Evolutionary and Swarm Algorithm Optimized Density-Based Clustering and Classification for Data Analytics

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy

BY

Chun Guan

Department of Computer Science

The University of Liverpool

Liverpool, United Kingdom

April, 2018

Abstract

Clustering is one of the most widely used pattern recognition technologies for data analytics. Density-based clustering is a category of clustering methods which can find arbitrary shaped clusters. A well-known density-based clustering algorithm is Density-Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN has three drawbacks: firstly, the parameters for DBSCAN are hard to set; secondly, the number of clusters cannot be controlled by the users; and thirdly, DBSCAN cannot directly be used as a classifier.

With addressing the drawbacks of DBSCAN, a novel framework, Evolutionary and Swarm Algorithm optimised Density-based Clustering and Classification (ESA-DCC), is proposed. Evolutionary and Swarm Algorithm (ESA), has been applied in various different research fields regarding optimisation problems, including data analytics. Numerous categories of ESAs have been proposed, such as, Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Differential Evaluation (DE) and Artificial Bee Colony (ABC).

In this thesis, ESA is used to search the best parameters of density-based clustering and classification in the ESA-DCC framework to address the first drawback of DBSCAN. As method to offset the second drawback, four types of fitness functions are defined to enable users to set the number of clusters as input. A supervised fitness function is defined to use the ESA-DCC as a classifier to address the third drawback. Four ESA-DCC methods, GA-DCC, PSO-DCC, DE-DCC and ABC-DCC, are developed. The performance of the ESA-DCC methods is compared with K-means and DBSCAN using ten datasets. The experimental results indicate that the proposed ESA-DCC methods can find the optimised parameters in both supervised and unsupervised contexts. The proposed methods are applied in a product recommender system and image segmentation cases.

Acknowledgments

I would first like to thank my primary supervisor, Dr. Kevin Kam Fung Yuen, for all the guidance, patience and invaluable encouragement to me. I am really fortunate and honoured to be his first PhD student. I would like to thank my second supervisor, Prof. Frans Coenen, for his support for my work, especially when I studied in the University of Liverpool during my second year. I would also like to take this opportunity to thank my examiners, Prof. Steven Guan and Prof. Jenq-Shiou Leu, for their very helpful comments and suggestions to improve my thesis.

I am very grateful to have been a PhD student of University of Liverpool based on Xi'an Jiaotong-Liverpool University. I am thankful to everyone who have helped me during my PhD study, especially for Prof. Yong Yue and the other staff at the department of Computer Science and Software Engineering, and staff at Research and Graduate Studies Office.

This research is partially supported by the PGR Scholarship of Xi'an Jiaotong-Liverpool University and the University of Liverpool, the National Natural Science Foundation of China (Project Number 61503306) and the Natural Science Foundation of Jiangsu Province (Project Number BK20150377), China.

Last but not least, I would like to thank my family as well as all of my friends for their continuous supports during my life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1 Introduction

Data analytic has attracted significant attention in the information industry and society, due to the huge amounts of data and the increasing need for turning such data into useful information and knowledge [Han, Pei, and Kamber, 2011]. Clustering Analysis is a widely used pattern recognition technology in the field of data analytic. Clustering methods can be categorized into Partional-based methods, Hierarchical-based methods, Density-based methods, Grid-based methods, Model-based methods and so on. Density-based clustering was proposed to deal with spatial datasets such as satellite images, facial images and medical images [Ester et al., 1996]. The best-known density-based clustering algorithm is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al., 1996]. DBSCAN uses two parameters, the radius of hyper-spheres (ε) and the minimum number of points in each hyper-sphere (*minpts*). The main advantage of DBSCAN is its ability to find arbitrary shaped clusters through detecting the high-density hyper-spheres and merging the close hyper-spheres into clusters. In contrast, the centroid-based clustering method cannot be used to find arbitrary shaped clusters in spatial datasets. Conversely, Hierarchical-based clustering can be applied to spatial datasets, although, the clustering results are sensitive to noise.

However, DBSCAN has some limitations. Firstly, the parameters for DBSCAN are hard to set, the two parameters, radius (ε) and the minimum number of points (*minpts*), are often set by manual testing. Although, some parameter-tuning methods require one pre-defined parameter to calculate another parameter, for instance, the k-distance plot method requires a pre-defined *minpts* to find the corresponding suitable ε. Secondly, the known number of clusters cannot be used in the clustering process. For some clustering methods, the number of clusters is known beforehand which can be used to aid the clustering process. For example, the number of clusters is the only input parameter for K-means. Similarly, the known number of clusters can be used to decide how to cut the tree structure produced by hierarchical clustering method. However, the number of clusters cannot be controlled by the users in DBSCAN. Thirdly, DBSCAN cannot be directly applied for classification purposes.

The central motivation for this research is the intuition that Evolutionary and Swarm Algorithms (ESAs) [Fogel, 2006] could be used as a parameter-tuning tool for DBSCAN. Evolutionary Computing (EC) and Swarm Intelligence (SI) can generally be described as Evolutionary and Swarm Algorithms (ESAs). Numerous categories of ESAs have been proposed, these include: Genetic Algorithms (GAs) [Golberg,1989], Particle Swarm Optimization (PSO) [Kennedy, 1995], Differential Evaluation (DE) [Storn & Price, 1997], Artificial Bee Colony (ABC) [Karaboga, 2005], etc. GA and DE are two typical types of evolutionary algorithms. PSO and ABC are two examples of ESA methods. The development of ESAs was inspired by the idea of natural selection and observations of animal behaviour. For example, GA is inspired by the mechanism of natural selection; the operators in GA algorithms and evolutionary strategies are the motivators behind DE; the social conduct of groups of animals, such as flocks of birds, schools of fish and herds of mammals, stimulated the development of PSO; and similarly, ABC is inspired by the behaviour of bees when searching for food sources The main advantage of ESA is the highly robust search performance of such algorithms. In this thesis, ESA methods are applied as parameter tuning tools to offset the first drawback of DBSCAN. ESAs have been applied as the optimisation methods for various clustering methods. Various ESA optimal clustering methods have been reviewed in Section 2 of this work.

Based on the above observations, a novel framework, Evolutionary and Swarm Algorithms Optimised Density-based Clustering and Classification (ESA-DCC), is proposed directed at optimising the performance of density-based clustering by finding the best parameter settings through a search of the entire parameter space using ESAs. In this framework, two types of fitness functions are designed on the basis of the current clustering validation indices; and penalty functions are designed to minimise the number of noises and to control the number of clusters. In this thesis, four types of ESAs (GA, PSO, DE and ABC) are adopted in the ESA-DCC framework. The Four ESA-DCC methods are evaluated by both experimental cases and real-world problems.

The main contributions of this work are the proposal of the structure of applying ESA to tune the parameters for density-based clustering methods and the proposal of fitness functions for the ESA optimised density-based clustering methods framework. With the structure and these fitness functions, any ESA could be applied to optimise a density-based clustering method. The ESA-DCC framework could be extended by adopting original and revised ESAs as well as density-based clustering methods. The proposed fitness functions are reasonable combinations of available components: the clustering index, the penalty function to minimize the number of noises and the function to control the number of clusters. The various functions of the different choices of components and clustering indices are be flexibly applied to clustering problems on a case-by-case basis.

The proposed ESA-DCC methods can be applied to find the arbitrary shaped clusters in spatial datasets. For the datasets which are not well understood, the proposed ESA-DCC method can be applied to search for possible clustering patterns by testing with different number of clusters. For the datasets with a known number of clusters, the proposed ESA-DCC method can find a set of parameters to produce the optimum clustering results.

Some limitations of the proposed ESA-DCC framework are evident. Firstly, the computational complexity of the proposed ESA optimised DBSCAN framework (ESA-DCC) is limited by the complexity of the DBSCAN method. The computational complexity of a standard DBSCAN method is as high as $O(n\log n)$. Since ESA-DCC adopts the standard DBSCAN, and the DBSCAN runs multiple times to reach the optimal solution in ESA-DCC, the complexity of ESA-DCC is higher than that of

DBSCAN. Secondly, the clustering indices applied in the fitness functions may not be suitable for non-centroid clusters since the indices were proposed for measuring the goodness of centroid-based clusters. A proposal for a clustering index for density-based clustering results will be explored in the future work of this research. Thirdly, the weights for the components of a fitness function need to be further investigated.

Six academic papers presenting the reseach work in this thesis have been accomplished and listed below.

(1) Guan, C., & Yuen, K. K. F., Towards a hybrid approach of primitive cognitive network process and agglomerative hierarchical clustering for music recommendation. In Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conference on (pp. 206-209), IEEE, 2015.

(2) Guan, C., Yuen, K. K. F., & Coenen, F., Towards an intuitionistic fuzzy agglomerative hierarchical clustering algorithm for music recommendation in folksonomy. In Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on (pp. 2039-2042), IEEE, 2015

(3) Guan, C., Yuen, K. K. F., & Chen, Q. (2017, June). Towards a Hybrid Approach of K-Means and Density-Based Spatial Clustering of Applications with Noise for Image Segmentation. In Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on (pp. 396-399). IEEE.

(4) Guan, C., & Yuen, K. K. F., The Cognitive Pairwise Rating Agglomerative Hierarchical Clustering for A Recommender System: An Application of Laptop Recommendation. Submitted to journal.

(5) Guan, C., Yuen, K. K. F., & Coenen, F., Particle Swarm Optimized Density-Based Clustering and Classification: Supervised and Unsupervised Learning Approaches. Submitted to journal.

(6) Guan, C., Yuen, K. K. F., & Yue, Y., Towards A Personalized Item Recommendation Approach in Social Tagging Systems Using Intuitionistic Fuzzy DBSCAN. Submitted to conference.

This thesis consists of 6 chapters and organized as below.

- Chapter 1 is the introduction, describing the context of the thesis and briefly introducing the background and motivation of the proposed framework.

- Chapter 2 is the literature review of the research area. This chapter reviews the details of clustering analysis and Evolutionary and Swarm Algorithms (ESAs). Three major types of clustering technologies and the corresponding representative algorithms are examined. Four mainstream ESAs, GA, PSO, DE and ABC, which will be used along this work are studied in this chapter. A number of representative hybrid methods of Clustering and ESA are also reviewed and summarized. The motivation of this work is proposed in this chapter by comparing and discussing the current methods.

- Chapter 3 proposes the framework of ESA optimised density-based clustering methods. The four ESA methods that are reviewed in Chapter 2 are applied in the framework. The design and implementation of the proposed ESA optimised density-based clustering methods are described in detail.

- Chapter 4 presents the experimental design and results for the proposed methods. The propositioned methods are evaluated by 10 datasets and compared with K-means and DBSCAN. By analyzing the experimental results, the strengths and limitations of the methods are highlighted.

- Chapter 5 presents two types of applications for the proposed methods for a number of real world problems. The suggested methods are demonstrated for the applications of recommender systems and image segmentation. Furthermore, the proposed method integrated with Cognitive Pairwise Rating (CPR), an ideal alternative of AHP, is applied to the personalized recommender system.
- Chapter 6 concludes the thesis and summarizes the future research works.

Chapter 2 Literature Review

The literature review covers three main sections: a review of clustering algorithms, an study of Evolutionary and Swarm Algorithms (ESAs), and a appraisal of current hybrid methods of clustering and ESAs.

2.1 Clustering Analysis

Clustering is an example of unsupervised learning in the machine learning field. The process of clustering can be described as grouping a set of objects into clusters with respect to the dissimilarities between them. The data objects in one cluster are similar to each other and dissimilar from the objects in other clusters. Cluster analysis has many functions in numerous data analytic applications, such as market analysis, pattern recognition and image processing. Clustering methods can generally be categorized into several classifications, such as Partional-based methods, Hierarchical-based methods, Density-based methods, Model-based methods and etc. The three mainstream categories of clustering methods are reviewed in Sections 2.1.1-2.1.3. Some of the basic conceptions and terminologies for clustering methods are reviewed according to the descriptions in [Han et al., 2011].

- **Data Matrix**

A dataset to be clustered can be represented as a data matrix. Each row of the matrix represents an object with its attributes. The structure of a n-by-p data matrix which contains $n \times p$ objects is shown in the form below.

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \tag{2.1}$$

- **Different Types of Attributes**

The attributes of each object vary depending on the meaning of the attribute value. The typical attribute types for measuring an variable are introduced as below.

Interval-Scaled Attributes are continuous measurements of a roughly linear scale, such as weight, height, weather temperature, latitude and longitude.

Binary Attributes have only two possible values, 0 or 1. The values indicate that the variable is absent (represented by 0) or present (represented by 1). It can be regarded as an “Yes or No” answer to a question for each individual. For example, for the attribute “marital status”, 0 means single and 1 means married. The meaning of 0 or 1 can also be pre-defined, such as for the attribute “gender”, 0 can be defined as female whilst 1 would be male and vice versa.

Categorical Attributes can be regarded as the generalizations of binary variables which can take on more than two states, such as color, brand, shape and so on.

Ordinal Attributes are a number of values which can be ordered in a meaningful sequence. One of the most famous example of ordinal variables is the three kinds of medals given out for a sporting competition: gold, silver and bronze.

- **Dissimilarity Matrix**

The proximities for all pairs of *n* objects can be represented by an *n*-by-*n* table shown as below, where *d(i, j)* is the measured by the dissimilarity between objects *i* and *j*.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \tag{2.2}$$

As a wildly used distance measure, **Euclidean distance** is suitable for measuring the dissimilarities between objects with multiple attributes. The computation of Euclidean distance is defined as below.

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \tag{2.3}$$

Note that this type of distance measure can only be applied to the interval-scaled attributes. The other types of attributes can be transferred into interval-scaled attributes in a preprocessing stage, before using the Euclidean Distance. In this research, the data preprocessing steps are mainly conducted by PCNP [Yuen, 2009; 2012; 2014$^{(1)}$; 2014$^{(2)}$] to deal with the difference types of attributes. The preprocessing steps are introduced in Section 5.1.2 with real-world cases.

Arbitrary Shaped Clusters

Some datasets include arbitrary shaped clusters, which means that the clusters are not centroid based, for instance, a dataset transferred from a digital facial image. Figure 2.1 presents two examples of a dataset consisting of arbitrary shaped clusters. Since most of the partitional based clustering methods are centroid based, such as K-means and K-centroid, the other types of clustering methods are applied to process this particular kind of dataset, such as hierarchical and density-based clustering methods.

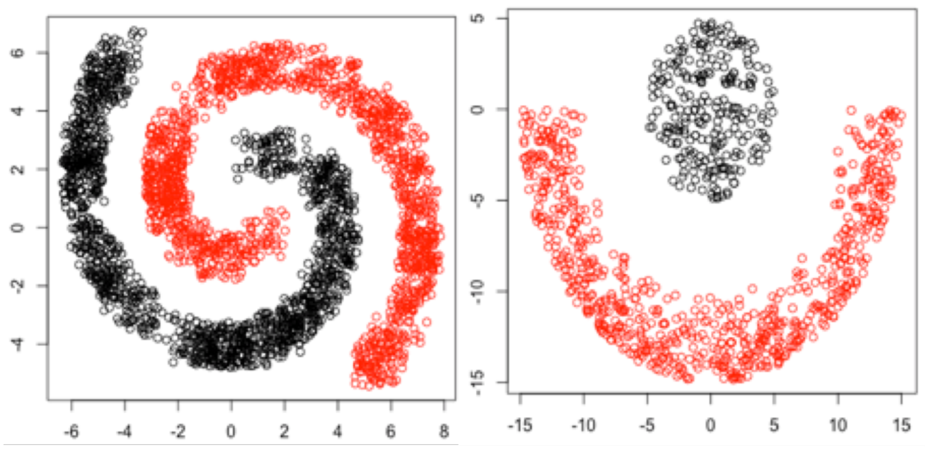


Figure 2.1 Two Datasets with Arbitrary Shaped Clusters

- **Noisy Data \ Noise**

In this work, the term noise (noisy data) refers to an object which is not assigned to any cluster. The detection of noise is a topic in clustering analysis. Some clustering methods, such as K-means, do not directly deal with noise, as a consequence, the noise in the ground truth partitions may lead to poor clustering results in the pattern of results when using such a method.

2.1.1 Partitioning Clustering

A partitioning clustering algorithm can organize a data set *D* of *n* objects into *k* clusters, where k ≤ n. The clusters are formed to optimise an objective partitioning criterion, for example, to minimize a dissimilarity value based on distance to ensure the objects within a cluster are similar, whilst the objects of different clusters are dissimilar. The most widely used and classical partitioning clustering method is K-means.

K-means

The K-means algorithm [MacQueen, 1967; Jain, 2009] takes a parameter, *k*, as input, and assigns *n* objects into *k* clusters resulting in the similarity of the objects within each cluster being high but the similarity between the objects in different clusters being low. Typically, the aim of the K-means process is to minimize the total mean-square quantization error (MSE) until the criterion function converges. The function to compute MSE is defined as below.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i| \tag{2.4}$$

where *E* is the sum of the square error for all objects in the data set; p is the point in space representing a given object; and m_i is the mean of cluster C_i (both *p* and m_i are multi-dimensional). The mean value of the objects in a cluster is regarded as the cluster's centroid. The criterion is the sum of the squared distance from the object to its cluster center. The pseudo code of classical K-means [MacQueen, 1967] is provided in Algorithm 2.1 as below.

Algorithm 2.1: K-means

```
Input: k: the number of clusters, D: a data set containing n objects.
Output: A set of k clusters.
1. Arbitrarily choose k objects from D as the initial cluster centers;
2. Assign each object to the cluster to which the object is the most similar, based
   on the mean value of the objects in the cluster;
3. Update the cluster means, i.e., calculate the mean value of the objects for each
   cluster;
4. Repeat steps 2-3 until no change.
```

2.1.2 Hierarchical Clustering

The hierarchical clustering method works by building a tree structure of the objects in a dataset. There are two major methods for Hierarchical clustering methods:agglomerative and divisive. For agglomerative methods, tree structures are built from the bottom up, and for divisive ones, the trees are built from the top down.

- **Agglomerative hierarchical clustering** starts by regarding each object as an atomic cluster and then merges these atomic clusters into larger clusters. This step is repeated until all of the objects are in a single cluster or until a certain termination condition is satisfied. The majority of hierarchical clustering methods belong to this category, they differ only in the computational style of the inter-cluster similarity. The details of typical AHC are introduced in this section.

- **Divisive hierarchical clustering** does the reverse of agglomerative hierarchical clustering by starting with putting all objects in one cluster, and then dividing the cluster into increasingly smaller clusters, until each cluster only contains one object, or until a certain termination conditions is satisfied. There are less hierarchical clustering algorithms which follow this strategy.

Agglomerative Hierarchical Clustering

The original Agglomerative Hierarchical Clustering (AHC) [Ward, 1963] was proposed more than half a century ago. In AHC, the pairs of closest clusters are iteratively merged into larger clusters until all of the objects are in a single cluster or a termination condition is satisfied [Han et al, 2011]. The three main steps of hierarchical clustering methods are summarized below with reference to [Murtagh, 1983].

i. **Initialization**: each object is initialized as an atomic cluster. The dissimilarities between atomic clusters can be computed in different ways, this was introduced in Section 2.1.1.

ii. **Merging**: the two closest clusters, C_i and C_j,are combined to form a larger cluster. The four mainstream measurements for choosing the closest pair of clusters for inter-cluster similarity are shown below, where p is an object, m_i is the centroid of clusters C_i, and n_i is the number of objects in cluster C_i

   - Minimun\Single-linkage:

$$d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'| \tag{2.5}$$

   - Maximun\Complete-linkage:

$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'| \tag{2.6}$$

   - Centroid-linkage:

$$d_{mean}(C_i, C_j) = |m_i - m_j| \tag{2.7}$$

   - Average-linkage

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'| \tag{2.8}$$

This step should be repeated until all objects are in one cluster.

iii. **Clusters generation:** a dendrogram is used to illustrate the arrangement of the merged clusters. The objects are divided into different clustering patterns by cutting the branches at an appropriate height, which is represented by the dissimilarity between clusters. For example, the dendrogram shown in Figure 2.2 can be cut by Line A and then three clusters are generated. Similarly, the dataset can be divided into five clusters if the dendrogram is cut by Line B.

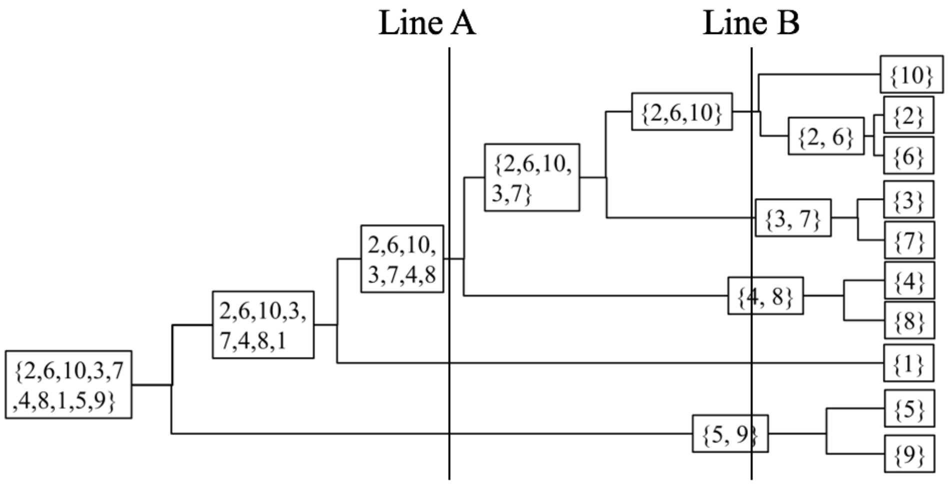


Figure 2.2: An Example of Clustering by Dendrogram.

2.1.3 Density-based Clustering

Density-based clustering methods can be used to discover clusters with arbitrary shape. The dense regions of objects in the data space are recognized as clusters, and the regions of low density are marked as noisy points (or noises). Thus, the DBSCAN grows clusters according to a density-based connectivity analysis. A number of hybrid and enhanced density-based clustering methods have been developed, for example: l-DBSCAN [Viswanath & Pinkesh, 2006], ST-DBSCAN [Birant & Kut, 2007], Rough-DBSCAN [Viswanath & Babu, 2009], P-DBSCAN [Kisilevich, Mansmann & Keim, 2010], MR-DBSCAN [He, Tan, Luo, Mao, Ma, Feng & Fan, 2011], PDS-DBSCAN [Patwary, Palsetia, Agrawal, Liao, Manne & Choudhary, 2012], Revised DBSCAN

[Tran, Drab & Daszykowski, 2013] and G-DBSCAN [Andrade, Ramos, Madeira, Sachetto, Ferreira & Rocha, 2013]. The details of DBSCAN are reviewed in this section.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was originally proposed in 1996 [Ester et al., 1996]. DBSCAN can easily find the arbitrary shape of clusters by detecting the high-density hyper-spheres and merging the close hyper-spheres into clusters. As already noted, DBSCAN uses two critical parameters, the radius of hyper-spheres (ϵ) and the minimum number of points in each hyper-sphere (*Minpts*), the clustering results of DBSCAN are sensitive to the values of these two parameters. The pseudo code for the DBSCAN algorithm is given in Algorithm 2.2.

Algorithm 2.2: DBSCAN

```
Input: Dataset $S$, Radius of each hyper-sphere ($\epsilon$), the minimum number of points in
the hyper-sphere, $MinPts$.
Output: Pattern Result, ($PR$).

1. Initialize $cid = 0$;
2. For each individual in dataset, i.e. $s \in S$,
   If s is not marked as "seen", then Mark s as "seen" and find $N\epsilon(s; S)$,
        If $card(N\epsilon(s; S)) < MinPts$, then $(PR)_{sid(s)} = 0$;
        else $cid = cid + 1$, $(PR)_{sid(s)} = cid$;
            For $s' \in N\epsilon(s; S)$ and $s'$ is not marked as "seen",
                Mark $s'$ as "seen";
                Find $N\epsilon(s'; S)$;
                If $card(N\epsilon(s; S)) \geq MinPts$, then
                    $(PR)_{sid(s')} = cid$;
   else continue next point
3. Return (PR).
```

The clustering Pattern Result (*PR*) is a list $[c_1, c_2, ..., c_n]$ where each element c_i is a cluster identifier (identifier 0 indicates the noise cluster), n is the number of records in the input dataset *S*, and the indexes indicate individual record numbers for each record *s* in *S*. A mark *seen* is used to distinguish between the records which have been processed and those which still need to be processed. $N_\epsilon(s; S)$ is a function that returns the subset of records in S, that are present in a particular cluster (hyper-sphere) of radius ϵ, that *s* in *S* belongs to $card(N_\epsilon(s; S))$ returns the cardinality of the set $N_\epsilon(s; S)$; whilst *sid*(*s*) returns the index in *PR* of *s* in *S*. The drawbacks of DBSCAN are discussed in Section 2.4 as a part of the research limitations.

2.2 Evolutional and Swarm Algorithms

2.2.1 Introduction

Evolutionary and Swarm Algorithms (ESAs) are some Computational Intelligence (CI) methods were inspired by the evolution of species and the behaviors of animals in swarms. This work covers four typical and widely used ESAs, Particle Swarm Optimization (PSO) [Kennedy, 1995], Artificial Bee Colony (ABC) [Karaboga, 2005], Genetic Algorithms (GAs) [Golberg,1989] and Differential Evaluation (DE) [Storn & Price, 1997]. The common use of each ESA is to be applied in an optimisation problem which find a set of parameter values that minimize or maximize a function in a pre-defined search space.

GA is the mainstream algorithm of evolutionary algorithms. The initial conception that the evolution could be applied in the process of optimisation has been proposed since the 1960s [Holland, 1962]. The theory of GA has been further developed by the team led by John Holland in the following decades [Holland, 1975; Holland, Holyoak, Nisbett, & Thagard, 1986]. The applications of GAs were further investigated during the 1980s [Goldberg, 1989; Grefenstette, 1985, 1987; Goldberg & Holland, 1988]. In this work, the canonical genetic algorithm is applied to develop the initial approaches and compare with other methods.

PSO was inspired by the swarming behavior that was displayed by a flock of birds, a school of fish, or even human social behavior being influenced by other individuals [Kennedy, 1995]. The developments, applications and resources of PSO before the year 2001 were summarised in [Shi, 2011]. The PSO methods developed for solving constrained optimisation problems were summarized in [Parsopoulos & Vrahatis, 2002]. Some PSO variant algorithms were proposed since the initial PSO was suggested. A standard of PSO was defined in 2007 [Bratton & Kennedy, 2007] to compare and summarize three types of PSO including original PSO, Constricted GBest and Constricted LBest. The variants were implemented and summarized in a famous R package hydroPSO [Zambrano-Bigiarini & Rojas, 2013; Zambrano-Bigiarini, Clerc & Rojas, 2013]. The different PSO algorithms presented in hydroPSO are Standard PSO 2011 (spso2011) [Clerc, 2012], Standard PSO 2007 (spso2007) [Clerc, 2012], Fully Informed Particle Swarm (fips) [Mendes, 2004], Weighted Fully Informed Particle Swarm (wfips) [Mendes, 2004], Improved PSO (IPSO) [Zhao, 2006] and Canonical PSO [Clerc, 2009]. In this work, Canonical PSO and SPSO 2011 are applied in the proposed framework. An application of this package including a detailed illustration was presented in 2013 [Zambrano-Bigiarini & Rojas, 2013]. The two reviews of PSO presented in [Banks, Vincent & Anyakoha, 2007, 2008] covers the development, hybridization and application of PSO. [García-Gonzalo & Fernández-Martínez, 2012] is a recent summary of PSO methods in 2012.

As an evolution strategy, the DE algorithm was introduced by Storn and Price in the 1990s [Storn & Price, 1997; 1995]. DE is particularly compatible to find the global optimum of a real-valued function of real-valued parameters and does not require that the function to be either continuous or differentiable. In the roughly fifteen years since its invention, DE has been successfully applied in a wide variety of fields, from computational physics to operations research [Price, Storn & Lampine, 2006]. A recent review of DE was presented in [Das, Mullick & Suganthan, 2016].

ABC were firstly defined in 2005 by Karaboga [Karaboga, 2005]. The computational processes and application areas of ABC were further extended in 2007 [Karaboga & Basturk, 2007]. The performance of ABC was compared to other EC methods such as DE, PSO and GA in [Karaboga & Basturk, 2008]. The optimisation results of the five functions demostrate that ABC algorithm performed better than the aforementioned algorithms in [Karaboga & Basturk, 2008]. Due to several insufficiencies in classical ABC, some improved ABC algorithms have been proposed. To improve the exploitation of classical ABC, the Gbest-guided artificial bee colony (GABC) was developed by incorporating the information of the global best (gbest) solution into the solution search equation in 2010 [Zhu & Kwong, 2010]. A modified ABC was developed for real parameter optimisation [Akay & Karaboga, 2012]. The development and application of ABC were reviewed in [Karaboga & Akay, 2009; Karaboga, Gorkemli, Ozturk & Karaboga, 2014].

Some comparative research has been studied to discuss the superiority of one EC method over the others. For example, [Eberhart & Shi, 1998] is a comparison between GA and PSO in 1998, similarly, [Civicioglu & Besdok, 2013] reviewed and compared PSO, DE and ABC in 2011. More reviews are mentioned in Section 2.3.1 and are conducted with respect to certain applications of ESA methods, such as optimising the performance of clustering analysis.

2.2.2 Basic Concepts

The general procedures of ESAs could be summarized as a group of candidate solutions moving in a pre-defined space in particular patterns and finally the best solution among all the candidate solutions is produced as the optimised solution. The steps of all the ESAs consist of two phases: representing the solutions as a swarm or genetics and searching for the best member of a swarm or the best chromosome of a gene pool in a search space. The common or similar terminologies applied in ESAs are introduced in this section, they include encoding, search space, fitness function, stopping criteria, etc. The details of the common terminologies are explained below.

- **Solution Representation**

There are three major ways of representing candidate solutions which are applied in ESAs: binary representation, integer representation and real-valued representation. All the representation forms can be applied in GA and the real-valued representation can be applied to PSO, DE and ABC.

In GA, each candidate solution can be encoded as a “chromosome” consisting of a number of “genes”, in comparison, for the binary representation, each solution is denoted as a bit-string. The integer representation is applied in GA, whilst the candidate solutions include ordinal parameters or cardinal attributes. Each solution is represented by a vector of integers which signify a particular meaning. This occurs when the values to be represent as genes come from a continuous rather than a discrete distribution. For example, if they represent physical quantities, such as the length, width, height, or weight of a component of a design, that can be specified within a tolerance smaller than integer values.

In PSO, each candidate solution is represented as a particle; and all the particles form the swarm. In DE, each agent represents one candidate solution. Similarly, food sources are the candidate solutions in ABC.

- **Objective Function and Fitness Function**

For an ESA, the problem to be solved is represented as an objective function. A fitness function is designed as a particular type of the objective function, to evaluate how well the candidate solutions solve the problem.

For example, GA often requires a fitness function that assigns a score (fitness) to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand. Similarly, PSO requires a fitness for each particle; ABC needs a fitness for each food source; and DE requires a fitness for each agent.

- **Search Space**

All the candidate solutions are randomly generated in the search space, which means a pre-defined range of each parameter of solutions. The dimension of the search space is the number of parameters in each candidate solution and the minimum and maximum values for each dimension should be pre-defined to determine the whole scope of the search space.

- **Stopping Criteria or Convergence Tolerance**

According to the summarization presented in [Eiben & James, 2003], the four major types of stopping criteria shown below can be applied in all the ESAs.

1. The maximum allowed CPU time elapses.
2. The total number of fitness evaluations reaches a given limit.
3. The fitness improvement remains under a threshold value for a given period of time, which means the algorithm's solution has been converged.
4. The population diversity drops under a given threshold.

2.2.3 Genetic Algorithm

Genetic algorithms (GAs) were created by John Holland in the 1960s and were further developed by his research group at the University of Michigan in the following decades. During the past half century, researchers have studied and developed the concept of GAs and broke the boundaries between GAs, evolution strategies, evolutionary programming, and other evolutionary approaches. Nowadays, the term "Genetic Algorithm" can be used to describe a number of various algorithms which could vary from the original GA method. In this work, the GA method mentioned mainly follows the Genetic Algorithms defined by [Mitchell, 1989] and [Eiben & James, 2003].

Algorithm 2.3: A Simple GA

```
1    Start with a randomly generated population of $n$ chromosomes.
2    Calculate the fitness of each chromosome in the population.
3    Repeat the following steps until $n$ offspring have been created:
     3.1  Select a pair of parent chromosomes from the current population.
     3.2  With the crossover probability, cross over the pair at a randomly chosen
          point to generate two offspring. If no crossover takes place, copy the
          parents as the two offspring.
     3.3  Mutate the two offspring at each locus with the mutation probability and
          place the resulting chromosomes in the new population.
4    Replace the current population with the new population.
5    Loop step 2-4 until the stop criteria is reached.
6    Output the best population with highest fitness.
```

A simple procedure of GA is presented in Algorithm 2.3 with respect to the context of [Mitchell, 1998]. Each iteration (Steps 2-4 in Algorithm 2.3) of this process is called a generation and a GA is typically iterated for hundreds of generations. The entire set of generations is called a run, at the end of the run, there are often one or more highly fit chromosomes in the population. As the algorithm demonstrates, the simplest form of a genetic algorithm involves three types of operators: selection, crossover, and mutation.

- **Selection**

Selection is the process to find the individuals with a higher fitness to produce the next generation. Typically, parent selection technologies are probabilistic in GAs. The high-quality individuals have high probabilities to be selected as parent, whilst the low-quality individuals have a chance to be selected but the chance is very small, such that the search cannot be too greedy to get stuck in a local best solution. For example, some widely used selections are roulette-wheel selection (also named as fitness proportionate selection), tournament selection, reward-based selection, and etc. In this work, the roulette-wheel selection is applied in the proposed GA-based clustering method described in Section 3.2.

- **Crossover**

Crossover (or sometimes referred to as recombination) is an operator which merges two individuals (which refers to the parents selected by the selection operator) into offspring individuals. The principle aim of the crossover is to mate two individuals with different but desirable features to produce an offspring that combines both of those features. This principle is inspired by produce species that give higher yields or have other desirable features in the area of plant and livestock breeders.

In GAs, the offspring (next generation) are produced by a random recombination which is named as a crossover. A crossover is a stochastic operator, which means that the choices of what parts of each parent are combined are randomly decided with respect to a pre-defined crossover rate. A crossover is also applied probabilistically, which means the parents have a small predefined chance not to be performed crossover. The offspring of a pair of parents are the same as themselves if no crossover is performed.

Various crossover methods are proposed with respect to the different genotypes (decoding style) of chromosomes. For example, the bit-string chromosomes, single-point, two-point, and uniform crossover. In this work, the single-point crossover is applied in the proposed GA-based clustering method described in Section 3.2.

- **Mutation**

Mutation is a unary variation operator in GAs. A mutation operator is stochastically preformed to one bit (genotype) of the offspring generated by the stage of crossover, a slightly changed mutant can be generated by the mutation operator.

2.2.4 Particle Swarm Optimization

PSO is a population based stochastic optimisation technique to find the best fitting solution. PSO has a number of advantages, such as flexibility, easy computational implementation, low computational requirements, low number of parameters, and efficiency [Eberhart and Shi, 1998; Shi and Eberhart, 1999; Eberhart and Shi, 2001; Poli et al., 2007]. Numerous variants of the original PSO algorithm have been proposed to further improve the performance of PSO (see Section 2.2.1 for details). The general procedures of PSO are presented as Algorithm 2.4. Two versions, the canonical PSO and the Standard PSO proposed in 2011 (SPSO-2011), are reviewed and applied in this work (see Section 3.3 for detail).

Algorithm 2.4: General PSO Procedures

```
1   Initialize the velocity and position of each particle.
2   Loop until the maximum iterations or minimum error criteria is reached.
    2.1  Calculate fitness value
    2.2  If the fitness value is better than the best fitness value (pBest) in history,
         then set current value as the new pBest
    2.3  Choose the particle with the best fitness value of all the particles as the
         gBest
    2.4  Calculate and update the velocity and position of each particle.
3   Output the best particles.
```

2.2.5 Differential Evolution

Differential Evolution (DE) was originally developed in 1995 by Storn and Price. The main procedure of DE is shown in Figure 2.3. Similar to that of GA, DE also has three operators, mutation, recombination (also can be named as crossover), and selection, but in a different order.

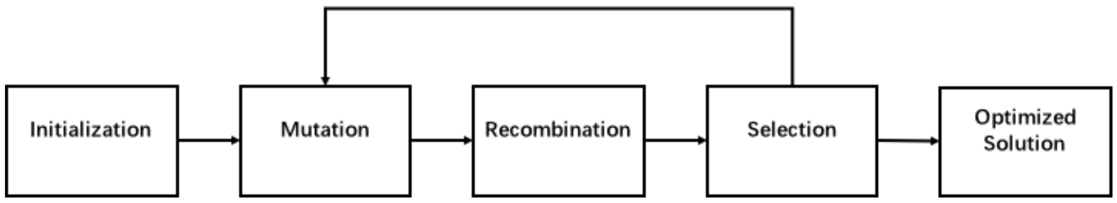


Figure 2.3 The Procedure of DE [Das & Suganthanm, 2011].

The advantages of DE have been summarized in [Storn et al. 1997] and are as follows:

- ability to handle non-differentiable, nonlinear and multimodal cost functions.
- parallelizability to cope with computation intensive cost functions.
- ease of use, since few control variables are required to steer the minimization.
- good convergence propertie.

The details about the DE algorithm are presented within the proposed DE-DCC method in Section 3.4.

2.2.6 Artificial Bee Colony Algorithm

ABC is proposed to model the specific intelligent behaviors of honey bee swarms and applies to solving combinatorial type problems. In the ABC algorithm, the candidate solutions of object functions are the food sources which can be selected or discarded by bees. The colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. A bee waiting in the dance area to make a decision to choose a food source, is called an onlooker, whereas, a bee going to the food source visited by itself previously is termed an employed bee. The bee that searches around randomly is called scout. In the ABC algorithm, half of the bees are employed artificial bees and the other half are the onlookers. One employee bee is in charge of one food source. In other words, the number of employed bees is equal to the number of food sources. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout. The main steps of ABC algorithm (which revises and improves the presentation of the paper [Karaboga, 2005]) are given in Algorithm 2.5.

Algorithm 2.5: ABC Algorithm

```
1   Send the scouts onto the initial food sources
2   Repeat from 2 until requirements are met.
    2.1  Send the employed bees onto the food sources and determine their nectar
         amounts
    2.2  Calculate the probability value of the sources with which they are preferred
         by the onlooker bees
    2.3   Send the onlooker bees onto the food sources and determine their nectar
          amounts
    2.4   Stop the exploitation process of the sources exhausted by the bees
    2.5   Send the scouts into the search area for discovering new food sources,
          randomly
    2.6   Memorize the best food source found so far
3   Output Best food source.
```

2.3 Hybrid Approaches of ESAs and Clustering

2.3.1 General Review

ESA algorithms have been widely applied in data mining fields such as clustering analysis. The fact that they are easy to be trapped in local best result is one of the main problems existing in classical clustering method. The main advantage of ESA is the stochastic optimisation, which can overcome the drawback of the search strategy in the classical clustering method. To improve the efficiency and accuracy of the classical clustering analysis algorithms, the clustering problems could be represented as functions to be optimised by ESA technologies. Some reviews have been made in the past two decades to summarize the various ESA optimised clustering algorithms.

A more detailed review of GA-based clustering methods was accomplished by presenting the framework of GA clustering methods step by step [Naldi, Carvalho & Campell, 2008]. The fitness functions employed in the different GA clustering algorithms are summarized and explained in detail. The work presented in [Hruschka, Campello & Freitas, 2009] is a survey of GA clustering which is similar to [Naldi et al., 2008], but more methods were covered. Some more Evolutionary Algorithms (EAs) applied in data mining were reviewed in [Freitas, 2008], which summarise the applications of EAs in the field of Data Mining including Clustering. The different types of fitness evaluations in EAs clustering methods can be roughly summarized into two types: minimize the intra-cluster (within-cluster) distance and maximize the inter-cluster (between-cluster) distance.

PSO has been used to support clustering in a number of studies. In [Van der Merwe & Engelbrecht, 2003], two PSO methods were proposed: one is to find the centroids of clusters and another one is to use K-means clustering to seed the initial swarm. In [Chen & Ye, 2004], PSO was applied to search the cluster centres in the arbitrary data set automatically, although, in [Potok & Palathingal, 2005], PSO was coupled with the K-means clustering to cluster document collections. In [Niknam & Amiri, 2010], a hybrid method, FAPSO-ACO-K, was proposed which combined Fuzzy Adaptive Particle Swarm Optimization (FAPSO), Ant Colony Optimization (ACO) and K-means so as to find the best cluster partition in the nonlinear partitional clustering problem. In [Xu, Xu & Wunsch, 2012], a framework was proposed for Differential Evolution Particle Swarm Optimization (DEPSO) based clustering, which combined DE with PSO. However, to the best knowledge, no work has been directed at using PSO for the purpose of DBSCAN parameter optimisation.

A review of PSO algorithms applying to clustering problems was presented in [Rana, Jasola & Kumar, 2011]. The variants of classical PSO methods applied in clustering were covered in this paper, however, the details of algorithms are not included. One of the recent reviews of PSO clustering methods was presented in [Alam, Dobbie, Koh, Riddle & Rehman, 2014]. In [Alam et al, 2014], PSO clustering methods were classified into two types, PSO hybridized for data clustering and PSO as a data clustering method. A number of papers presenting PSO clustering algorithms were summarized. UCI machine learning datasets [Dua & Karra Taniskidou, 2017] are used for testing and validation, the experimental results indicate that almost all PSO clustering methods have a higher efficiency and accuracy than classical clustering. However, the limitation is that the details of PSO clustering algorithms, such as the choice of fitness functions, were not mentioned.

A new SI clustering method, MEPSO clustering algorithms, was proposed in [Abraham, Das & Roy, 2008]. The procedures and the Fitness Functions adopted in each method of the three kinds of SI clustering algorithms mentioned were presented in detail. An experiment was presented to compare the performance of Fuzzy C-means (FCM), Fuzzy clustering with Variable length Genetic Algorithm (FVGA) and MEPSO-clustering algorithm. The test results show the superiority of the MEPSO-clustering algorithm, both in terms of accuracy and efficiency. The limitation of this review is that only two types of SI clustering methods are covered. More ESA optimised clustering methods are reviewed in more detail in next section (Section 2.3.2) with the fitness functions applied in the hybrid methods.

In the context of DBSCAN, the clustering approach of interest, with respect to the work presented in this paper, is the research that has been directed at applying ESAs to optimise the performance of DBSCAN One example is that of [Jiang, Li, Yi, Wang & Hu, 2011] where a hybrid partitioning-based DBSCAN method is proposed that uses a modified ant clustering algorithm but they did not consider parameter optimisation. One example, where the nature of the parameters used in DBSCAN was considered, can be found in [Lin, Chang & Lin, 2005], where a Genetic Algorithm with a Density-Based Approach for Clustering (GADAC) was proposed to determine the nature of the parameters used by DBSCAN to provide satisfactory clustering results. GADAC determines the range of parameters in the pre-processing step before GA operations; the entire parameter space is not considered. The above methods have a number of limitations. Firstly, the encoding methods of clustering results are based on numerating all items, which are complex to search the optimal solution especially when the data size is large. Secondly, the number of identified clusters cannot be controlled.

2.3.2 Fitness Functions Applied in Current ESA-Clustering Methods

There are two approaches in applying PSO to clustering problem solving which were developed in 2003 [Van der Merwe & Engelbrecht, 2003]. Particle x_i is defined as x_i =

m_{i1} , ..., m_{ij} , ..., m_{iN_c} , where m_{ij} is the j^{th} cluster centroid vector of the i^{th} particle in cluster *Xij*. The fitness function is designed as below.

$$J_e = \frac{\sum_{j=1}^{N_c} [\sum_{\forall Z_p \in C_{ij}} d(z_p, m_j)/|C_{ij}|]}{N_c} \tag{2.9}$$

$$d(z_p, m_j) = \sqrt{\sum_{k=1}^{N_d} (z_{pk} - m_{jk})^2}, m_j = \frac{1}{n_j} \sum_{\forall z_p \in C_j} z_p \tag{2.10}$$

where *d(zp,mj)* and *mj* are the centroid of cluster *j* defined as below; |*Cij*| is the number of data vectors belonging to cluster *Cij*; *Nc* is the number of clusters; *zp* is the p^{th} data vector; *nj* is the number of data vectors in cluster *j*.

Two artificial classification problems and four data sets from the UCI depository were applied in the experiment to compare the performance of K-means, PSO and the proposed hybrid clustering method. The four UCI datasets are Iris, Wine, Breast cancer and Automotives.

The hybrid method of PSO and K-means clustering is presented in [Cui et al., 2005]. The fitness function in this hybrid method is designed with respect to the document clustering case and termed the Average Distance of Documents to the cluster Centroid (ADDC) as in the equation below.

$$f = \frac{\sum_{i=1}^{N_c} \frac{\sum_{j=1}^{p_i} d(o_i, m_{ij})}{p_i}}{N_c} \tag{2.11}$$

where *mij* is the j^{th} document vector of the i^{th} cluster; *Oi* is the centroid of the i^{th} cluster; *d(oi,mij)* is the distance between m_{ij} and *Oi*; *Pi* is the number of documents in cluster *Ci* and *Nc* is the number of clusters. The four document datasets derived from Text REtrieval Conference (TREC) collections are used in the experiments to compare the performance of K-means and PSO clustering algorithm.

Intra-cluster distance was used to compute the fitness in HPSO-clustering [Alam, Dobbie, Riddle & Naeem, 2010], which is the hybrid method of PSO and AHC. In the HPSO-clustering method, each cluster centroid is modelled as a particle of the PSO process. The particles are merged following the average attribute values shown as below.

$$X_i = \frac{X_i(nearest) + X_i(loser)}{2} \tag{2.12}$$

where X_i is the newly formed particle after merging; $X_i(nearest)$ is the particle which is more populated; $X_i(loser)$ is the particle which is less populated. Five popular UCI datasets, including Iris, Breast Cancer, Wine, Vowel and Glass, are used in the comparison of HPSO clustering, PSO clustering and K-means.

DE, PSO and GA were applied in partitional clustering problems in [Paterlini & Krink, 2004]. In this hybrid method, the fitness function was defined as below.

$$F(X_{nxp}, m) = \begin{cases} f(X_{nxp}, H) & if\ H \subset G = \{G^1, G^2, ..., G^{N(n,g)}\} \\ K & if\ H \not\subset G = \{G^1, G^2, ..., G^{N(n,g)}\} \end{cases} \tag{1}$$

[Paterlini & Krink, 2006] presented the further development of [Paterlini et al., 2004].

An ABC clustering was developed in 2010 [Zhang, Ouyang & Ning, 2010]. The total mean-square quantization error (MSE) [Güngör & Ünler, 2007], which can also be described as the total within-cluster variance shown below, was applied in this method as the fitness function.

$$Perf(O, G) = \sum_{i=1}^{N} \min\{||o_i - C_l||^2 | l = 1, ..., K\} \tag{2.13}$$

where $||o_i - C_l||$ is the distance between object o_i and center C_l; the distance could be computed by Euclidean distance.

Several EC and clustering methods such as GA, tabu search (TS) [Al-Sultan, 1995], Simulated Annealing (SA) [Selim & Alsultan, 1991], ACO, K-NM-PSO [Kao et al. 2008] were used to compare with ABC in three clustering problems (Iris, Thyroid and Wine from UCI datasets).

The famous clustering measurement function, MSE, was also employed in the hybrid method of Cooperative ABC (CABC) and K-Means clustering [Zou, Zhu, Chen & Sui, 2010].

MSE was also used as the fitness function of the Hybrid Artificial Bee Colony (HABC) and was employed in clustering problems [Yan, Zhu, Zou & Wang, 2012]. The HABC is a hybrid method of GA and ABC. ABC, PSO, GA, CABC [Yan et al., 2012], Cooperative Particle Swarm Optimization (CPSO) [Van den Bergh & Engelbrecht, 2004] and the K-means algorithm were tested on six real clustering problems selected from UCI, such as Iris, Wine, CMC, WBC, Glass and LD.

ABC was proposed as a clustering approach in 2011 [Karaboga & Ozturk, 2011], the clustering problem was stated as the process of minimizing the sum of the squared Euclidean distances between each object and the center of the cluster. The fitness function to be minimized was designed as below.

$$J(w,z) = \sum_{i=1}^{N}\sum_{j=1}^{K} w_{ij}||x_i - z_j||^2 \tag{2.14}$$

where $z_j, j = 1,...,K$ is the center of the j^{th} clusters which can be computed as below.

$$z_j = \frac{1}{N_j}\sum_{i=1}^{N} w_{ij}x_i \tag{2.15}$$

where N_j is the number of objects in the j^{th} cluster, w_{ij} is the association weight of object x_i with the j^{th} cluster; w_{ij} is 1 if object i is allocated to cluster j, otherwise w_{ij} is 0.

A hybrid clustering method of Fuzzy adaptive PSO, ACO, and K-means was developed in 2010 [Niknam & Amiri, 2010]. The fitness function, (defined as performance function *Perf*(*X,C*) of this hybrid method, is the total within-cluster variance or the total mean-square quantization error shown as below.

$$Perf(X,C) = \sum_{i=1}^{N} Min\{||X_i - C_l||^2 | l = 1, ..., K\} \tag{2.16}$$

where $\{X_i|i = 1,2,...n\}$ is the set of points to be clustered; $\{C_l|l = 1,2,...K\}$ is the set of clusters

A number of EC clustering algorithms such as PSO, ACO, GA, Simulated Annealing (SA), Tabu search (TS), honey bee mating optimization (HBMO), and several hybrid methods such as PSO-SA, ACO-SA, PSO-ACO are tested in [Niknam et al., 2010] to compare with the proposed FAPSO-ACO-K clustering method. Four artificial data sets and six real-life data sets are used in the experiment. The real-life data sets include Iris, Wine, Vowel, Contraceptive Method Choice (CMC), Wisconsin breast cancer and Ripley's glass. The experiment results illustrates that the proposed FAPSO-ACO-C method could find a better cluster pattern than the other methods tested in the experiment.

To search the clusters in arbitrary shape, a PSO-Clustering method was developed in 2004 [Chen & Ye, 2005]. The cluster process obeys the following rules. A point x_i is assign to cluster C_j where i = 1, 2, ..., N and j = 1,2,...,K if

$$||x_i - z_j|| < ||x_i - z_p||, p = 1, 2, ..., K \ and \ j \neq z. \tag{2.17}$$

where z_p is the centre of cluster C_j. The fitness function adopted in this clustering method is given as below.

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} ||x_i - z_j||^2 \tag{2.18}$$

$$fitness = k/(J + J_o) \tag{2.19}$$

where *k* is a positive constant, and J_O is a small-valued constant. A hybrid clustering method of DE and K-means was developed in 2008 [Das, Abraham & Konar, 2008]. The similar hybrid method of PSO and K-means was also implemented for comparison with the DE clustering method. In this hybrid method framework, two famous clustering validation indices, the DB index and the CS index, were used to build the fitness functions in the DE-clustering methods. [Das et al, 2008]. The clustering index-based fitness functions are shown as below.

$$f_i = \frac{1}{CS_i(K) + eps} \tag{2.20}$$

$$f_i = \frac{1}{DB_i(K) + eps} \tag{2.21}$$

where *i* indicates each partition yielded by the i^{th} chromosome of DE or each particle of PSO.

The criteria Trace within criterion (TRW) and Variance ratio criterion (VRC) are applied to the fitness function of the latest hybrid clustering method of DE and K-means [Tvrdík & Křivý, 2015]. The functions for computing TRW are shown below.

$$TRW = TR(\mathbf{W}) \tag{2.22}$$

$$\mathbf{W} = \sum_{l=1}^{k} \mathbf{W}_l \tag{2.23}$$

$$\mathbf{W}_l = \sum_{j=1}^{n_l} (z_j^{(l)} - \overline{z}^{(l)})(z_j^{(l)} - \overline{z}^{(l)})^T \tag{2.24}$$

$$\text{where } z_j^{(l)} = (\sum_{j=1}^{n_l} z_j^l)/n_l, n_l = |C_l| \tag{2.25}$$

where $\overline{z}$ is the vector of attributes for the j^{th} object of the cluster. The functions for computing VRC are shown below.

$$VRC = \frac{tr(\mathbf{B})/(k-1)}{tr(\mathbf{W})/(n-k)} \tag{2.26}$$

$$\mathbf{B} = \sum_{l=1}^{k} n_l(\overline{z}^{(l)} - \overline{z})(\overline{z}^{(l)} - \overline{z})^T \tag{2.27}$$

$$\text{where } \overline{z} = (\sum_{i=1}^{n} z_i)/n, n = \sum_{l=1}^{k} n_l \tag{2.28}$$

The framework of differential-evolution-particle-swarm-optimization (DEPSO)-based clustering was proposed in 2012 [Xu et al., 2012]. This hybrid method was developed by combining DE and PSO. Some specific clustering validation indices are applied as fitness functions in DEPSO-based clustering frameworks, such as the Calin`ski-Harabasz (CH) index, the CS index, the Davies-Bouldin (DB) index, the Dunn indices (DI), the I index, and the silhouette statistic (SIL) index. Given a set of N data points, $X = (x_1, .., x_N)$ is assigned to K clusters $C = \{C_1, ...C_K\}$ and the set of the centroids of all clusters is $\{m_i: i = 1, ..., K\backslash\}$, the aforementioned indices are defined in Table 2.1.

Table 2.1: Clustering Validation Indices Used as Fitness Functions

CI	Formulae	Details	Min or Max
CH		$Tr(\mathbf{S}_B) = \sum_{i=1}^{K} \|\|m_i - m\|\|^2$ and $Tr(\mathbf{S}_W) = \sum_{i=1}^{K} \sum_{j=1}^{N_i} \|\|x_j - m_i\|\|^2$	Max
CS	$CS(K) = \frac{\sum_{i=1}^{K} (\frac{1}{N_i} \sum\limits_{x_i \in C_i} \max\limits_{x_j \in C_i} D(\mathbf{x}_l, \mathbf{x}_j))}{\sum_{i=1}^{K} (\max\limits_{j \in K, j \neq i} D(m_i, m_j))}$	/	Min
DB	$DB(K) = \frac{1}{K} \sum_{i=1}^{K} R_i$	$R_i = \max\limits_{j, j \neq i} (\frac{e_i + e_j}{D_{ij}})$ $D_{ij} = \|\|m_i - m_j\|\|^2$ e_i is the average errors for clusters C_i $e_i = (1/N_i) \sum_{x \in C_i} \|\|x - m_i\|\|^2$	Min
DI	$Du(K) = \min\limits_{i=1,\ldots,K} (\min\limits_{j-1,\ldots,K, j \neq i} (\frac{D(C_i, C_j)}{\max l=1,\ldots,K \delta(C_l)}))$	$\delta(C_i) = \max\limits_{\mathbf{x}, \mathbf{y} \in C_i} D(\mathbf{x}, \mathbf{y})$ $D(C_i, C_j) = \min\limits_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} D(\mathbf{x}, \mathbf{y})$	Max
GD_{33}	$D_3(C_i, C_j) = \frac{1}{N_i N_j} \sum\limits_{x \in C_i, y \in C_j} D(x, y)$	$\delta_3(C_i) = 2 \frac{\sum\limits_{\mathbf{x} \in C_i} D(\mathbf{x}, \mathbf{m}_i)}{N_{ij}}$ $D(C_i, C_j) = \min\limits_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} D(\mathbf{x}, \mathbf{y})$	Max
GD_{53}	$D_5(C_i, C_j) = \frac{1}{N_i + N_j} (\sum\limits_{x \in C_i} D(x, m_j) + \sum\limits_{y \in C_j} D(y, m_i))$	$\delta_3(C_i) = 2 \frac{\sum\limits_{\mathbf{x} \in C_i} D(\mathbf{x}, \mathbf{m}_i)}{N_{ij}}$ $D(C_i, C_j) = \min\limits_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} D(\mathbf{x}, \mathbf{y})$	Max
I	$I(K) = (\frac{1}{K} \times \frac{E_1}{E_K} \times \max\limits_{i,j=1,\ldots,K} \|\|m_i - m_j\|\|)^p$	$p \geq 1$ and $p \in \mathbb{R}$ $E_K = \sum\limits_{i=1}^{K} \|\|x - m_i\|\|$	Max
SIL	$SIL = \frac{1}{N} \sum\limits_{j=1}^{N} s_j$	$s_j = \frac{b_j - a_j}{\max(a_j, b_j)}$ For cluster C_i and C_h, $h = 1, \ldots, K$ and $i \neq h$ $a_j = (1/N_i - 1) \sum\limits_{l=1,\ldots,N_i, l \neq j} \|\|\mathbf{x}_j - \mathbf{x}_l\|\|$ $b_j = \min\limits_{h=1,\ldots,K, h \neq i} (1/N_h) \sum_{\mathbf{x}_l \in C_h} \|\|\mathbf{x}_j - \mathbf{x}_l\|\|$	Max

2.4 Research Gap

- **Limitations of Partitioning Clustering and Hierarchical Clustering**

Partitioning Clustering methods were designed for centroid-based problems. The arbitrary shaped clusters cannot be detected and the complexity of hierarchical clustering is higher than most of the clustering methods. Subsequently, the results of both hierarchical and partitioning clustering are easily influenced by the noises in a dataset.

- **Limitations of Density-based Clustering**

DBSCAN and the density-based clustering developed on the basis of DBSCAN have three drawbacks. Firstly, it lacks a method to determine the appropriate settings of the two parameters. Thus, manual tuning appears to be the only option. Secondly, unlike K-means, the number of clusters cannot be controlled by the users since DBSCAN does not support the idea of fixing the number of clusters upon start up. Thirdly, DBSCAN cannot be used directly as a supervised learning method to perform classification. The three drawbacks of DBSCAN are illustrated by a simple problem of clustering a dataset of 10 items as shown in Table 2.2.

Example 2.1

Table 2.2: Sample Dataset

Item ID	Attribute 1	Attribute 2	Cluster Label
1	-8.055	-2.913	1
2	7.111	3.188	2
3	6.953	-4.693	3
4	-3.627	-7.416	4
5	5.732	3.648	2
6	6.988	-3.216	3
7	-0.041	-9.207	4
8	-1.983	-8.748	4
9	6.827	5.266	2
10	-1.306	-8.633	4

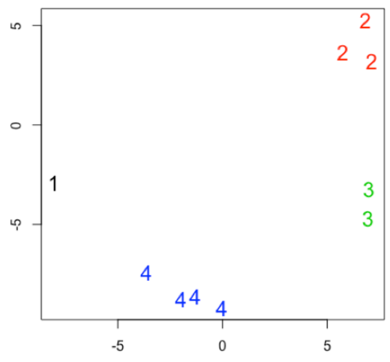


Figure 2.4 Sample Dataset

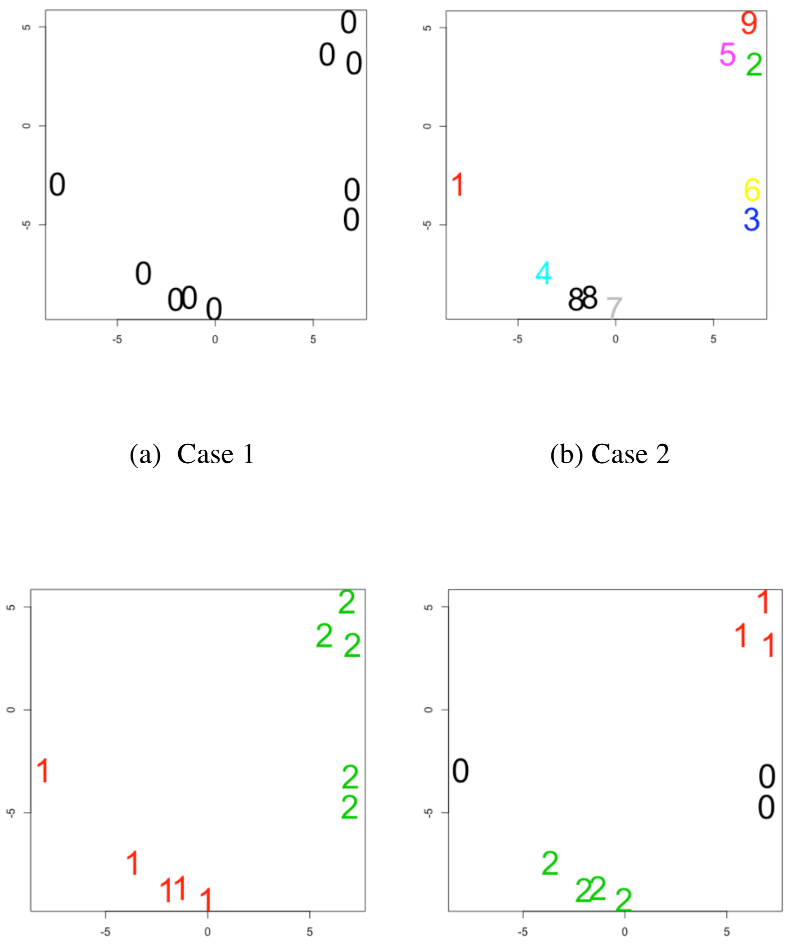
Firstly, to demonstrate the parameter setting problem, the two parameters are randomly set for four different cases as shown in Table 2.3. The clustering result for each case, generated using DBSCAN, is shown in Figure 2.5. Inspection of Figure 2.5 indicates that the known clustering shown in Table 2.3 is not found. The results are also presented in Table 2.3. The second column of the table gives the clustering pattern result PR, the third column gives the parameter settings, the fourth column gives the obtained Czekanowski Dice (CD) coefficient and the last column gives the number of clusters. Note that with respect to the CD coefficient, the higher the value, the better the clustering result in comparison with ground truth clustering.

Secondly, to demonstrate the problem of the number of clusters, it can be observed that the numbers of clusters of the four cases differ and are different from the number of ground truth clusters which is four as shown in Table 2.3.

Thirdly, the cluster labels in Table 2.3 cannot be optionally used in the training to perform classification.

Table 2.3: Clustering Results of Sample Dataset

Case	PR	$(\epsilon, MinPts)$	CD Coef.	No. of Clusters
1	0 0 0 0 0 0 0 0 0 0	(9, 7)	0.364	0
2	1 2 3 4 5 6 7 8 9 8	(1, 1)	0.182	9
3	1 2 2 1 2 2 1 1 2 1	(7, 0)	0.667	2
4	0 1 0 2 1 0 2 2 1 2	(6, 3)	0.909	2



(a) Case 1 (b) Case 2

(b) Case 3 (d) Case 4

Figure 2.5 Four Different Clustering Results of the Sample Dataset

To overcome the first drawback, this thesis proposes a novel framework ESA-DCC to search for the most appropriate parameters for DBSCAN. To overcome the second and third issues, this thesis also presents a number of fitness functions, for the use with ESA-DCC. The details of the proposed framework and fitness functions are introduced in chapter 3.

Chapter 3 Evolutionary and Swarm Algorithm Optimised Density-based Clustering and Classification

3.1 ESA-DCC Framework

The framework of Evolutionary and Swarm Algorithm optimised Density-based Clustering and Classification (ESA-DCC) is proposed in this chapter. The general procedures in the flowchart are presented in Figure 3.1.

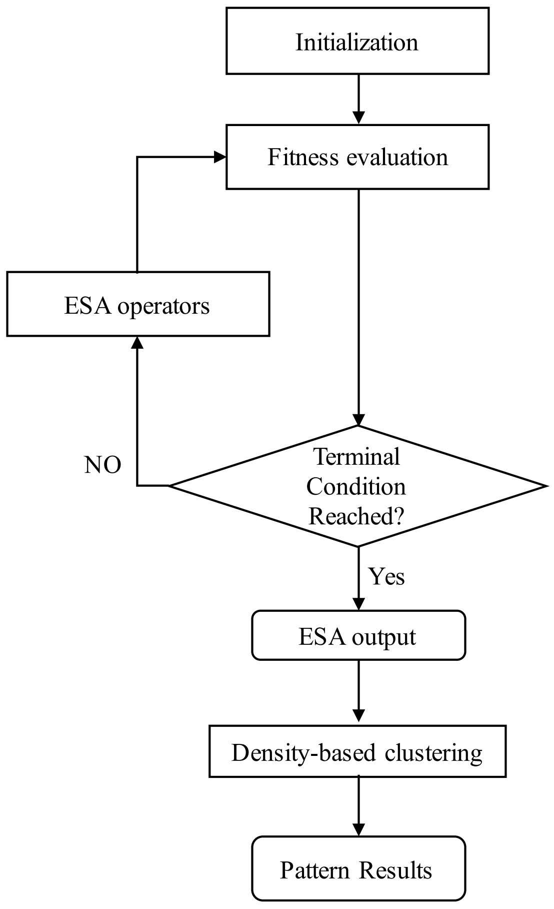


Figure 3.1 The General Procedure of Evolutionary and Swarm Algorithm optimised Density-based Clustering and Classification

- **Solution Representation**

Each candidate solution consists of two parameters, i.e. *minpts* and radius. The two parameters are essential for producing the clustering results by DBSCAN.

- **Fitness Functions**

The candidate solution is a pair of parameters required by DBSCAN to produce pattern results. A pattern result is a vector of integers, whilst one integer is labelled as one individual of the dataset with a cluster identifier, one cluster identifier refers to the cluster in which the individual belongs to. The fitness functions are designed to measure how good the clustering result is. A series of fitness functions are proposed for this framework and presented in this section. The fundamental distinction is that if the "ground truth" target class values of the records in the dataset are not used in ESA-DCC we have unsupervised learning (clustering), if they are used we have supervised learning (classification). Both Internal and External Indices are used to measure clustering results. Class labels (ground truth values) are needed for the calculation of the external index, whilst calculations of the internal indices do not require ground truth values. The unsupervised fitness function for PSO-DCC, F_{usp}, is defined as follows:

$$F_{usp} = f_{Int} + f_{NK} \tag{3.1}$$

where f_{Int} is an internal clustering index function, f_{NK} (Eq.15) is the sum of the function to control the number of clusters (f_K) and the noise minimization function (f_{Noise}).

Two widely used clustering indices, the Davies-Bouldin (DB) index [Davies & Bouldin, 1979] and Silhouette (SIL) index [Rousseeuw, 1987], are applied for f_{Int} in this thesis. Given a set of N data points $S = (s_1, \ldots, s_N)$ assigned to K clusters $C = \{C_1,\ldots,C_i,\ldots,C_K\}$ and the centroid of each cluster m_i, $i = 1,\ldots,K$, $C_i = \{s_{i1},\ldots,s,\ldots,s_{in_i}\}$ is the i^{th} cluster, where n_i is the number of the data points in C_i. The DB index is calculated as follows:

$$f_{DB} = \frac{1}{K}\sum_{i=1}^{K} \max_{i' \in \{1,\ldots,K\}, i' \neq i} \{\frac{e_i + e_{i'}}{||m_i - m_i'||^2}\}, e_i = (1/n_i)\sum_{j=1}^{n_i} ||s_j^i - m_i||^2 \quad (3.2)$$

where e_i and $e_{i'}$ are the measures of scatters within the clusters C_i and $C_{i'}$ respectively. The silhouette statistic (SIL) index is calculated using:

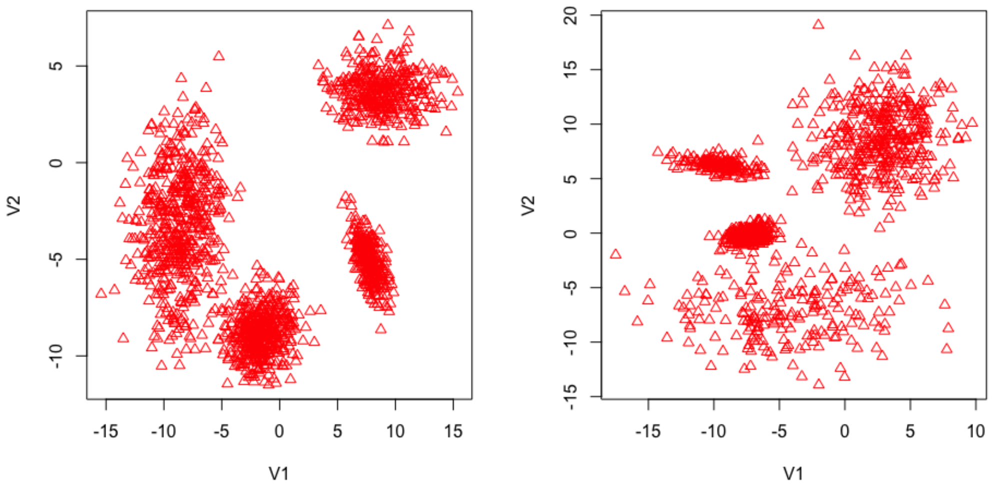
$$f_{SIL} = \frac{1}{K}\sum_{i=1}^{K}\Big(\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{b_j^i - a_j^i}{\max\,(a_j^i, b_j^i)}\Big), \text{where} \quad (3.3)$$

$$a_j^i = \frac{1}{n_i - 1}\sum_{k=1, k\neq j}^{n_i} ||s_j^i - s_k^i|| \quad (3.4)$$

$$b_j^i = \min_{h\in\{1,\ldots,K\}, h\neq i}\Big\{\frac{1}{n_h}\sum_{k=1}^{n_h}||s_j^i - s_k^h||\Big\} \quad (3.5)$$

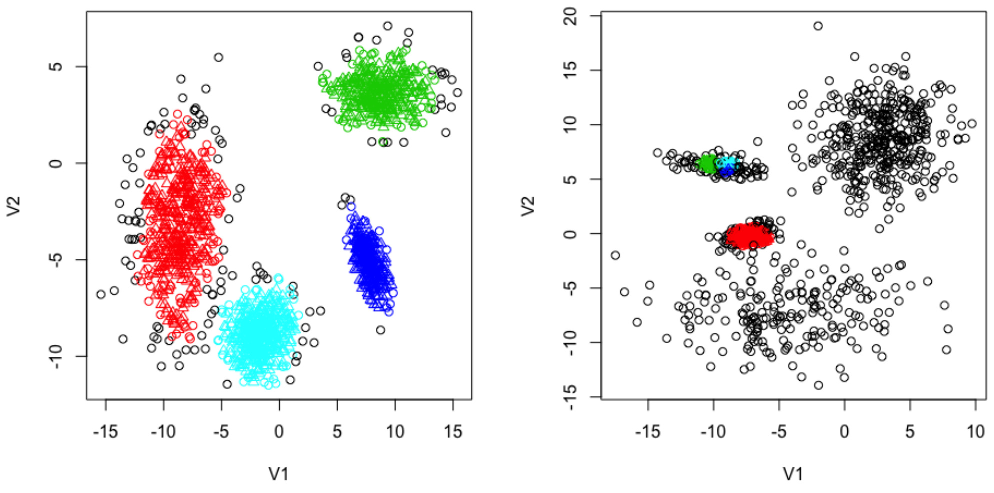
where: (i) a^i_j is the average distance between a data point s^i_j belonging to a cluster C_i and all other data points in C_i, and (ii) b^i_j is the minimum average distance between the j^{th} data point in the cluster C_i and all the data points in the other clusters $\{C_h : h \neq i\}$. The lower the DB index the better the clustering result, whilst the higher the SIL index the better the clustering result. By default, the ESA-DCC is used to minimize the objective function, therefore f_{Int} is f_{DB} or $-f_{SIL}$ in ESA-DCC.

f_{Int} cannot be solely used as a fitness function since the best internal indices for ESA-DCC do not lead to the best pattern results. As shown in Figure 3.2, all the data points are clustered in one cluster when either $f_{Int} = -f_{SIL}$ or $f_{Int} = f_{DB}$ is minimized.



(a) Case 1 (b) Case 2

Figure 3.2: Clustering Pattern Results Using ESA-DCC with f_{Int} or f_{Noise} as Fitness Function



(a) Dataset 07 (b) Dataset 08

Figure 3.3: Results by ESA-DCC with f_K as Fitness Function

To address this problem, f_{NK} is also used. f_{NK} returns the sum of the function for the number of clusters (f_K) and the noise minimization function (f_{Noise}). f_{NK} is defined as follows:

$$f_{NK} = f_K + f_{Noise}, \text{ where} \tag{3.6}$$

$$f_K = \frac{abs(\max(PR) - K)}{K} \tag{3.7}$$

$$f_{Noise} = \frac{card(\{PR_i \in PR : PR_i = 0\})}{N} \tag{3.8}$$

f_K is used to overcome the drawback of DBSCAN, which is that the number of clusters cannot be controlled by users. f_K is further used to calculate the ratio of the absolute difference between the number of clusters during the ESA-DCC procedure (i.e. *max*(*PR*)) and the number of clusters determined by user (i.e. *K*) to *K*. f_K can be minimized to 0 when *max*(*PR*) = *K*; therefore, the number of clusters in ESA-DCC can be controlled by the users. However, f_K cannot be solely used as a fitness function since the pattern results shown in Figure 3.3 may be generated by ESA-DCC. f_{Noise} is used to compute the percentage of noises in pattern results during ESA-DCC, such that it can be used to minimize the number of noises in the pattern results of ESA-DCC. f_{Noise} cannot be solely used as a fitness function since all data points are grouped into one cluster (Figure 3.2). f_{NK} can be used as a fitness functions for unsupervised ESA-DCC (i.e. $F_{usp} = f_{NK}$) if no suitable internal index is appropriate for the dataset to be clustered.

If the ground truth target class values of dataset are used in ESA-DCC, then ESA-DCC can be performed as classification which is supervised learning. A supervised fitness function for ESA-DCC, F_{spd}, is defined as follows:

$$F_{spd} = f_{Ext} \tag{3.9}$$

where f_{Ext} is the external clustering index function. An external index function measures the similarity between two partitions, (Partition 1 and Partition 2). In this case, the set of classes represents the set of the clusters in Partition 1, while Partition 2 signifies some other pattern results which we want to determine the quality of. In other words, the similarity of Partition 2 compared to the "ground truth" of Partition 1 indicates the accuracy of Partition 2. When considering a pair of points, α and β, in Partitions 1 and 2, there are four possibilities:

- αα: the two points belong to the same cluster in both partitions.
- αβ: the two points belong to the same cluster in Partition 1 but not in Partition 2.
- βα: the two points belong to the same cluster in Partition 2 but not in Partition 1.
- ββ: the two points do not belong to the same cluster in either partition.

A widely used external index is the Czekanowski-Dice index [Czekanowski, 1909], this was thus adopted in the supervised fitness function for ESA-DCC. The CD index is defined as follows:

$$f_{CD} = \frac{2\alpha\alpha}{2\alpha\alpha + \alpha\beta + \beta\alpha} \tag{3.10}$$

The higher the CD index the better pattern the result. Given that ESA-DCC is designed to minimize the fitness value, by default $f_{Ext} = f_{CD}$ was used.

The computational complexities of the proposed fitness function components and the four fitness functions applied in this work are presented in Table 3.1.

Table 3.1: Computational Complexities of Proposed Fitness Functions

Fitness function component / Fitness function	Computational Complexity
f_{Noise}	$O(n)$
f_K	$O(n)$
f_{DB}	$O(n)$, if $K \ll n$
f_{SIL}	$O(n^2)$
f_{CD}	$O(n^2)$
$F_{usp} = f_{NK} = f_{Noise} + f_K$	$O(n)$
$F_{usp} = f_{NK} + f_{DB}$	$O(n)$, if $K \ll n$
$F_{usp} = f_{NK} + f_{SIL}$	$O(n^2)$
$F_{spd} = f_{CD}$	$O(n^2)$

A fitness function is chosen on a case-by-case basis. More details regarding the choosing of fitness functions is discussed in Section 5. $F_{usp} = f_{NK}$ is the lowest complexity unsupervised function. The time complexities of $F_{usp} = f_{NK} + f_{Int}$ vary from the different internal indices applied in the function.

For computing an external clustering index, it is required to check all of the pairs of points in two partitions and it takes $O(n^2)$ time, such that the complexity of any supervised fitness function is equal to or higher than $O(n^2)$.

- **Search Space**

Since each one of the candidate solutions has two parameters, the best solution will be searched in a 2-dimension space.

- **Stop Criteria or Convergence Tolerance**

Two stop criteria are applied in the proposed method by default. When the total number of fitness evaluations reaches a given limit, or while the fitness improvement remains under a threshold value for a given period of time, the algorithm will be stopped and the optimised solution will be produced.

3.2 Genetic Algorithm optimised Density-based Clustering and Classification

The proposed Genetic Algorithm optimised Density-based Clustering and Classification (GA-DCC) method is based on the idea of applying GA and cluster measurement indices to optimise the input parameter settings for the algorithm. The procedure of GA-DCC is illustrated in Algorithm 3.1. The details are illustrated as follows.

Algorithm 3.1. GA-DCC

```
1   Initialize the population with random candidate solutions;
2   Evaluate the fitness of each chromosome in the population
3   Create a new population by repeating following steps until the new population
    is complete.
    3.1 **Selection**: Select two parent chromosomes from a population according to
        their fitness.
    3.2 **Crossover:** With a crossover, probability cross over the parents to form a
        new offspring. If no crossover was performed, offspring is an exact copy
        of parents.
    3.3 **Mutation:** With a mutation probability mutate new offspring at each
        position in chromosome.
    3.4 Place new offspring in a new population.
4   Use newly generated population for a further run of algorithm.
5   If the end condition is satisfied, stop, and return the best solution in current
    population.
6   Generate the DBSCAN result by best solution.
```

1 Initialize population with random candidate solutions.

A population of 10 chromosomes are randomly generated. Each chromosome represents a pair of parameters required by DBSCAN, which refers to the Minpts and Radius. The pairs of parameters are initially randomly generated in the range of [0, 10]. The binary representation is applied in the encoding process of the DBSCAN parameters. The two parameters are converted into a binary format and then combined into a binary string. For example, given a pair of random numbers with an accuracy to three decimal places, (0.94, 0.04), the two values are firstly converted into integers by multiplying the first value to 10^2 as (94, 4). The integers are then converted into the binary format as (01011110, 00000100). Finally, a binary string, 0101111000000100, is constructed by simply combining the two values.

2 Evaluate each candidate

A fitness function is selected from the candidate fitness functions provided in Section 3.1 with respect to the real case. After the initial population is generated, the fitness of each chromosome will be evaluated by the fitness function.

3 Genetic operations in iteration

3.1 Selection

Fitness proportionate selection is applied in this method and the selection strategy is introduced as Algorithm 3.2. Repeat the selection strategy until two difference chromosomes are selected as the parents.

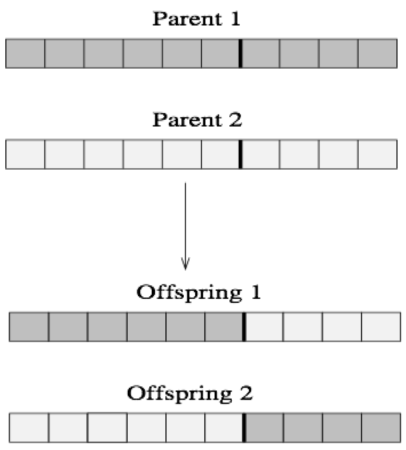
Algorithm 3.2: Fitness Proportionate Selection

```
1. Normalize the fitness of all the chromosomes in the population.
2. Sorted the chromosomes into descending fitness values as an array reorder.fitness.
3. Compute the accumulated normalized fitness values (accumulated.fitness) of each chromosome $i$ as the function below.
   $accumulated.fitness[i] = reorder.fitness[i] + Sum(reorder.fitness[1:(i-1)])$
4. Generate a random value in range of [0,1]. The chromosome for which accumulated normalized fitness values exceeds the random value is the selected chromosome.
```

3.2 Crossover

For each pair of parent chromosomes, a random value in the range of (0, 1) is generated. If the random value is less than the pre-defined crossover rate, then the crossover operation will be carried out upon to the corresponding chromosome.

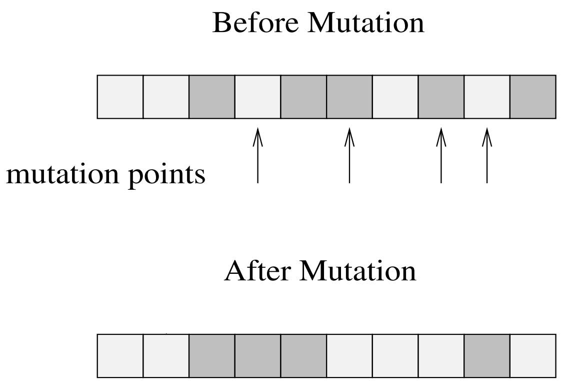
By default, the single point crossover is applied. A single crossover point on both parents is selected randomly. All the data beyond that point in either parent chromosome is swapped between the two chromosomes. The resulting chromosomes are the children.



Figures 3.4 Example of Single Point Crossover [Van den Bergh et al., 2004]

3.3 Mutation

Flip Bit is the default mutation type applied in the proposed method. For each bit of a chromosomes, a random value in the range of (0, 1) is generated. If the random value is less than the pre-defined mutation rate, the flip-over operation will be executed on this bit of the chromosome. The flip-over operation in the method means to change the original bit from 0 to 1and vice versa.



Figures 3.5 Example of Flip Bit Mutation [Van den Bergh et al., 2004]

3.4 Place new offspring generated by step 3.1-3.4 in a new population.

4 Use the newly generated population to replace the last generation of the population for a further run of algorithm

5 Check whether the terminal condition is satisfied. If satisfied, produce the best chromosomes from the population as the solution and then move to Step 6; If not satisfied, go to step 2 to start next generation.

6 Decoding the best chromosomes into a pair of parameters. Using the best pair of parameters to produce the best pattern results by DBSCAN.

3.3 Particle Swarm Optimisation optimised Density-based Clustering and Classification

The proposed Particle Swarm Optimised Density-based Clustering and Classification (PSO-DCC) method is based on the idea of applying PSO and cluster measurement indices to optimise the input parameter settings for the algorithm. The procedures of PSO-DCC are illustrated in Algorithm 3.3. The details of steps are demonstrated as follows.

```
Algorithm 3.3: PODCC
Input: A fitness function $F$, dataset $\mathcal{S}$, the swarm size $M$
and maximum iteration number $T$;
Output: Best Pattern Result $BPR$;
1. For all particles in the swarm, $\forall i \in \{1, ..., M\}$
    1.1 Initialise particles' positions $\overrightarrow{X_i}$ and velocities $\overrightarrow{V_i}$;
    1.2 Initialise personal/previous best $\overrightarrow{P_i}$ and local best $\overrightarrow{L}$;
2. For all particles in the swarm, $\forall i \in \{1, ..., M\}$
    2.1 Update particle's velocity
    2.2 Update particle's position
    2.3 Generate the Pattern Results by
        $(PR)_{\overrightarrow{X_i}} = DBSCAN(\mathcal{S}, x_{i1}, x_{i2})$;
        $(PR)_{\overrightarrow{P_i}} = DBSCAN(\mathcal{S}, p_{i1}, p_{i2})$;
        $(PR)_{\overrightarrow{L_i}} = DBSCAN(\mathcal{S}, l_1, l_2)$;
    2.4 If $F((PR)_{\overrightarrow{X_i}}) < F((PR)_{\overrightarrow{P_i}})$, then
        Update particle's best-known position $\overrightarrow{P_i} = \overrightarrow{X_i}$;
    2.5 If $F((PR)_{\overrightarrow{P_i}}) < F((PR)_{\overrightarrow{L}})$, then
        Update the neighbourhood's best-known position $\overrightarrow{L} = \overrightarrow{P_i}$;
3. Repeat step 2 until maximum iteration number $T$ or the other stop condition is met;
4. Generate the best Pattern Result,
    i.e. $BPR = (PR)_{\overrightarrow{L}} = DBSCAN(\mathcal{S}, l_1, l_2)$.
```

A parameter value pair is a particle, which means a possible solution. A group of particles are generated in a 2-dimension search space. After the positions ($X_i = (x_{i1}, x_{i2})$) and velocities ($V_i = (v_{i1}, v_{i2})$) of particles, the previous best particles ($P_i = (p_{i1}, p_{i2})$) and the local best particle ($L = (l_1, l_2)$) are initialized, a loop is executed to find the best particle of the highest fitness value. The flow chart features a loop, which starts by updating X_i and V_i . The updated particles are passed to the DBSCAN function to produce a cluster pattern results (PR). In this thesis, a pattern result either means the clustering result or the classification result. The fitness values of the *PR*s are computed by chosen fitness functions. Two types of fitness function, unsupervised and supervised, are considered in this thesis (the nature of these functions are considered further in Section 4). The best particles are updated with respect to the fitness values. To ensure that the process terminates, a maximum iteration (T) is specified, consequently, the loop will be terminated when T is reached. Finally, L is returned and used in DBSCAN to produce the Best Pattern Results (BPR).

In this work both the Canonical PSO and the Standard Particle Swarm Optimisation algorithm, defined in 2011 (SPSO-2011) [Zambrano-Bigiarini et al. 2013], are applied in PSO-DCC. SPSO-2011 was used because the adaptive random topology and rotational invariance featured in SPSO-2011 has been shown to achieve faster convergence to the global optimum than pervious PSO variants. The algorithm of PSO-DCC shown in Algorithm 3.3 are represented as below in detail.

In Step 1, the particles are initialized using the following equations.

$$x_{i,d} = U(min_d, max_d) \tag{3.11}$$

$$v_{i,d} = \frac{U(min_d, max_d) - x_{i,d}^0}{2} \tag{3.12}$$

$$p_{i,d} = x_{i,d}^0 \tag{3.13}$$

$$l_{i,d} = \min\left(f(p_{i,d}^0)\right) \tag{3.14}$$

where $U(min_d, max_d)$ is a random value in $[min_d, max_d]$ where the subscript $d \in \{1,2\}$ denotes the dimension of the particle.

In Step 2.1, the velocity is updated using the following function. For Canonical PSO, the equation 3.15 is applied, whilst for SPSO-2011, the equations 3.16-19 are used.

$$\overrightarrow{V_i} = \omega\overrightarrow{V_i} + x' - \overrightarrow{X_i} \tag{3.15}$$

$$\overrightarrow{V_i} = \overrightarrow{V_i} + c_1\overrightarrow{U_1} \otimes (\overrightarrow{P_i} - \overrightarrow{X_i}) + c_2\overrightarrow{U_2} \otimes (\overrightarrow{L} - \overrightarrow{X_i}) \tag{3.16}$$

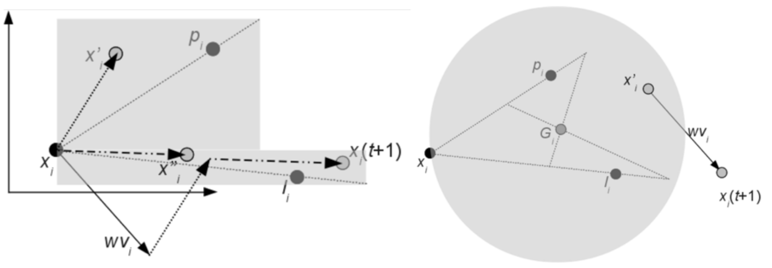
where x is a random point defined in the hypersphere: $H_i(G_i, \|G_i - X_i\|)$. For the i^{th} particle, the centre of gravity (G_i) is calculated by three points: the current position (X_i), a point slightly beyond the best previous personal position (p_i), and a point slightly beyond the best previous position in the neighbourhood (l_i), as shown below.

$$\overrightarrow{p_i} = \overrightarrow{X_i} + c_1\overrightarrow{U_1} \otimes (\overrightarrow{P_i} - \overrightarrow{X_i}) \tag{3.17}$$

$$\overrightarrow{l_i} = \overrightarrow{X_i} + c_2\overrightarrow{U_2} \otimes (\overrightarrow{L} - \overrightarrow{X_i}) \tag{3.18}$$

$$\overrightarrow{G_i} = \frac{\overrightarrow{X_i} + \overrightarrow{p_i} + \overrightarrow{l_i}}{3} \tag{3.19}$$

where c_1 and c_2 are the cognitive and social acceleration coefficients respectively. U_1 and U_2 are the predefined independent and uniformly distributed random vectors respectively within the range [0, 1]. $\otimes$ denotes the element-wise vector multiplication. ω is a predefined inertia weight. The differences between the two variants of PSO is shown in Figure 3.6 as below.



(a) Canonical PSO (b) SPSO-2011

Figure 3.6 The Geometrical Interpretation of PSOs [Zambrano-Bigiarini et al., 2013]

In Step 2.2, the position of the i^{th} particle is updated according to the equation:

$$\overrightarrow{X_i} = \overrightarrow{X_i} + \overrightarrow{V_i} \tag{3.20}$$

In Step 2.3, by using the particles x_{i1} and x_{i2} as ε and *MinPts* respectively, the items in the dataset *S* can be clustered or classified by DBSCAN.

In Steps 2.4 and 2.5, the fitness value of the pattern results can be computed using the different fitness functions as defined in Section 3.1. After the stop condition is met, the best pair of parameters, l_1 and l_2 can be found as the output in Step 3. Finally, the optimised pattern results are returned by passing the parameters, l_1 and l_2, as ε and *MinPts* to DBSCAN.

3.4 Differential Evolution optimised Density-based Clustering and Classification

The proposed Differential Evolution optimised Density-based Clustering and Classification (DE-DCC) method is founded on the idea of applying DE and cluster measurement indices to optimise the input parameter settings for the algorithm. The procedures of DE-DCC are illustrated in Algorithm 3.4.

Algorithm 3.4: DE-DCC.

```
1   Initialize all agents with random positions in the predefined search space.
2   Repeat until a termination criterion is met. For each agent x,
    2.1  Randomly pick three different agents $a$, $b$ and $c$ which are also distinct
         from agent $x$. Randomly set $R$ as 1 or 2. Set crossover probability ($CR$) as
         a random number in range of [0,1].
    2.2  Compute the potentially new position $y=[y_1, y_2]$ as follows:
         For each dimension, pick a uniformly distributed number $r_i$ in the range of
         (0, 1)
         If $r_i<CR$ or $i=R$ then set $y=a_i+F*(b_i-c_i)$, otherwise set $y_i=x_i$
    2.3  If $f(y_i)<f(x_i)$ then replace the agent with the improved candidate solution.
3  Pick the agent from the population that has the highest fitness or lowest cost and
   return it as the best-found candidate solution.
4  Produce the optimised clustering patterns by applying the best-found candidate
   solution to DBSCAN process.
```

3.5 Artificial Bee Colony optimised Density-based Clustering and Classification

The proposed Artificial Bee Colony optimised Density-based Clustering and Classification (ABC-DCC) method is based on the idea of applying ABC and cluster measurement indices to optimise the input parameter settings for the algorithm. The procedures of ABC-DCC are illustrated in Algorithm 3.5 and the details of the steps are illustrated as follows.

Algorithm 3.5 ABC-DCC.

```
1  Send the scouts onto the initial food sources.
2  Repeat until requirements are reached.
    2.1  Send the employed bees onto the food sources and determine their nectar
         amounts
    2.2  Calculate the probability value of the sources with which they are
         preferred by the onlooker bees
    2.3  Send the onlooker bees onto the food sources and determine their nectar
         amounts
    2.4  Stop the exploitation process of the sources exhausted by the bees
    2.5  Send the scouts into the search area for discovering new food sources,
         randomly
    2.6  Memorize the best food source found so far
3  3. Produce the best clustering patterns by the best food source.
```

1 Initialization

Each food source ($food_i$) is initialized with respect to a random value generated by function *randomValue*() and the search range of the possible solutions [*min*, *max*] is as below.

$$food_i = randomValue()*\ (max\ -\ min)\ +\ min \qquad (3.21)$$

The trials index for all the food sources are initialised as 0. The global best parameter will be selected from the initial food sources, with respect to the corresponding fitness values calculated by the selected fitness function. The global best parameter is initialised as the food source which reaches the best fitness, whilst the best fitness value is regarded as the global best value.

2 Repeat until the pre-defined number of loops is reached.

2.1 For each the food sources ($food_i$), randomly select a parameter to change and a neighbor source ($food_n$), and then change the selected parameter of the solution of this food source by the function below:

$$food_{i,1}=food_{i,1}+(food_{i,1}-\ food_{n,1})*(randomValue()-0.5)*2 \qquad (3.22)$$

$$food_{i,2}=food_{i,2}+(food_{i,2}-\ food_{n,2})*(randomValue()-0.5)*2 \qquad (3.23)$$

where *randomValue()* is a function to generate random value in certain range. The newly produced value should not be out of the search space.

2.2 The fitness of all the updated food sources are calculated by the fitness function. The highest fitness value is saved as the *maxfit* for calculating the probabilities of each food source to be selected by the onlooker bees as the function below.

$$Prob_i=(fitness_i/maxfit)*0.9+0.1 \qquad (3.24)$$

2.3 For all the food sources which the probabilities are higher than a random value (which means the food sources chosen by the onlooker bees are decided with respect to the probabilities calculated in step 2.2):

Randomly select a parameter to change and a neighbour source, and then change the selected parameter of the solution on this food source by the equations 3.21-3.22. The new produced value also should not be out of the search space. The fitness of all the updated food sources are calculated by the fitness function.

2.4 The exploitation process of the sources exhausted by the bees will stop. Once a food source is searched for by the onlooker bees, the number of trials for this source is added by one. The source, which has been searched for a pre-defined number of times, will be discarded.

2.5 Generate new food sources to replace the exhausted food sources judged in Step 2.4. The new food source is generated in the way described in Step 1.

2.6 Memorize the best food source found so far.

3 Apply the best food source to DBSCAN to produce the best patterns.

Chapter 4 Experiments

4.1 Experimental Settings

- **Hardware and Software Settings**

The experimental tests are all run by a MacBook Pro with OS X EI Capitan (Version 10.11.6) system. The proposed methods are implemented by R language and run in R studio (Version 1.0.153). Some R packages are applied in the codes of proposed methods, including R.utils [R Core Team, 2016], clusterCrit [Desgraupes, 2016], and R.matlab [Bengtsson, 2016] and hydroPSO [Zambrano-Bigiarini & Rojas, 2013; 2014] (only used in PSO-DBC applied SPSO-2011). The GA-DBC, PSO-DBC applied Canonical PSO, DE-DBC and ABC-DBC are implemented in R language.

- **Datasets**

This section presents the results obtained from a sequence of experiments used to evaluate the proposed ESA-DBC framework. For the experiments 10 datasets were used. Comparisons were conducted using common clustering and classification methods. The nature of the datasets used is presented in Table 4.1. Six of the datasets featured challenging known arbitrary shaped clusters (Datasets 1-6): (i) Two spirals, (ii) Cluster in cluster, (iii) Corners, (iv) Half-kernel, (v) Crescent & Full Moon and (vi) Outlier. The remaining four datasets (Datasets 7-10) were synthetic datasets generated using software provided in [Handl &Knowles, 2005]. The ground truth partitions of Datasets 1-6 are shown in Figure 4.1, and Datasets 7-10 are shown in Figure 4.2. By observing the figures of the datasets, it can be found that the clusters in Datasets 1-6 have clear boundaries and are evenly distributed. Comparing to the first six datasets, the clusters in Datasets 7-10 have fuzzy boundaries and centroid-based shapes; and the data points are not evenly distributed.

Table 4.1: Description of Datasets

ID	Dataset Name	No. of Clusters	No. of Data Points
1	Two spirals	2	3000
2	Cluster in cluster	2	1024
3	Corners	4	1000
4	Half-kernel	2	1000
5	Crescent & Full Moon	2	1000
6	Outlier	4	600
7	2d-4c-no0	4	1572
8	2d-4c-no2	4	1064
9	2d-10c-no0	10	2972
10	2d-10c-no2	10	3073

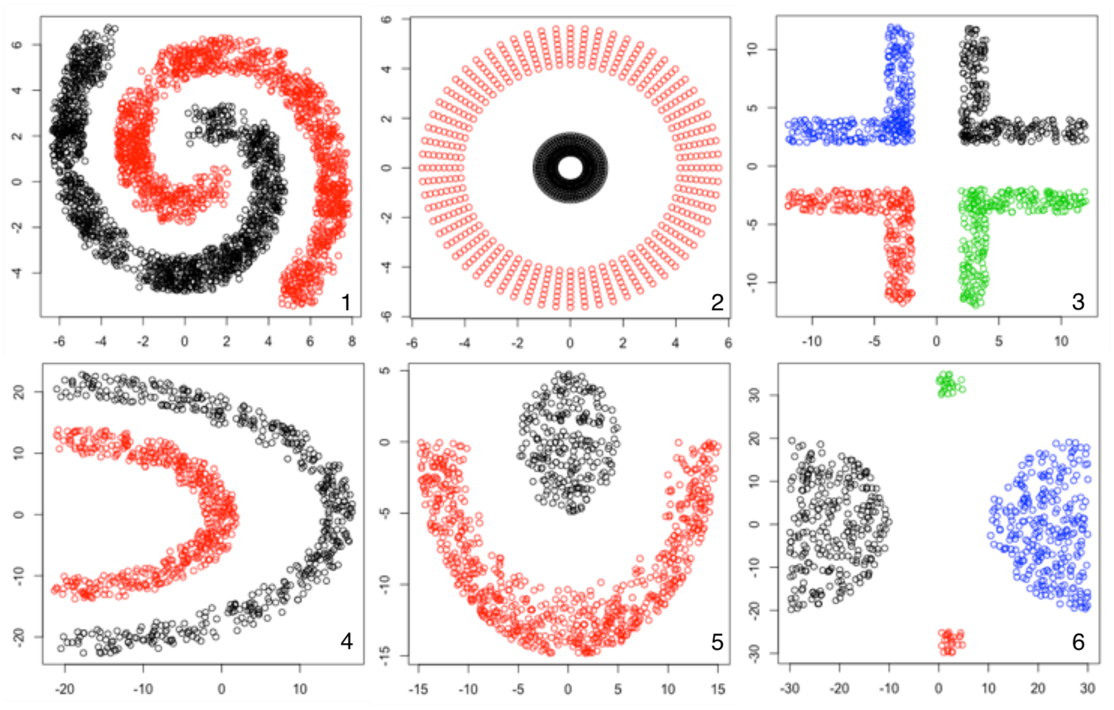


Figure 4.1 Dataset 1-6

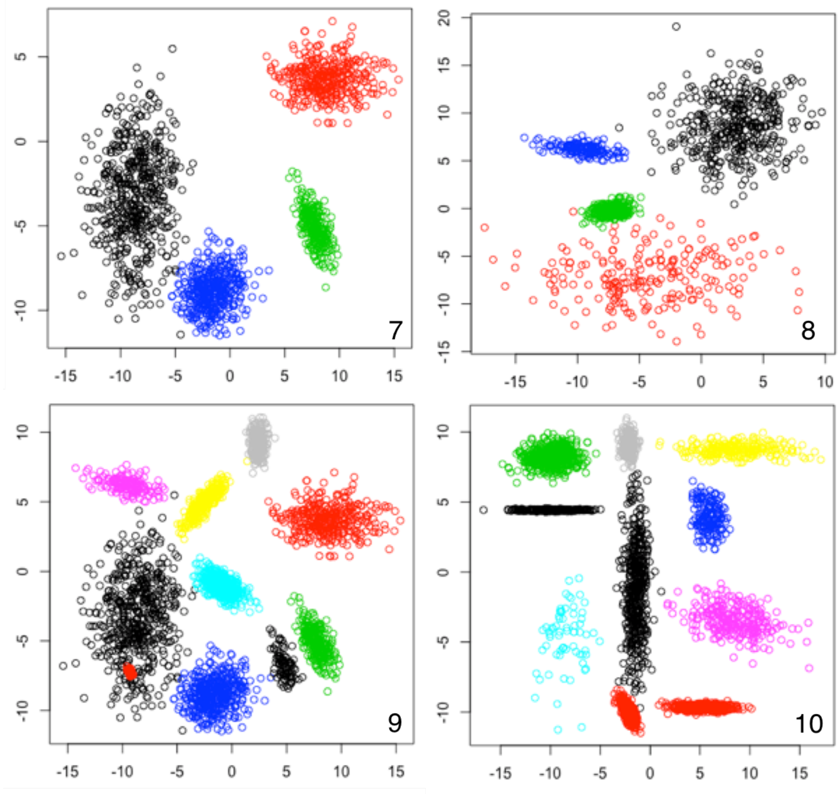


Figure 4.2 Dataset 7-10

4.2 Evaluations of PSO-DCC

Four individual sets of experiments were conducted to evaluate the performance of PSO-DCC. The first three were designed for evaluating PSO-DCC (SPSO-2011) in the context of unsupervised learning, for each experiment one of the unsupervised fitness functions presented on Section 4 was used, namely: F_{NK} (Equation 3.6), F_{usp} with Davies Bouldin Index (Equation 3.2) and F_{usp} with Silhouette Index (Equations 3.3 to 3.5). For the evaluation the performance of PSO-DCC was compared with both DBSCAN and K-means. The fourth set of experiments was designed to evaluate the performance of supervised PSO-DCC using the supervised fitness function F_{spd} with the Czekanowski-Dice Index (Equations 3.9 and 3.10). The performance of supervised PSO-DCC was compared with Support Vector Machine (SVM) classification [Cortes & Vapnik, 1995].

4.2.1 Parameter Settings

For the experiments the following PSO-DCC (SPSO-2011) settings were used: (i) swarm size of 40, (ii) maximum values of the two particles 10 and the minimum values 0, (iii) c_1, c_2 (the cognitive and social acceleration coefficients) and w (predefined inertia weight) to 1.193, 1.193 and 0.721 respectively, and (iv) the maximum number of iterations to 50. For DBSCAN, the two parameters were set to random values in the range from 0 to 10. For K-means, the values of K were set according to the given number of clusters for each dataset. For SVM, the default settings using the e1071 package [Meyer, Dimitriadou, Hornik, Weingessel & Leisch, 2014] were adopted. For each dataset, the number of times that DBSCAN and K-means were run was determined on the basis of the convergence point of PSO-DCC.

Since the dimension of search space is low (only 2-dimension), it will take a short amount of time to find an optimised pair of parameters. The number of iterations for the proposed framework is suggested to be set as a number less than 50. Given that the PSO-DCC swarm size was set to 40, forty applications of PSO-DCC would be performed on each iteration of PSO-DCC, therefore the number of times PSO-DCC and K-means was run should be the same, namely the product of 40 and the number of iterations required for PSO-DCC to reach convergence. For example, Figure 6b shows that Dataset 8 reaches convergence at the third iteration when PSO-DCC is applied; therefore DBSCAN and K-means were run 120 times so as to give a fair comparison.

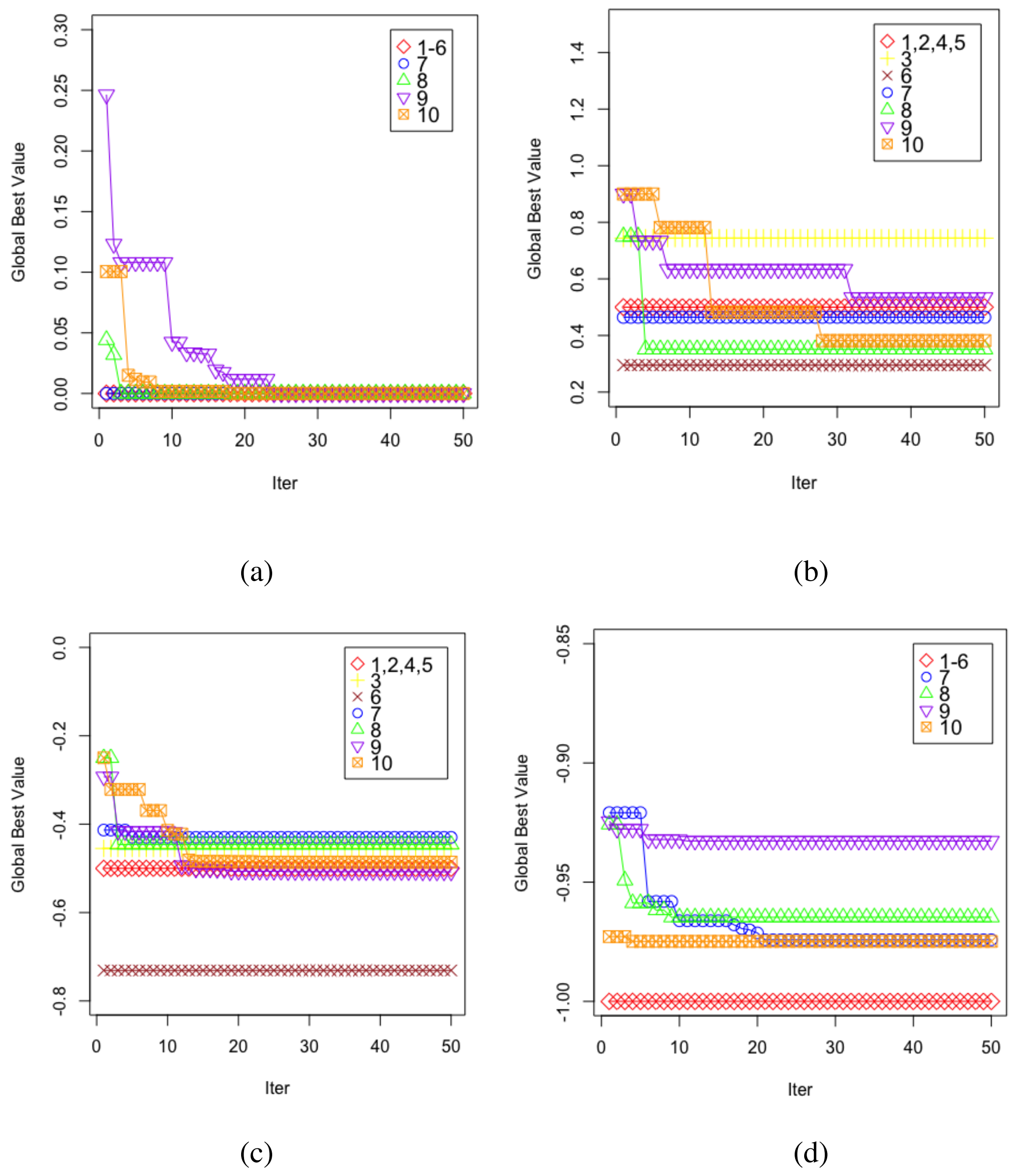


Figure 4.3 Convergence Performance of PSO-DCC (SPSO-2011)

4.2.2 Convergence

The convergence performance of PSO-DCC in terms of the number of iterations required to reach a stable point is illustrated in Figure 4.3. From the figure it can be seen that the solution of PSO-DCC is converged to a stable state after about 30 iterations. In some cases, PSO-DCC requires less than 10 iterations to reach convergence. The convergence speed of PSO-DCC should be much faster than in the case of the Genetic

Algorithm Density-Based Approach for Clustering (GADAC) [Lin et al., 2005] which converges at about 100 iterations.

4.2.3 Experimental Results and Analysis

To compare the performance of unsupervised PSO-DCC with DBSCAN and K-means, in terms of accuracy, for each dataset, the average fitness values of the selected fitness function for all the clustering results (*fit*$_{avg}$), the fitness value for the best clustering results (*fit*$_{best}$) and the Czekanowski-Dice indices for the best clustering results (CD) were used. The results are presented in Tables 4.2-4.4.

Table 4.2: Unsupervised PSO-DCC with F_{NK} versus DBSCAN and K-means

Dataset	PSO-DCC			DBSCAN			K-means		
	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD
1	0.475	0.000	1.000	1.174	0.000	1.000	0.000	0.000	0.501
2	0.400	0.000	1.000	0.475	0.000	1.000	0.000	0.000	0.645
3	0.716	0.000	1.000	1.045	0.000	1.000	0.000	0.000	0.399
4	1.716	0.000	1.000	0.839	0.000	1.000	0.000	0.000	0.500
5	1.719	0.000	1.000	15.877	0.000	1.000	0.000	0.000	0.567
6	2.019	0.000	1.000	9.392	0.000	1.000	0.000	0.000	0.874
7	4.755	0.000	0.744	0.540	0.000	0.744	0.000	0.000	0.973
8	0.840	0.000	0.715	1.714	0.002	0.836	0.000	0.000	0.929
9	3.441	0.000	0.385	2.340	0.000	0.386	0.000	0.000	0.902
10	1.112	0.000	0.597	1.519	0.002	0.782	0.000	0.000	0.937

Table 4.2 presents the results obtained using PSO-DCC applying F_{NK} as fitness function, in comparison to the operation of DBSCAN and K-means. For Datasets 1-6, the CD values show that PSO-DCC performs better than DBSCAN and K-means. According to the clustering results shown in Figure 4.4 and the convergence curve shown in Figure 4.3(a), PSO-DCC applying F_{NK} can cluster Datasets 1-6 perfectly and in a shorter time. The fitness values of all the K-means results are 0 using F_{NK}, as K is known and no noise is contained in K-means results. Even though the fitness values of K-means results are minimized, the CD values obtained by K-means show that the clustering results are not satisfactory with respect to the ground truth partitions. For Datasets 7-10, the CD results using PSO-DCC applying F_{NK} are not better than K-means, as Datasets 7-10 are obviously centroid-based clustering problems. Although the K-means approach is specifically directed at centroid-based clustering, the operation of k-means

is subject to the manner in which the initial cluster centroids are generated, and thus the generated cluster results may not be optimal. Examples of the clustering results obtained for Datasets 7-10 are given in Figure 4.6.

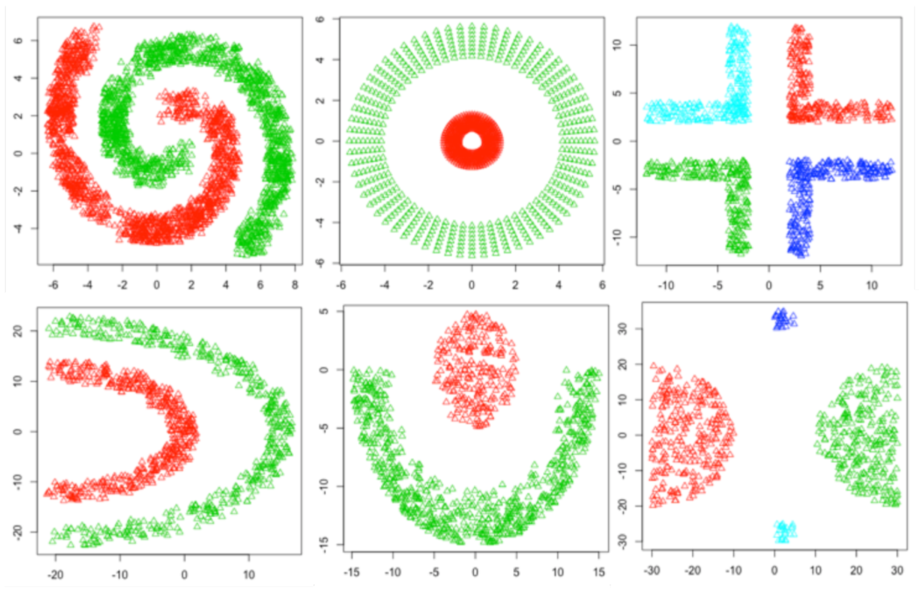


Figure 4.4 Clustering Pattern Results of Datasets 1-6 of Unsupervised PSO-DCC (F_{NK})

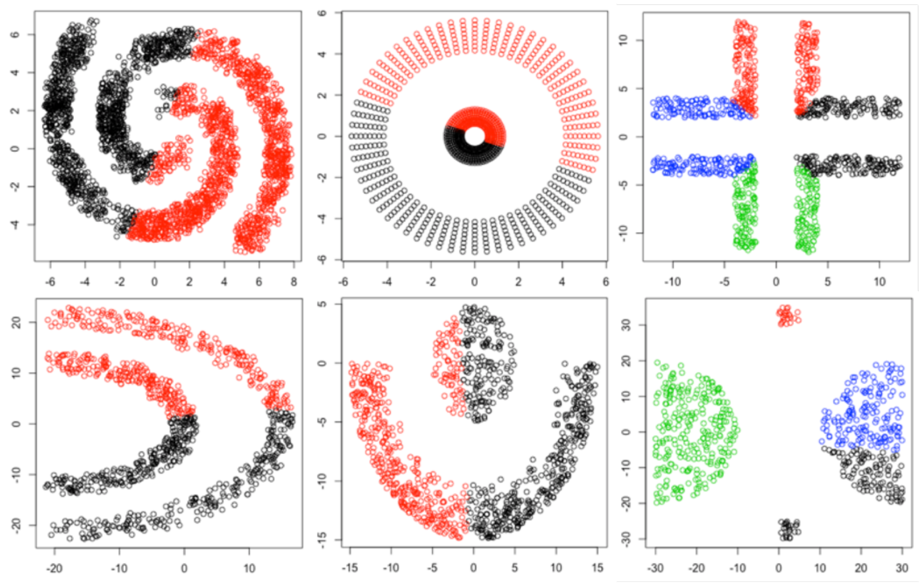


Figure 4.5 Clustering Pattern Results of Datasets 1-6 of K-means

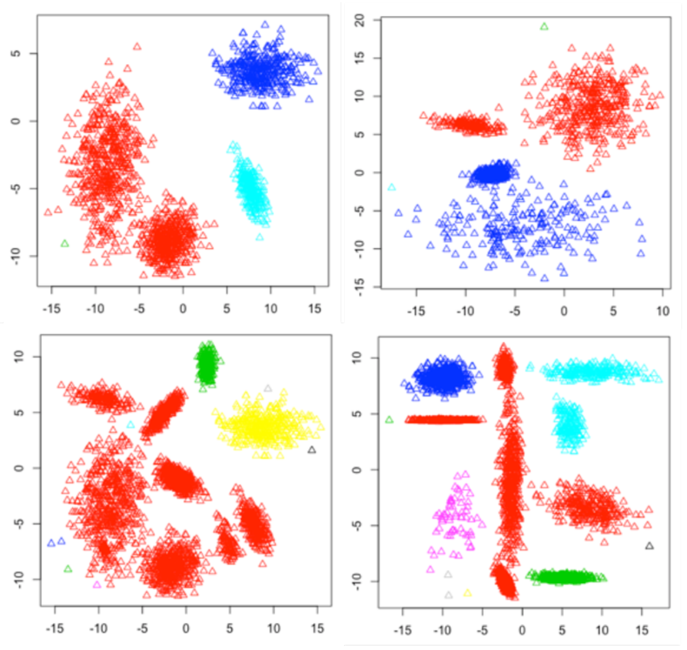


Figure 4.6 Clustering Results of Datasets 7-10 Produced by Unsupervised PSO-DCC (F_{NK})

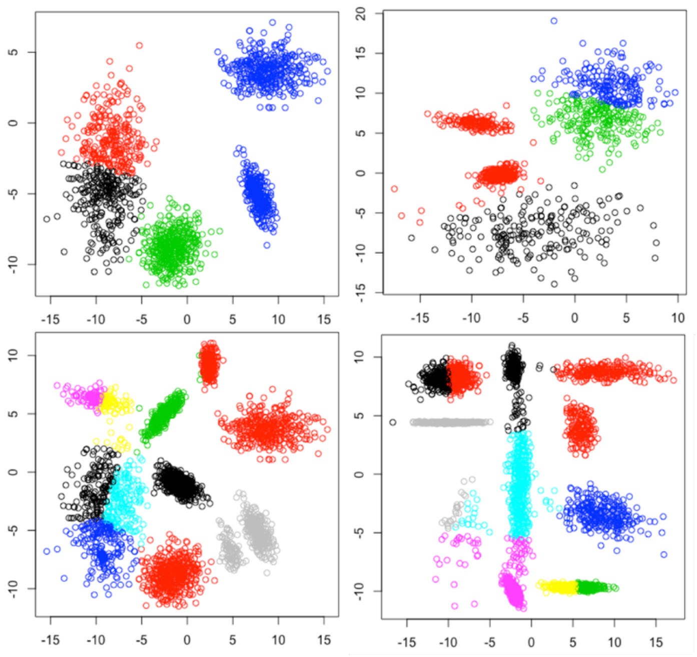


Figure 4.7 Clustering Results of Datasets 7-10 Produced by K-means

The limitation of F_{NK} is that there always clusters containing only one point in the optimized results. The reason is that such clusters can make the fitness to be minimized to zero (which is the best value) and the search process will be stop.

With respect to the shapes of clusters in datasets, different methods can be used to solve these problems. If the clusters are centroid-based, a clustering index can be added to the fitness function. In this research two fitness functions applying Davies Bouldin and Silhouette Indices are proposed and evaluated later in this section. If the clusters are in arbitrary shapes, a constraint for the minimum value of *minpts* parameters should be applied to this case such that the outlier and small clusters whose size is less than the constraint can be determined as noises. The values of the minimum minpts should be set on a case-by-case basis. The constraint(s) will be further investigated in the future research.

Table 4.3: Unsupervised PSO-DCC with Silhouette Index versus DBSCAN and K-means

Dataset	PSO-DCC			DBSCAN			K-means		
	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD
1	5.637	-0.500	0.666	-0.453	-0.500	0.666	-0.431	-0.432	0.501
2	-0.435	-0.500	0.666	-0.377	-0.500	0.666	-0.321	-0.376	0.645
3	-0.105	-0.455	1.000	-0.242	-0.455	1.000	-0.419	-0.455	1.000
4	0.724	-0.500	0.666	2.402	-0.500	0.666	-0.413	-0.413	0.500
5	-0.267	-0.500	0.769	0.606	-0.500	0.769	-0.415	-0.420	0.567
6	0.295	-0.731	1.000	0.982	-0.731	1.000	-0.523	-0.594	0.850
7	0.393	-0.430	0.962	1.465	-0.415	0.742	-0.599	-0.686	0.973
8	-0.070	-0.446	0.836	0.542	-0.446	0.836	-0.494	-0.595	0.929
9	0.127	-0.507	0.926	1.125	-0.499	0.909	-0.564	-0.621	0.896
10	0.209	-0.486	0.974	0.367	-0.480	0.971	-0.577	-0.630	0.833

Table 4.3 presents the results of PSO-DCC when a fitness function F_{usp} with the Silhouette Index is used. It can be seen that PSO-DCC performs better than DBSCAN for all the datasets considered. For Datasets 1-6, the fit_{best} values using PSO-DCC is better than those associated with K-means, but the CD values are still not satisfactory. For Datasets 7-10, although the fit_{best} values of K-means are higher than PSO-DCC, the CD values of PSO-DCC are better than the corresponding K-means values for Datasets 9-10.

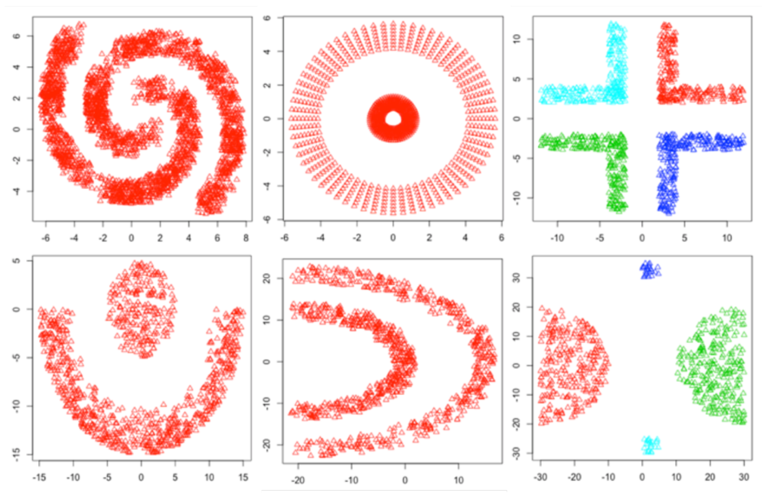


Figure 4.8 Pattern Results of Datasets 1-6 Produced by Unsupervised PSO-DCC(SIL)

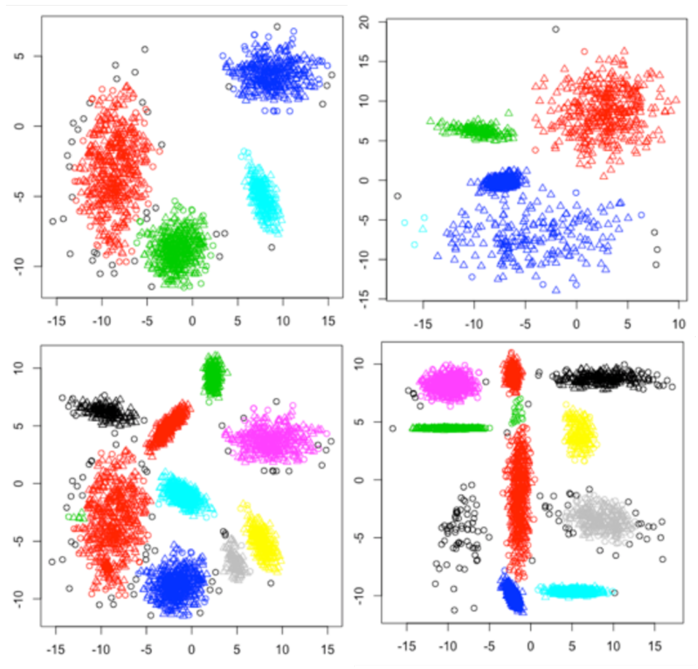


Figure 4.9 Pattern Results of Datasets 7-10 Produced by Unsupervised PSO-DCC(SIL)

Table 4.4: Unsupervised PSO-DCC with Davies Bouldin Index versus DBSCAN and K-means

Dataset	PSO-DCC			DBSCAN			K-means		
	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD	fit_{avg}	fit_{best}	CD
1	1.632	0.500	0.666	0.834	0.500	0.666	0.931	0.930	0.501
2	4e+15	0.500	0.666	3e+15	0.500	0.666	1.281	1.191	0.645
3	12.794	0.744	1.000	1.226	0.744	1.000	0.750	0.691	1.000
4	16.833	0.500	0.666	2.375	0.500	0.666	0.997	0.997	0.500
5	1.659	0.500	0.769	2.618	0.500	0.769	0.985	0.977	0.567
6	1.112	0.295	0.998	0.659	0.359	1.000	0.652	0.281	0.863
7	1.348	0.465	0.744	5.401	0.714	0.744	0.519	0.310	0.973
8	3.917	0.352	0.715	2.342	0.592	0.836	0.668	0.401	0.929
9	3.312	0.534	0.385	2.046	0.534	0.386	0.512	0.231	0.849
10	1.466	0.382	0.597	1.642	0.482	0.597	0.498	0.258	0.787

Table 4.4 presents the results of PSO-DCC applying fitness function F_{usp} with the Davies Bouldin clustering index. By reading the *CD* values in the table it can be seen that the results of PSO-DCC are more accurate than results of DBSCAN for most of datasets considered. The results demonstrate that PSO-DCC can still optimize the fitness values with respect to the chosen fitness function; however, the presented CD values associated with PSO-DCC are not satisfactory for most of datasets. Figures 4.10 and 4.11 shows the clusters resulting from PSO-DCC with Davies Bouldin as the fitness function. Inspection of the figures indicates that the results could be much improved.

Comparing the results shown in Table 4.3 and 4.4, using the Silhouette index as the basis for the fitness function achieves a better performance than using the Davies Bouldin Index. The clustering results presented in Figure 4.9 shows that PSO-DCC using the Silhouette index based fitness function can address satisfactorily centroid-based clustering problems. The penalty function guarantees that the number of clusters is close to the real K value and the amount of noises is minimized. The Silhouette index can make sure the clusters are good enough and are centroid-based. The Silhouette value is in the range of [0, 1] which is the same as the penalty function values, whilst the range of DB values could be much wider than the range of penalty functions values. In consequence, for Silhouette index basis fitness function, both the index and penalty function can contribute to the optimization process; and the DB index basis fitness function sometimes can only optimise the DB value of clustering results, the effect of penalty function is reduced. As a result, the fitness function applying the Silhouette index achieves a better performance than the fitness function applying the DB index. It can thus be concluded that the DB index is not appropriate in the case of unsupervised PSO-DCC.

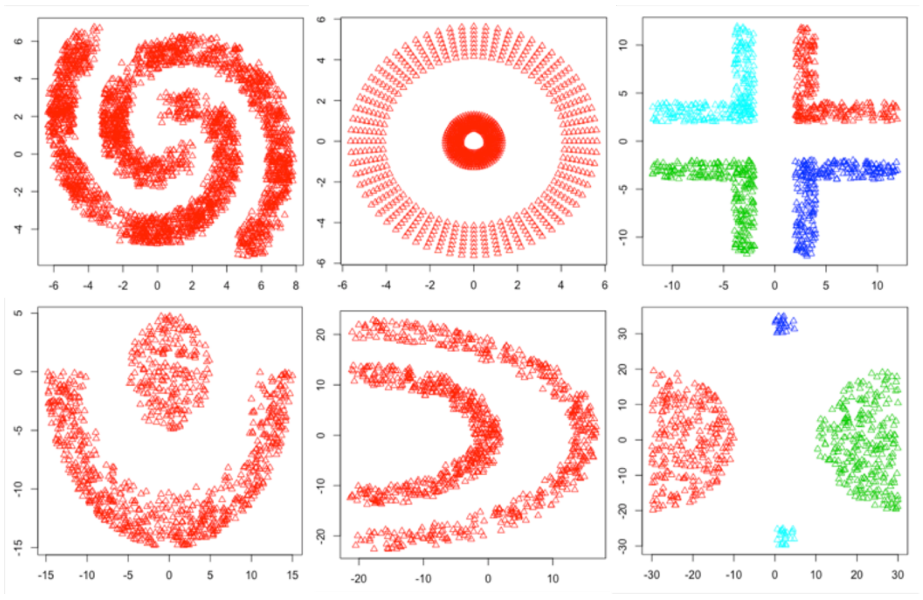


Figure 4.10 Clustering Results of Datasets 1-6 Produced by Unsupervised PSO-DCC(DB)

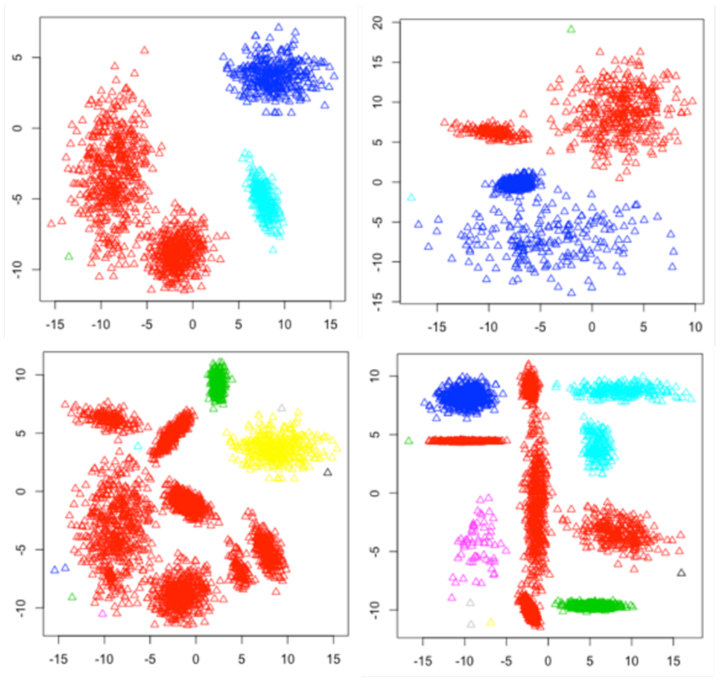


Figure 4.11 Clustering Results of Datasets 7-10 Produced by Unsupervised PSO-DCC (DB)

Table 4.5: Supervised PSO-DCC with Czekanowski-Dice Index versus SVM

Dataset	PSO-DCC			SVM
	CD_{avg}	BP	CD_{best}	CD
1	0.694	(0.576, 5)	1.000	0.980
2	0.733	(1.456, 9)	1.000	1.000
3	0.522	(1.566, 6)	1.000	0.976
4	0.752	(2.180, 4)	1.000	1.000
5	0.846	(3.929, 8)	1.000	1.000
6	0.844	(8.098, 1)	1.000	1.000
7	0.678	(1.189, 10)	0.974	0.922
8	0.659	(1.480, 5)	0.965	0.969
9	0.457	(0.913, 6)	0.933	0.677
10	0.615	(0.628, 4)	0.975	0.542

As supervised PSO-DCC can be used for classification purposes, to evaluate this its operation was compared to SVM in terms of: (i) the average Czekanowski-Dice Index of all the predicted pattern results (CD_{avg}) in PSO-DCC, (ii) the Best Parameters (BP), (iii) the corresponding CD index values of the best predicted patterns (CD_{best}) of PSO-DCC, and (iv) the CD index values of the SVM prediction results (CD). The results are presented in Table 4.5. From the Table it can be seen that PSO-DCC performs better than SVM for all datasets except for Dataset 8. However, for Dataset 8 the CD value produced by PSO-DCC result is very close to the SVM result. By comparing the pattern results of Dataset 8 given in Figures 4.14 and 4.15, it can be seen that the boundaries of the classes produced by PSO-DCC are clearer than in the case of the SVM results, as PSO-DCC inherits the character of DBCSAN which can find the presence of noises in the datasets. Such character could be either advantage or limitation subject to different situations.

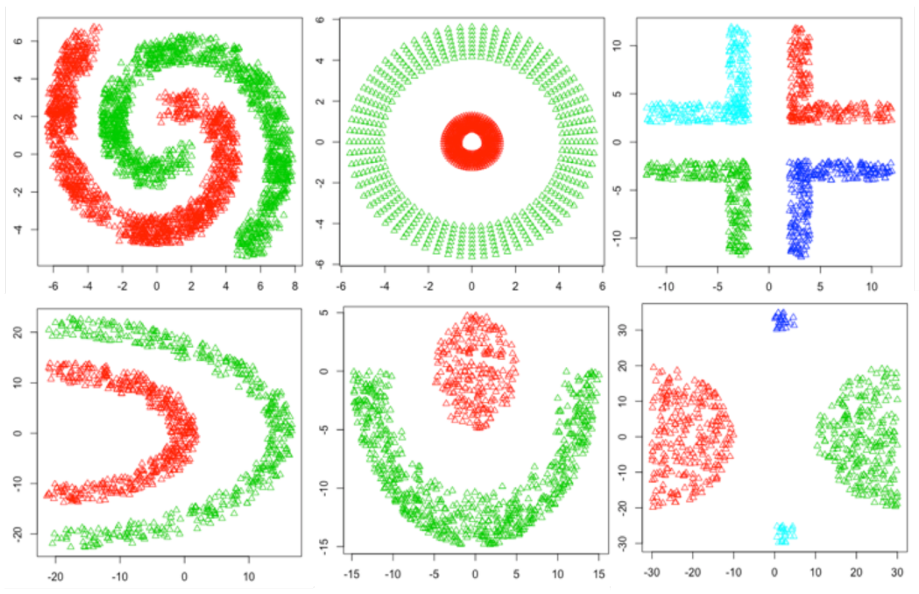


Figure 4.12 Prediction Results of Datasets 1-6 Produced by Supervised PSO-DCC

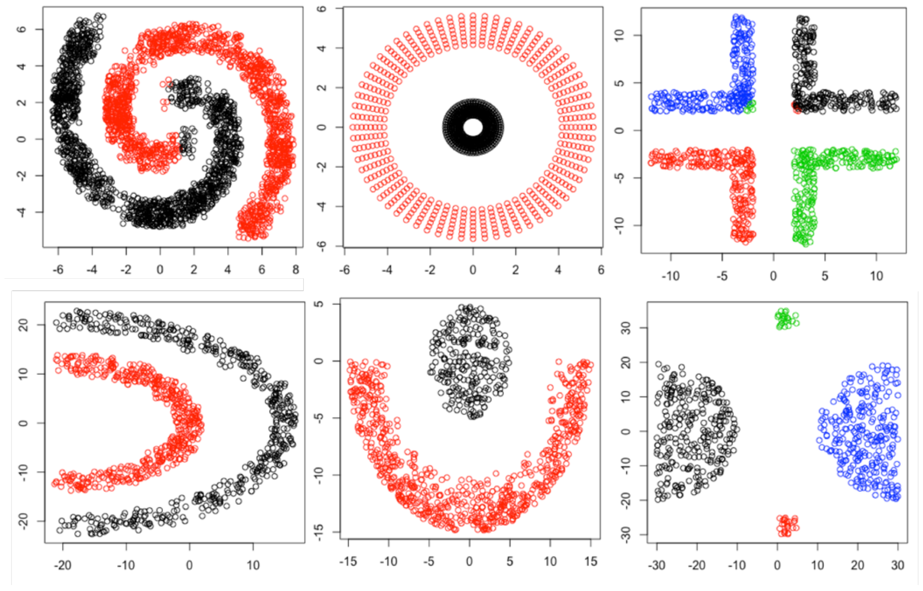


Figure 4.13 Prediction Results of Datasets 1-6 Produced by SVM

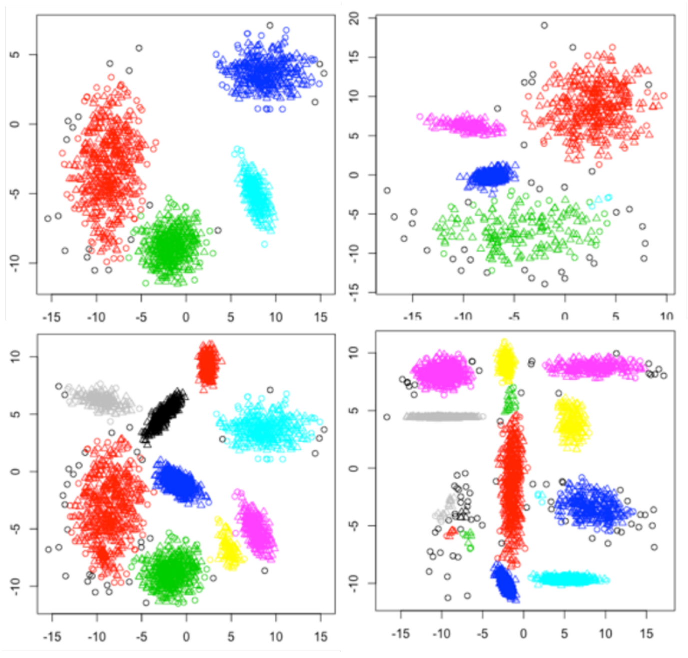


Figure 4.14 Prediction Results of Datasets 7-10 Produced by Supervised PSO-DCC

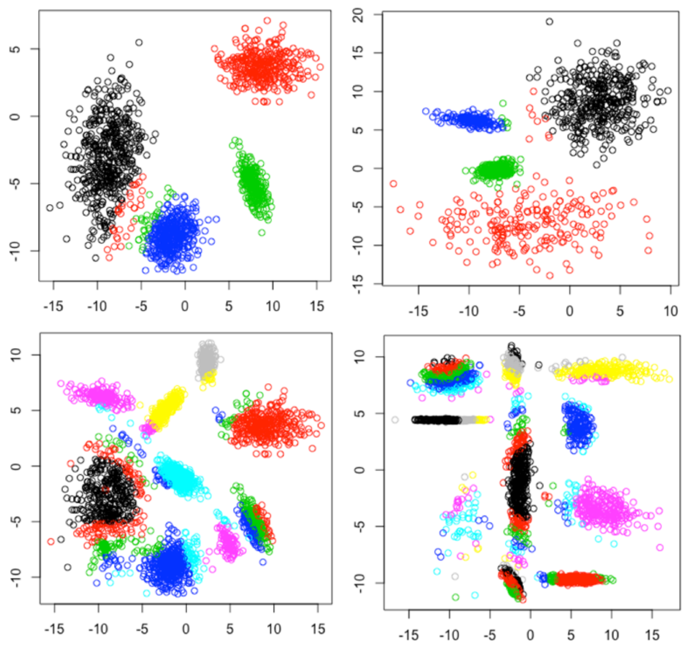
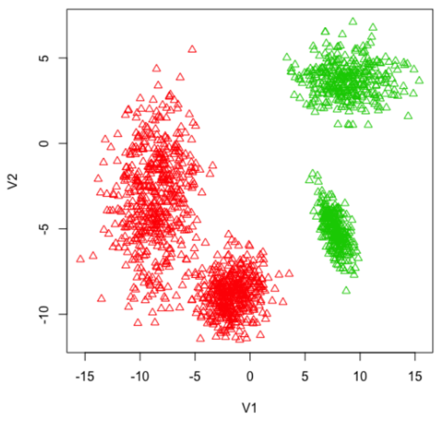
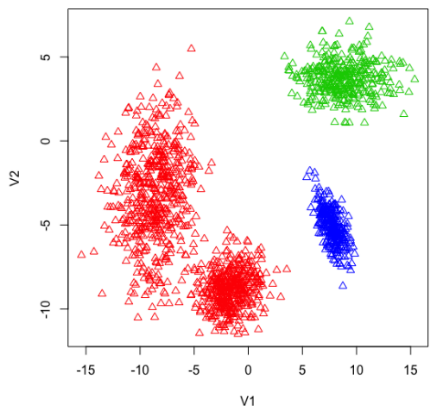


Figure 4.15 Prediction Results of Datasets 7-10 Produced by SVM

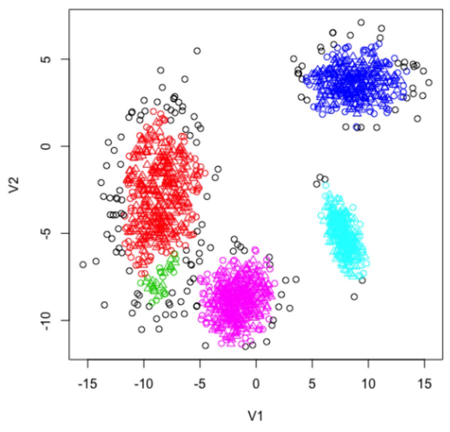
For the datasets which are not well understood, the proposed ESA-DCC method can be applied to search for possible clustering patterns by testing with different number of clusters. Figures 4.17-4.18 demonstrate that different user-defined K values for PSO-DCC can lead to different pattern results. Once K is smaller than the ground truth number of clusters, 4 in the examples, some small clusters are merged to give bigger clusters, for example, Figures 4.17(a), 4.17(b), 4.18(a), and 4.18(b). Similarly, when K is larger than than the ground truth number of clusters, bigger clusters are divided into some smaller clusters, for example, Figures 4.17(c), 4.17(d), 4.18(c), and 4.18(d).



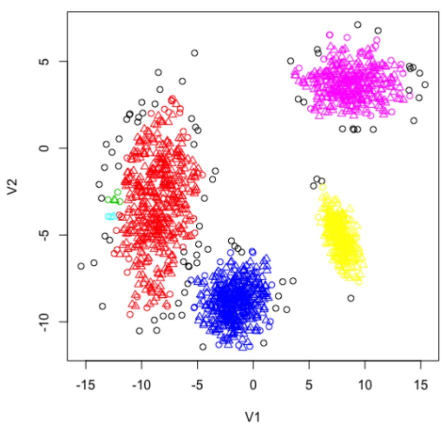
(a)



(b)

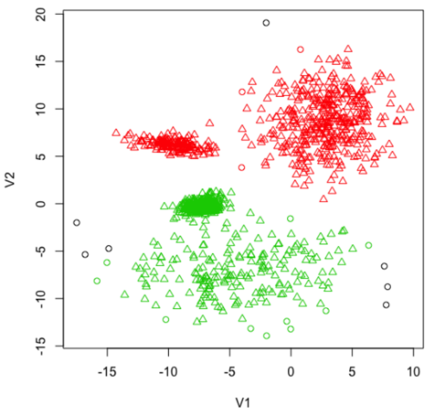


(c)

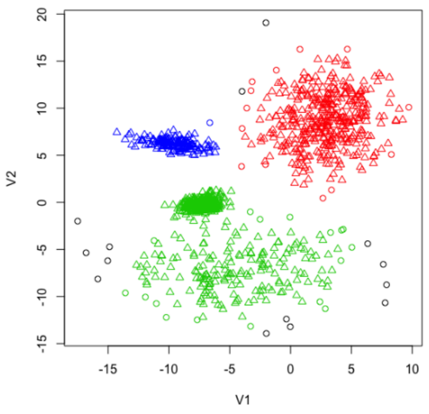


(d)

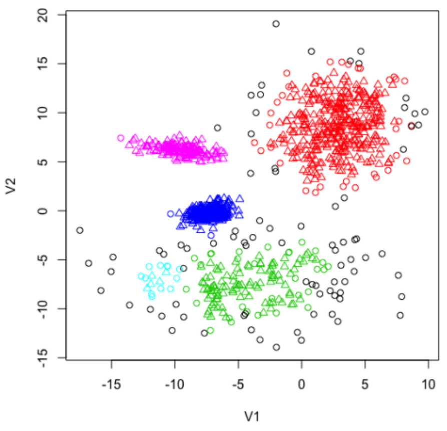
Figure 4.16 PSO-DCC Results of Dataset 7 with Different Numbers of Clusters



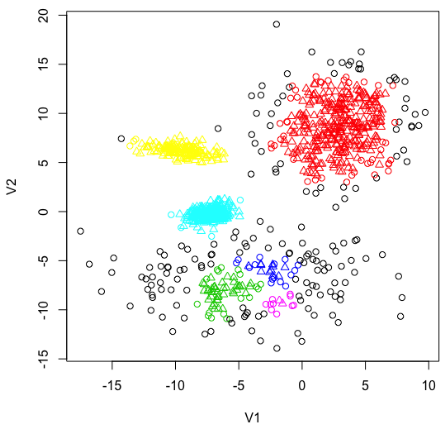
(a)



(b)



(b)



(d)

Figure 4.17 PSO-DCC Results of Dataset 8 with Different Numbers of Clusters

To summarize, PSO-DCC, when using the fitness function F_{NK},can be applied to datasets featuring arbitrary shaped clusters; Classical DBSCAN requires a manual generate-and-test process to find the appropriate parameters to produce the correct results and K-means cannot find clusters of arbitrary shape at all. PSO-DCC ,when using $Fusp$ with the Silhouette Index, can cluster centroid-based datasets; and supervised PSO-DCC can deal with all types of labelled datasets. Whilst suitable fitness function is chosen with respect to dataset, PSO-DCC performs better than DBSCAN and K-means for unsupervised learning and better than SVM for supervised learning.

4.3 Evaluations of Four ESA-DCC Methods

4.3.1 Parameter Settings

The common parameters are set as below for all the four methods, GA-DCC, PSO-DCC (Canonical PSO), DE-DCC, and ABC-DCC.

- Search Space: number of parameters is 2, minimal value is 0 and maximal value is 10.
- Number of iterations:10 iterations are set for all the tests.
- Stop criteria or Convergence tolerance: the algorithm will stop once the number of iterations is reached.

The detailed settings for each method is defined as Table 4.6 below.

Table 4.6: Parameter Settings for Four ESA-DCC methods

	GA-DBC	PSO-DBC	DE-DBC	ABC-DBC

Settings	Population=10
	Crossover Rate=0.8
	Mutation Rate=0.1
		Swarm size=10
		w=0.2
		c.p=2
		c.g=2
			Population =10
			Crossover Probability=0.8
			Differential.weight=1.2
				Colony size=20
				No.food=Colony.size/2
				Trials Limit=10

4.3.2 Convergence

The convergence performance of four proposed methods is summarized in Table 4.7 as below. The F_{usp} with the Davies Bouldin clustering index is short named as F_{DBNK} and the F_{usp} with the Silhouette clustering index four fitness functions are short named as F_{SINK} for following tables in this chapter.

Table 4.7 shows that the number of iterations before the optimal solution is found by each method applying different fitness functions for each dataset. For example, by reading the first line of Table 4.7, it can be found that 3 iterations are required for PSO-DCC method to find the best solution when F_{NK} is applied for Dataset 1. The sum of each fitness functions is computed to show the difference convergence performance of all the method applying difference functions. It can be found that ESA-DCC applying F_{DBNK} can converge faster than the other functions.

For each functions, different methods reach convergence in different speed. For example, GA-DCC applying F_{DBNK} can converge faster than the other tests according to the average coverage performance. The performance of different methods for the same fitness function is compared in the table. GA-DCC is the most efficient method among the four proposed methods according to the sum of all the iterations for each method required for all the tests. GA-DCC requires 89 iterations in total whilst PSO-DCC requires 121 iterations, DE-DCC requires 109 iterations and ABC-DCC requires 100 iterations. To summarize, the best one of the ESA-DCC methods can get coverage in 10 iterations.

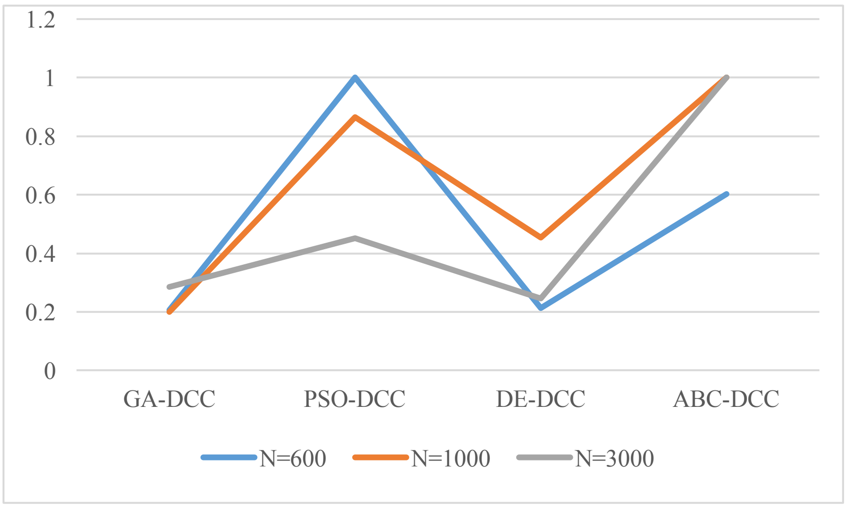


Figure 4.18 Nomarilized Proceed Time of Four ESA-DCC for 3 Datasets of Different Sizes

Table 4.7: Coverage of Four Proposed Methods

Fitness function	Dataset	GA-DBC	PSO-DBC	DE-DBC	ABC-DBC
F_{NK}	1	1	3	1	1
	2	1	1	1	1
	3	1	1	1	1
	4	1	2	1	1
	5	1	1	1	1
	6	1	1	1	1
	7	5	5	6	1
	8	6	7	10	9
	9	2	4	9	1
	10	6	9	3	9
	Avg	2.5	3.4	3.4	2.6
	Sum	119			
F_{SINK}	1	1	1	1	1
	2	1	1	1	1
	3	1	3	1	1
	4	1	1	1	1
	5	1	1	1	1
	6	1	1	1	1
	7	1	5	8	8
	8	10	1	1	1
	9	10	7	3	6
	10	1	10	7	10
	Avg	2.8	3.1	2.5	3.1
	Sum	115			
F_{DBNK}	1	1	1	1	1
	2	1	1	1	1
	3	2	1	1	7
	4	1	5	1	1
	5	1	1	1	1
	6	1	1	8	1
	7	5	3	4	4
	8	1	1	1	1
	9	1	1	4	1
	10	1	5	3	1
	Avg	1.5	2.0	2.5	1.9
	Sum	79			
F_{CD}	1	1	1	1	3
	2	1	1	1	1
	3	1	1	1	1
	4	1	1	1	1
	5	1	1	1	1
	6	1	1	1	1
	7	3	9	6	1
	8	6	9	4	8
	9	5	6	8	10
	10	1	6	1	7
	Avg	2.1	3.6	2.5	2.4
	Sum	106			

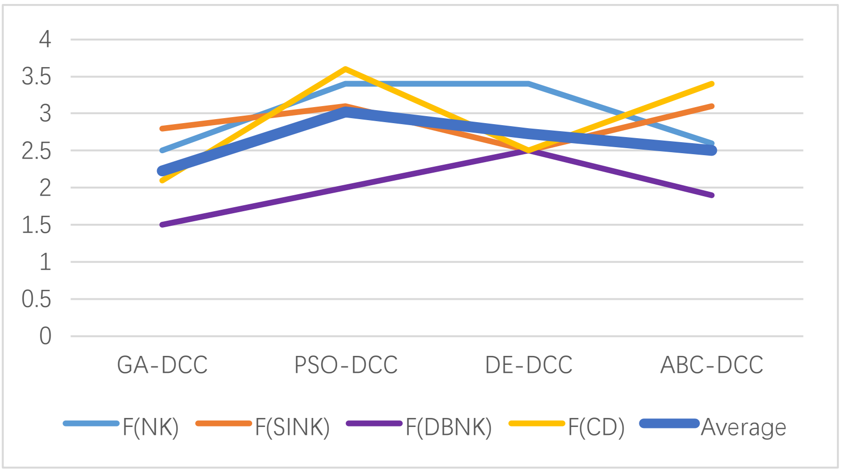


Figure 4.19 The Average Convergence Performance of Four ESA-DCC Methods

4.3.3 Experimental Results and Analysis

The accuracy of four proposed ESA-DCC methods applying different fitness functions for 10 datasets is evaluated by the best fitness value and an external index *Rand*.

Table 4.8 shows the ESA-DCC algorithms applying the function F_{NK}. The first 6 datasets can be perfectly clustered by all the four methods. The best fitness value 0 is reached for all the 10 datasets by PSO-DCC method, whilst the other three methods cannot reach the best fitness values for all datasets in 10 iterations. Overall, the accuracy of all the tests of DE-DCC method is higher than the other methods regarding the Rand values.

Table 4.8: Unsupervised ESA-DCCs with F_{NK}

Dataset	GA-DCC		PSO-DCC		DE-DCC		ABC-DCC	
	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$
1	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
2	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
3	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
4	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
5	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
6	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
7	0.000	0.819	0.000	0.819	0.000	0.819	0.000	0.818
8	0.005	0.892	0.000	0.778	0.000	0.778	0.005	0.892
9	0.023	0.980	0.000	0.607	0.040	0.976	0.000	0.606
10	0.100	0.837	0.000	0.837	0.110	0.974	0.004	0.910
Ave	0.017	0.953	0.000	0.904	0.015	0.955	0.001	0.923

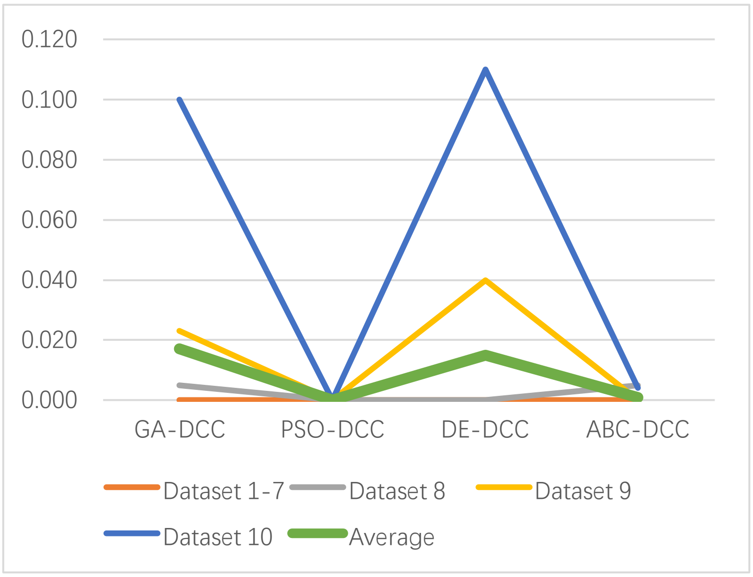


Figure 4.20 Optimized Fitness Values of 4 ESA-DCCs

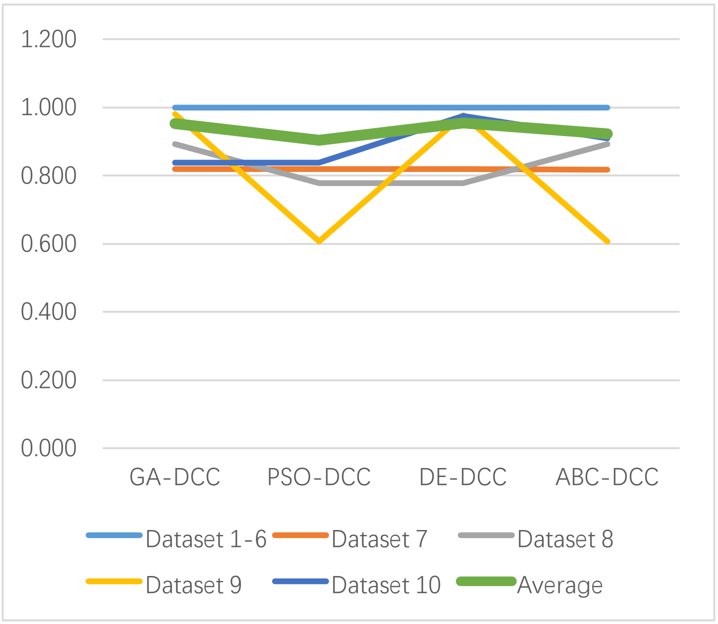


Figure 4.21 Rand Index Values of 4 ESA-DCCs (FNK) Clustering Results

The experimental results of ESA-DCC applying F_{usp} with the Silhouette clustering index (short for F_{SINK} as mentioned before) are presented in Table 4.9. The results of the ESA-DCC applying F_{usp} with the Davies Bouldin clustering index (F_{DBNK}) are shown in Table 4.10. According to the average *Rand* values for each group of experiments, it can be concluded that the accuracy of ESA-DCC methods applying F_{SINK} is higher than the methods applying F_{DBNK}. The overall performance of ESA-DCC applying F_{usp} is not satisfactory, especially for the datasets 1, 2, 4 and 5 since the internal indices are designed for centroid-based clusters, not arbitrary shaped cluster. For datasets 7-10, the accuracies of GA-DCC, DE-DCC and ABC-DCC are tolerable when F_{SINK} is applied. The accuracies of all the ESA-DCC methods applying F_{DBNK} is not acceptable. To summarize, F_{SINK} is a better fitness function than F_{DBNK} for dealing with the datasets consisting of centroid-based clusters.

Table 4.9: Unsupervised ESA-DCC with F_{SINK}

Dataset	GA-DCC		PSO-DCC		DE-DCC		ABC-DCC	
	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$
1	-0.500	0.500	-0.500	0.500	-0.500	0.499	-0.500	0.499
2	-0.500	0.500	-0.500	0.500	-0.500	0.499	-0.500	0.499
3	-0.455	1.000	-0.455	1.000	-0.455	1.000	-0.455	1.000
4	-0.500	0.499	-0.500	0.499	-0.500	0.499	-0.500	0.499
5	-0.500	0.625	-0.500	0.625	-0.500	0.625	-0.500	0.625
6	-0.731	1.000	-0.731	1.000	-0.731	1.000	-0.731	1.000
7	-0.413	0.819	-0.413	0.819	-0.427	0.982	-0.414	0.818
8	-0.471	0.892	-0.250	0.280	-0.394	0.891	-0.445	0.892
9	-0.391	0.974	-0.500	0.980	-0.414	0.980	-0.504	0.980
10	-0.375	0.909	-0.402	0.992	-0.195	0.910	-0.439	0.949
Avg	-0.484	0.772	-0.471	0.720	-0.462	0.789	-0.499	0.776

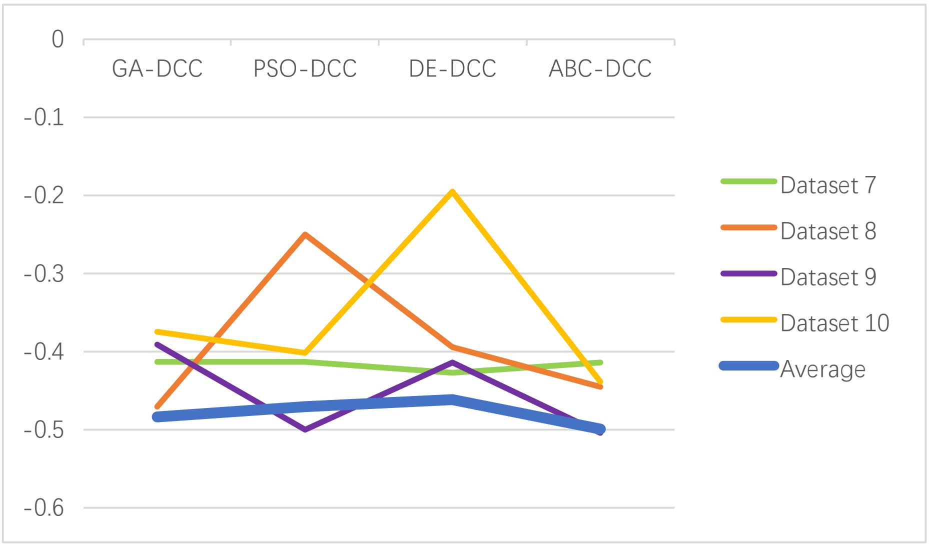


Figure 4.22 Fitness Values of ESA-DCC (FSINK) for Datasets 7-10 and Average

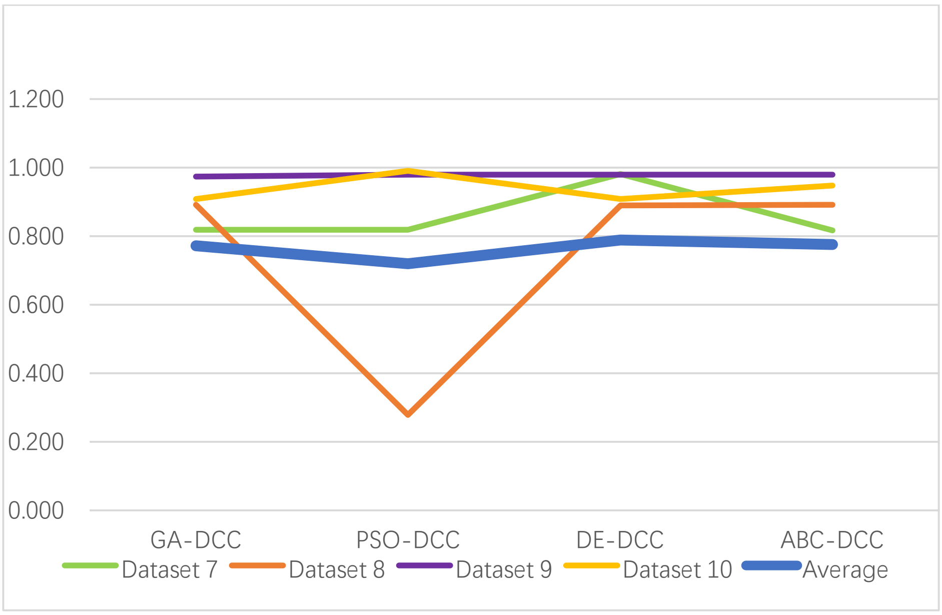


Figure 4.23 Rand Values of ESA-DCC (FSINK) for Datasets 7-10 and Average

Table 4.10: Unsupervised ESA-DBC with F_{DBNK}

Dataset	GA-DCC		PSO-DCC		DE-DCC		ABC-DCC	
	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$
1	0.500	0.500	0.500	0.500	0.500	0.499	0.500	0.499
2	0.500	0.500	0.500	0.500	0.500	0.499	0.500	0.499
3	0.658	1.000	0.744	1.000	0.744	1.000	0.652	0.999
4	0.500	0.499	2.000	0.499	0.500	0.499	0.500	0.499
5	0.500	0.625	0.500	0.625	0.500	0.625	0.500	0.625
6	0.295	1.000	0.359	1.000	0.295	0.998	0.295	0.998
7	0.465	0.819	0.465	0.819	0.465	0.819	0.465	0.819
8	0.750	0.280	0.750	0.280	0.750	0.280	0.750	0.280
9	0.900	0.124	0.900	0.124	0.824	0.904	0.900	0.124
10	0.900	0.121	0.382	0.837	0.482	0.837	0.900	0.121
Avg	0.597	0.547	0.710	0.618	0.556	0.696	0.596	0.546

The experimental results of supervised ESA-DCC methods are presented in Table 4.11 below. The accuracies of all the methods for all the datasets are satisfactory. The differences between the accuracies of the four methods are small. It can be concluded that all the supervised ESA-DCC methods implemented by any ESA methods can be used in classification problems.

Table 4.11: Supervised ESA-DBC with F_{CD}

Dataset	GA-DCC		PSO-DCC		DE-DCC		ABC-DCC	
	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$	fit_{best}	$Rand$
1	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
2	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
3	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
4	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
5	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
6	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000
7	-0.960	0.980	-0.971	0.985	-0.960	0.980	-0.960	0.979
8	-0.954	0.975	-0.959	0.978	-0.960	0.978	-0.963	0.980
9	-0.932	0.983	-0.932	0.983	-0.909	0.976	-0.932	0.983
10	-0.972	0.993	-0.971	0.993	-0.972	0.993	-0.974	0.994
Avg	-0.982	0.993	-0.983	0.994	-0.980	0.993	-0.983	0.994

The surfaces of two examples of fitness function are shown in Figures 4.24-4.25. Both of the fitness functions are non-linear. For the datasets with un-even density, such as the Dataset 8 shown in Figure 4.25, the surface of fitness functions could be very rugged and ruleless and the range of optimal values could be very small and hard to find the best value. GA and DE has the mutation feature which makes it perform better to find a good solution from the uneven surface. The movement of the particles in the PSO and bees in ABC follows some certain rules. The advantages of PSO and ABC are hard to be shown in proposed methods.

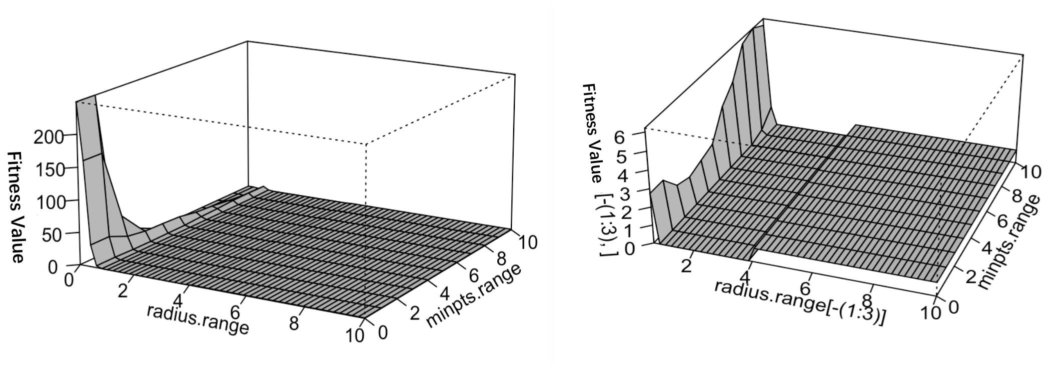


Figure 4.24. Surface of Fitness Function FNK for Dataset 3 in Range of [0,10] for Minpts and Radius (Left) and in Range of [0,10] for Minpts, [0.4,10] for Radius (Right)

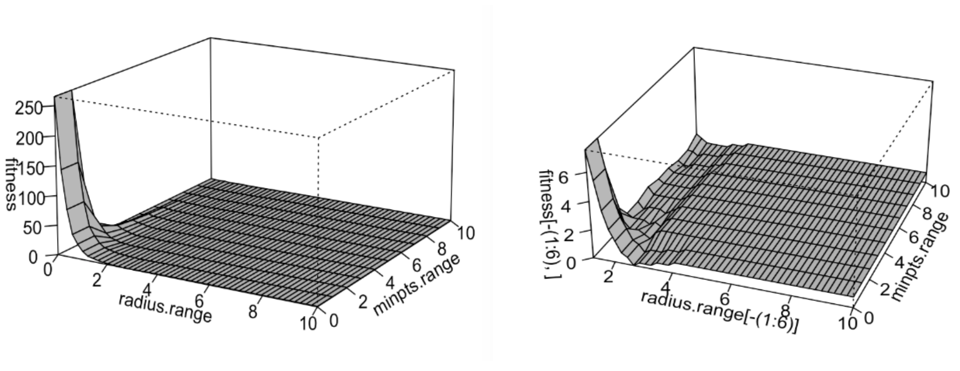


Figure 4.25 Surface of Fitness Function F_{NK} for Dataset 8 in Range of [0,10] for Minpts and Radius (Left) and in Range of [0,10] for Minpts, [0.7,10] for Radius (Right)

Chapter 5 Applications

5.1 Product Recommender System

E-commerce reaches the daily lives of people. Consumers face overloaded marketing information to select appropriate products. Recommender Systems (RSs) are able to retrieve suitable products for consumers with respect to their preferences. A highly personalized recommender service could improve consumer loyalty to increase sales [Schafer, Konstan & Riedl, 2001].

Recommender systems can be classified into various categories by using different criteria [Peis, Castillo & Delgado-López, 2008]. For example, according to the filtering techniques in RSs, four categories can be classified as content-based RSs, collaborative filtering RSs, economic factor-based RSs, and hybrid RSs combining content-based and collaborative filtering techniques [Peis et al., 2008]. As the aforementioned RSs rely on user profiles consisting of their historical ratings to various products [Lika, Kolomvatsos & Hadjiefthymiades, 2014], the major limitation of the established RSs is the cold start problem, which means lack of information for RSs to learn the profiles of new users [Burke, 2002; Balabanović & Shoham, 1997; Lika et al., 2014]. A review of approaches solving the cold start problem is presented in [Lika et al., 2014].

Since the new launches of consumer electronics, such as laptops, smartphones, digital cameras, audio equipment and video game consoles, rapidly replace the old models, the product recommendations derived by exploring historical user profiles for the old model may not be suitable for the latest launch of consumer products. This thesis proposes a RS model which does not rely on users' rating history but an individual users' rating preferences elicited from pairwise ratings to make recommendations.

5.1.1 Data Preprocessing Tools

The process of evaluating user preferences to products could be complicated as some products have many attributes. Multi-Criteria Decision Making (MCDM) tools, which can evaluate user preferences with respect to multiple attributes for products, have been employed in RSs for making recommendations [Jannach, Karakaya & Gedikli, 2012; Adomavicius, Manouselis & Kwon, 2011; Adomavicius & YoungOk, 2007; Lakiotaki, Matsatsinis & Tsoukia, 2011; Porcel & Herrera-Viedma, 2010]. In this section, Cognitive Pairwise Rating (CPR), an MCDM tool, is introduced to evaluate user preferences. CPR applies pairwise comparisons instead of direct rating in classical rating methods.

In this chapter, a novel RS model is proposed. This RS relies on user preferences of product attributes, but is not related to user rating history. Since the user preferences are not learnt from historical data, the proposed RS model does not include the cold start problem. User preferences could be elicited by Multi-Criteria Decision Making (MCDM) tools to provide personalized recommender services. In pervious works, Analytic Hierarchy Process (AHP) has been applied to evaluate user preferences with respect to multiple attributes for products [Byun, 2001; Liu & Shih, 2005; Lee & Kozar, 2006; Wang & Tseng, 2013: Liu & Shih, 2005]. In this section, an ideal alternative of AHP, Cognitive Pairwise Rating (CPR), is employed in the proposed RS model to evaluate user preferences. CPR is based on Cognitive Network Process (CNP) [Yuen, 2009; 2012; 2014(1); 2014(2)], which is an approach rectifying the mathematical representation problem of the perception of the paired differences in AHP.

5.1.2 Cases studies

- **Product Recommendation System**

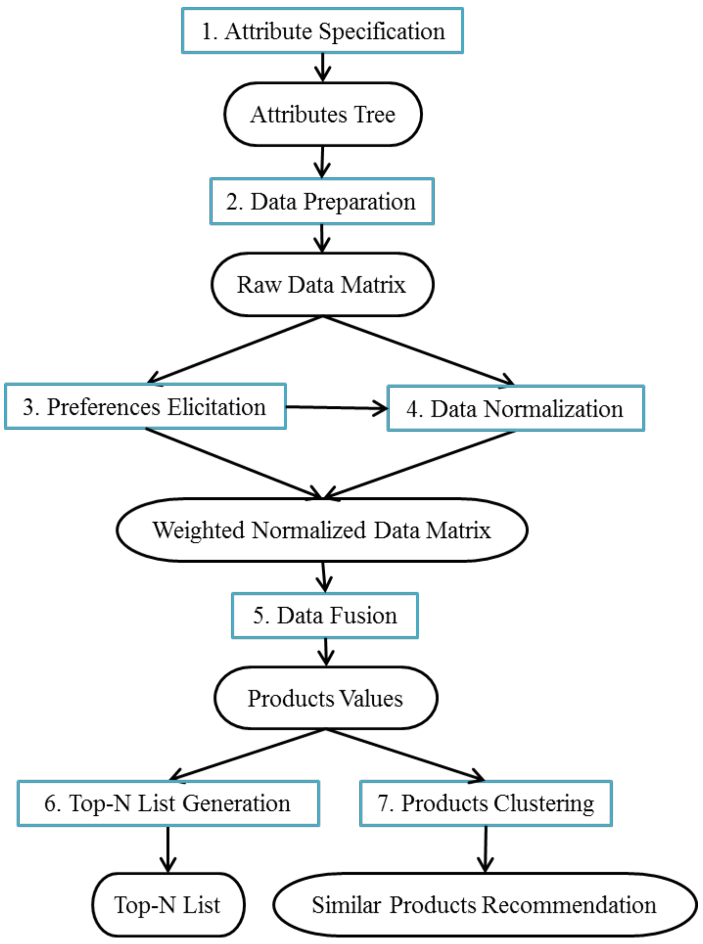


Figure 5.1 CPR-AHC Framework

The steps of the proposed Cognitive Pairwise Rating Agglomerative Hierarchical Clustering (CPR-AHC) approach are illustrated in Figure 5.1. In the first four steps, the product attributes are organized as a data schema called attribute tree. With the schema, the raw data matrix is obtained from various sources. Cognitive Pairwise Rating (CPR) is used to evaluate the attribute weights and nominal attribute values in the raw data matrix according to customer preferences. The raw data matrix within preference values is normalized. In steps 5 and 6, the normalized data matrix and attribute weights are aggregated into a vector of product values. A Top-N list is generated according to the product values. In step 7, products are grouped by AHC algorithm according to their similarities. Similar products recommendation could be provided according to product clusters.

Step 1 Attribute Specification

Product information can be collected from various sources, such as retailers, product engineers and customers. A product can be represented by a vector of its attributes $\{\delta_i\} = (\delta_1, \delta_2 ... \delta_i ... \delta_n)$ where δ_i denotes the i^{th} attribute. Some attributes can be divided into a set of sub-attributes. For example, the attribute δ_i has n_i sub-attributes, $\{\delta_{i,j}\} = (\delta_{i,1}, \delta_{i,2} ... \delta_{i,j} ... \delta_{i,n_i})$, where $\delta_{i,j}$ is the j^{th} sub-attribute of δ_i. Some sub-attributes can be further divided. For example, $\delta_{i,j}$ has $n_{i,j}$ sub-attributes $\{\delta_{i,j,k}\} = (\delta_{i,j,1}, \delta_{i,j,2} ..., \delta_{i,j,k} ... \delta_{i,j,n_{i,j}})$ where $\delta_{i,j,k}$ is the k^{th} sub-attribute of $\delta_{i,j}$. The relationships between attributes and their sub-attributes can be organized as an attributes tree. An attribute is represented by a node whilst its sub-attributes are represented by its children. An example of laptop attribute tree is shown in Figure 5.2.

Step 2 Data Preparation

The attributes with no sub-attributes are leaf nodes of the attribute tree. A leaf attribute, denoted by *L*, is a measureable attribute obtained from various sources mentioned above. A product dataset of *m* products and *l* leaf attributes is organized as an $m \times l$ raw data matrix, $D = \{d_{\alpha\beta} | \forall \alpha \in (1, ..., m), \forall \beta \in (1, ..., l)\}$. An example of collecting product information as data matrix is presented in Step 2 of Example 5.1. The raw product data matrix *D* cannot be directly used in the clustering process since it may contain nominal label values. Using CPR, the nominal values can be represented by numerical values using the method presented in Step 3.

Step 3 Preferences Elicitation

User preferences could be collected in various ways. The preferences for different attributes and nominal scales could be measured by Cognitive Pairwise Rating. An example of CPR interface is shown as Figure. A1 in the Appendix.

Table 5.1: Measurement Scale Schema for CPR

Label ($\aleph$)	Notation	Paired Interval Scale ($\overline{X}$)
Equally	0	0
Slightly	1	$\kappa/8$
Moderately	2	$2\kappa/8$
Fairly	3	$3\kappa/8$
Highly	4	$4\kappa/8$
Strongly	5	$5\kappa/8$
Significantly	6	$6\kappa/8$
Outstandingly	7	$7\kappa/8$
Absolutely	8	κ

A measurement scale schema $(\aleph, \overline{X})$ is used for the paired comparisons as shown in Table 5.1. $\aleph$ is the space of linguistic labels of the paired interval scales such as {Equally, Slightly, Moderately, Fairly, Highly, Strongly, Significantly, Outstandingly, Absolutely}. The numerical representation of paired interval scales $\overline{X}$ is the form below.

$$\overline{X} = \left\{ \overline{x}_q = \frac{q\kappa}{\tau} \middle| \forall q \in \{-\tau, ..., -1, 0, 1, ..., \tau\}, \kappa > 0 \right\} \tag{5.1}$$

Normal utility κ is the subjective perception of the difference between pairs and $\kappa = \max\left(\overline{X}\right)$ by default. τ is the number of linguistic scale. $2\tau + 1$ is the number of scales in $\left(\aleph, \overline{X}\right)$.

A Pairwise Opposite Matrices (POMs) of the form below is used to evaluate user preferences by pairwise comparison in paired interval scales.

$$B = [b_{ij}] = \begin{bmatrix} 0 & v_1 - v_2 & v_1 - v_n \\ v_2 - v_1 & 0 & v_2 - v_n \\ v_n - v_1 & v_n - v_2 & 0 \end{bmatrix} \cong \begin{bmatrix} 0 & b_{12} & b_{1n} \\ b_{21} & 0 & b_{2n} \\ b_{n1} & b_{n2} & 0 \end{bmatrix} = [b_{ij}] = B \tag{5.2}$$

where B is a POM, i and j are local indices. v_i is the priority value of the i^{th} object, and $b_{ij} \cong \left[v_i - v_j \right]$ is the approximate comparison value between the i^{th} and j^{th} objects. b_{ij} is the rating score obtained from the survey. For instance, $b_{12} = 2$ means that a consumer thinks that the first object is moderately more important than the second object.

Accordance Index (*AI*) of the form below is used to evaluate the validity of POM. If *AI*=0, then *B* is perfectly accordant. If $0 < AI \leq 0.1$, then *B* is satisfactory. If $AI > 0.1$, then *B* is unsatisfactory, and the corresponding survey should be re-assessed again.

$$AI = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{\frac{1}{n}\sum_{p=1}^{n}\left(\frac{b_{ip}+b_{pj}-b_{ij}}{\kappa}\right)^2} \tag{5.3}$$

Row Average plus the normal Utility (RAU) of the form below is used to calculate the priorities of objects.

$$RAU(B,\kappa) = \left[v_i : v_j = \left(\frac{1}{n}\sum_{j=1}^{n} b_{ij} + \kappa,\ \forall i \in \{1,...,n\} \right) \right] \tag{5.4}$$

The vector of normalized RAU values W is shown in the form below.

$$W = \left\{ w_i : w_i = \frac{v_i}{n\kappa}, \forall i \in \{1,...,n\} \right\},\ \ which \sum_{i \in \{1,...,n\}} v_i = n\kappa \tag{5.5}$$

The normalized RAU values can be used to represent various things including attribute weights, preference values for scales, priorities of options, and item utilities.

Step 4 Data Normalization

The raw data matrix is rescaled or normalized in this step. Two normalization methods are introduced for the criteria whether the higher or lower value is preferred. If the higher value reflects the higher preference, Dividing Maximal Function $\Delta_{\max}$ is applied to rescale a vector of raw attribute values, i.e. $D_\beta^T = \{d_{1,\beta}, ..d_{\alpha,\beta}.., d_{m,\beta}\}$, which means a column vector of the raw data matrix. If the lower value is preferred, Minimal Dividing Function $\Delta_{\min}$ is used for the normalization. Normalized data matrix, $D' = \{x_{\alpha\beta} | \forall \alpha \in (1,...,m), \forall \beta \in (1,...,l)\}$, is produced for following steps.

$$x_{\alpha\beta} = \Delta_{\max}(d_{\alpha\beta}) = \frac{d_{\alpha\beta}}{\max(D_\beta^T)}, \forall \alpha \in (1,...,m), \forall \beta \in (1,...,l) \tag{5.6}$$

$$x_{\alpha\beta} = \Delta_{\min}\left(d_{\alpha\beta}\right) = \frac{\min\left(D_{\beta}^{T}\right)}{d_{\alpha\beta}}, \forall \alpha \in (1,...,m), \forall \beta \in (1,...,l) \qquad (5.7)$$

Step 5 Data Fusion

The product values $\{\rho^{(\alpha)} : \forall \alpha \in (1,...,m)\}$ are aggregated from the sub-ordinate attributes in the weighted attribute tree, where (α) indicates the product index. The weights of δ_i, $\delta_{i,j}$ and $\delta_{i,j,k}$ are denoted as r_i, $r_{i,j}$ and $r_{i,j,k}$ respectively. The values of leaf attributes could be obtained from normalized data matrix D'. The attribute value is the weighted summation of its low-level attributes by the Eqs.5.8-5.10.

$$\delta_{i,j}^{(\alpha)} = \sum_{k=1}^{n_{i,j}} r_{i,j,k} \cdot \delta_{i,j,k}^{(\alpha)}, \forall i \in (1,...,n), \forall j \in (1,...,n_i), \forall \alpha \in (1,...,m) \qquad (5.8)$$

$$\delta_{i}^{(\alpha)} = \sum_{j=1}^{n_i} r_{i,j} \cdot \delta_{i,j}^{(\alpha)}, \ \forall i \in (1,...,n), \forall \alpha \in (1,...,m) \qquad (5.9)$$

$$\rho^{(\alpha)} = \sum_{i=1}^{n} r_i \cdot \delta_{i}^{(\alpha)}, \forall \alpha \in (1,...,m) \qquad (5.10)$$

Step 6 Top-N List Generation

A Top-N products list includes the products of the *N* highest values. The Top-N list is recommended in descending order. Algorithm 5.1 shows the calculation details. The Top-N list with different users may be different since product values are calculated with respect to their preference inputs.

Algorithm 5.1*: *Top-N List

```
Procedure Top-N ($\{\rho^{(\alpha)}\}$,N)
For i=1 to N,
    M =Max($\{\rho^{(\alpha)}\}$)
    TopN[i]= M
    $\{\rho^{(\alpha)}\} = M / \{\rho^{(\alpha)}\}$, where / is a complement operator.
Return TopN
```

Step 7 Products Clustering

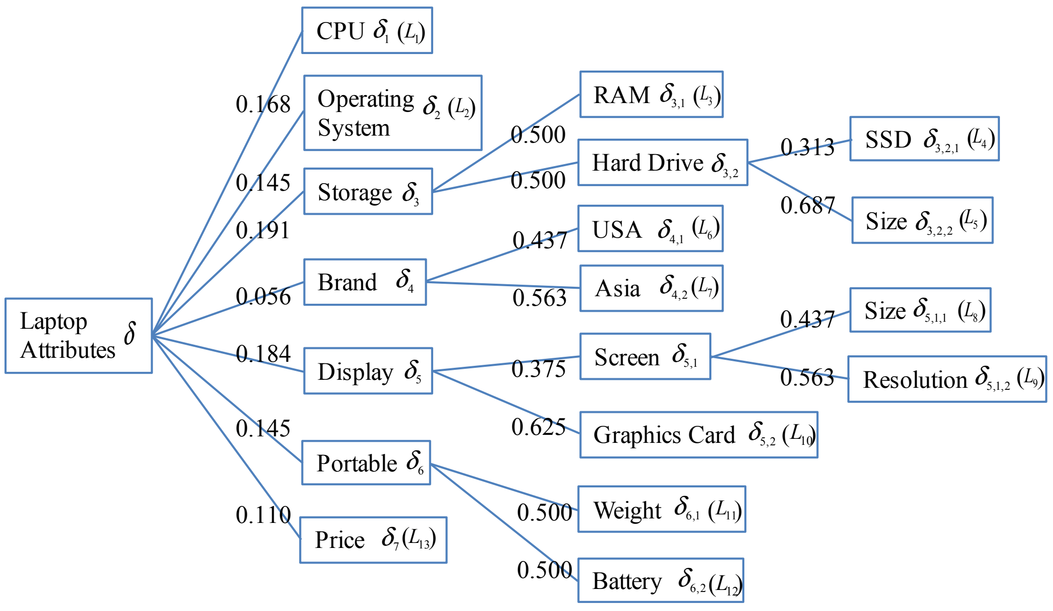
The products are clustered by proposed ESA-DCC method taking the product values as input.

Example 5.1: Application to Laptop Recommender System

Laptops are constituted of many attributes. When consumers search for a laptop, they search for a set of suitable attributes matching their requirements. To demonstrate the validity and applicability of proposed CPR-AHC approach, a laptop recommender system for a user is presented in this section. The laptops dataset from the market information can be collected from various sources such as websites of online retail shops, magazines and manufacturers. A small size of the dataset used in this section was manually collected in 2015.

Step 1 Attribute Specification

The websites for selling, introducing or comparing laptops provide a tremendous amount of laptop information. A set of distinct attributes to select an ideal laptop are presented in a tree structure in Figure 5.2. Some consumers may be unacquainted with some attributes, such as wireless type and video output. Some attributes may not be important to the users, such as the number of USB ports, DVD/CD burner, and speakers. These attributes are not considered in the recommender system.



Laptop Attributes δ
CPU δ_1 (L_1)
Operating System δ_2 (L_2)
Storage δ_3
Brand δ_4
Display δ_5
Portable δ_6
Price δ_7(L_{13})
0.168
0.145
0.191
0.056
0.184
0.145
0.110
RAM $\delta_{3,1}$ (L_3)
Hard Drive $\delta_{3,2}$
USA $\delta_{4,1}$ (L_6)
Asia $\delta_{4,2}$(L_7)
Screen $\delta_{5,1}$
Graphics Card $\delta_{5,2}$ (L_{10})
Weight $\delta_{6,1}$ (L_{11})
Battery $\delta_{6,2}$(L_{12})
0.500
0.500
0.437
0.563
0.375
0.625
0.500
0.500
SSD $\delta_{3,2,1}$ (L_4)
Size $\delta_{3,2,2}$ (L_5)
Size $\delta_{5,1,1}$ (L_8)
Resolution $\delta_{5,1,2}$ (L_9)
0.313
0.687
0.437
0.563

Figure 5.2 Attribute Tree for Laptops

The first level of the attributes tree of laptops constitutes CPU (δ_1), Operation System (δ_2), Storage (δ_3), Brand (δ_4), Display (δ_5), Portable (δ_6) and Price (δ_7). Five of them have sub-attributes, such as the storage including Hard Drive and Random-Access Memory (RAM). These sub-attributes are represented in the second level of the attribute tree, including {RAM ($\delta_{3,1}$), Hard Drive ($\delta_{3,2}$)}, {USA ($\delta_{4,1}$), Asia ($\delta_{4,2}$)}, {Screen ($\delta_{5,1}$), Graphics Card ($\delta_{5,2}$)}, and {Weight ($\delta_{7,1}$), Battery ($\delta_{7,2}$)}.Some sub-attributes can be further divided. Hard Drive, for example, can be divided into Solid State Drive (SSD) and Size. Level three consists of the sub-attributes including {SSD ($\delta_{3,2,1}$), Size ($\delta_{3,2,2}$)} and {Size ($\delta_{5,1,1}$), Resolution ($\delta_{5,1,2}$)}.

Step 2 Data Preparation

According to the attribute tree, a laptop can be represented by a vector of 13 leaf attributes. The raw data matrix *D* of 27 laptops are complied and presented in Table A2 in Appendix. Some leaf attribute values are numerical values whilst some nominal attribute values are labels and need to be further converted into numerical values. The calculation methods to quantify leaf attributes are summarized in Table 5.2. The attributes CPU and Graphics Card are measured by the test scores of 3DMark06 [Benchmarks, 2016]. The SSD attribute has three kinds of values: SSD means that the hard disk in a laptop is SSD, No SSD means that the laptop does not have an SSD and Hybrid means that part of the hard disks is SSD. The three values of SSD attribute are respectively rated as 2, 0 and 1. The numerical values of screen resolution attribute is measured by the product of width pixel and height pixel. The nominal values will be converted to numerical values by CPR in Step 3.

Step 3 User Preferences Elicitation

The user preferences data can directly be collected by the CPR surveys. An example of a CPR survey questionnaire to measure user preferences is shown in Figure A1 in the Appendix. The measurement scale schema is defined as Table 5.1. κ is set to 8. According to Eq.2, the POM obtained from the survey results in Figure A1 is shown in Table 5.3. *AI* is computed by Eq.5.3 and is less than 0.1, which is within an acceptable range. Weights of laptop attributes are computed by Eqs.5.4 and 5.5, where the computational steps are presented in Table 3. The POMs, *AI* and weights of other sub-attributes are shown in Table 5.4. The weights are summarized in the attribute tree exhibited in Figure 5.2. The nominal scales of Operating System (L_2), Asia Brand (L_6) and USA Brand (L_7) are measured by CPR and shown in Table 5.5. The preference values of the attributes are the prioritization results of POMs.

Table 5.2: Schema of Leaf Attributes of Laptops

Measurable attribute	Leaf attribute	Measurement scale	Quantification method	Normalization method
L_1	CPU δ_1	Nominal: CPU model	3DMark06 Score	$\Delta_{\max}$
L_2	OS δ_2	Nominal: Linux OS X Windows 7 Windows 8	CPR	$\Delta_{\max}$

L_3	RAM $\delta_{3,1}$	GB	GB	$\Delta_{\max}$
L_4	SSD $\delta_{3,2,1}$	Nominal: SSD Hybrid No SSD	SSD: 2 Hybrid:1 No SSD: 0	$\Delta_{\max}$
L_5	Hard Drive Size $\delta_{3,2,2}$	GB	GB	$\Delta_{\max}$
L_6	Brand (USA) $\delta_{4,1}$	Nominal: Alienware Apple Dell Microsoft	CPR	$\Delta_{\max}$
L_7	Brand (Asia) $\delta_{4,2}$	Nominal: Acer ASUS HP Lenovo Sansung	CPR	$\Delta_{\max}$
L_8	Screen Size $\delta_{5,1,1}$	Inch	Inch	$\Delta_{\max}$
L_9	Screen Resolution $\delta_{5,1,2}$	DPI	Width pixel by Height pixel	$\Delta_{\max}$
L_{10}	Graphics Card $\delta_{5,2}$	Nominal: Graphics Card model	3DMark06 Scores	$\Delta_{\max}$
L_{11}	Weight $\delta_{6,1}$	Kg	Kg	$\Delta_{\min}$
L_{12}	Battery $\delta_{6,2}$	Hour	Hour	$\Delta_{\max}$
L_{13}	Price δ_7	RMB	Thousand RMB	$\Delta_{\min}$

Table 5.3: Comparison Matrices For Laptop 1st Level Attributes of User A

B_0	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	$\sum_{j=1}^{7} b_{ij}$	$\frac{1}{7}\sum_{j=1}^{7} b_{ij}$	$v_i = \frac{1}{7}\sum_{j=1}^{7} b_{ij} + 8$	$r_i = w_i = \frac{v_i}{7 \cdot 8}$
δ_1	0	1	-1	7	-1	1	3	10	1.429	9.429	0.168
δ_2	-1	0	-3	5	-2	0	2	1	0.143	8.143	0.145
δ_3	1	3	0	7	0	3	5	19	2.714	10.714	0.191
δ_4	-7	-5	-7	0	-7	-5	-3	-34	-4.857	3.143	0.056
δ_5	1	2	0	7	0	2	4	16	2.286	10.286	0.184
δ_6	-1	0	-3	5	-2	0	2	1	0.143	8.143	0.145
δ_7	-3	-2	-5	3	-4	-2	0	-13	-1.857	6.143	0.110
						AI=0.051					

Table 5.4: Comparison Matrices of User A for Laptop Sub-attributes

B_3	$\delta_{3,1}$	$\delta_{3,2}$	$r_{3,i}$	$B_{3,2}$	$\delta_{3,2,1}$	$\delta_{3,2,2}$	$r_{3,2,i}$	B_4	$\delta_{4,1}$	$\delta_{4,2}$	$r_{4,i}$
$\delta_{3,1}$	0	0	0.5	$\delta_{3,2,1}$	0	-6	0.313	$\delta_{4,1}$	0	-2	0.437
$\delta_{3,2}$	0	0	0.5	$\delta_{3,2,2}$	6	0	0.687	$\delta_{4,2}$	2	0	0.563
		AI=0				*AI*=0				*AI*=0	
B_5	$\delta_{5,1}$	$\delta_{5,2}$	$r_{5,i}$	$B_{5,1}$	$\delta_{5,1,1}$	$\delta_{5,1,2}$	$r_{5,1,i}$	B_6	$\delta_{6,1}$	$\delta_{6,2}$	$r_{6,i}$
$\delta_{5,1}$	0	-4	0.375	$\delta_{5,1,1}$	0	-2	0.437	$\delta_{6,1}$	0	0	0.5
$\delta_{5,2}$	4	0	0.625	$\delta_{5,1,2}$	2	0	0.563	$\delta_{6,2}$	0	0	0.5
		AI=0				*AI*=0				*AI*=0	

Table 5.5: Comparison Matrices of User A for Nominal Attribute of L_2, L_6 and L_7

B_2	Linux	OS X	Windows 7	Windows 8		Preference Value
Linux	0	-2	-3	0		0.211
OS X	2	0	-1	2		0.273
Windows 7	3	1	0	3		0.305
Windows 8	0	-2	-3	0		0.211
			AI=0			
B_6	Alienware	Apple	Dell	Microsoft	HP	Preference Value
Alienware	0	1	3	4	4	0.260
Apple	-1	0	2	3	3	0.235
Dell	-3	-2	0	1	2	0.190
Microsoft	-4	-3	-1	0	0	0.160
HP	-4	-3	-2	0	0	0.155
			AI=0.043			
B_7	Acer	ASUS	Lenovo	Sansung		Preference Value
Acer	0	-1	-3	2		0.234
ASUS	1	0	-2	3		0.266
Lenovo	3	2	0	4		0.320
Sansung	-2	-3	-4	0		0.180
			AI=0.042			

Step 4 Data Normalization

The chosen normalization method for each leaf attribute is presented in Table 5.2. For example, the higher performance score of CPU (L_1) should be more preferable, and therefore $\Delta_{\max}$ (shown in Eq. 5.6) is chosen to normalize the vector of CPU. The lower price (L_{13}) should be more preferable, and therefore $\Delta_{\min}$ (shown in Eq. 7) is chosen to normalize price attribute. The results of normalized data D' are shown in Table A3 in appendix. For example, the calculation details for normalizing the attribute values of CPU and Price for the laptop ID1 are shown below.

$$x_{1,1} = \Delta_{\max}(d_{1,1}) = \frac{d_{1,1}}{\max(D_1^T)} = \frac{3367}{7060} = 0.477 \tag{5.11}$$

$$x_{1,13} = \Delta_{\min}(d_{1,13}) = \frac{\min(D_{13}^T)}{d_{1,13}} = \frac{2}{7} = 0.285 \tag{5.12}$$

Step 5 Data Fusion

According to the weights presented in Tables 5.3-5.4 and the normalized data matrix D' presented in Table A3 in appendix, for each laptop, the attribute values of second level are computed by Eq.8. For example, the computation of $\delta_{3,2}^{(1)}$ is shown as below.

$$\begin{aligned}\delta_{3,2}^{(1)} &= \sum_{k=1}^{2} r_{3,2,k} \cdot \delta_{3,2,k}^{(1)} = (r_{3,2,1} \cdot \delta_{3,2,1}^{(1)}) + (r_{3,2,2} \cdot \delta_{3,2,2}^{(1)}) \\ &= (0.313 \cdot x_{1,4}) + (0.687 \cdot x_{1,5}) = (0.313 \cdot 1.000) + (0.687 \cdot 0.169) \\ &= 0.429\end{aligned} \tag{5.13}$$

Similarly, the value of $\delta_{5,1}^{(1)}$ is computed as 0.556. The attribute values of first level are computed by Eq.9. the computation of $\delta_3^{(1)}$ is shown as below.

$$\begin{aligned}\delta_3^{(1)} &= \sum_{j=1}^{2} r_{3,j} \cdot \delta_{3,j}^{(1)} = (r_{3,1} \cdot \delta_{3,1}^{(1)}) + (r_{3,2} \cdot \delta_{3,2}^{(1)}) = (r_{3,1} \cdot x_{1,5}) + (r_{3,2} \cdot \delta_{3,2}^{(1)}) \\ &= (0.500 \cdot 0.250) + (0.500 \cdot 0.429) = 0.340\end{aligned} \tag{5.14}$$

Similarly, the values of $\delta_4^{(1)}$, $\delta_5^{(1)}$ and $\delta_6^{(1)}$ are computed as 0.563, 0.327 and 0.550 respectively. The product values of laptop are aggregated by Eq.10. For example, the product value of the laptop ID1 is calculated as below. The product values of all the laptops are presented in Table 6.

$$\begin{aligned}\rho^{(1)} &= \sum_{i=1}^{7} r_i \cdot \delta_i^{(1)} \\ &= (r_1 \cdot \delta_1^{(1)}) + (r_2 \cdot \delta_2^{(1)}) + (r_3 \cdot \delta_3^{(1)}) + (r_4 \cdot \delta_4^{(1)}) + (r_5 \cdot \delta_5^{(1)}) + (r_6 \cdot \delta_6^{(1)}) + (r_7 \cdot \delta_7^{(1)}) \\ &= (r_1 \cdot x_{1,1}) + (r_2 \cdot x_{1,2}) + (r_3 \cdot \delta_3^{(1)}) + (r_4 \cdot \delta_4^{(1)}) + (r_5 \cdot \delta_5^{(1)}) + (r_6 \cdot \delta_6^{(1)}) + (r_7 \cdot x_{1,1}) \\ &= 0.448\end{aligned} \tag{5.14}$$

Table 5.6: Laptop Product Values for User A

ID (α)	Laptop Model	Product Value ($\rho^{(\alpha)}$)
1	Lenovo Yoga3 14-IFI	0.448
2	Lenovo Y430p AT-ISE	0.553
3	Lenovo ThinkPad E540 20C60019CD	0.465
4	Dell XPS 11 (XPS11D-1508T)	0.403
5	Dell Inspiron 15 (INS15UD-1748S)	0.447
6	Dell Inspiron 15 7000 (Ins15BD-1748)	0.459
7	MacBook 256GB	0.507
8	MacBook Pro 15’	0.660
9	MacBook Air (MJVE2CH/A)	0.478
10	ASUS A550JK4200	0.476
11	ASUS GFX71JY4710	0.662
12	ASUS U305FA5Y71 (8GB/256GB)	0.475
13	Acer VN7-591G-56BD	0.427
14	Acer E1-470G-33212G50Dnkk	0.419
15	Acer VN7-791G-78KL	0.603
16	HP Envy 15-k222tx	0.431
17	HP ProBook 440 G2 (J7W06PA)	0.535
18	HP Pavilion 11-h112tu x2 (G0A07PA)	0.380
19	Sansung 910S3G-K04	0.378
20	Sansung 930X2K-K01	0.431
21	Sansung 900X3K-K01	0.488
22	Surface Pro 3(i3/64GB)	0.445
23	Surface Pro 3 (i7/512GB/Profession)	0.493
24	Surface 3 (4GB/128GB)	0.457
25	Alienware 15 (ALW15ED-1718)	0.643
26	Alienware 13 (ALW13ED-2708)	0.462
27	Alienware 17 (ALW17ED-2728)	0.676

Step 6 Top-N List Generation

By using the product values, a Top-N list recommendation is generated by Algorithm 1. With respect to the user preference inputs for the attribute data, a Top-10 list for laptops, {27, 11, 8, 25, 15, 2, 17, 7, 23, 21}, is shown in Table 5.7. Therefore, the system will generate the recommended web links and product information for the laptops to customers in descending order after the users simply submit the CPR survey as search queries.

Table 5.7. The Top-10 Laptops for User A

Rank	ID (α)	Laptop Model	Product Value ($\rho^{(\alpha)}$)
1	27	Alienware 17 (ALW17ED-2728)	0.676
2	11	ASUS GFX71JY4710	0.662
3	8	MacBook Pro 15’	0.660
4	25	Alienware 15 (ALW15ED-1718)	0.643
5	15	Acer VN7-791G-78KL	0.603
6	2	Lenovo Y430p AT-ISE	0.553
7	17	HP ProBook 440 G2 (J7W06PA)	0.535
8	7	MacBook 256GB	0.507
9	23	Surface Pro 3 (i7/512GB/Profession)	0.493
10	21	Sansung 900X3K-K01	0.488

Step 7 Products Clustering

The clustering results produced by ESA-DCC applied F_{NK} are shown in Table 5.8 below.

Table 5.8. Clustering Results of Laptops

K	Parameter pair	Pattern Results	Best fitness values
3	0.030, 0.040	1 1 1 1 1 1 1 2 1 1 2 1 1 1 3 1 1 1 1 1 1 1 1 1 2 1 2	0
5	0.021, 0.100	1 2 1 1 1 1 1 3 1 1 3 1 1 1 4 1 2 5 5 1 1 1 1 1 3 1 3	0
7	0.016, 0.066	1 2 1 1 1 1 1 3 1 1 3 1 1 1 4 1 5 6 6 1 1 1 1 1 7 1 3	0
9	0.014, 0.008	1 2 1 3 1 1 1 4 1 1 4 1 1 1 5 1 6 7 7 1 1 1 1 1 8 1 4	0.111

For example, if the user wants to divide the laptops into 5 groups, the pattern results in the second line of Table 5.8 will be produced as the result of ESA-DCC. The laptops are divided into five clusters: {1, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 16, 20, 21, 22, 23, 24, 26}, {2, 17}, {8, 11}, {15}, and {18, 19}. Once the user chooses any laptop which belong to a cluster, the system will recommend the other laptops in this cluster. For example, once the user takes a look on the webpage of laptop 8, the system will recommend laptop 11 to him/her.

5.2 Image Segmentation

Image segmentation is an important topic in the field of computer vision. The representation of images can be simplified by segmenting a digital image into multiple regions for further analysis [Felzenszwalb & Huttenlocher, 2004]. The pixels of an image can be clustered as regions with respect to their color similarities and spatial relationships [Felzenszwalb et al., 2004; Jain & Flynn, 1996; Chuang, Tzeng, Chen, Wu & Chen, 2006].

Density-based clustering methods can be applied for image segmentation due to their ability to discover arbitrary shaped clusters [Celebi, Aslandogan & Bergstresser, 2005; Shen, Hao, Liang, Liu, Wang & Shao, 2016; Ye, Gao & Zeng, 2003]. DBSCAN is the widely used density-based clustering method. The main disadvantage of the proposed ESADCC method is the high time-complexity. As the size of an image dataset is normally very large, it is hard to be directly processed by ESADCC.

K-means [Macqueen, 1967] has been applied for image segmentation [Felzenszwalb et al., 2004; Jain et al., 1996;] by clustering the pixels with respect to their color similarities. As K-means was designed for centroid-based cases, it can not be used to detect the arbitrary shaped regions in an image.

A hybrid approach, Kmeans-ESADCC is proposed by combining K-means and ESADCC for image segmentation. For image preprocessing, the color and spatial information for all the pixels are represented in a 2-D data matrix. K-means algorithm is then adopted to cluster the pixels into a number of small clusters whilst the centroid of each cluster is to group the data points with similar aggregated features values. The cluster centroids produced by K-means are further clustered by ESADCC. The image segmentation results are finally provided by fusing the results of K-means and ESADCC. To demonstrate the usability of the proposed method, four images selected from Berkeley Segmentation Dataset and Benchmark (BSDB) [Martin, Fowlkes, Tal, & Malik, 2007] were used in the experiment.

5.2.1 Data Preprocessing

- **Matrix Combinations**

An RGB color image can be represented as a 3-D raw dataset by using the image reading functions such as *readJPEG()* provided by an R package called jpeg [Urbanek, 2014]. The 3-D raw dataset is a color matrix consisting of three additive primary color (Red, Green and Blue) values of each pixel. In the preprocessing stage, the raw 3-D image dataset is reconstructed as a 2-D data matrix by combining the primary color and spatial information. The i^{th} pixel is represented by the vector (X_i, Y_i, R_i, G_i, B_i) where X_i, Y_i are the spatial values computed by Eqs. (5.20-5.21) and (R_i, G_i, B_i) is the color vector from the raw image dataset.

$$X_i = \frac{1 + i \setminus x}{x} \tag{5.20}$$

$$Y_i = \frac{i \bmod y}{y} \tag{5.21}$$

where x and y are the numbers of pixels in the x and y axes of the image respectively; the notation, \, is integer division.

- **K-means**

The general procedure of K-means for grouping the pixels into clusters are presented in Algorithm 5.2. The MacQueen K-means [MacQueen, 1967] has been applied in the proposed Kmeans-ESADCC approach. A number of cluster centroids are randomly initialized and then every point is assigned to the nearest centroid. The mean value of data points assigned to a new cluster is computed. The loop should be executed until the centroids of clusters converge. The similarities of pixels are computed as Euclidean distance by Eq. (5.22).

$$d_{ii'} = \sqrt{(X_i - X_{i'})^2 + (Y_i - Y_{i'})^2 + (R_i - R_{i'})^2 + (G_i - G_{i'})^2 + (B_i - B_{i'})^2} \tag{5.22}$$

Algorithm 5.2: K-means for Image Dataset

```
Input: Image data matrix $D$, the number of clusters $K$;
Output: a set of clusters $C$ and clusters centroids $P$;
1. Randomly choose $K$ pixels from $D$ as the initial cluster centroid s;
2. Assign each pixel to the closest cluster on the basis the similarities between the
   pixels computed by Eq. (5.22);
3. Update each cluster centroid as the mean value of the pixels in the cluster;
4. Repeat steps 2 and 3 until the centroids stop changing;
```

5.2.2 Case Studies

Four images shown in Figure 5.3 are selected from Berkeley Segmentation Dataset and Benchmark (BSDB) [Martin et al, 2007] to evaluate the usability of proposed Kmeans-ESADCC for image segmentation. The proposed method is implemented by R language with packages stats [Team & Worldwide, 2012] and fpc [Hennig, 2014]. The processing details and results of the four images are presented as follows.

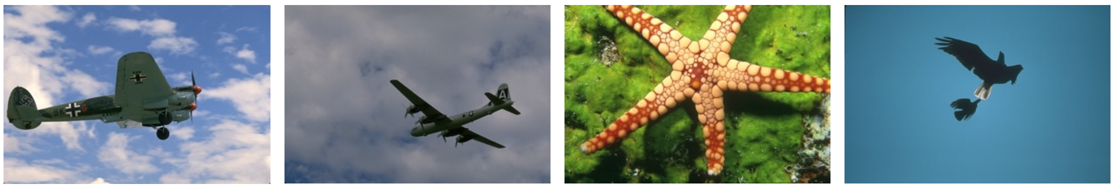

(a) Image 3063 (b) Image 3069 (c) Image 12003 (d) Image 135069

Figure 5.3 Orginal Images.

Step 1 Image Dataset Preprocessing

Each image has 481 pixels in x-axis and 321 pixels in y-axis, and thus it has 154401 pixels in total. The raw image datasets have 2-dimension space matrix and each element for the matrix is the 3-dimension color values (R, G, B). For the example of Image 3063, the color values of the first pixel are 0.3451, 0.4627 and 0.6667 in the raw image dataset. In the 2-D data matrix, the first two attributes values are computed as (1\481+1)/481=0.0021 and (1mod321)/321=0.0.0031. The attribute values of the first 4 pixels in Image 3063 are shown in Table 5.9.

Table 5.9 A Sample of Preprocessed Image 3063 Data Matrix

Pixel ID	X_i	Y_i	R_i	G_i	B_i
1	0.0021	0.0031	0.3451	0.4627	0.6667
2	0.0021	0.0063	0.3686	0.4784	0.6745
3	0.0021	0.0093	0.4039	0.4980	0.6862
4	0.0021	0.0125	0.4392	0.5216	0.6902

Step 2 Pixel Clustering by K-means

The parameter settings of four sets of experiments are shown in Table II. For each image, two tests are performed by setting K as 50 and 100 respectively and the results are shown in Figures 2-3. By inspection of the segmentation results in Figures 2-3, K-means cannot solely detect the arbitrary shapes in the images, whilst the same color at the same area in a image is forced to be divided into small centroid-based clusters due to the X_i and Y_i values. The next step by DBSCAN is to solve this problem.

Step 3 Centroid Clustering by ESA-DCC

The parameters of DBSCAN are set in the procedure of ESA-DCC parameter. In these cases, each set of centroids produced cluster patterns *S* of two parts and few centroids are detected as noise by ESA-DCC.

Step 4 Clustering Results Fusion

The image segmentation results of Kmeans-ESADCC by fusing the clustering results produced are visualized in Figures 5.4-5.5. Each image has been divided into two parts, the object and the background.

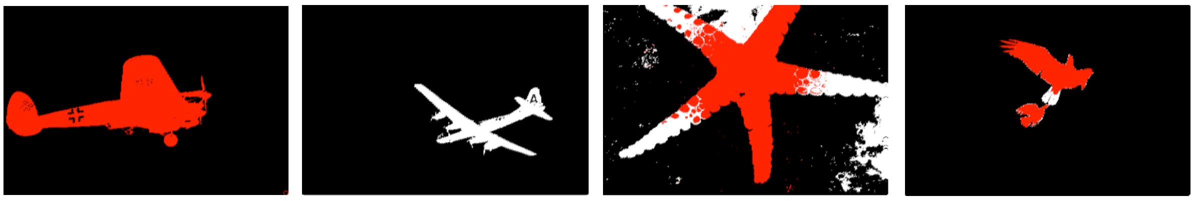

(a) Image 3063 (b) Image 3069 (c) Image 12003 (d) Image 135069

Figure 5.4: Segmentation Results of Kmeans-DBSCAN by Setting K as 50, DBSCAN Parameter Pairs as: (a) (0.40, 2); (b) (0.35, 3); (c) (0.31, 2); (d) (0.33, 3)

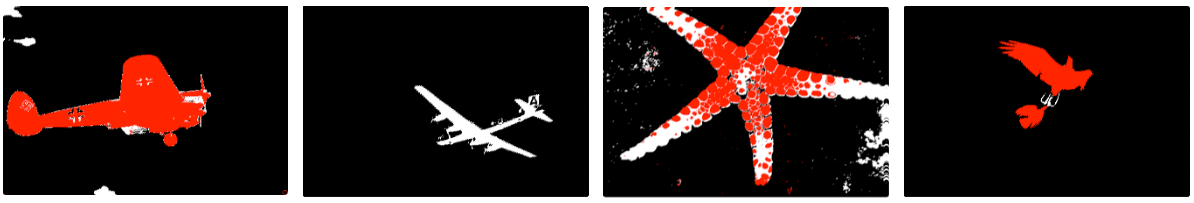

(a) Image 3063 (b) Image 3069 (c) Image 12003 (d) Image 135069

Figure 5.5: Segmentation Results of Kmeans-DBSCAN by Setting K as 100, DBSCAN Parameter Pairs as: (a) (0.25, 3); (b) (0.35, 5); (c) (0.31, 4); (d) (0.33, 5)

Chapter 6 Conclusions

6.1 Summary

Evolutionary and Swarm Algorithms (ESAs) can be used to optimize clustering methods. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an exemplar of density-based clustering which can easily find the arbitrary shapes of clusters by detecting the high-density hyper-spheres and merging the close hyper-spheres into clusters. The main research objective of this work is to use ESA methods to optimize DBSCAN.

The review section of this work includes the details of three main categories of clustering methods, partitioning, hierarchical and density-based clustering, four mainstream ESAs, Genetic Algorithm, Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony Algorithm. A number of ESA optimized clustering methods and the fitness functions applied in the methods are summarized in the review part.

A novel framework, Evolutionary and Swarm Algorithms Optimized Density-based Clustering and Classification (ESA-DCC), is proposed to overcome the drawbacks of classical Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Firstly, the two critical parameters for DBSCAN are difficult to set. To address this drawback, Evolutionary and Swarm Algorithms (ESA) are proposed to search the entire parameter space for DBSCAN. Secondly, when using DBSCAN the number of desired clusters cannot be pre-specified by the user. The second drawback is addressed by using the proposed fitness functions that take the number of desired clusters into account. Thirdly, DBSCAN cannot be used in a supervised learning context where the labelled training data can be used to drive the clustering process. This issue is addressed by a proposed fitness function which operates with the labelled training data, so that ESA-

DCC can also perform in a supervised learning context. Four ESA-DCC methods including GA-DCC, PSO-DCC, DE-DCC and ABC-DCC are implemented and evaluated in this thesis.

The first part of experiment is the evaluation of PSO-DCC using 10 datasets; comparisons are undertaken so as to analyze the operation of the proposed four fitness functions, F_{NK}, F_{SINK}, F_{DBNK}, and F_{CD}, and to analyze the operation of PSO-DCC in comparisons to DBSCAN, K-means and SVM. The experimental results indicate that F_{NK} is suitable for clustering datasets featured arbitrary shaped clusters and any shaped datasets with clear boundaries. PSO-DCC applying F_{SINK} can be used to deal with the general centroid-based clustering problems whilst F_{DBNK} has no advantage in each type of clustering problems. By comparing the usage of DB and SIL indices in the proposed fitness function, it can be concluded that the clustering index in range of [0,1] is suitable to be used in the fitness function for unsupervised ESA-DCC method. As a supervised method, PSO-DCC applying F_{CD} overperforms SVM for most of the 10 datasets. For the datasets which are not well understood, the proposed ESA-DCC method can be applied to search possible clustering patterns by testing with different numbers of clusters.

The second part experiment is to compare the performance of four ESA-DCC methods: GA-DCC, PSO-DCC, DE-DCC and ABC-DCC. The ESA-DCCs are tested using the ten datasets and proposed four fitness functions. The experiment results showed that GA-DCC is the most efficient one among the four methods, whilst the convergence of PSO-DCC is the slowest. The accuracy of DE-DCC is the highest among all the unsupervised ESA-DCC methods, whilst the accuracies of all the four supervised ESA-DCC methods are similar.

The four proposed fitness functions are non-linear. For the datasets with un-even density, the surface of fitness functions could be very rugged and ruleless. GA and DE have the mutation feature which performs better to find a good enough solution from the uneven surface. The features of PSO and ABC are more suitable for optimizing linear functions; therefore, the advantages of PSO and ABC are hard to be shown in proposed methods.

Some limitations of the proposed ESA-DCC framework are evident. Firstly, the computational complexity of ESA-DCC is limited by the complexity of the DBSCAN method which is as high as $O(n\log n)$. Since ESA-DCC adopts the standard DBSCAN, and the DBSCAN runs multiple times to reach the optimal solution in ESA-DCC, the complexity of ESA-DCC is higher than DBSCAN. Secondly, the clustering indices applied in the fitness functions may not be suitable for arbitrary shaped clusters since the indices were proposed for measuring the goodness of centroid-based clusters. Thirdly, the weights for the components of a fitness function could be further investigated.

In the application chapter, the proposed methods were applied in two real world cases, a laptop recommender system and a group of image segmentation cases. The future works regarding to limitations of ESA-DCC and its applications are discussed in next subsection.

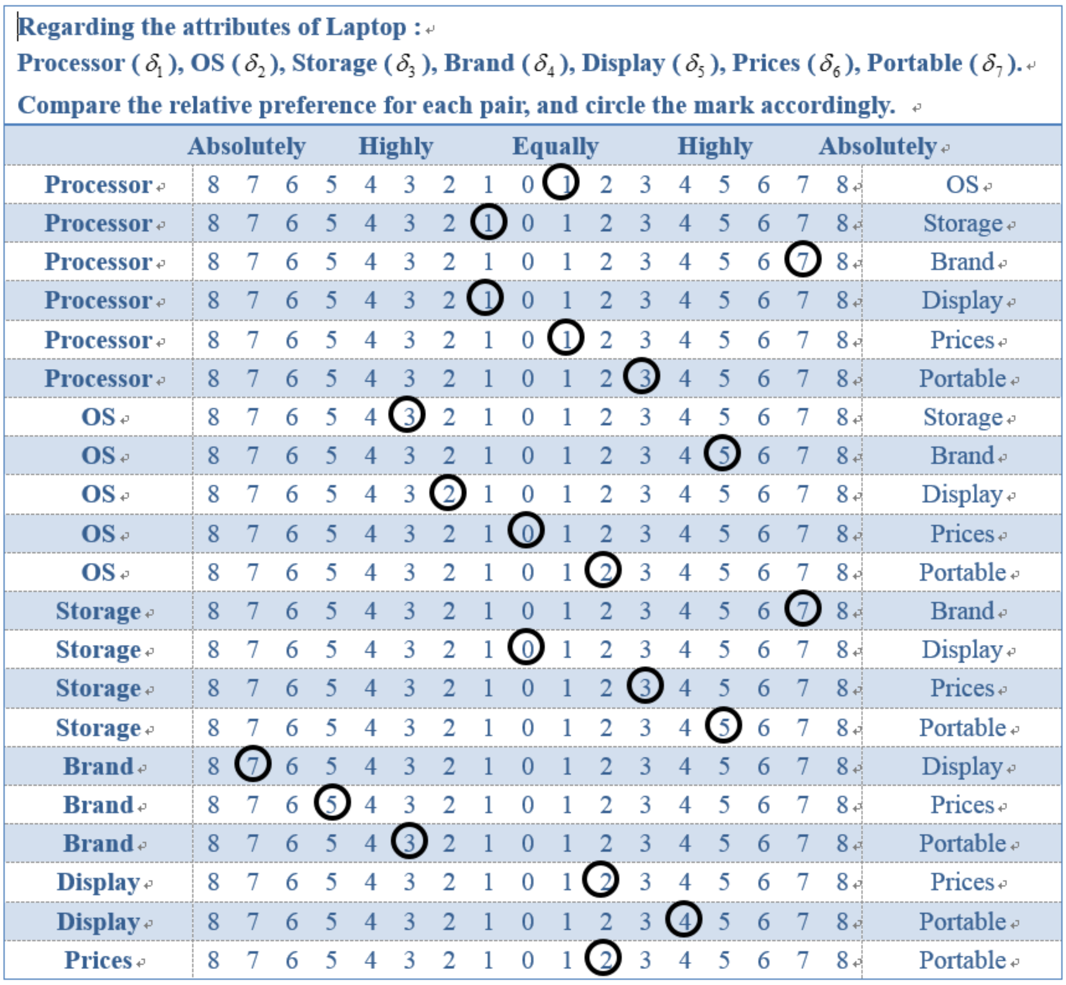
6.2 Future Works

The future work mainly consists of two parts: enhancing the performance of ESA-DCC methods and expanding the application of proposed ESA-DCC framework.

To enhance the performance of proposed methods, the algorithm of proposed framework should be improved. Firstly, a fast fall mechanism should be introduced in the algorithm. Once the clustering pattern results are detected as poor results, the current loop should stop to save time. For example, if too many noises have already been detected, the clustering process should be terminated and start the next iteration. Secondly, since the experiment results of ESA-DCC methods may widely vary with the selected fitness function using different clustering indices, more clustering indices will be tested in the proposed framework for comparisons. Thirdly, as mentiond in Section 4.2.3, the constraint to the minimum value of *minpt* parameter should be proposed to avoid possible unsatisfied clustering results. Fourly, a clustering index for density-based clustering methods will be proposed for the fitness function of ESA-DCC. The evaluation of revised proposed method should include the comparson between ESA-DCC and other existing schemes.

Since the advantage of DBSCAN is to find the arbitrary shaped clusters, the application area of ESA-DCC will be extended to high-dimension spatial dataset processing cases, such as face recognition and medical image segmentation.

Appendix



Regarding the attributes of Laptop :

Processor (δ_1), OS (δ_2), Storage (δ_3), Brand (δ_4), Display (δ_5), Prices (δ_6), Portable (δ_7).

Compare the relative preference for each pair, and circle the mark accordingly.

		Absolutely				**Highly**			**Equally**				**Highly**				**Absolutely**	
Processor	8	7	6	5	4	3	2	1	0	[✓ 1]	2	3	4	5	6	7	8	OS
Processor	8	7	6	5	4	3	2	[✓ 1]	0	1	2	3	4	5	6	7	8	Storage
Processor	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	[✓ 7]	8	Brand
Processor	8	7	6	5	4	3	2	[✓ 1]	0	1	2	3	4	5	6	7	8	Display
Processor	8	7	6	5	4	3	2	1	0	[✓ 1]	2	3	4	5	6	7	8	Prices
Processor	8	7	6	5	4	3	2	1	0	1	2	[✓ 3]	4	5	6	7	8	Portable
OS	8	7	6	5	4	[✓ 3]	2	1	0	1	2	3	4	5	6	7	8	Storage
OS	8	7	6	5	4	3	2	1	0	1	2	3	4	[✓ 5]	6	7	8	Brand
OS	8	7	6	5	4	3	[✓ 2]	1	0	1	2	3	4	5	6	7	8	Display
OS	8	7	6	5	4	3	2	1	[✓ 0]	1	2	3	4	5	6	7	8	Prices
OS	8	7	6	5	4	3	2	1	0	1	[✓ 2]	3	4	5	6	7	8	Portable
Storage	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	[✓ 7]	8	Brand
Storage	8	7	6	5	4	3	2	1	[✓ 0]	1	2	3	4	5	6	7	8	Display
Storage	8	7	6	5	4	3	2	1	0	1	2	[✓ 3]	4	5	6	7	8	Prices
Storage	8	7	6	5	4	3	2	1	0	1	2	3	4	[✓ 5]	6	7	8	Portable
Brand	8	[✓ 7]	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	Display
Brand	8	7	6	[✓ 5]	4	3	2	1	0	1	2	3	4	5	6	7	8	Prices
Brand	8	7	6	5	4	[✓ 3]	2	1	0	1	2	3	4	5	6	7	8	Portable
Display	8	7	6	5	4	3	2	1	0	1	[✓ 2]	3	4	5	6	7	8	Prices
Display	8	7	6	5	4	3	2	1	0	1	2	3	[✓ 4]	5	6	7	8	Portable
Prices	8	7	6	5	4	3	2	1	0	1	[✓ 2]	3	4	5	6	7	8	Portable

Figure A1. Interface for Comparison Pairwise Rating of Laptop Attributes

Table A1: Computational Details of AI for Nominal Attribute of L_7

(i,j)	b_{ij}	$B_i + B_j^T$	$B_i + B_j^T - b_{ij}$	$\left(\frac{1}{\kappa}\left(B_i + B_j^T - b_{ij}\right)\right)^2$	$d_{ij} = \sqrt{Mean\left(\left(\frac{1}{\kappa}\left(B_i + B_j^T - b_{ij}\right)\right)^2\right)}$
(1,1)	0	(0,0,0,0)	(0,0,0,0)	(0,0,0,0)	0
(1,2)	-1	(-1,-1, -1, -1)	(0,0,0,0)	(0,0,0,0)	0
(1,3)	-3	(-3,-3,-3,-2)	(0,0,0,1)	(0,0,0, 0.016)	0.0625
(1,4)	2	(2,2,1,2)	(0,0,-1,0)	(0,0,0.016,0)	0.0625
(2,1)	1	(1,1,1,1)	(0,0,0,0)	(0,0,0,0)	0
(2,2)	0	(0,0,0,0)	(0,0,0,0)	(0,0,0,0)	0
(2,3)	-2	(-2,-2,-2,-1)	(0,0,0,1)	(0,0,0,0.016)	0.0625
(2,4)	3	(3,3,2,3)	(0,0,-1,0)	(0,0,0.016,0)	0.0625
(3,1)	3	(3,3,3,2)	(0,0,0,-1)	(0,0,0, 0.016)	0.0625
(3,2)	2	(2,2,1,1)	(0,0,-1,-1)	(0,0, 0.016,0.016)	0.088
(3,3)	0	(0,0,0,0)	(0,0,0,0)	(0,0,0,0)	0
(3,4)	4	(5,5,4,4,)	(1,1,0,0)	(0.016,0.016,0,0)	0.088
(4,1)	-2	(-2,-2,-1,-2)	(0,0,1,0)	(0,0,0.016,0)	0.0625
(4,2)	-3	(-3,-3,-2,-3)	(0,0,1,0)	(0,0,0.016,0)	0.0625
(4,3)	-4	(-5,-5,-4,-4)	(-1,-1,0,0)	(0.016,0.016,0,0)	0.088
(4,4)	0	(0,0,0,0)	(0,0,0,0)	(0,0,0,0)	0

$$AI = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} = \frac{0.676}{16} = 0.042$$

Table A2: Raw Matrix *D* of 27 Laptops

ID	Laptop Model	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}	L_{11}	L_{12}	L_{13}
1	Lenovo Yoga3 14-IFI	3367	Win8	4	1	256	0	Lenovo	14	2073600	2385	1.6	6	7
2	Lenovo Y430p AT-ISE	6830	Win8	8	0	1000	0	Lenovo	14	1049088	4385	2.5	5	6
3	Lenovo ThinkPad E540 20C60019CD	3882	Linux	4	0	1000	0	Lenovo	15.6	1049088	1848	2.44	6	4
4	Dell XPS 11 (XPS11D-1508T)	2039	Win8	4	1	256	Dell	0	11.6	3686400	638	1.13	6	8
5	Dell Inspiron 15 (INS15UD-1748S)	3807	Win8	8	0	1000	Dell	0	15.6	1049088	1705	2.3	4	5
6	Dell Inspiron 15 7000 (Ins15BD-1748)	3807	Win8	8	0	1000	Dell	0	15.6	1049088	1857	2.11	7	7
7	MacBook 256GB	2589	OS	8	1	256	Apple	0	12	3317760	658	0.92	9	9
8	MacBook Pro 15'	6990	OS	16	1	512	Apple	0	15.4	5184000	2543	2.02	9	17
9	MacBook Air (MJVE2CH/A)	3393	OS	4	1	128	Apple	0	13.3	1296000	1333	1.35	9	7
10	ASUS A550JK4200	4361	Win8	4	0	1000	0	ASUS	15.6	2073600	4385	2.35	4	5
11	ASUS GFX71JY4710	6980	Win8	16	0.5	1256	0	ASUS	17.3	2073600	12632	4.8	3	19
12	ASUS U305FA5Y71 (8GB/256GB)	2503	Win8	8	4	256	0	ASUS	13.3	2073600	658	1.2	10	6
13	Acer VN7-591G-56BD	3367	Win8	4	0	500	0	Acer	15.6	2073600	4385	2.4	4	5
14	Acer E1-470G-33212G50Dnkk	2229	Linux	2	0	500	0	Acer	14	1049088	1213	2.1	4	2
15	Acer VN7-791G-78KL	7060	Win8	8	0.5	1064	0	Acer	17.3	2073600	9840	3	3	8
16	HP Envy 15-k222tx	3367	Win8	4	0	1000	HP	0	15.6	1049088	4385	2.34	4	5
17	HP ProBook 440 G2 (J7W06PA)	3420	Win7	8	0	1500	HP	0	14	1440000	1784	1.83	9	6
18	HP Pavilion 11-h112tu x2 (G0A07PA)	2071	Win8	4	1	128	HP	0	11.6	1049088	638	1.49	6	5
19	Sansung 910S3G-K04	1375	Win8	4	1	128	0	Sansung	13.3	1049088	638	1.44	5	4
20	Sansung 930X2K-K01	2492	Win8	4	1	128	0	Sansung	12.2	4096000	658	0.95	7	8
21	Sansung 900X3K-K01	3807	Win8	8	1	256	0	Sansung	13.3	5760000	968	1.07	6	10
22	Surface Pro 3(i3/64GB)	1675	Win8	4	1	64	Microsoft	0	12	3110400	638	0.8	9	4
23	Surface Pro 3 (i7/512GB/Profession)	3249	Win8	8	1	512	Microsoft	0	12	3110400	1033	0.8	9	12
24	Surface 3 (4GB/128GB)	2320	Win8	4	1	128	Microsoft	0	10.8	2457600	638	0.887	10	4
25	Alienware 15 (ALW15ED-1718)	6980	Win8	16	0.5	1128	Alienware	0	15.6	2073600	9809	3.207	4	15
26	Alienware 13 (ALW13ED-2708)	3807	Win8	8	1	384	Alienware	0	13.3	2073600	5249	2.058	3	13
27	Alienware 17 (ALW17ED-2728)	7060	Win8	16	0.5	1512	Alienware	0	17.3	2073600	12632	3.78	3	21

Table A3: Normalized data matrix D' of 27 Laptops for User A

ID	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}	L_{11}	L_{12}	L_{13}
1	0.477	0.692	0.250	1.000	0.169	0.000	1.000	0.809	0.360	0.189	0.500	0.600	0.286
2	0.967	0.692	0.500	0.000	0.661	0.000	1.000	0.809	0.182	0.347	0.320	0.500	0.333
3	0.550	0.692	0.250	0.000	0.661	0.000	1.000	0.902	0.182	0.146	0.328	0.600	0.500
4	0.289	0.692	0.250	1.000	0.169	0.672	0.000	0.671	0.640	0.051	0.708	0.600	0.250
5	0.539	0.692	0.500	0.000	0.661	0.672	0.000	0.902	0.182	0.135	0.348	0.400	0.400
6	0.539	0.692	0.500	0.000	0.661	0.672	0.000	0.902	0.182	0.147	0.379	0.700	0.286
7	0.367	0.897	0.500	1.000	0.169	0.983	0.000	0.694	0.576	0.052	0.870	0.900	0.222
8	0.990	0.897	1.000	1.000	0.339	0.983	0.000	0.890	0.900	0.201	0.396	0.900	0.118
9	0.481	0.897	0.250	1.000	0.085	0.983	0.000	0.769	0.225	0.106	0.593	0.900	0.286
10	0.618	0.692	0.250	0.000	0.661	0.000	0.829	0.902	0.360	0.347	0.340	0.400	0.400
11	0.989	0.692	1.000	0.500	0.831	0.000	0.829	1.000	0.360	1.000	0.167	0.300	0.105
12	0.355	0.692	0.500	1.000	0.169	0.000	0.829	0.769	0.360	0.052	0.667	1.000	0.333
13	0.477	0.692	0.250	0.000	0.331	0.000	0.732	0.902	0.360	0.347	0.333	0.400	0.400
14	0.316	0.692	0.125	0.000	0.331	0.000	0.732	0.809	0.182	0.096	0.381	0.400	1.000
15	1.000	0.692	0.500	0.500	0.704	0.000	0.732	1.000	0.360	0.779	0.267	0.300	0.250
16	0.477	0.692	0.250	0.000	0.661	0.466	0.000	0.902	0.182	0.347	0.342	0.400	0.400
17	0.484	1.000	0.500	0.000	0.992	0.466	0.000	0.809	0.250	0.141	0.437	0.900	0.333
18	0.293	0.692	0.250	1.000	0.085	0.466	0.000	0.671	0.182	0.051	0.537	0.600	0.400
19	0.195	0.692	0.250	1.000	0.085	0.000	0.561	0.769	0.182	0.051	0.556	0.500	0.500
20	0.353	0.692	0.250	1.000	0.085	0.000	0.561	0.705	0.711	0.052	0.842	0.700	0.250
21	0.539	0.692	0.500	1.000	0.169	0.000	0.561	0.769	1.000	0.077	0.748	0.600	0.200
22	0.237	0.692	0.250	1.000	0.042	0.328	0.000	0.694	0.540	0.051	1.000	0.900	0.500
23	0.460	0.692	0.500	1.000	0.339	0.328	0.000	0.694	0.540	0.082	1.000	0.900	0.167
24	0.329	0.692	0.250	1.000	0.085	0.328	0.000	0.624	0.427	0.051	0.902	1.000	0.500
25	0.989	0.692	1.000	0.500	0.746	1.000	0.000	0.902	0.360	0.777	0.249	0.400	0.133
26	0.539	0.692	0.500	1.000	0.254	1.000	0.000	0.769	0.360	0.416	0.389	0.300	0.154
27	1.000	0.692	1.000	0.500	1.000	1.000	0.000	1.000	0.360	1.000	0.212	0.300	0.095

Bibliography

A. Abraham, S. Das, and S. Roy, "Swarm intelligence algorithms for data clustering," in Soft computing for knowledge discovery and data mining. Springer, pp. 279–313, 2008.

A. A. Freitas, "A review of evolutionary algorithms for data mining," in Soft Computing for Knowledge Discovery and Data Mining. Springer, pp. 79–111, 2008.

A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: background and development," Natural Computing, vol. 6, no. 4, pp. 467–484, 2007.

A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part ii: hybridisation, combinatorial, multi- criteria and constrained optimization, and indicative applications," Natural Computing, vol. 7, no. 1, pp. 109–124, 2008.

A. E. Eiben, and E. S. James, Introduction to evolutionary computing, Heidelberg: springer, vol. 53, 2003.

A. K. Jain, R. C. Dubes, "Algorithms for clustering data," Prentice-Hall, Inc., 1988.

A. K. Jain and P. J. Flynn, "Image segmentation using clustering". IEEE Press, 1996.

A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Lett, 2009.

A. Shepitsen, J. Gemmell, B. Mobasher and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," Proceedings of the 2008 ACM conference on Recommender systems, ACM, pp: 259-266, 2008.

B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," Information Sciences, vol. 192, pp. 120–142, 2012.

Benchmarks, http://www.futuremark.com/benchmarks/legacy, Accessed March 2016.

B. Desgraupes, clusterCrit: Clustering Indices. R package version 1.2.7. URL: https://CRAN.R-project.org/package=clusterCrit, 2016.

B. Lika, K. Kolomvatsos and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," Expert Systems with Applications, vol. 41, no. 4, pp. 2065-2073, 2014.

B. Zhao, "An Improved Particle Swarm Optimization Algorithm for Global Numerical Optimization," in International Conference on Computational Science. Springer, pp. 657–664, 2006.

C. Cortes and V. Vapnik, “Support-vector networks,” Machine learning, vol. 20, no. 3, pp.273–297, 1995.

C. Hennig, “fpc: Flexible procedures for clustering,” R package version 2-1, 2014.

C. Porcel, and E. Herrera-Viedma, “Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries,” Knowledge-Based Systems, vol. 23, no.1, pp. 32-39, 2010.

C.-Y. Chen and F. Ye, “Particle swarm optimization algorithm and its application to clustering analysis,” in Networking, Sensing and Control, 2004 IEEE International Conference on, vol. 2. IEEE, pp. 789–794, 2004.

C.-Y. Lin, C.-C. Chang, and C.-C. Lin, “A new density-based scheme for clustering based on genetic algorithm,” Fundamenta Informaticae, vol. 68, no. 4, pp. 315–331, 2005.

C. Zhang, D. Ouyang, and J. Ning, “An artificial bee colony approach for clustering,” Expert Systems with Applications, vol. 37, no. 7, pp. 4761–4767, 2010.

D. Birant and A. Kut, “St-dbscan: An algorithm for clustering spatial–temporal data,” Data & Knowledge Engineering, vol. 60, no. 1, pp. 208–221, 2007.

D. B. Fogel, Evolutionary computation: toward a new philosophy of machine intelligence. John Wiley & Sons, vol. 1, 2006.

D. Bratton and J. Kennedy, “Defining a standard for particle swarm optimization,” in 2007 IEEE swarm intelligence symposium. IEEE, pp. 120–127, 2007.

D. Dua and E. Karra Taniskidou, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2017.

D. E. Goldberg, “Genetic algorithms and rule learning in dynamic system control,” Grefenstette, vol. 876, pp. 8–15, 1985.

D. E. Golberg, “Genetic algorithms in search, optimization, and machine learning,” Addion wesley, vol. 1989, p. 102, 1989.

D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” Machine learning, vol. 3(2), pp. 95-99, 1988.

D. H. Byun, “The AHP approach for selecting an automobile purchase model,” Information & Management, vol. 38, no. 5, pp. 289-297, 2001.

D. Jannach, Z. Karakaya, and F. Gedikli Accuracy, “Improvements for Multi-criteria Recommender Systems,” Proceedings of the 13th ACM Conference on Electronic Commerce, ACM, pp. 674-689, 2012.

D. Karaboga, “An idea based on honey bee swarm for numerical optimization,” Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Tech. Rep., 2005.

D. Karaboga and B. Akay, “A comparative study of artificial bee colony algorithm,” Applied mathematics and computation, vol. 214, no. 1, pp. 108–132, 2009.

D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” Journal of global optimization, vol. 39, no. 3, pp. 459–471, 2007.

D. Karaboga and B. Basturk, “On the performance of artificial bee colony (ABC) algorithm,” Applied soft computing, vol. 8, no. 1, pp. 687–697, 2008.

D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, “A comprehensive survey: artificial bee colony (ABC) algorithm and applications,” Artificial Intelligence Review, vol. 42, no. 1, pp. 21–57, 2014.

D. Karaboga and C. Ozturk, “A novel clustering approach: Artificial bee colony (abc) algorithm,” Applied soft computing, vol. 11, no. 1, pp. 652–657, 2011.

D. L. Davies and D. W. Bouldin, “A cluster separation measure,” IEEE transactions on pattern analysis and machine intelligence, no. 2, pp. 224–227, 1979.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, “e1071: Misc functions of the department of statistics (e1071), tu wien. r package version 1.6-3,” Retrieved from, 2014.

D. Martin, C. Fowlkes, D. Tal, and J. Malik, “The berkeley segmentation dataset and benchmark,” University of California, Berkeley, http://www. cs. berkeley. edu/projects/vision/grouping/segbench, 2007.

D. R. Liu, and Y. Y. Shih, “Integrating AHP and data mining for product recommendation based on customer lifetime value,” Information & Management, vol. 42, no. 3, pp. 387-400, 2005.

D. R. Liu and Y. Y. Shih, “Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences,” Journal of Systems and Software, vol.77, no. 2, pp. 181-191, 2005.

D. Van der Merwe and A. P. Engelbrecht, “Data clustering using particle swarm optimization,” in Evolutionary Computation, 2003. CEC’03. The 2003 Congress on, vol. 1. IEEE, pp. 215–220, 2003.

E. Portmann, “A fuzzy grassroots ontology for improving social semantic web search,” Proceedings of 6th international summer school on aggregation operators, Benevento, 2011.

E. Peis, J. M. M. del Castillo, and J. A. Delgado-López, “Semantic Recommender Systems. Analysis Of The State Of The Topic.” Hipertext. net 6, vol. 1-5, 2008.

E. R. Hruschka, R. J. Campello, A. A. Freitas et al., “A survey of evolutionary algorithms for clustering,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 39, no. 2, pp. 133–155, 2009.

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

F. Van den Bergh and A. P. Engelbrecht, “A cooperative approach to particle swarm opti- mization,” IEEE transactions on evolutionary computation, vol. 8, no. 3, pp. 225–239, 2004.

G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, and L. Rocha, “G-dbscan: A GPU accelerated algorithm for density-based clustering,” Procedia Computer Science, vol. 18, pp. 369–378, 2013.

G. Adomavicius, N. Manouselis, Y. O. Kwon, “Multi-criteria recommender systems,” Recommender systems handbook, Springer US, pp. 769-803, 2011.

G. Adomavicius, and K. YoungOk, “New Recommendation Techniques for Multicriteria Rating Systems,” Intelligent Systems, IEEE, vol. 22, no. 3, pp. 48-55, 2007.

García-Gonzalo, E., and J. L. Fernández-Martínez, “A brief historical review of particle swarm optimization (pso),” Journal of Bioinformatics and Intelligent Control, vol. 1, no. 1, pp. 3–16, 2012.

G. Zhu and S. Kwong, “Gbest-guided artificial bee colony algorithm for numerical function optimization,” Applied Mathematics and Computation, vol. 217, no. 7, pp. 3166–3173, 2010.

H. Bengtsson, R.matlab: Read and Write MAT Files and Call MATLAB from Within R. R package version 3.6.1. URL: https://CRAN.R-project.org/package=R.matlab, 2016.

H. Jiang, J. Li, S. Yi, X. Wang, and X. Hu, “A new hybrid method based on partitioning-based dbscan and ant clustering,” Expert Systems with Applications, vol. 38, no. 8, pp. 9373–9381, 2011.

J. B. Schafer, J.A. Konstan, and J. Riedl, “E-commerce recommendation applications”, in Applications of Data Mining to Electronic Commerce. Springer. pp. 115-153, 2001.

J. Czekanowski, Zur differentialdiagnose der Neandertalgruppe. Friedr. Vieweg & Sohn, 1909.

J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques. Elsevier, 2011.
J. Kennedy, “Particle swarm optimization,” in Encyclopedia of machine learning. Springer, 1995, pp. 760–766.

J. Handl and J. Knowles, “Improvements to the scalability of multi-objective clustering,” in 2005 IEEE Congress on Evolutionary Computation, IEEE, vol. 3., pp. 2372–2379, 2005.

J. H. Ward Jr, “Hierarchical grouping to optimize an objective function,” Journal of the American statistical association, vol. 58, no. 301, pp. 236-244, 1963.

J. H. Holland, “Outline for a logical theory of adaptive systems,” Journal of the Association for Computing Machinery, 3, 297-314, 1962.

J. H. Holland, Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press, 1975.

J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. Induction: Processes of inference, learning, and discovery. Cambridge, MA: MIT Press, 1986.

J. J. Grefenstette, Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Pittsburgh, PA: Lawrence Erlbaum, 1985.

J. J. Grefenstette, “Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms,” Cambridge, MA: Lawrence Erlbaum, 1987.

J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang and L. Shao, “Real-Time Superpixel Segmentation by DBSCAN Clustering Algorithm.,” IEEE Transactions on Image Processing, vol. 25(12), pp. 5933-5942, 2016.

J. Tvrdík and I. Křivý, “Hybrid differential evolution algorithm for optimal clustering,” Applied Soft Computing, vol. 35, pp. 502–512, 2015.

J. MacQueen, “Some methods for classification and analysis of multivariate observations,” Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1(14), pp. 281-297, 1967.

K. E. Parsopoulos and M. N. Vrahatis, “Particle swarm optimization method for constrained optimization problems,” Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies, vol. 76, no. 1, pp. 214–220, 2002.

K. K. F. Yuen, “Cognitive network process with fuzzy soft computing technique for collective decision aiding,” The Hong Kong Polytechnic University, vol. Ph.D. thesis, 2009.

K. K. F. Yuen, “Pairwise opposite matrix and its cognitive prioritization operators: Comparisons with pairwise reciprocal matrix and analytic prioritization operators,” Journal of the Operational Research Society, vol. 63, no. 3, pp. 322-338, 2012.

K. K. F. Yuen(1), “The Primitive Cognitive Network Process in healthcare and medical decision making: Comparisons with the Analytic Hierarchy Process,” Applied Soft Computing, vol. 14, Part A, no. 0, pp. 109-119, 2014.

K. K. F. Yuen(2), “Fuzzy Cognitive Network Process: Comparisons With Fuzzy Analytic Hierarchy Process in New Product Development Strategy,” Fuzzy Systems, IEEE Transactions on, vol. 22, no. 3, pp. 597-610, 2014.

K. Lakiotaki, N. F. Matsatsinis, and A. Tsoukia, “Multicriteria User Modeling in Recommender Systems,” Intelligent Systems, IEEE, vol. 26, no. 2, pp. 64-76, 2011.

K. Price, R.M. Storn, and J.A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006

K. S. Al-Sultan, “A tabu search approach to the clustering problem,” Pattern Recognition, vol. 28, no. 9, pp. 1443–1451, 1995.

K. S. Chuang, H. L. Tzeng, S. Chen, J. Wu and T. J. Chen, “Fuzzy c-means clustering with spatial information for image segmentation,” computerized medical imaging and graphics, vol. 30(1), pp. 9-15, 2006.

L. Specia and E. Motta, “Integrating folksonomies with the semantic web. The semantic web: research and applications,” Springer Berlin Heidelberg, pp. 624-639, 2007.

M. Balabanović, and Y. Shoham, “Fab: content-based, collaborative recommendation,” Communications of the ACM,. vol. 40, no. 3, pp. 66-72, 1997.

M. Clerc, “Standard particle swarm optimisation,” 2012.

M. Clerc, “A method to improve standard PSO,” 2009.

M. C. Naldi, A. C. de Carvalho, R. J. G. B. Campell et al., “Genetic clustering for data mining,” in Soft computing for knowledge discovery and data mining. Springer, pp. 113–132, 2008.

M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 26, no. 1, pp. 29–41, 1996.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in Kdd, vol. 96, no. 34, pp. 226–231, 1996.

M. E. Celebi, Y. A. Aslandogan and P. R. Bergstresser, “Mining biomedical images with density-based clustering,” In Information Technology: Coding and Computing International Conference on IEEE, vol. 1, pp. 163-168, 2005.

M. Mitchell, An introduction to genetic algorithms[M]. MIT press, 1998.

M. M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary, “A new scalable parallel dbscan algorithm using the disjoint-set data structure,” Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press, p. 62, 2012.

M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, “Standard particle swarm optimisation 2011 at cec-2013: A baseline for future PSO improvements,” in 2013 IEEE Congress on Evolutionary Computation. IEEE, pp. 2337–2344, 2013.

M. Zambrano-Bigiarini and R. Rojas, “A model-independent particle swarm optimization software for model calibration,” Environmental Modelling & Software, vol. 43, pp. 5–25, 2013.

M. Zambrano-Bigiarini and R. Rojas, “hydropso: Particle swarm optimisation, with focus on environmental models,” URL http://www. rforge. net/hydroPSO, http://cran. r-project. org/web/packages/hydroPSO. R package version 0.3-3, 2013.

P. Civicioglu and E. Besdok, “A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms,” Artificial Intelligence Review, vol. 39, no. 4, pp. 315–346, 2013.

P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” International journal of computer vision, vol.59(2), pp. 167-181, 2004.

P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” Journal of computational and applied mathematics, vol. 20, pp. 53–65, 1987.

P. Viswanath and V. S. Babu, “Rough-dbscan: A fast hybrid density based clustering method for large data sets,” Pattern Recognition Letters, vol. 30, no. 16, pp. 1477–1488, 2009.

P. Viswanath and R. Pinkesh, “l-dbscan: A fast hybrid density based clustering method,” in 18th International Conference on Pattern Recognition (ICPR’06), IEEE, vol. 1, pp. 912–915, 2006.

Q. Ye, W. Gao and W. Zeng, “Color image segmentation using density-based clustering,” In Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on IEEE, vol. 2, pp. II-401, 2003.

R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331-370, 2002.

R. Core Team and C. Worldwide, “The R stats package. R Foundation for Statistical Computing,” Vienna, Austria: Available from: http://www. R-project.org, 2002.

R Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL:https://www.R-project.org/, 2016

R. Eberhart and Y. Shi, “Comparison between genetic algorithms and particle swarm optimization,” in International Conference on Evolutionary Programming. Springer, pp. 611–616, 1998.

R. Eberhart, and Y. Shi, “Particle swarm optimization: developments, applications and resources,” in Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 81–86, 2001.

R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” Swarm Intelligence, vol. 1(1), pp. 33–57, 2007.

R. Storn and K. Price, “Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces,” Journal of global optimization, vol. 11, no. 4, pp. 341–359, 1997.

R. Mendes, “Population topologies and their influence in particle swarm performance,” Ph.D. dissertation, Citeseer, 2004.

R. Xu, J. Xu, and D. C. Wunsch, “A comparison study of validity indices on swarm-intelligence- based clustering,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernet- ics), vol. 42, no. 4, pp. 1243–1256, 2012.

S. Alam, G. Dobbie, P. Riddle, and M. A. Naeem, “Particle swarm optimization based hier- archical agglomerative clustering,” in Web Intelligence and Intelligent Agent Technology (WI- IAT), 2010 IEEE/WIC/ACM International Conference on, vol. 2. IEEE, pp. 64–68, 2010.

S. Alam, G. Dobbie, Y. S. Koh, P. Riddle, and S. U. Rehman, “Research on particle swarm optimization based clustering: a systematic review of literature and techniques,” Swarm and Evolutionary Computation, vol. 17, pp. 1–13, 2014.

S. Das, A. Abraham, and A. Konar, “Automatic clustering using an improved differential evolution algorithm,” IEEE Transactions on systems, man, and cybernetics Part A: Systems and Humans, vol. 38, no. 1, pp. 218–237, 2008.

S. Das, P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,”. IEEE transactions on evolutionary computation, vol. 15(1), pp. 4-31, 2011.

S. Das, S. S. Mullick, and P. N. Suganthan, “Recent advances in differential evolution– an updated survey,” Swarm and Evolutionary Computation, vol. 27, pp. 1–30, 2016.

S. Kisilevich, F. Mansmann, and D. Keim, “P-dbscan: a density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos,” in Proceedings of the 1st international conference and exhibition on computing for geospatial research & application. ACM, p. 38, 2010.

S. Niwa and S. Honiden, “Web page recommender system based on folksonomy mining for ITNG'06 submissions,” Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on. IEEE, pp. 388-393, 2006.

S. Paterlini and T. Krink, “High performance clustering with differential evolution,” in Evolu- tionary Computation, 2004. CEC2004. Congress on, vol. 2. IEEE, 2004.

S. Rana, S. Jasola, and R. Kumar, “A review on particle swarm optimization algorithms and their applications to data clustering,” Artificial Intelligence Review, vol. 35, no. 3, pp. 211–222, 2011.

S. Urbanek, “jpeg: Read and write JPEG images,” R package version 0.1–8, 2014.

S. Z. Selim and K. Alsultan, “A simulated annealing algorithm for the clustering problem,” Pattern recognition, vol. 24, no. 10, pp. 1003–1008, 1991.

T. N. Tran, K. Drab, and M. Daszykowski, “Revised dbscan algorithm to cluster data with dense adjacent clusters,” Chemometrics and Intelligent Laboratory Systems, vol. 120, pp. 92– 96, 2013.

T. Niknam and B. Amiri, “An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis,” Applied Soft Computing, vol. 10, no. 1, pp. 183–197, 2010.

T. V. Wal, Folksonomy coinage and definition, 2007.

W. Zou, Y. Zhu, H. Chen, and X. Sui, “A clustering approach using cooperative artificial bee colony algorithm,” Discrete Dynamics in Nature and Society, vol. 2010, 2010.

X. Cui, T. E. Potok, and P. Palathingal, “Document clustering using particle swarm opti- mization,” in Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, pp. 185–191, 2005.

X. Yan, Y. Zhu, W. Zou, and L. Wang, “A new approach for data clustering using hybrid artificial bee colony algorithm,” Neurocomputing, vol. 97, pp. 241–250, 2012.

X. Cui, T. E. Potok, and P. Palathingal, “Document clustering using particle swarm opti- mization,” in Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, pp. 185–191, 2005.

Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, “Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce,” in Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. IEEE, pp. 473–480, 2011.

Y. Lee, and K. A.Kozar, “Investigating the effect of website quality on e-business success: An analytic hierarchy process (AHP) approach,” Decision support systems, vol.42. no.3, pp. 1383-1401, 2006.

Y.-T. Kao, E. Zahara, and I.-W. Kao, “A hybridized approach to data clustering,” Expert Systems with Applications, vol. 34, no. 3, pp. 1754–1762, 2008.

Y. Shi, “Particle swarm optimization: developments, applications and resources,” in evolutionary computation, 2001. Proceedings of the 2001 Congress on, vol. 1. IEEE, pp. 81–86, 2001.

Y. Shi, and R. Eberhart, “Empirical study of particle swarm optimization,” in Proceedings of the 1999 Congress on Evolutionary Computation- CEC99 (Cat. No. 99TH8406), p. 1945, 1999.

Y. Wang and M. M. Tseng, “Customized products recommendation based on probabilistic relevance model,” Journal of Intelligent Manufacturing, vol.24, no.5, pp. 951-960, 2013.

Z. Güngör, A. Ünler, “K-harmonic means data clustering with simulated annealing heuristic,” Applied mathematics and computation, vol. 184, no. 2, pp. 199–209, 2007.