can get better results. However, the results shown in Fig. 41 (c1) and (c2) are very poor. The WK score decreased swiftly while the Agreement Fitness value was going up. It means that only maximising the Agreement Fitness value is not good for clustering this data set. After analysing Fig. 41, it is clear that ABMDLO generated the best result for E-coli, which is also confirmed by the Validation Vector.

Secondly, the Image Segmentation data set is taken as an example for demonstrating the contribution of EPMDLO. In Fig. 42, the first four pictures, (a1), (a2), (b1) and (b2), show that WK scores of both EPMDLO and ABMDLO are going up when the AWDL value is going down. Fig. 42 (c1) and (c2) display that the WK score is risen considerably when the Agreement Fitness value is increasing.

As a whole, Fig. 42 shows that all of the three optimisation phases, EPMDLO, ABMDLO and EPAFO, can increase the clustering accuracy for the Image data set. Consequently, we need to validate which optimisation is the most efficient. From Fig. 42 (a2), (b2) and (c2), we can see that the result of EPMDLO has the highest WK score. Moreover, the linear correlation between the AWDL value and the WK score for the EPMDLO process is -0.96501, which is better than those of the ABMDLO and EPAFO processes. Therefore we can reach the conclusion that EPMDLO is the most efficient for the Image data set, which is confirmed by the Validation Vector.

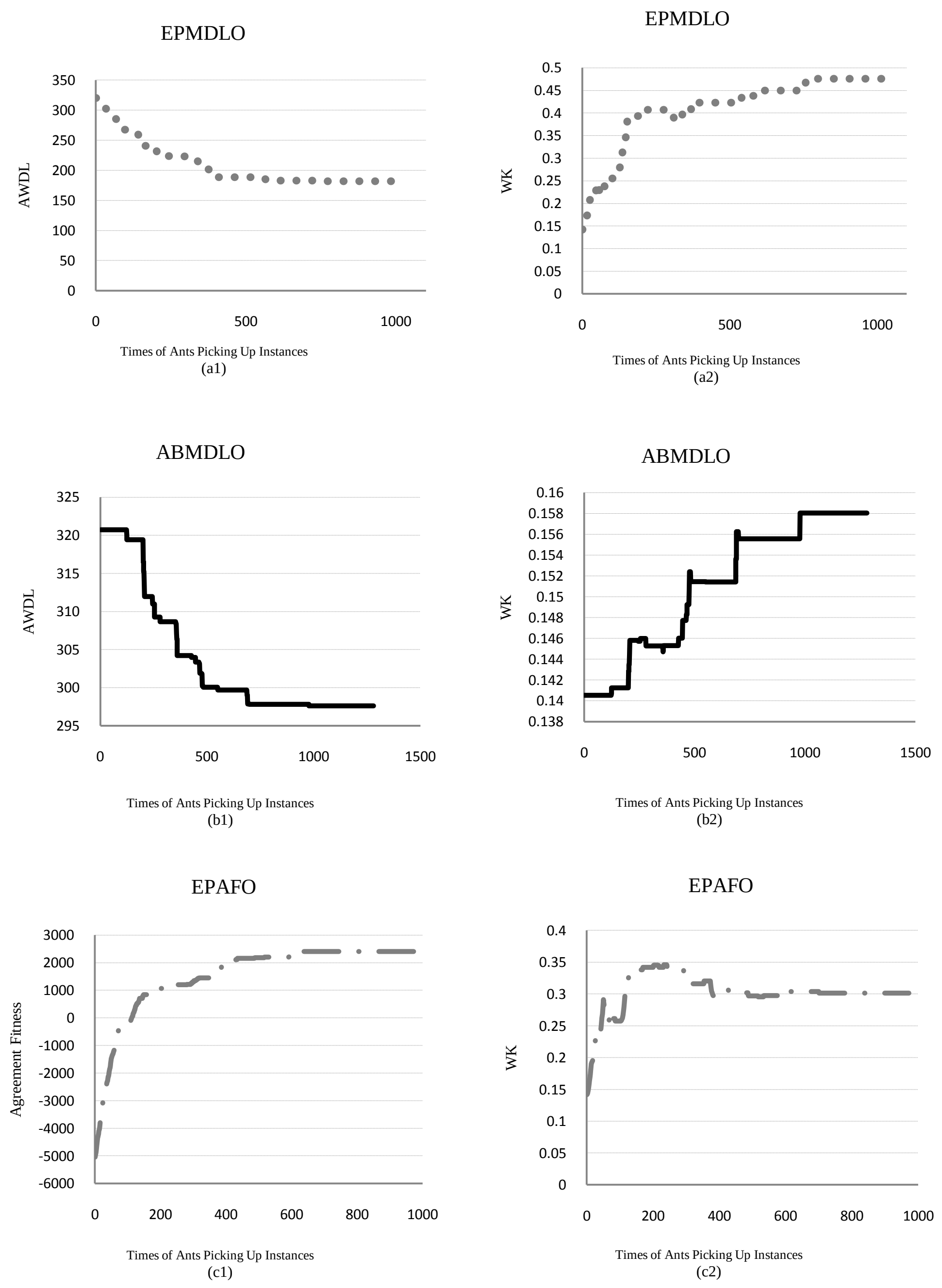


Fig. 42: Take the Image Segmentation data set as an example for illustrating the contribution of EPMDLO.

Finally, Fig. 43 illustrates the contribution of EPAFO by taking the Normal data set as an example. Fig. 43 (a1) and (a2) show that the WK score declines sharply during the EPMDLO process. In other words, EPMDLO is not suitable for Normal. If we compare Fig. 43 (a2) with (b2), it can be seen that ABMDLO has a much better trace of WK scores than EPMDLO (although the WK score of ABMDLO decreases remarkably). The best clustering result is obtained from EPAFO and shown by Fig. 43 (c1) and (c2). We can see that, when the Agreement Fitness value increases, the WK score only has a slight undulation at the beginning, and then it remains stable until it finally converges at 0.8. Therefore, the result of EPAFO has been selected by the Validation Vector to be the final result of KACC.

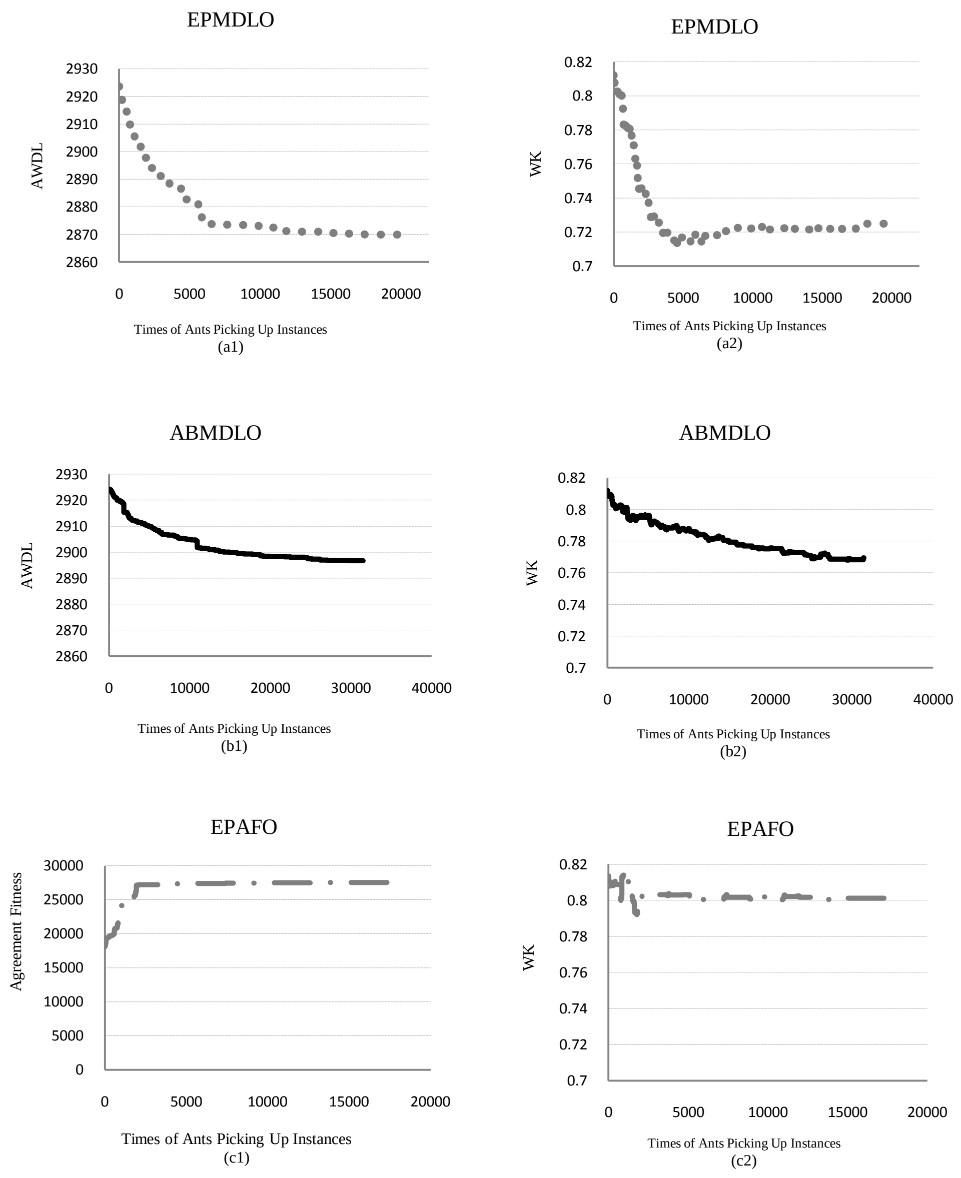


Fig. 43: Take the Normal data set as an example for illustrating the contribution of EPAFO.

5.6 Summary

This chapter has introduced a new Ensemble Clustering method called K-Ants Consensus Clustering (KACC), which is developed by integrating the MDL principle and the Ant Colony clustering techniques. KACC consists of three optimisation phases: EPMDLO, ABMDLO and EPAFO. The synthesis of these optimisation phases achieves the combination of internal and external clustering information.

The results of KACC were firstly compared with the input clusterings. The comparison shows that KACC generated significantly better results than the average accuracy of the input clusterings. Secondly, KACC was compared with a comparable ant colony-based Ensemble Clustering method called ESIC and seven consensus clustering algorithms of CLUE respectively. The results demonstrate that KACC clearly performed better than these clustering algorithms. Finally, this chapter illustrates the contributions of the three optimisation phases (EPMDLO, ABMDLO and EPAFO), and demonstrates that they can be good complement to each other.

A possible limitation of KACC is that it requires prior knowledge of the number of clusters. KACC uses the estimated number of clusters *k* (shown in Fig. 34) as one of its input parameters. Since *k* is an estimated number, the quality of clustering results of KACC could be affected by the accuracy of *k*. It is very important to have an appropriate estimation to predict the number of clusters *k*. There are many different ways to estimate the number of clusters for a given data set. Some widely used estimation techniques can be found in the following literatures: [Choi, Kim and Choi, 2006], [Tibshirani, Walther and Hastie, 2001], [Fraley, Adrian and Raftery, 1998], [Fridlyand, Dudoit and Dudoit, 2001], and [Cuevas, Febrero and Fraiman, 2000].

In this chapter, we used ideally clean data sets to test different clustering methods. Only using clean data sets is the limitation of our experiments. This limitation is the same as in Chapter 3. Therefore, in future work, we will analyse how robust KACC is for dealing with noisy data by testing KACC on a large range of data sets that have noise and outliers.

6

Chapter 6: Performance Comparison between CC, MOCC & KACC

In order to enhance the clustering accuracy based on an existing heuristic optimisation-based Ensemble Clustering method - Consensus Clustering (CC), we developed a novel method called Multi-Optimisation Consensus Clustering (MOCC), which was presented in Chapter 3; in Chapter 5, we presented another novel Ensemble Clustering method called K-Ants Consensus Clustering (KACC). The reason why we developed KACC after MOCC is that MOCC has very high computational costs. In order to understand the advantages of KACC and differences between performances of KACC, MOCC and CC, this chapter gives a performance comparison amongst the three Ensemble Clustering methods.

We achieve the performance comparison from two aspects: the accuracy of clustering, and the efficiency of clustering. For the accuracy comparison, we still use the Weighted-Kappa (WK) [Viera and Garrett, 2005] validation index to evaluate clustering results. For the efficiency comparison, we focus on analysing the time complexity and time cost (i.e. time of implementation) of the three methods. In the following sections, we will describe data sets and experiments, and then discuss the accuracy comparison and the efficiency comparison in Section 6.2 and Section 6.3 respectively. The final section gives a summary of our discussion.

6.1 Data sets and experiments

We continue to use the ten data sets listed in Table 11 to test performances of CC, MOCC and KACC. The designed experiments are outlined as follows:

- Each of CC, MOCC and KACC is run 20 times on each data set.
- We record clustering results and time cost for each run.
- WK is used to evaluate accuracies of clustering results.
- We analyse the average performance across the 20 individual runs for each Ensemble Clustering method.
- The number of iterations N_C of CC is set to be the default value i.e. 1,000,000 based on the discussion in Chapter 4.
- The number of iterations N_M of MOCC is also set to be 1,000,000.
- The maximal number of iterations N_K of KACC is set to be 30, which is the same setting as in Chapter 5.4.

The details of experimental facilities are as follows:
- The experimental facility: Laptop
- Manufacturer: Sony Corporation
- Type: VGN-CR Series
- Operating System: Windows Vista, 32 bits
- RAM: 1GB
- CPU: Intel(R) Core(TM)2 Duo, T5450, 1.67GHz

6.2 Comparison of the Accuracy of Clustering

Hirsch *et al.* [2007] compared CC with seven consensus clustering algorithms provided by the CLUE [Hornik, 2005] package, and claime that CC generated results comparable to these methods. In Chapter 5, we compared KACC with the same seven consensus clustering algorithms, and the results demonstrate that KACC clearly generated better results than these algorithms. Thus we can draw a conclusion that the clustering accuracy of KACC can be better than the one of CC. On the other hand, the clustering accuracy comparison between CC and

MOCC was described in Chapter 3, where the results demonstrate that MOCC generated comparable or even better results than CC. Therefore we can find out which method has the best clustering accuracy by only comparing KACC with MOCC. The accuracies of clustering results for KACC and MOCC are illustrated in Fig. 44.

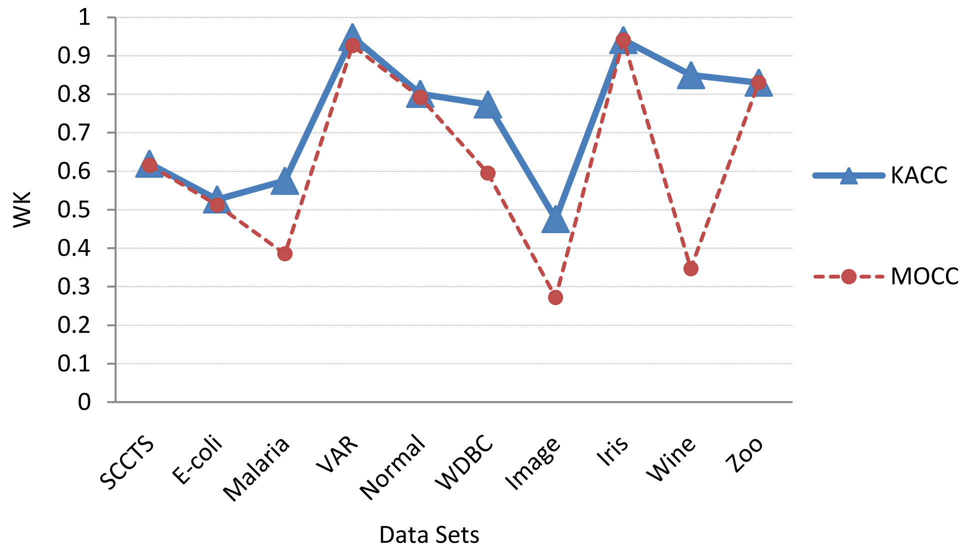


Fig. 44: The accuracy comparison between the clustering results of KACC and MOCC (tested on ten data sets).

The WK scores in Fig. 44 indicate accuracies of clustering results. For the data sets Malaria, WDBC, Image and Wine, Fig. 44 shows that the results of KACC are much more accurate than those of MOCC. For other data sets, the results of KACC are comparable to or have the same accuracy as those of MOCC. Consequently, we can say that KACC generated comparable or better results than MOCC in general.

Both MOCC and KACC aim to enhance the accuracy of clustering by combining internal and external information. MOCC uses the Agreement Matrix, the Agreement Fitness function, and the Separation index to describe and combine external and internal information. KACC utilises three optimisation phases, ABMDLO, EPMDLO, and EPAFO, to achieve the information integration. The results in Fig. 44 demonstrate that the integration strategy of KACC performs better than the one of MOCC for improving the accuracy of clustering.

From the above discussion and the extensive results analysed in Chapter 3 and Chapter 5, we can reach the conclusion that KACC has the best clustering accuracy among the three Ensemble Clustering methods, and MOCC is not far in the second place.

6.3 Comparison of the Clustering Efficiency

6.3.1 The Time Complexity

When using CC to analyse a given data set, time is mostly spent for implementing the heuristic optimisation part. CC utilises SA to achieve the heuristic optimisation, where the number of iterations and the quantity of computations for iteration are key factors for the time complexity. Suppose we have a data set with n instances. The number of iterations is N_C, and each iteration has m operations. Hence the time complexity of CC is $O(mN_C)$, where m is less than or equal to n. CC has three different operations, Move, Split and Merge, for generating new candidate solutions. The worst case for m is that splitting the whole data set (with n instances) into two clusters or vice versa. Therefore, in the worst case, the time complexity of CC is $O(nN_C)$.

MOCC also utilise SA for its heuristic optimisation part, and the solution generator also implements three different operations that are the same as those of CC. Thus the time complexity of MOCC seems to be similar to the one of CC. However, the key difference between MOCC and CC is that MOCC implements multiple optimisations, which are the agreement fitness optimisation and the separation optimisation. Therefore, for each iteration, MOCC requires double operations to evaluate candidate solutions. According to these characteristics of MOCC, its time complexity can be $O(2mN_M)$, where N_M is the number of iterations, and $m \leqslant n$. In the worst case, the time complexity of MOCC is $O(2nN_M)$.

KACC contains three heuristic optimisation phases, which are the main factors for its time complexity. These optimisation phases have the same maximal number of iterations. During each iteration, each instance needs to be selected by an ant. Thus, for a data set with *n* instances, there will be *n* operations for each iteration. The time complexity of KACC, therefore, will be $O(3nN_K)$, where N_K is the maximal number of iterations.

From the above discussion, it seems that CC, MOCC and KACC have similar time complexities. However, it is important to note that N_C and N_M could be much larger than N_K. For example, in this thesis, N_C and N_M have the same default value that is 1,000,000, but N_K is 30. In this case, the time complexity of KACC is much lower than those of CC and MOCC, moreover, the time complexity of MOCC is a double of the one of CC.

6.3.2 Time Cost

In order to further understanding how different time complexities the three Ensemble Clustering methods have, we recorded their time costs when these methods analysing ten data sets respectively. Table 13 lists average time costs across 20 runs for each method analysing each data set.

Table 13: The comparison of time costs between CC, MOCC and KACC

Data Sets	**Time of the Computation (minutes)**		
	CC	**MOCC**	**KACC**
SCCTS	33.03	403.33	**2.03**
E-coli	11.27	54.67	**0.62**
Malaria	11.97	155.33	**1.06**
VAR	14.33	1521.67	**5.32**
Normal	21.17	2948.86	**17.67**
WDBC	256.67	551.67	**2.8**
Image	10.2	48.17	**0.16**
Iris	19.33	31.33	**0.08**
Wine	22.33	42.94	**0.12**
Zoo	14.8	35.83	**0.04**

First we compare time costs of CC with those of MOCC. The second and third columns of Table 13 clearly show that MOCC has much higher time costs than CC especially for data sets SCCTS, VAR, and Normal. From the time costs of KACC, it is obvious that KACC has extremely lower time costs than CC and MOCC. In general, Table 13 demonstrates that KACC has the lowest time costs, and MOCC has the most expensive time costs. Under different experimental conditions, time costs of the three methods may be different from those listed in Table 13. The conclusion, however, will be the same.

There are two reasons why MOCC has expensive time costs. One is that MOCC employs SA for its heuristic optimisation search. In order to find global optimal solutions, SA requires a long time to run the search process; moreover, SA seeks for optimal solutions by evaluating random candidate solutions. Hence it is inevitable for MOCC to have some redundant or pleonastic operations during the search process. The other reason is that MOCC contains a multiple optimisation framework, where the Separation optimisation increases the computational cost of MOCC. It is also the main reason why the time cost of MOCC is higher than the one of CC (because CC only implements a single heuristic optimisation).

For KACC having the lowest time cost among the three methods, there are also two reasons. The first one is that KACC aims to provide a sufficient accuracy of clustering in the application context instead of searching for global optima, which saves huge time for optimisation. The second reason is that, in KACC, intelligent agents (ants) have “memory” to trace the search process so that KACC can avoid having redundant or pleonastic operations.

Based on the above discussions of the time complexity and time cost, we can get a conclusion that KACC has the best clustering efficiency among the three methods, while MOCC has the worst clustering efficiency. Since we have described the accuracy comparison between CC, MOCC and KACC in Section 6.2 and known that KACC has the best clustering accuracy, we can draw another conclusion that KACC can not only provide a high accuracy of clustering but also have a very good clustering efficiency. In addition, expensive time costs of MOCC suggest that MOCC improves clustering accuracies by sacrificing its clustering efficiency.

6.4 Summary

This chapter has discussed the performance comparison between CC, MOCC and KACC. Among the three heuristic optimisation-based Ensemble Clustering methods, the performance of KACC is the best. There are two advantages of KACC. On the one hand, KACC has significantly improved the accuracy of clustering by a good combination of internal and external clustering information. On the other hand, KACC aims to provide a sufficient accuracy of clustering in the application context instead of searching for global optima to reduce computational costs effectively.

MOCC implements a combination of internal and external information to have a better clustering accuracy than CC, but its time cost is too high. Further improvements are needed for MOCC in order to reduce its computational costs while improving the accuracy of clustering. Moreover, the clustering accuracy of MOCC cannot be comparable to the one of KACC.

For the CC algorithm, its clustering efficiency is better than the one of MOCC, but much worse than the one of KACC. Furthermore, CC has no capability of providing a clustering accuracy that is comparable to those of MOCC and KACC.

7

Chapter 7: Conclusions & Future Work

7.1 Conclusions

In this thesis, our work was focused on employing heuristic optimisation techniques to achieve Ensemble Clustering. We highlighted two difficulties with existing Heuristic Optimisation-Based Ensemble Clustering (HOBEC) methods.

One difficulty is that the consensus clustering step of existing HOBEC methods only relies on input clusterings; the clustering accuracy of these methods depends heavily on the quality of input clusterings. It is inevitable that, if input clusterings are noisy or differ significantly from each other, it will be difficult to guarantee the quality of results when using these HOBEC methods. Results of these methods could be worse than the mean accuracy of input clusterings.

The second difficulty is that heuristic optimisation has very high computational costs especially for analysing large data sets. In order to find a global optimal solution, traditional heuristic optimisation methods require a long time to implement the heuristic search. Therefore, it is often the case that existing HOBEC methods have very high computational costs, and global optima cannot be guaranteed even after extensive search.

To address the first difficulty, we use multiple optimisations to integrate information of input clusterings (i.e. external information) with information of given data (i.e. internal information) during the consensus clustering step. In this way, we can reduce the effect of biases of input clusterings, and enhance the accuracy of clustering. The second difficulty is handled from two aspects.

One is that optimising initial candidate clustering solutions for the heuristic search; the other is that aiming to provide a sufficient accuracy of clustering in the application context instead of searching for global optima. These enable our Ensemble Clustering methods to not only meet requirements of applications but also reduce computational costs. Two novel HOBEC methods, MOCC and KACC, were presented in this thesis to overcome these difficulties. The main achievements are outlined as follows:

- **Multi-Optimisation Consensus Clustering**

  Based on the Consensus Clustering (CC) algorithm proposed by Swift *et al.* (2004), we developed a novel Ensemble Clustering method called Multi-Optimisation Consensus Clustering (MOCC). MOCC employs Agreement Matrix and an Improved Agreement Fitness (IAF) function to describe external information (provided by input clusterings), and utilises the Separation index to describe internal information of given data. MOCC integrates the internal and external information by combining the IAF function with the Separation index to evaluate candidate solutions. Heuristic optimisation of MOCC is achieved by a multi-optimisation framework, which employs a well known heuristic method, Simulated Annealing (SA), to seek optimal solutions based on the IAF function and the Separation validation. The results demonstrate that MOCC has better stability for clustering and clearly performs better than the original CC algorithm. In other words, MOCC has successfully achieved the combination of internal and external clustering information, and improved the accuracy of clustering.

- **Analysing cooling functions for Ensemble Clustering using SA**

  Simulated Annealing has played an important role in optimisation search and decision making. One of the key components of SA is the Cooling Function. This thesis presented insights into the behaviour of cooling functions in the context of Ensemble Clustering. According to these insights, we can choose appropriate cooling functions for the optimisation within the Ensemble Clustering context. Ten cooling functions were selected and studied in this thesis, and two heuristic optimisation-based Ensemble Clustering methods (CC and MOCC) were used as representatives to test the ten cooling functions on thirteen different data sets.

These cooling functions are sorted into three groups. We tested them by different numbers of input methods, and different data sets with different sizes and dimensionalities. After analysing the results obtained from the experiments, we have the following key conclusions: 1) different cooling functions have different convergence speeds; 2) the number of iterations for the optimisation process should be set up carefully in order to obtain convergent results; 3) the larger a data set is (or the higher the dimensionality of a data set is), the slower the convergence speed is; 4) adding more input methods can improve the accuracy of clustering only when most of the input methods are appropriate for a given clustering problem.

These insights are important and can be very useful in assisting with the choice of cooling functions for different clustering problems. To sum up, for choosing a suitable cooling function or adjusting settings of a cooling function, we can combine all information from these aspects to make an informed decision.

- **K-Ants Consensus Clustering**

  The other new Ensemble Clustering method introduced in this thesis is called K-Ants Consensus Clustering (KACC). Based on the MDL principle and the Ant Colony Clustering theory, KACC combines external and internal information by multiple optimisation phases, which are EPMDLO, ABMDLO and EPAFO. It has been proved that these optimisation phases can be good complements to each other for combining clustering information. Each phase starts from an initial clustering solution based on input clusterings. The three optimisation phases generate three different clustering solutions. These solutions will be evaluated by a Validation

Vector to find out which solution is the best. After that, the best solution is selected to be the final clustering result of KACC.

Ten data sets were used for testing the performance of KACC. The experiments reveal that the results of KACC are much more accurate than the average of input clusterings. In addition, KACC was compared with some other comparable Ensemble Clustering methods such as ESIC and seven consensus clustering algorithms of the R CLUE package. The experimental results demonstrate that KACC clearly performed better than these Ensemble Clustering algorithms.

Among the two novel Ensemble Clustering methods proposed in this thesis, KACC is the ideal solution for solving the two difficulties of existing HOBEC methods. KACC not only improves the accuracy of clustering significantly by combining internal and external clustering information, but also has excellent clustering efficiency achieved by providing a sufficient accuracy of clustering in the application context instead of global optima. Moreover, this thesis has demonstrated that KACC clearly performs better than both CC and MOCC. MOCC only achieves the combination of internal and external clustering information to solve the first difficulty. For solving the second difficulty, further improvements are needed for MOCC.

In addition, all insights and conclusions summarised in this thesis are based on the experiments that test our methods and relevant algorithms on clean and high-dimensional data sets. In other words, having clean and high-dimensional data is the condition of using our methods. This thesis has demonstrated that our methods have clearly better performance for analysing cleanly high-dimensional data. In order to evaluate how robust our methods are for dealing with noisy data, in future work, we will test our methods on a large range of data sets that have noise and outliers.

7.2 Future Work

In this thesis, we have presented successful applications of integrating two well known heuristic algorithms, Simulated Annealing and the Ant Colony Clustering theory, with Ensemble Clustering. This thesis has demonstrated that our Ensemble Clustering methods generated very promising results. These meaningful achievements open up some possible further research directions for improving the performance of Heuristic Optimisation-Based Ensemble Clustering methods.

1) MOCC combines the Separation index with the Agreement Fitness Evaluation (AFE) for its multi-optimisation structure. The combination, however, is not limited to the Separation index. Since the Separation index has expensive computational costs, to combine other appropriate clustering evaluation indexes with AFE would be a good way forward for further research.

2) The function achieved by the multi-optimisation framework of MOCC is similar to the one of the Multi-Objective Simulated Annealing (MOSA) algorithm [Bandyopadhyay *et al.*, 2008], which aims to generate global optimal solutions by optimising multiple objectives. Therefore, it is good to further explore these two multi-optimisation structures to see whether MOCC can benefit from MOSA.

3) Based on the Minimum Description Length (MDL) principle, we developed the Attributed Weighed Description Length (AWDL) criterion for KACC to describe characteristics of given data. Rissanen [2001] proposed a criterion called Normalized Maximum Likelihood (NML) [Kontkanen *et al.*, 2006] for applying the MDL principle to describe data, and he claims that NML could be a universal code for describing all types of data sets. In future work, to integrate the advantages of the NML criterion with AWDL could be beneficial for refining the AWDL criterion of KACC.

4) This thesis has introduced promising results of using our novel methods MOCC and KACC to analyse different types of data sets. However, the limitation of our experiments is that all data sets are clean (i.e. there are no noise and outliers in data) and some data sets are relative small. In order to further analyse the performance of our methods, in future work, we will test them on a wider range of data sets that contain noise and outliers.

5) There are some other heuristic optimisation techniques such as Tabu Search [Glover and Laguna, 1997], Genetic Algorithm [Mitchell, 1998], and so on. In future work, we will analyse these techniques and try to find possible ways of integrating these heuristic methods with Ensemble Clustering to improve the accuracy of clustering and the clustering efficiency.

References

Andresen, B. and Gordon, J.M., “Analytic Constant Thermodynamic Speed-Cooling Strategies in Simulated Annealing,” *Open Systems & Information Dynamics*, vol. 2, pp. 1-12, 1993.

Ankerst, M., Breunig, M., Kriegel, H.P. and Sander, J., “OPTICS: Ordering Points to Identify the Clustering Structure,” *Management of Data Mining*, pp. 49–60, 1999.

Asuncion, A. and Newman, D.J., “UCI Machine Learning Repository,” [http://www.ics.uci.edu/~mlearn/MLRepository.html], *Irvine,* CA: University of California, School of Information and Computer Science, 2007.

Asur, S., Ucar, D., and Parthasarathy, S., "An Ensemble Framework for Clustering Protein-Protein Interaction Networks," *Bioinformatics,* vol. 23, no. 13, pp. I29-I40, Jul. 2007.

Atiqullah, M.M., “An Efficient Simple Cooling Schedule for Simulated Annealing,” *International Conference on Computational Science and Its Applications 3,* pp. 396-404, 2004.

Au, W., Chan, K. C., Wong, A. K., and Wang, Y., “Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data,” *IEEE/ACM Trans. Comput. Biol. Bioinformatics,* vol. 2, no. 2, pp. 83-101, 2005.

Avogadri, R. and Valentini, G., "Ensemble clustering with a fuzzy approach," *Supervised and Unsupervised Ensemble Methods and their Applications, Springer*, pp. 49-69, 2008.

Ayad, H.G., and Kamel, M.S., “Cumulative voting consensus method for partitions with a variable number of clusters,” *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 30, no. 1 pp. 160-173, 2008.

Azimi, J., Abdoos, M. and Analoui, M., “A New Efficient Approach in Clustering Ensembles,” *In IDEAL, International Conference on Intelligent Data Engineering and Automated Learning,* 2007.

Azimi, J., Cull, P. and Fern, X., “Clustering Ensembles Using Ants Algorithm,” *Proceedings of the 3rd International Work-Conference on The Interplay Between Natural and Artificial Computation: Part I: Methods and Models in Artificial and Natural Computation*, pp. 295-304, 2009.

Bandyopadhyay, S., Saha, S., Maulik, U., and Deb, K., “A Simulated Annealing based Multi-objective Optimization Algorithm: AMOSA,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269-283, 2008.

Barron, A.R., Rissanen, J. and Yu, B., “The Minimum Description Length Principle in coding and modelling,” *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743-2760, 1998.

Ben-Dor, A., Shamir, R., and Yakhini, Z., “Clustering gene expression patterns,” *J. Comput. Biol.*, vol. 6, pp. 281–297, 1999.

Ben-Hur, A., Horn, D., Siegelmann, H., and Vapnik, V., “Support vector clustering,” *J. Mach. Learn. Res.*, vol. 2, pp. 125–137, 2001.

Berkhin, P., “Survey of clustering data mining techniques,” *Technical Report*, Accrue Software, 2002.

Bezdek, J.C., Keller, J.M., Krishnapuram, R. and Pal, N.R., “Fuzzy Models and Algorithms for Pattern Recognition and Image Processing,” *Kluwer Academic Publishers,* 1999.

Bezdek, J.C., “Pattern Recognition with Fuzzy Objective Function Algorithms,” *Plenum Press*, New York, 1981.

Bidgoli, B.M., Topchy, A. and Punch, W.F., "A Comparison of Resampling Methods for Clustering Ensembles," *IC-AI*, pp. 939-945, 2004.

Bonabeau, E., Dorigo, M. and Theraulaz, G., “Inspiration for optimisation from social insect bebavior,” *Nature*, vol. 406, no. 6, pp. 39-42, 2000.

Bozdech, Z., Llinás, M., Pulliam, B.L., Wong, E.D., Zhu, J. and DeRisi, J.L., "The Transcriptome of the Intraerythrocytic Developmental Cycle of Plasmodium Falciparum," *PLoS Biology*, vol. 1, pp. 85-100, 2003.

Brendan, J.F. and Dueck, D., "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, pp. 972–976, 2007.

Buckham, B.J. and Lambert, C., “Simulated Annealing Applications,” http://www.me.uvic.ca/˜zdong/courses/mech620/SA_App.pdf, 1999.

Carpenter, G.A. and Grossberg, S., “Adaptive Resonance Theory,” In Michael A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, Second Edition, Cambridge, MA: MIT Press, pp. 87-90, 2003.

Carpenter, G.A. and Grossberg, S., “ART 2: Self-organization of stable category recognition codes for analog input patterns,” *Applied Optics*, vol. 26, no. 23, pp. 4919-4930, 1987.

Carpenter, G.A. and Grossberg, S., “ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures,” *Neural Networks*, vol. 3, pp. 129-152, 1990.

Causton, H., Quackenbush, J. and Brazma, A., “Microarray Gene Expression Data Analysis: A Beginner’s Guide,” *UK: Blackwell Science*, 2003.

Cheeseman, P. and Stutz, J., “Bayesian Classification (AutoClass): Theory and Results,” *Advances in Knowledge Discovery and Data Mining*, Cambridge, 1996.

Chen, G., Banerjee, N., Jaradat, S.A., Tanaka, T.S., Ko, M.S.H., and Zhang, M.Q., "Evaluation and Comparison of Clustering Algorithms in Analyzing ES Cell Gene Expression Data," *Statistica Sinica*, vol. 12, pp. 241-262, 2002.

Cheng, Y., "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

Choi, K., Kim, D.H., and Choi, T., "Estimating the Number of Clusters Using Multivariate Location Test Statistics," *Lecture Notes in Computer Science*, vol. 4223, pp. 373-382, 2006.

Colorni, A., Dorigo, M. and Maniezzo, V., "Distributed Optimization by Ant Colonies," *actes de la première conférence européenne sur la vie artificielle*, Paris, France, Elsevier Publishing, pp. 134-142, 1991.

Comaniciu, D. and Meer, P., "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

Cuevas, A., Febrero, M. and Fraiman, R., "Estimating the number of clusters," *the Canadian Journal of Statistics*, vol. 28, no. 2, pp. 367-382, 2000.

D'andrade, R., "U-Statistic Hierarchical Clustering," *Psychometrika*, vol. 4, pp. 58-67, 1978.

Davies, D.L. and Bouldin, D.W., "A cluster separation measure," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 1, pp. 224–227, 1979.

Deneubourg, J.L., Goss, S. and Franks, N., "The dynamics of xollective sorting: robot-like ants and ant-like robots," *Proceedings of the 1st Intenational Conference on Simulation of Adaptive Behavior: from Animals to Animals*, pp. 356-363, 1991.

Dictionary.com, "robust," *The Free On-line Dictionary of Computing*, Source location: Denis Howe, http://dictionary.reference.com/browse/robust, Available: http://dictionary.reference.com, Accessed: July 19, 2010.

Dimitriadou, E., Weingessel, A. and Hornik, K., “A Combination Scheme for Fuzzy Clustering,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, pp. 901–912, 2002.

Dorigo, M. and Caro, G. Di, "The Ant Colony optimization metaheuristic,” *New Ideas in Optimization,* pp. 11-32, 1999.

Dorigo, M., Birattari, M. and Stutzle, T., “Artificial ants as a computational intelligence technique,” *IEEE Computational Intelligence Magazine*, 2006.

Dorigo, M., Caro, G. Di and Gambardella, M., “Ant algorithms for discrete optimization,” *Artificial Life*, vol. 5, no. 2, pp. 137-172, 1999.

Dorigo, M., Maniezzo, V. and Colorni, A., “Ant System: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, vol. 26, pp. 29–41, 1996.

Duda, R., Hart, P. and Stork, D., “Pattern Classification,” *John Wiley & Sons*, 2001.

Dunn, J.C., "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, pp. 32-57, 1973.

Eiben, A.E. et al, "Genetic Algorithms with Multi-Parent Recombination," *PPSN III: Proceedings of the International Conference on Evolutionary Computation*, pp. 78-87. 1994.

Elster, K.H., "Modern Mathematical Methods of Optimization," *Akadamie Verlag Berlin*, pp. 415, 1993.

Ester, M., Kriegel, H.P. and Xu, X., “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *In: Proc. 2nd ACM SIGKDD*, pp. 226-231, 1996.

Everitt, B., “Cluster Analysis,” *Third Edition. Edward Arnold: a member of the Hodder Headline Group.* 1993.

Everitt, B., Landau, S. and Leese, M., “Cluster Analysis,” *London: Arnold*, 2001.

Faceli, K., Carvalho, A.C.P.L.F. de, Souto, M.C.P. de, Sagot, M.F. and Walter, M.E.M.T., “Multi-objective clustering ensemble with prior knowledge,” *Advances in Bioinformatics and Computational Biology*, vol. 4643, pp. 34-45, 2007.

Fanty, M. and Cole, R., “Spoken Letter Recognition,” *Advances in Neural Information Processing Systems 3.* San Mateo, CA: Morgan Kaufmann, 1991.

Fern, X.Z. and Brodley, C.E., "Solving Cluster Ensemble Problems by Bipartite Graph Partitioning," *Proc. the Twenty-First International Conference on Machine Learning (ICML '04)*, vol. 69, ACM Press, New York, NY, pp. 36, 2004.

Fisher, D.H., “Knowledge Acquisition via Incremental Conceptual Clustering,” *Machine Learning*, vol. 2, pp. 139-172, 1987.

Fraley, C., Adrian and Raftery, E., “How many clusters? Which clustering method? Answers via model-based cluster analysis,” *the Computer Journal*, vol. 41, pp. 578-588, 1998.

Fred, A.L.N. and Jain, A.K., "Combining Multiple Clusterings using Evidence Accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835-850, Jun. 2005.

Frey, B.J. and Dueck, D., “Clustering by Passing Messages Between Data Points,” *Science*, vol. 315, pp. 972-976, 2007.

Fridlyand, J., Dudoit, R. and Dudoit, S., “Applications of Resampling Methods to Estimate the Number of Clusters and to Improve the Accuracy of a Clustering Method,” *Technical Report 600*, Statistics Department, UC Berkeley, 2001.

Fukunaga, K. and Hostetler, L.D., “The Estimation of the Gradient of a Density Function with Applications in Pattern Recognition,” *IEEE Trans. Info. Theory*, vol. IT-21, pp. 32-40, 1975.

Garey, M.R. and Johnson, D.S., “Computers and Intractability: A Guide to the Theory of NP-Completeness,” *New York: W.H. Freeman*, ISBN 0-7167-1045-5, 1979.

Georgescu, B., Shimshoni, I. and Meer, P., “Mean Shift based Clustering in High Dimensions: A Texture Classification Example,” *In International Conference on Computer Vision*, pp. 456–463, 2003.

Glover, F. and Laguna, M., “Tabu Search,” *Kluwer Academic Publishers*, Boston, ISBN 0-7923-9965-X, July 1997.

Goder, A. and Filkov, V., “Consensus clustering algorithms: Comparison and refinement,” *In ALENEX '08: Procs. of 10th Workshop on Algorithm Enginering and Experiments*, pp. 109-117, 2008.

Goldberg, D.E., “Genetic Algorithm in Search, Optimization and Machine Learning,” *Addison-Wesley*, Reading, MA, pp. 41, 1989.

Gordon, A.D. and Vichi, M., “Fuzzy Partition Models for Fitting a Set of Partitions.” *Psychometrika*, vol. 66, no. 2, pp. 229–248, 2001.

Granville, V., Kfivanek, M. and Rasson, J.P., “Simulated Annealing: A proof of Convergence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 652-656, June 1994.

Graunwald, P.D., Myung, I.J. and Pitt, M.A. (Eds.), “Advances in Minimum Description Length: Theory and Applications,” *MIT Press*, 2004.

Grossberg, S., “Competitive learning: From interactive activation to adaptive resonance,” *Cognitive Science*, vol. 11, pp. 23-63, 1987.

Guha, S., Rastogi, R. and Shim, K., “CURE: An efficient clustering algorithm for large databases,” *in Proc. ACM SIGMOD Int. Conf. Management of Data*, pp. 73-84, 1998.

Gu, Y., Hall, L.O. and Goldgof, D.B., “Ant Clustering Using Ensembles of Partitions,” *Lecture Notes in Computer Science*, vol. 5519, pp. 283-292, 2009.

Hajek, B., “Cooling Schedules for Optimal Annealing,” *Mathematics of Operations Research*, vol. 13, pp. 311-329, 1988.

Halkidi, M., Batistakis, Y. and Vazirgiannis, M., “On clustering validation techniques,” *J. Intell. Inform. Syst.*, 17, 107–145, 2001.

Hastie, T., Tibshirani, R., and Friedman, J., “The Elements of Statistical Learning: Data Mining, Inference and Prediction,” *Springer*, 2001.

He, Z., Xu, X. and Deng, S., "A Cluster Ensemble Method for Clustering Categorical Data," *Information Fusion*, vol. 6, no. 2, pp. 143-151, Jun. 2005.

Hirsch, M., Swift, S. and Liu, X., "Optimal Search Space for Clustering Gene Expression Data Via Consensus," *The Journal of Computational Biology*, vol. 14, no. 10, pp. 1327-1341, 2007.

Hoffmann, K.H. and Salamon, P., “The optimal simulated annealing schedule for a simple model,” *Journal of Physics A: Mathematical and General*, vol. 23, pp. 3511-3523, 1989.

Hong, Y. and Kwong, S., “To combine steady-state genetic algorithm and ensemble learning for data clustering,” *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1416-1423, 2008.

Hore, P., Hall, L.O. and Goldgof, D.B., “A scalable framework for cluster ensembles,” *Pattern Recognition*, vol. 42, no. 5, pp. 676-688, 2009.

Hornik, K., "A Clue for Cluster Ensembles," *Journal of Statistical Software*, vol. 14, no. 12, available at http://www.jstatsoft.org/v14/i12/, 2005.

Hornik, K., “Cluster Ensembles,” In C Weihs, W Gaul (eds.), “Classification – The Ubiquitous Challenge,” *Proceedings of the 28th Annual Conference of the Gesellschaft f ür Klassifikation e.V.*, Springer-Verlag, Heidelberg, pp. 65–72, 2005.

Hubert, L. and Arabie, P., “Comparing Partitions,” *Journal of Classification*, pp. 193–218, 1985.

Hubert, L. and Schultz, J., “Quadratic assignment as a general data-analysis strategy,” *Br. J. Math. Statist. Psychol*, vol. 29, pp. 190-241, 1976.

Hu, T., Yu, Y., Xiong, J., and Sung, S.Y., “Maximum likelihood combination of multiple clusterings • SHORT COMMUNICATION,” *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1457-1464, 1 October 2006.

Hu, X. and Yoo, I., "Cluster Ensemble and its Applications in Gene Expression Analysis," *Proc. the Second Conference on Asia-Pacific Bioinformatics*, vol. 55, pp. 297-302, 1 Jan. 2004.

Hu, X., Zhang, X. and Zhou, X., "Integration of Cluster Ensemble and EM Based Text Mining for Microarray Gene Cluster Identification and Annotation," *Proc. the 15th ACM International Conference on Information and Knowledge Management (CIKM '06),* ACM Press, New York, NY, pp. 824-825, 2006.

Jain, A.K. and Dubes, R.C., "Algorithms for Clustering Data," *Prentice Hall,* 1988.

Jain, A.K., Murty, M.N., and Flynn, P.J., "Data clustering: a review," *ACM Comput. Surv.* Vol. 31, no. 3, pp. 264-323, 1999.

Jenkins, A.M., "Research Methodologies and MIS Research," *In Research Methods in Information Systems*, E. Mumford *et al.* (Ed.), Elsevier Science Publishers B.V., Amsterdam, Holland, pp. 103-117, 1985.

Jenner, R.G., Maillard, K., Cattini, N., Weiss, R.A., Boshoff, C., Wooster, R. and Kellam, P., "Kaposi's sarcoma-associated herpesvirus-infected primary effusion lymphoma has a plasma cell gene expression profile," *Proc Natl Acad Sci USA*, vol. 100, pp. 10399-10404, 2003.

Jiang, D., Tang, C. and Zhang, A., "Cluster Analysis for Gene Expression Data: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1370-1386, Nov. 2004.

Jiang, H.F., Chen, S.F. and Liu, S.F., "A new ant colony algorithm for a general clustering," *Proceedings of 2007 IEEE International Conference on Grey Systems And Intelligent Services*, vol. 1-2, pp. 1158-1162, 2007.

Johnson, S.C., "Hierarchical Clustering Schemes," *Psychometrika*, vol. 2, pp. 241-254, 1967.

Karypis, G., Aggarwal, R., Kumar, V. and Shekhar, S., "Multilevel hypergraph partitioning: applications in vlsi domain," *IEEE Trans. Very Large Scale Integr. Syst.* Vol. 7, pp. 69-79, 1999.

Kaufman, L. and Rousseeuw, P.J., “Clustering by Means of Medoids,” *Statistical Data Analysis Based on the L1–Norm and Related Methods, edited by Y. Dodge, North-Holland*, pp. 405–416, 1987.

Kaufman, L. and Rousseeuw, P.J., “Finding Groups in Data: An Introduction to Cluster Analysis,” *Wiley-Interscience: New York* (*Series in Applied Probability and Statistics*), ISBN 0-471-87876-6, 1990.

Kayypis, G. and Kumar, V., “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs,” *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359-392, 1998.

Kirkpatrick, S., Gelatt, Jr.C.D. and Vecchi, M.P., "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

Kohonen, T., “Automatic formation of topological maps of patterns in a self-organizing system,” *Proceedings of 2SCIA*, pp. 214-220, 1981a.

Kontkanen, P., Myllymäki, P., Buntine, W., Rissanen, J. and Tirri, H., “An MDL Framework for Data Clustering,” in Advances in Minimum Description Length: Theory and Applications, Gr ̈unwald, P., Myung, I.J., and Pitt, M., Eds. *The MIT Press*, pp. 323–354, 2006.

Kuncheva, L.I., Hadjitodorov, S.T., and Todorova, L.P., “Experimental comparison of cluster ensemble methods,” *In Proceedings of the International Conference on Information Fusion*, pp. 1-7, 2006.

Laarhoven, P.J. and Aarts, E.H., “Simulated Annealing: Theory and Applications,” *Kluwer Academic Publishers*, pp. 100-138, 1987.

Lee, D.D. and Seung, H.S., “Learning the parts of objects with nonnegative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

Lee, P., “Bayesian Statistics - an introduction,” Arnold & Oxford University Press, 1997.

Leone, M., Sumedha, and Weight, M., “Clustering by soft-constraint affinity propagation: Applications to gene-expression data,” *Bioinformatics,* vol. 23, no. 2708, 2007.

Li, J., Swift, S. and Liu, X., “Multi-Optimisation Consensus Clustering,” *Lecture Notes in Computer Science*, Springer Berlin, vol. 5772, pp. 345-356, 2009.

Li, T., Ogihara, M. and Ma, S., "On Combining Multiple Clusterings," *Proc. ACM International Conference on Information and Knowledge Management (CIKM '04),* ACM Press, New York, NY, pp. 294-303, 2004.

Lloyd, S.P., “Least Squares Quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

Luke, B.T., “Simulated Annealing Cooling Schedults,” http://fconyx.ncifcrf.gov/lukeb/simanf1.html, Last Updated 14 Mar. 2002.

Lumer, E. and Faieta, B., “Diversity and adaptation in populations of clustering ants,” *Proceeding of the 3rd international conference on simulaton of adaptive behaviour*: *from animals to animals*, pp. 499-508, 1994.

Lütkepohl, H., “New Introduction to Multiple Time Series Analysis”. *1st ed.* ISBN: 978-3-540-26239-8, pp. 49, 2007.

Lv, T., Huang, S., Zhang, X., and Wang, Z., “Combining Multiple Clustering Methods Based on Core Group”. *Semantics, Knowledge, and Grid*. SKG '06. Second International Conference on, pp. 29-29, 2006.

March, S. and Smith, G., "Design and Natural Science Research on Information Technology,” *Decision Support Systems*, vol. 15, pp. 251-266, 1995.

Mcqueen, J., “Some methods for classification and analysis of multivariate observations,” *In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

Mitchell, M., “An Introduction to Genetic Algorithms,” *Cambridge, Mass.: MIT Press*, ISBN: 0262631857, 1998.

Monti S., Tamayo P., Mesirov J., Golub T., “Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data,” *Machine Learning*, vol. 52, pp. 91-118, 2003.

Mozer, M., Jordan, M., and Petsche, T., “Clustering sequences with hidden Markov models,” *In Advances in Neural Information Processing*, Eds. Cambridge, MA: MIT Press, vol. 9, pp. 648–654, 1997.

Mullen, R.J., Monekosso, D. and Barman, S., “A review of ant algorithms,” *Expert Systems with Applications*, vol. 36, no. 6, pp. 9608-9617, 2009.

Murtagh, F., “A survey of recent advances in hierarchical clustering algorithms,” *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983.

Myers, M.D., "Qualitative Research in Information Systems," *MIS Quarterly*, vol. 21, no. 2, pp. 241-242, June 1997.

Ng, R., and Han, J., “CLARANS: A method for clustering objects for spatial data mining,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.

Nourani, Y. and Andresen, B., “A Comparison of Simulated Annealing Cooling Strategies,” *Journal of Physics A: Mathematical and General*, vol. 31, pp. 8373–8385, 1998.

O'Connell, M., "Differential expression, class discovery and class prediction using S-PLUS and S+ArrayAnalyzer," *SIGKDD Explor. Newsl.* Vol. 5, no. 2, pp. 38-47, 2003.

Pandey, G., Kumar, V. and Steinbach, M., "Computational Approaches for Protein Function Prediction," *Supported by the National Science Foundation under Grant Nos.* IIS-0308264 and ITR-0325949, 2007.

Parr, W.C. and Schucany, W.R., "Minimum distance and robust estimation," *Journal of the American Statistical Association*, vol. 75, pp. 616–624, 1980.

Punera, K. and Ghosh, J., "Consensus-based Ensembles of Soft Clusterings," *Applied Artificial Intelligence*, vol. 22, no. 7-8, pp. 780-810, 2008.

Punera, K. and Ghosh, J., "Soft Cluster Ensembles," *Advances in Fuzzy Clustering and Its Applications*, 2007.

Quarteroni, A. and Fausto, S., "Scientific Computing with MATLAB and Octave," *Springer*, 2006.

R Development Core Team, "R: A Language and Environment for Statistical Computing," *R Foundation for Statistical Computing*, Vienna, Austria, ISBN: 3-900051-07-0, URL: http: //www.R-project.org, 2005.

Rijsbergen, C.J., "Information Retrieval," *2nd edn*, London: Butterworths, http://www.dcs.gla.ac.uk/Keith/Preface.html, 1979.

Rissanen, J., "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, vol. 11, pp. 416-431, 1983.

Rissanen, J., "Fisher information and stochastic complexity," *IEEE Transac-tions on Information Theory*, vol. 42, no. 1, pp. 40-47, 1996.

Rissanen, J., “Modeling by the shortest data description,” *Automatica*, vol. 14, pp. 465-471, 1978.

Rissanen, J., “Strong optimality of the normalized ML models as universal codes and information in data,” *IEEE Transactions on Information Theory*, vol. 47, pp. 1712–1717, 2001.

Rousseeuw, P.J., “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *J. Comput. Appl.* Math., vol. 20, pp. 53–65, 1987.

Russell, S.J. and Norvig, P., “Artificial Intelligence: A Modern Approach (2nd ed.),” *Upper Saddle River*, NJ: Prentice Hall, pp. 111-114, 2003.

Shamir, R. and Sharan, R., “Algorithmic Approaches to Clustering Gene Expression Data,” *Current Topics in Computational Molecular Biology*, pp. 269-300, 2002.

Sheikholeslami, G., Chatterjee, S. and Zhang, A., “WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases,” *Proceedings of the 24rd International Conference on Very Large Data Bases*, pp.428-439, 1998.

Sims, C.A., "Macroeconomics and Reality," *Econometrica*, pp. 1-48, Jan. 1980.

Snedecor, G.W., and Cochran, W.G., “Statistical Methods,” *7th ed. Ames*, IA: Iowa State Press, pp. 175-178, 1980.

Straub, D., Gefen, D. and Boudreau, M.C., "The ISWorld Quantitative, Positivist Research Methods Website," http://www.dstraub.cis.gsu.edu:88/quant/. Last updated: January 7, 2005.

Strehl, A. and Ghosh, J., "Cluster Ensembles --- a Knowledge Reuse Framework for Combining Multiple Partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583-617, Mar. 2003.

Swift, S., Tucker, A. and Hirsch, M., “Improving the Performance of Consensus Clustering through Seeding: An Application to Visual Field Data,” *Proc. the Intelligent Data Analysis in Biomedicine and Pharmacology Workshop*, Amsterdam, the Netherlands, pp. 29-34, 2007.

Swift, S., Tucker, A., Crampton, J., and Garway-Heath, D., “An Improved Restricted Growth Function Genetic Algorithm for the Consensus Clustering of Retinal Nerve Fibre Data,” *Proc. the 9th Annual Conference on Genetic and Evolutionary Computation* (*GECCO '07*), ACM, New York, NY, pp. 2174-2181, 2007.

Swift, S., Tucker, A., Vincotti, V., Martin, N., Orengo, C., Liu, X. and Kellam, P., “Consensus Clustering and Functional Interpretation of Gene-Expression Data,” *Genome Biology*, vol. 5, no. 11, pp. R94.1-R94.16, 2004.

Taylor, J.R., “An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements,” *University Science Books*, pp. 128–129, 1999.

Tibshirani, R., Walther, G. and Hastie, T., “Estimating the number of clusters in a dataset via the Gap statistic,” *Journal of the Royal Statistical Society*, vol. 63, pp. 411-423, 2001.

Topchy, A., Jain, A.K. and Punch, W., "Clustering Ensembles: Models of Consensus and Weak Partitions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1866-1881, Dec. 2005.

Topchy, A., Jain, A.K. and Punch, W., "Combining Multiple Weak Clusterings," *IEEE International Conference on Data Mining (ICDM 2003)*, pp. 331-338, 2003.

Topchy, A.P., Law, M.H.C., Jain, A.K. and Fred, A.L., "Analysis of Consensus Partition in Cluster Ensemble," *Proc. Fourth IEEE International Conference on Data Mining (ICDM '04),* pp. 225-232, Nov. 2004.

Tseng, V. S. and Kao, C., “Efficiently Mining Gene Expression Data via a Novel Parameterless Clustering Method,” *IEEE/ACM Trans. Comput. Biol. Bioinformatics,* vol. 2, no. 4, pp. 355-365, 2005.

Tumer, K. and Agogino, A.K., “Ensemble clustering with voting active clusters,” *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1947-1953, 2008.

Vaishnavi, V. and Kuechler, W., “Design Research in Information Systems,” January 20, 2004, last updated January 18, 2006. URL: http://www.isworld.org/Researchdesign/drisISworld.htm

Vega-Pons, S., Correa-Morris, J. and Ruiz-Shulcloper, J., “Weighted Cluster Ensemble Using a Kernel Consensus Function,” *Progress in Pattern Recognition, Image Analysis and Applications*, vol. 5197, pp. 195-202, 2008.

Viera, A.J., and Garrett, J.M., “Understanding Interobserver Agreement: the Kappa Statistic,” *Fam Med* 37, pp. 360-363, 2005.

Viswanath, P. and Jayasurya, K., "A Fast and Efficient Ensemble Clustering Method," *18th International Conference on Pattern Recognition (ICPR'06),* pp. 720-723, 2006.

Wang, J., Thiesson, B., Xu, Y. and Cohen, M., “Image and Video Segmentation by Anisotropic Kernel Mean Shift,” *In European Conference on Computer Vision*, vol. 2, pp. 238–249, 2004.

Wang, K., Zhang, J., Li, D., Zhang, X. and Guo, T., “Adaptive Affinity Propagation Clustering,” *Acta Automatica Sinica*, vol. 33, no. 12, pp. 1242-1246, 2007.

Wang, W., Yang, J. and Muntz, R., "STING: A Statistical Information Grid Approach to Spatial Data Mining," *In: Proc. of the 23rd Very Large Databases Conf*, 1997.

Wang, X. and Li, J., "Hybrid Particle Swarm Optimization with Simulated Annealing," ISBN: 0-7803-8403-2, vol. 4, pp. 2402-2405, 2004.

Ward, J.H., "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association*, vol. 58, pp. 236-244, 1963.

Webb, A., "Statistical Pattern Recognition," *Arnold: a member of the Hodder Headline Group*, pp. 275-317, 1999.

Wei, X., "Research and Application of a Multi-Ant Colony Clustering Combination Algorithm," *International Conference on Computer Science & Education*, pp. 27-30, 2009.

Witten, I.H. and Frank, E., "Data Mining: practical machine learning tools and techniques," San Francisco: Morgan Kaufmann Publishers, Second Edition. pp183-184, 2005.

Wu L.F., Hughes T.R., Davierwala A.P., Robinson M.D., Stoughton R., Altschuler S.J., "Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters," *Nat Genet*, vol. 31, pp. 255-265, 2002.

Xia, L.N. and Jing, J.W., "An ensemble density-based clustering method," *Proceedings Of The International Conference On Intelligent Systems And Knowledge Engineering (ISKE 2007),* ATlantic Press, pp. 1209-1209, 2007.

Xu, R., and Wunsch, D., "Survey of clustering algorithms," *II, IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645- 678, 2005.

Yang, X.S., “Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics,” *Cambridge Int. Science Publishing*, 2008.

Yang, Y. and Kamel, M., “An Aggregated Clustering Approach Using Multi-Ant Colonies Algorithms,” *Pattern Recognition*, vol. 7, pp. 1278-1289, 2006.

Yang, Y. and Kamel, M., “Clustering Ensemble Using Swarm Intelligence,” *IEEE Swarm Intelligence Symposium*, Indianapolis, USA, pp. 65-71, April, 2003.

Yang, Y., Kamel, M. and Jin, F., “Clustering Ensemble Using ANT and ART,” *Swarm Intelligence in Data Mining*, Springer Berlin, vol. 34, pp. 243-264, 2006.

Yeung, K.Y., and Ruzzo, W.L., “Principal Component Analysis for Clustering Gene Expression Data,” *Bioinformatics*, vol. 17, no. 9, pp. 763-774, 2001.

Yu, Z.W., Wong, H.S. and Wang, H.Q., “Graph-based consensus clustering for class discovery from gene expression data,” *Bioinformatics*, vol. 23, no. 21, pp. 2888-2896, 2007.

Zhang, T., Ramakrishnan, R., and Livny, M., “BIRCH: An efficient data clustering method for very large databases”. *In Proc. ACM SIGMOD Conf. Management of Data*, pp. 103–114, 1996.

Zhang, X., Furtlehner, C. and Sebag, M., “Data streaming with affinity propagation,” *In ECML/PKDD*, pp. 628-643, 2008.

Zhao, B., “An ant colony clustering algorithm,” *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, vol. 7, pp. 3933-3938, Aug. 2007.