Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Clustering Analysis using Swarm Intelligence

Settore/i scientifico disciplinari di afferenza: ING-INF/05

Mohammad Reza Farmani

Coordinator: Fabio Roli

Tutor: Giuliano Armano

XXVIII Cycle

Esame finale anno accademico 2014-2015

Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Clustering Analysis using Swarm Intelligence

Settore/i scientifico disciplinari di afferenza: ING-INF/05

Mohammad Reza Farmani

Coordinator: Fabio Roli

Tutor: Giuliano Armano

XXVIII Cycle

Esame finale anno accademico 2014-2015

I dedicate this dissertation to my lovely parents

Acknowledgements

This dissertation would not have been possible without the help of several people who, in one way or another, have extended their valuable assistance in completion of my studies.

First of all, I would like to express my sincere thanks to my supervisor, Prof. Giuliano Armano, for giving me the opportunity to be a part of his research group. I am especially thankful for his continuous support of my study and research, and also for his patience, motivation, enthusiasm, and immense knowledge.

I would like to give a great thank my friends who have supported me throughout the process. My friends in IASC group: Alessandro Giuliani, Andrea Manconi, Emanuele Tamponi, Amir Mohammad Amiri, Maria Giovanna, Matteo Baire, and Emanuele Manca. I will never forget the wonderful environment that we had from the very early days.

Last but not least, I thank to my close friends Simone Porru, Mansour Ahmadi, Mehran Zareh, Farideh Tavazoee, Matteo Orru, and Bahram Lavi. I need to thank so many other friends that for lack of space I cannot list here.

Abstract

This thesis is concerned with the application of the swarm intelligence methods in clustering analysis of datasets. The main objectives of the thesis are

- Take the advantage of a novel evolutionary algorithm, called artificial bee colony, to improve the capability of K-means in finding global optimum clusters in nonlinear partitional clustering problems.
- Consider partitional clustering as an optimization problem and an improved ant-based algorithm, named Opposition-Based API (after the name of Pachycondyla APIcalis ants), to automatic grouping of large unlabeled datasets.
- Define partitional clustering as a multiobjective optimization problem. The aim is to obtain well-separated, connected, and compact clusters and for this purpose, two objective functions have been defined based on the concepts of data connectivity and cohesion. These functions are the core of an efficient multiobjective particle swarm optimization algorithm, which has been devised for and applied to automatic grouping of large unlabeled datasets.

For that purpose, this thesis is divided is five main parts:

- The first part, including Chapter 1, aims at introducing state of the art of swarm intelligence based clustering methods.
- The second part, including Chapter 2, consists in clustering analysis with combination of artificial bee colony algorithm and K-means technique.
- The third part, including Chapter 3, consists in a presentation of clustering analysis using opposition-based API algorithm.
- The fourth part, including Chapter 4, consists in multiobjective clustering analysis using particle swarm optimization.
- Finally, the fifth part, including Chapter 5, concludes the thesis and addresses the future directions and the open issues of this research.

Contents

List of Figures

1

Introduction and structure of the thesis

1.1 Clustering and Swarm Intelligence

Research investigations in different organizations have recently shown that huge amount of data are being stored and collected in databases and this large amount of stored data continues to grow fast. The main reasons of the dramatical increment in this data volume can be listed as explosive growth in the generation of electronic information, rapid advancement in computer network, improvement in computer performance, and technology advances in data acquisition. Valuable knowledge which is hidden in this large amount of stored data should be revealed to improve the decision-making process in organizations. Therefore, a field called knowledge discovery and data mining in databases has been emerged due to such large databases Han et al. (2011). Extracting or mining knowledge from large amounts of data is referred by data mining approaches. These methodologies apply data analysis techniques to discover previously unknown, valid patterns and relationships in large datasets. Data mining analysis includes a number of technical approaches such as classification, data summarization, finding dependency networks, clustering, regression, and detecting anomalies. The process of grouping data into classes or clusters such that the data in each cluster share a high degree of similarity while being very dissimilar to data from

other clusters is called data clustering. Attribute values which describe the objects are used for assessing the dissimilarities among clusters. Different areas such as data mining, machine learning, biology, and statistics include the roots of data clustering. Generally speaking, hierarchical and partitional clustering are the two main categories of clustering methods Kao et al. (2008); Alsabti et al. (1997); Nguyen and Cios (2008); Niknam et al. (2008a,b, 2009); Chahine (2012); Fathian et al. (2007); Krishna and Murty (1999); Maulik and Bandyopadhyay (2000); Shelokar et al. (2004). Each category contains various algorithms for finding the clusters. Hierarchical clustering results in a tree which presents a sequence of clustering while each cluster is a group of dataset Leung et al. (2000). However hierarchical clustering algorithms do not need the number of clusters and are independent from the initial conditions, but they are static. That means data points that belong to a cluster cannot be assigned to other clusters in the process of clustering. Moreover, due to lack of information about the global shape or size of the clusters, these algorithms may not be successful to separate overlapping clusters Jain et al. (1999). On the other hand, partitional clustering decomposes a dataset into a set of disjoint clusters. Many partitional clustering algorithms try to minimize some measure of dissimilarity in the samples within each cluster while maximizing the dissimilarity of different clusters. The drawbacks of hierarchical algorithms can be considered as the advantages of partitional algorithms, and vice versa Frigui and Krishnapuram (1999).

Swarm Intelligence (SI) is an innovative artificial intelligence category inspired by intelligent behaviors of insect or animal groups in nature, such as ant colonies, bird flocks, bee colonies, bacterial swarms, and so on. Over the recent years, the SI methods like ant-based clustering algorithms were successful dealing with clustering problems. So, the research community has given them special attention recently. This attention is mainly because ant-based approaches are particularly proper to perform exploratory analysis, and also because there is still a lot of investigation to perform on this field. Different features such as the performance and convergence capability, stability, robustness, etc allow us to apply these methods in real world applications. Application of the SI methods in partitional clustering has been investigated by re-

searchers all over the globe Bandyopadhyay and Maulik (2002); Das et al. (2008); Selim and Alsultan (1991); Omran et al. (2005). Kao et al. have introduced a hybrid method based on combining k-means, Nelder-Mead simplex, and Particle Swarm Optimization (PSO) for cluster analysis Kao et al. (2008). A hybrid algorithm according to the combination of Genetic Algorithm (GA), k-means, and logarithmic regression expectation maximization has been presented by Cao et al.Nguyen and Cios (2008). Zalik has proposed the performance of correct clustering without pre-assigning the exact number of clusters within k-means Alsabti et al. (1997). Krishna and Murty have shown an approach called genetic k-means algorithm for clustering analysis Krishna and Murty (1999). A GA based method, which contains a basic mutation operator specific to clustering called distance-based mutation, has been introduced by Mualik Maulik and Bandyopadhyay (2000). This method is used to solve the clustering problem on real life datasets to evaluate the performance. An algorithm named HBMO has been proposed by Fathian et al. to solve clustering problems Fathian et al. (2007). A GA that exchanges neighboring centers for k-means clustering has demonstrated by Laszlo and Mukherjee Chahine (2012). Shelokar et al. have introduced an evolutionary algorithm based on Ant Colony Optimization (ACO) for clustering problems Shelokar et al. (2004). A combination of two evolutionary algorithms, ACO and Simulated Annealing (SA), has been proposed by Niknam et al. to solve clustering problems in Niknam et al. (2008a,b). They also have presented a hybrid evolutionary algorithm based on PSO and SA to find optimal cluster centers Niknam et al. (2009).

The Artificial Bee Colony (ABC) algorithm is developed through simulation of intelligent foraging behavior of honey bees, and has been found to be robust in solving continuous nonlinear optimization problems. Since the ABC algorithm is simple in concept, easy to implement, and has fewer control parameters, it has attracted the attention of researchers and been widely used in solving many numerical optimization Karaboga and Basturk (2007a,b) and engineering optimization problems Baykasoglu et al. (2007); Tasgetiren et al. (2011). As mentioned earlier, the main drawback of the k-means algorithm is that the result is sensitive to the selection of the initial cluster cancroids and may converge to the local optima Selim and Ismail (1984).

Therefore, the initial selection of the k-means cancroids affects the main processing of the k-means and the partition result of the dataset as well. In the current study, the ABC algorithm is utilized to find the optimal initial cluster cancroids for the k-means algorithm. Contrary to the localized searching of the k-means algorithm, the ABC performs a globalized search in the entire solution space.

Ants have an incredible optimizing capacity due to their ability to communicate indirectly by means of pheromone deposits Bonabeau et al. (1999); Dorigo et al. (1999, 2006). In the most research works, clustering analysis is considered as an optimization problem and solved by using the different types of ACO and ant-based algorithms. The idea is to make a group of ants to explore the search space of the optimization problems and find the best candidates of solutions. These candidates create clusters of the datasets and are selected according to a fitness function, which evaluate their quality with respect to the optimization problem. The API algorithm is inspired by a model of the foraging behavior of a population of primitive ants named P. apicalis (after apicalis in Pachycondyla apicalis) Monmarché et al. (2000). It is demonstrated in Monmarché et al. (2000); Ciornei and Kyriakides (2012); Aupetit et al. (2005) that API can be applied to continuous optimization problems and achieved robust performance for all the test problems. In order to improve the convergence of the ant-based clustering algorithm, a combination of the popular k-means algorithm and the stochastic and exploratory behavior of clustering ants is proposed in Monmarché et al. (1999a). This method, called AntClass algorithm, is mainly based on work of Lumer and Faieta Lumer and Faieta (1994). An ant system and ACO, which is based on the parameterized probabilistic model of the pheromone, is presented by Dorigo Dorigo et al. (1999).Monmarche et al. applies explorative and stochastic principles from the ACO meta-heuristic combined with deterministic and heuristic principles of k-means Monmarché et al. (1999b) . A novel strategy called ACLUSTER is developed in Ramos and Merelo (2004) to deal with unsupervised clustering as well as data retrieval problems. This algorithm was applied to textual document clustering and the authors proposed the use of bio-inspired spatial transition probabilities to avoid exploring non-interesting regions. Laborche et al. proposed a clustering algorithm,

called ANTCLUST Labroche et al. (2002). This algorithm is based on a modeling of the chemical recognition system of ants which allows a colonial odor construction used for determining the ants' nest membership. In this way, ants can discriminate between nest mates and intruders. A hybridization of the classical Fuzzy C-Means (FCM) algorithm with the ant systems is presented in Kanade and Hall (2003) to determine the number of clusters in a given dataset automatically. In this algorithm, the ant-based clustering is firstly refined using the FCM algorithm. Handl et al. argue that although many of the ant-based clustering algorithms have resulted promisingly so far, there is a lack of knowledge about the actual performance of many of them Handl et al. (2003). In their method, they applied the agglomerative hierarchical clustering algorithm to the positions of the data items on the grid to overcome those limitations. It is also shown that the developed method performs well comparing with the other algorithms such as k-means and agglomerative average link Handl and Meyer (2002). Two other ant-based clustering algorithms, named Ant-Clust and AntTree, are presented in Labroche et al. (2003); Azzag et al. (2003), respectively. In Ant-Clust, the ants proceed according to chemical properties and odors to recognize themselves as similar or not. Both algorithms are applied to unsupervised learning problems. Tsai et al. proposed a novel clustering algorithm called ACO with Different Favor (ACODF) Tsai et al. (2004). This algorithm applies a direct adaptation of the ACO for solving clustering problems. It is shown that ACODF performs better than some other combined meta-heuristic methods such as genetic k-means. An ACO methodology is described for optimally clustering N objects into K clusters in Shelokar et al. (2004). The proposed algorithm is tested on several simulated and real datasets and the obtained results show its effectiveness in terms of quality comparing to other heuristic methods. Hartmann added a neural network to each ant in his proposed algorithm which enables the ants to take the objects of their vicinity as input, and return the move action, the pick up or drop action, as outputs Hartmann (2005). In this way, the ants are trained to make annular clusters while one cluster would be encircling another. An approach called AntPart is introduced in Admane et al. (2006) to solve exclusive unsupervised classification problems. A particular

species of ants called Pachycondyla Apicalis is modeled to develop the model and its performance are compared with other methods such as AntClass, AntTree, and AntClust. To avoid the search of optimal clusters being trapped in local optimums, Huang et al. proposed a method named Chaotic Ant Clustering Algorithm (CACAS) Huang et al. (2007). In this method, chaotic perturbation is used to enables the ant to escape from local optimas. An advanced clustering algorithm called ant colony ISODATA is proposed in for applying in real time computer simulation Wang et al. (2007). An efficient and fast algorithm is proposed by Tao et al. Tao et al. (2007). This method is applied in aggregation analysis and obtained very promising experimental results. Boryczka used a modified version of the short-term memory Boryczka (2008), introduced in Lumer and Faieta (1994), and improved its convergence. This modified clustering algorithm is called ACA and applied in a knowledge discovery context. A new algorithm for clustering datasets is proposed in Ghosh et al. (2008) which is mainly based on the ants' aggregation pheromone property. This method is used to form homogeneous groups of data. A new clustering strategy is proposed in Sadeghi et al. (2008) which used artificial ants trying to do clustering by inserting and removing operations. In this work, clustering is performed with the aim of groups of ants which are as many as the number of clusters. It is shown that the algorithm outperforms k-means and an another ant clustering approach. Ant clustering algorithm is also used in Chen and Mo (2009) to improve k-means and optimize the rule of ant clustering algorithm. Weili introduced an algorithm named Improved Entropy-based Ant Clustering (IEAC) Weili (2009). In this algorithm, the information entropy is utilized to model behaviors of agents. The entropy function resulted in better quality in the obtained clusters.

Similar to other SI methods, PSO is based on a phenomenon occurring in nature –the social behavior of bird flocking or fish schooling Poli et al. (2007). Two PSO-based clustering methods are proposed in Rana et al. (2011). In the first, PSO is used to find the centroid of a user specified number of clusters. In the second, K-means is used to extend the algorithm to seed the initial swarm. It is shown that this algorithm has better convergence accuracy, compared to the classical version of K-means.

Ghali et al. propose a clustering method called exponential particle swarm optimization (EPSO) Ghali et al. (2009). Instead of using a linear inertia weight, exponential inertia weight is applied in EPSO and they show that EPSO has better performance in data clustering than PSO in terms of quantization error and accuracy. A dynamic PSO based clustering algorithm (DCPSO) is proposed in Omran et al. (2006) for tackling color image segmentation. Binary PSO is used in this algorithm to automatically determine the optimum number of clusters and simultaneously cluster the dataset. A dynamic binary PSO-based multiobjective clustering approach (DCBMPSO) is proposed in Latiff et al. (2008) to determine the number of clusters in the wireless sensor network problem. DCBMPSO finds the optimal number of clusters in the network and minimizes the total network energy dissipation simultaneously. In this method, two clustering metrics named total network energy consumption and intra-cluster distance are defined to select the best set of network cluster heads. Janson et al. Janson and Merkle (2005) introduce ClustMPSO as a multiobjective PSO-based clustering algorithm and apply it to predict the three dimensional structure in a molecule docking problem. In their algorithm, all particles are divided into several subswarms and new strategies are proposed for updating the personal and global best particles. It is presented that ClustMPSO outperforms another well-known clustering method dealing with the docking problem. A multiobjective PSO and simulated annealing clustering algorithm (MOPSOSA) is proposed in Abubaker et al. (2015). This method simultaneously optimizes three different objective functions as the cluster validity indices to find the proper values of the number of clusters and cluster of the datasets. Euclidean, point symmetry, and short distances are considered as the validity indices in MOPSOSA. The method obtains more promising results in comparison with some other conventional clustering algorithms. Several other PSO-based clustering algorithms are introduced in the clustering literature so far and for a comprehensive review about PSO-based clustering one can refer to Sarkar et al. (2013). However, they mostly consider a single function as the objective of the clustering problem and the recent works which use the term 'multiobjective' do not apply the concept of pareto optimal solutions Kasprzak and Lewis (2001).

1.2 Organization of the Dissertation

The dissertation is organized as follows:

In Chapter 2, the ABC algorithm, one of the new SI methods, is used to find the optimal initial cluster cancroids for the K-means algorithm. Contrary to the localized searching of the K-means algorithm, the ABC performs a globalized search in the entire solution space. The proposed algorithm, named ABCk, has developed a combined algorithm for solving the clustering problem. The algorithm has been implemented and tested on several well known real datasets and preliminary computational experience showed very encouraging results.

In Chapter 3, partitional clustering is considered as an optimization problem and an improved ant-based algorithm, named Opposition-Based API, is applied to automatic grouping of large unlabeled datasets. The proposed algorithm employs Opposition-Based Learning (OBL) for ants' hunting sites generation phase in API. Experimental results are compared with the classical API clustering algorithm and three other recently evolutionary-based clustering techniques.

Chapter 4 is confined to the application of particle swarm optimization (PSO) algorithm to clustering. Similar to other swarm intelligence methods, PSO is based on a phenomenon occurring in nature - the social behavior of bird flocking or fish schooling. Several PSO-based clustering algorithms are introduced in the clustering literature so far and for a comprehensive review about PSO-based clustering. However, they mostly consider a single function as the objective of the clustering problem and the recent works which use the term âĂŹmultiobjectiveâĂŹ do not apply the concept of pareto optimal solutions. We introduced a multiobjective clustering particle swarm optimization (MCPSO) framework to obtain well-separated, connected, and compact clusters in any unlabeled datasets with different dimensions and cluster characteristics. MCPSO also aims to determine the optimal number of clusters, automatically. To achieve this objectives, two contradictory objective functions are defined based on the concepts of connectivity and cohesion and MCPSO is used to find a set of non-dominated clustering solutions as a pareto front. Finally, we utilize

a simple decision maker to select the best solution along the obtained pareto solutions. A comprehensive comparison of the results of MCPSO with four conventional clustering approaches is investigated.

Chapter 5 summarizes the contributions and explains the open issues.

2

Clustering Analysis with Combination of Artificial Bee Colony Algorithm and K-means Technique

Among different proposed clustering methods, K-means clustering algorithm is an efficient clustering technique to cluster datasets, but this method highly depends on the initial state and usually converges to local optimum solution. This chapter takes the advantage of a novel evolutionary algorithm, called artificial bee colony, to improve the capability of K-means in finding global optimum clusters in nonlinear partitional clustering problems. Artificial bee colony algorithm is inspired by the intelligent foraging behavior of honey bees. The performance of the proposed algorithm is evaluated through several benchmark datasets. The simulation results show that the combination of artificial bee colony algorithm and K-means technique improves the performance K-means to find global optimums.

2.1 Introduction

In this chapter, a partitional clustering method is considered. One of the most popular partitional clustering methods, which is developed about three decades ago, is k-means algorithm. This algorithm is defined over continuous data and used in variety

of domains. However, as k-means needs initial partitions to start its process, better results are given only when the initial partitions are close to the final solution. In other words, the results of this technique highly depend on the initial sate and converge to local optimal solution. The remainder of this paper is organized as follows: Section 2.2 provides a general overview of the K-means. In section 2.3, the ABC algorithm is introduced. The combination of ABC and K-means for clustering problems is described in section 2.4. Section 2.5 provides the experimental results for comparing the performance of the proposed method with the simple K-means algorithm. The discussion of the experiments' results is also presented in this section. The conclusion is in section 2.6.

2.2 K-means Clustering Algorithm

The K-means is a simple algorithm which is proposed based on the firm foundation of analysis of variances. In this method, a group of data vectors will be clustered into a predefined number of clusters. The K-means starts with randomly initial cluster centroids and keeps reassigning the data objects in the dataset to cluster centroids based on the similarity between the data objects and the cluster centroids. The reassignment procedure will stops when a convergence criterion, such as the fixed iteration number, or no change in the cluster results after a certain number of iteration, is met. The K-means clustering process is described in the four following steps:

1- Create K cluster centroid vectors randomly to set an initial dataset partition.

2- Assign each document vector to the closest cluster centroids.

3- Recalculate the cluster centroid C_j by:

$$C_j = \frac{1}{n} \sum_{\forall d_j \in S_j} d_j \tag{2.1}$$

where d_j denotes the document vectors that belong to cluster S_j; C_j stand for the centroid vector; n_j is the number of document vectors that belong to cluster S_j.

4- Repeat step 2 and 3 until the convergence is achieved.

As the K-means' performance significantly depends on the selection of the initial cluster centroids, the algorithm may finally converge to the local optima. Therefore, the processing of the K-means is to search the local optimal solution in the vicinity of the initial solution and to refine partition result. The same initial cluster centroids in a dataset will always generate the same cluster results. However, if good initial clustering centroids can be obtained using any other techniques, the K-means would work well in refining the clustering centroids to find the optimal clustering centers.

2.3 Artificial Bee Colony Algorithm

Karaboga recently proposed a swarm intelligence algorithm inspired by the foraging behaviors of bee colonies Karaboga and Basturk (2007a). This algorithm then further developed by Karaboga, Baturk, and Akay et al. Karaboga and Basturk (2007b,b, 2008); Karaboga and Akay (2009). The Artificial Bee Colony (ABC) algorithm simulates the search space as the foraging environment and each point in the search space corresponds to a food source (solution) that the artificial bees could exploit. The fitness of the solution is represented as the nectar amount of a food source. In this algorithm, three kinds of bees exist in a bee colony: employed bees, onlooker bees, and scout bees. Employed bees exploit the specific food sources they have explored before and give the quality information of the food sources to the onlooker bees. Information about the food sources is received by onlooker bees, and then, a food source to exploit depending on the information of nectar quality will be chosen by them. The more nectar the food source contains, the larger probability the onlooker bees will choose it. A parameter, called "limit" controls the employed bees whose food should be abandoned. These food sources will become scout bees who search the whole environment randomly. In the ABC algorithm, half of the colony comprises of employed bees and the other half includes the onlooker bees. Each food source is exploited by only one employed bee. That is, the number of the employed bees or the onlooker bees is equal to the number of food sources. The details of the ABC

algorithm description are given below:

1- Initialization phase: All the vectors of the population of food sources $x_{i,j}$ are initialized by scout bees and control parameters are set. Where $i = 1, 2, ..., SN$, $j = 1, 2, ..., D$. SN is the number of food sources and equals to half of the colony size. D is the dimension of the problem, representing the number of parameters to be optimized. The following equation might be used for initialization purposes:

$$x_{i,j} = l_j + rand(0, 1)(u_j - l_j) \tag{2.2}$$

l_j and u_j are lower and upper bounds of the jth parameter. In this phase, the fitness of food sources (objective function values) will be evaluated and additional counters which store the numbers of trails of each bee are set to 0.

2- Employed bees phase: Employed bees search for new food sources having more nectar (better fitness value) within the neighborhood of the food sources $x_{i,j}$ in their memory. After finding a neighbor food source, they evaluate its fitness. Following equation is used to determine a neighbor food source $v_{i,j}$:

$$v_{i,j} = x_{i,j} + \phi(x_{i,j} - x_{k,j}) \tag{2.3}$$

Where k is a randomly selected food source different from i, and j is a randomly selected dimension. ϕ is a random number which uniformly distributed in range [-1,1]. As it can be seen, the new food source v determined by changing one dimension on x. If the new value in this dimension produced by this operation exceed its predetermined boundaries, it will set to be the boundaries. Then, the new food source is evaluated and a greedy selection is applied on the original food source and the new one. The better one will be kept in the memory. The trials counter of this food will be reset to zero if the food source is improved, otherwise, its value will be incremented by one.

3- Onlooker bees phase: Onlooker bees waiting in the hive receive the food source information from employed bees and then probabilistically choose their food sources depending on this information. By using the fitness values provided by employed bees,

the probability values of food sources will be calculated. An onlooker bee chooses a food source depending on its probability value. That is to say, there may be more than one onlooker bee choosing a same food source if that food source has a higher fitness. The probability is calculated according to:

$$p_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j} \tag{2.4}$$

After food sources have been probabilistically chosen for onlooker bees, each onlooker bee finds a new food source in its neighborhood using equation 2.3. Fitness values of these new food sources will be computed and, as in the employed bees phase, a greedy selection is applied between vi and xi. In other words, more onlooker bees will be recruited to richer food sources.

4- Scout bees phase: In this phase, if the value of trials counter of a food source is greater than "limit" parameter, the food source will be abandoned and the bee becomes a scout bee. According to equation 2.2, as in the initialization phase, a new food source will be produced randomly in the search space for each scout bee. And the trials counter of the bee will be reset to zero.

The tree employed, onlooker, and scout bees' phases will repeated until the termination criterion is met and best food source which shows the best optimal value will be selected as the final solution.

2.4 Combination of ABC and K-means

The K-means algorithm is a fast method due to its simple and small number of iterations. But the dependency of the algorithm on the initialization of centroids has been a major problem, and it usually gets stuck in local optimal. On the other hand, the ABC algorithm performs a global search in the entire solution space. If given time enough, the ABC can generate good and global results. A new combined algorithm is proposed here to use the merits of two algorithms. The new algorithm does not depend on the initial centroids and can avoid being trapped in a local optimal solution

to some extent as well.

In the proposed algorithm, each food source in the search environment represents a set of cluster centroids, that is, a food source represents one possible solution for clustering, and the position x_i is constructed as:

$$x_i = (C_{i1}, C_{i2}, ..., C_{iK}) \tag{2.5}$$

Where K is the number of clusters, C_{ij} is the jth cluster center vector of the ith food source. The procedure for the proposed algorithm can be summarized as follows:
Setp 1: Initialize the positions of food sources (a group of centroids) randomly and use the K-means algorithm to finish clustering task for all produced positions and compute the fitness value of each group of centroids.
Setp 2: Search for new food sources and update the place of food sources by employed bees. Apply the K-means algorithm and a greedy selection to evaluate new fitness values and compare them with the original ones. Better food sources will be delivered to onlooker bees.
Step 3: Calculate probability values of food sources and update their place according to the probability values by onlooker bees. Again, the K-means algorithm and a greedy selection will be applied to finish clustering, evaluate new fitness values and compare them with the original ones to update them.
Step 4: Check the trial counter of food sources and produce a new food source (set of centroids) in the search space for which exceed the "limit" parameter amount. To measure the overall clustering quality of each food source, a clustering criterion function should be defined. In this work, a simple Sum-of-Squares-Error (SSE) criterion is used as the clustering criterion function. SSE is the total sum of the squared distance between all samples and their cluster centers. The SSE criterion function for a group of clusters is given by:

$$E = \sum_{j=1}^{K} \sum_{z_j \in C_j} \parallel z_j - C_j \parallel^2 \tag{2.6}$$

Where z_j represents all patterns in cluster C_j. The goal in SSE clustering is to obtain

a partitioning of the data set, such that E is minimized. SSE criterion is valid for cluster sample dense as well as the small differences in the number of various clustering samples. However, if the shape and size of the cluster varies greatly, SSE rule may cause error clustering. In the proposed algorithm, SSE is used to calculate fitness of each food source.

2.5 Experimental Results and Discussion

The experimental results comparing the ABC+k-means with K-means algorithm are provided for three real-life datasets (Iris, Wine, and Contraceptive Method Choice (CMC)) which are described as follows:

Iris data (n = 150, d = 4, K = 3). These data with 150 random samples of flowers from the iris species setosa, versicolor, and virginica used by Fisher Fisher (1936). From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in cm.

Wine data (n = 178, d = 13, K = 3). These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars Forina et al. (1991). The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 instances with 13 numeric attributes in wine data set. All attributes are continuous and there is no missing attributes.

Contraceptive Method Choice (CMC) data (n = 1473, d = 10, K = 3). These data are a subset of the 1987 National Indonesia Contraceptive Prevalence Survey Lim et al. (2000). The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socioeconomic characteristics.

In this study, in the ABC algorithm, 10, 100, and 20 are determined as colony size, "limit" parameter, and number of iteration, respectively. The comparison of results for each dataset based on the best solution found in 100 distinct runs of each algorithm and the convergence processing time taken into attain the best solution. The

algorithms are implemented by using Matlab R2012b on a Intel Core i7, 2.4 GHz, 8 GB RAM computer. The comparison of results for each dataset based on the average solutions found in 100 distinct runs of each algorithm and the convergence processing time taken to attain the best solution. The quality of the respective clustering will also be compared, where the quality is measured by the following three criteria:

1- The SSE criterion as defined in equation 2.6. Clearly, the smaller the sum is, the higher the quality of clustering is.

2- The F-measure which uses the ideas of precision and recall from information retrieval Dalli (2003). Each class i (as given by the class labels of the used benchmark dataset) is regarded as the set of n_i items desired for a query; each cluster j (generated by the algorithm) is regarded as the set of n_j items retrieved for a query; n_{ij} gives the number of elements of cluster i within cluster j. For each class i and cluster j precision and recall are then defined as $p(i,j) = (n_{ij}/n_j)$ and $r(i,j) = (n_{ij}/n_i)$ and the corresponding value under the F-measure is $F(i,j) = ((b^2+1)p(i,j)r(i,j))/(b^2 p(i,j)r(i,j))$, where $b = 1$ is chosen here to obtain equal weighting for $p(i,j)$ and $r(i,j)$. The overall F-measure for the dataset of size n is given by:

$$F^* = \sum_i \frac{n_i}{n} MAX_i \{F(i,j)\} \tag{2.7}$$

Obviously, the bigger F-measure is, the higher the quality of clustering is.

3- Accuracy which is the percentage of correctly assigned instances on the real-life datasets.

The simulation results given in Tables 2.1-2.3 show that ABC+K-means is much more precise than K-means algorithm. In other words, it provides the optimum value and small standard deviation in compare to those of obtained by K-means. For instance, the results obtained on the Iris dataset show that ABC+K-means converges to the global optimum of 97.326 in all of runs while the average and standard deviation amounts of K-means are 102.728 and 10.518. Table 2.2 shows the results of algorithms on the Wine dataset. The average optimum values, which are obtained by ABC+K-means and K-means in all runs, are 16574.492 and 16890.162, respectively. As it

Table 2.1: Results obtained by the algorithms for 100 different runs on Iris data.

	Average	Std. Dev.	CPU times (sec)	Fmeasure	Accuracy
K-means	102.728	10.518	0.1	0.889	87.33%
ABC+K-means	97.326	0	23.7	0.892	89.25%

Table 2.2: Results obtained by the algorithms for 100 different runs on Wine data.

	Average	Std. Dev.	CPU times (sec)	Fmeasure	Accuracy
K-means	16890.162	718.65	0.2	0.715	70.22%
ABC+K-means	16574.492	188.13	31.1	0.715	71.47%

is presented, the ABC+K-means noticeably resulted in a smaller standard deviation value in comparison with the K-means. Table 2.3 provides the results of algorithms on the CMC dataset. As seen from the results, the ABC+K-means is far superior in term of the standard deviation value. Therefore, it is found that the ABC+K-means clustering algorithm is able to provide the same partition of data points in all the runs.

The simulation results of the tables also illustrate that the average of F-measure and the accuracy of the proposed algorithm is better than or equal to those obtained by the K-means algorithm on the all datasets. F-means is an indication that shows how the clusters are spatially well separated and the accuracy presents the ability of both algorithms to cluster the data into different partitions, correctly. To conclude, the simulation results in the tables demonstrate that the proposed algorithm converges to global optimum with a smaller standard deviation. However, in terms of computational costs, ABC+K-means significantly needs more evaluation times which is caused by the statistical behavior of all nature inspired optimization algorithms.

Table 2.3: Results obtained by the algorithms for 100 different runs on CMC data.

	Average	Std. Dev.	CPU times (sec)	Fmeasure	Accuracy
K-means	5864.22	51.32	0.4	0.402	39.71%
ABC+K-means	5711.27	3.41	121.1	0.400	42.31%

3

Clustering Analysis using Opposition-Based API Algorithm

In this chapter, partitional clustering is considered as an optimization problem and an improved ant-based algorithm, named Opposition-Based API (after the name of Pachycondyla APIcalis ants), is applied to automatic grouping of large unlabeled datasets. The proposed algorithm employs Opposition-Based Learning (OBL) for ants' hunting sites generation phase in API. Experimental results are compared with the classical API clustering algorithm and three other recently evolutionary-based clustering techniques. It is shown that the proposed algorithm can achieve the optimal number of clusters and, in most cases, outperforms the other methods on several benchmark datasets in terms of accuracy and convergence speed.

3.1 Introduction

Despite being powerful, the ant-based algorithms, including API, can remain trapped in local optimums. This situation can occur when a certain component is very desirable on its own, but leads to a sub-optimal solution when combined with other components. Considering the fact that implementations of the ant-based algorithms basically depend on positive reinforcement of good solutions, ants will tend to select similar paths after a certain number of iterations. Moreover, most of the reported

ant-based clustering methods need the number of clusters as an input parameter instead of determining it automatically on the run. Many practical situations show that it is impossible or very difficult to determine the appropriate number of groups in a previously unlabeled datasets. Also, if a dataset contains high-dimensional feature vectors, it is practically impossible to graph the data for determining its number of clusters. This paper contains two objectives. First, it attempts to show that application of the API algorithm in clustering problems, with a modification of using Opposition-Based Learning (OBL) in hunting sites generation, can achieve very promising results. The improvement is based on the idea of opposition numbers and attempt to increase the exploration efficiency of the solution space Tizhoosh (2006). The modification focuses on the initialization of sites' positions. In other words, the API algorithm is modified from its original form to a more intelligent approach to improve its exploration capability and increase its convergence speed. Second, it tries to determine the optimal number of clusters in any unlabeled dataset automatically. A comprehensive comparison of the proposed algorithm's results with classical API, and the reported results of three other automatic clustering methods including Genetic Algorithm (GA) Bandyopadhyay and Maulik (2002), Particle Swarm Optimization (PSO) Omran et al. (2005), and Differential Evolution (DE) Das et al. (2008) has been investigated. The accuracy of the final clustering results, the capability of the algorithms to achieve nearly similar results over randomly repeated runs (robustness), and the convergence speed are used as the performance metrics in the comparative analyses.

Organization of the rest of this chapter is as follows. In Section 3.2, the clustering problem is defined in a formal language. The API algorithm is shortly reviewed in Section 3.3. The proposed algorithm optimization algorithm and the clustering scheme used in this study are presented in Sections 3.4 and 3.5. A comprehensive set of experimental results are provided in Section 3.6. Finally, the work is concluded in Section 3.7.

3.2 Clustering Problem

The clustering problem consists of dividing a set of data into different groups, based on one or more features of the data Jain et al. (1999); Craigen et al. (1993). In the area of machine learning, clustering analysis is considered as an unsupervised learning method that constitutes a main role of an intelligent data analysis process. This tool explores the data structure and attempt to group objects into clusters such that the objects in the same clusters are similar and objects from different clusters are dissimilar. It is called unsupervised learning because, unlike classification (known as supervised learning), no a priori labeling of patterns is available to use in categorizing the cluster structure of the whole dataset. As the aim of clustering is to find any interesting grouping of the data, it is possible to define cluster analysis as an optimization problem in which a given function, called the clustering validity index, consisting of within cluster similarity and among clusters dissimilarities needs to be optimized.

In every optimization algorithm it is necessary to measure the goodness of candidate solutions. In this problem, the fitness of clusters must be evaluated. In order to achieve this, one given clustering definition called the clustering validity index has been considered, that is the objects inside a cluster are very similar, whereas the objects located in distinct clusters are very different. Thereby, the fitness function is defined according to the concepts of cohesion and separation:

1) Cohesion: The variance value of the objects in a cluster indicates the cluster's compactness. In other words, the objects within a cluster should be as similar to each other as possible.

2) Separation: The objects inside different clusters should be as dissimilar to each other as possible. To achieve this objective, different distance measures such as Euclidean, Minowsky, Manhatann, the cosine distance, etc are used as the cluster separation's indication Jain et al. (1999).

The clustering validity index is also used to determine the number of clusters. Traditionally, the clustering algorithms were run with a different number of clusters as

an input parameter. Then, based on the best gained validity measure of the dataset partitioning, the optimal number of clusters was selected Halkidi and Vazirgiannis (2001). Since the definitions of cohesion and separation are given, the fitness function of clustering can be introduced. There are some well-known clustering validity indexes in the literature which their maximum and minimum values indicate proper clusters. Therefore, these indexes can be used to define the fitness functions for optimization algorithms. In the current paper, two validity measures are employed in the study of automatic clustering algorithms. These two indexes are introduced as follows:

1- DB measure index Davies and Bouldin (1979a): This index is evaluated by division of within-cluster scatter by between-cluster separation. These two values are formulated as:

$$S_{i,q} = \left[\frac{1}{N_i} \sum_{X \in C_i} \| X - m_i \|^q \right]^{\frac{1}{q}} \tag{3.1}$$

and

$$d_{ij,t} = \left[\sum_{p=1}^{d} | m_{i,p} - m_{j,p} |^t \right]^{\frac{1}{t}} = \| m_i - m_j \| \tag{3.2}$$

where $S_{i,q}$ and $d_{ij,t}$ are the ith cluster scatter and the between ith and jth cluster distance values and X is a set of data points within C_i cluster. The ith cluster center is shown by $m_i, q, t \geq 1$, q is an integer, and q and t can be independently selected. The number of elements in the ith cluster C_i is N_i. By defining

$$R_{i,qt} = max \left\{ \frac{S_{i,q} + S_{j,q}}{d_{ij,t}} \right\}_{j \in k, j \neq i} \tag{3.3}$$

DB measure index is formulated as:

$$DB(K) = \frac{1}{K} \sum_{i=1}^{K} R_{i,qt} \tag{3.4}$$

where K is the number of clusters. The smaller $DB(K)$ value is, the more valid is the clustering process.

2- CS measure index Chou et al. (2004): First the centroid of the cluster C_i is

calculated as the average of the elements within that cluster:

$$m_i = \frac{1}{N_i} \sum_{x_j \in C_i} x_j \tag{3.5}$$

Then the CS measure can be formulated as:

$$CS(K) = \frac{\sum_{i=1}^{K} \left[\frac{1}{N_i} \sum_{X_i \in C_i} \{d(X_i, X_q)\} \right]}{\sum_{i=1}^{K} \left[min_{j \in K, j \neq i} \{d(m_i, m_j)\} \right]} \tag{3.6}$$

$d(X_i, X_q)$ is a distance metric between any two data points X_i and X_q. As the CS measure is also a function of the sum of within-cluster scatter to between-cluster separation, both the DB and CS measures has the same concept. It is stated in Chou et al. (2004) that while dealing with datasets of different densities and/or sizes the CS measure is more efficient than the other measures introduced in the literature.

3.3 API Algorithm

The API algorithm is inspired by the colonies of P. APIcalis ants in tropical forests near the Guatemala border in Mexico Monmarché et al. (2000). In this algorithm, a population of n_a ants $(a_1, a_2, ..., a_{n_a})$ is located in search space S to minimize objective function f. API contains two parameters named O_{rand} and O_{explo}. O_{rand} generates a random point (named nest N) that indicates a valid solution in search space S according to a uniform distribution and O_{rand} generates a new points in the neighborhood of N and also hunting sites. In the beginning, the nest location N placed randomly in the search space using parameter O_{rand}. Then, each ant ai of the na ants leaves the nest to create hunting sites randomly and utilizes O_{explo} with an amplitude $A_{site}(a_i)$ of the neighborhood centered in N. The $A_{site}(a_i)$ values are set as:

$$A_{site}(1) = 0.01, ..., A_{site}(i) = 0.01x^i, ..., A_{site}(n_a) = 0.01x^{n_a} \tag{3.7}$$

where $x = (1/0.01)^{(1/n_a)}$.

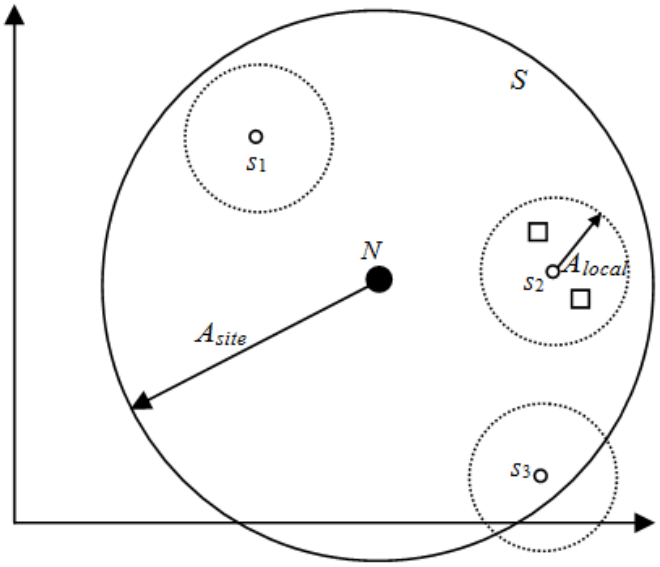

Figure 3-1: Search space of the API algorithm. s_1, s_2, and s_3 are sites randomly generated around nest N and their maximum distance from the nest being given by A_{site}. The small squares denote local exploration of site s_2 (points situated at a maximum distance of A_{local} from the site center s_2).

Afterwards, local search starts and each ant a_i goes to one of its p hunting sites s' in the neighborhood of its site s using O_{explo} with an amplitude $A_{local}(a_i)$. $A_{local}(a_i)$ is set to $A_{site}(a_i)/10$ based on the behavior of real ants. If $f(s') \leq f(s)$, the local search will be considered as successful (a prey has been caught) and ant a_i will memorize point s' and update its memory from s to s' and does a new exploration in the vicinity of the new site. On the contrary, a_i will randomly choose another site among its p sites saved in memory in the next exploration. If ant a_i cannot catch any prey in a hunting site which has been explored successively for more than $t_{local}(a_i)$ times, that hunting site will be forgotten and repeated by a new site created using O_{explo}. Then, nest N moves after T movements of the na ants (after every $n_a \times T$ individual moves) and goes to the best point found since its own last displacement. Finally, all sites will be erased from the ants' memories to avoid local minima. It is presented in Figure 3.1 how the initial solution space is divided into smaller search spaces in the AIP algorithm. The API algorithm usually terminates after a specific number of iterations or when the best-so-far solution achieves a desired value.

3.4 Opposition-Based API Algorithm

In most instances, Evolutionary Algorithms (EAs) start with random initial populations and attempt to lead them toward some optimal solutions. This searching process usually terminates when EAs meet some predefined criteria. However, the distance of these initial guesses from the optimal solutions has a significant effect on the computation effort and the obtained solutions' quality. The concept of Opposition-Based Learning (OBL) is introduced by Tizhoosh Tizhoosh (2006) to increase the chance of starting from fitter initial (closer to optimal solutions) points in the search space. In the proposed method, the opposition points of the initial guesses are found simultaneously. After making a comparison between initial solutions and their opposites in the search space, the fitter ones are chosen as the initial solutions. The judgment between a point and its opposite position is made based on their corresponding fitness function values. This procedure has the potential to improve the convergence speed and quality of solutions and can be applied not only to initial points but also continuously to each solution in the current population. The concept of opposite point can be defined as Tizhoosh (2006):
Let $X = (x_1, x_2, ..., x_D)$ be a point in a D-dimensional space, where $x_1, x_2, ..., x_D \in \mathbb{R}$ and $x_i \in [a_i, b_i] \forall i \in \{1, 2, ..., D\}$. The opposition point $\overline{X} = (\overline{x_1}, \overline{x_2}, ..., \overline{x_D})$ is defined by:

$$\overline{x_i} = a_i + b_i - x_i \tag{3.8}$$

Now assume that $f(X)$ and $f(\overline{X})$ are the fitness function values which are evaluated simultaneously to measure the fitness of the main point X and its opposition position X in the search space. Making a comparison between these two fitness values we continue the optimization process with the fitter one. In other words, if $f(X) \leq f(\overline{X})$ then point X can be replaced with $\overline{X}$; otherwise, the process will be continued by X.

In this study, we enhance the hunting sites' creation step of the API algorithm by using OBL scheme. We choose the original API as the main algorithm and the proposed opposition-base idea is embedded in API to improve its performance and

convergence speed.

In this part, we explain the OBL approach added to the original API algorithm. Based on optimization literature, the common method to create initial solutions, in absence of a priori knowledge, is random number generation. Therefore, as explained previously, by applying the OBL concept, fitter starting candidate solutions can be obtained when there is no a priori knowledge about the solutions. The following steps show the implementation of opposition-based initialization for API:

1- Create hunting sites $S = \{s_1, s_2, ..., s_{n_a}\}$ randomly using O_{explo} where $s_j = (x_{ij}, ..., x_{Dj})$ and $x_{ij} \in [a_i, b_i] \forall i \in \{1, 2, ..., D\}, j \in \{1, ..., n_a\}$.

2- Calculate opposite points $S_o = \{so_1, so_2, ..., so_{n_a}\}$ of the initialized random sites:

$$\overline{xo_{ij}} = a_i + b_i - x_{ij} \tag{3.9}$$

where $so_j = (xo_{ij}, ..., xo_{Dj})$.

3- Select n_a fittest hunting sites from $\{S \cup S_o\}$ as initial hunting sites using fitness function values.

A similar approach is applied to the algorithm when an ant loses all of its p sites and needs to create new hunting sites. Therefore, after making new sites by that ant, hunting sites which are ideally fitter than current created ones will be established in each iteration.

3.5 Clustering Formulation and Fitness Functions

The clustering method we applied in this work is the scheme proposed in Das et al. (2008), in which the chromosomes of a Differential Evolution (DE) algorithm are assigned to vectors of real numbers. These vectors contain $2K_{max}$ entries, where K_{max} is the maximum number of clusters specified by user.

To control the activation of each cluster during the clustering process, first K_{max} elements of the defined vectors are assigned to random positive floating numbers $T_{i,j}$ (for jth cluster center in the ith vector) in [0,1]. These floating numbers are called

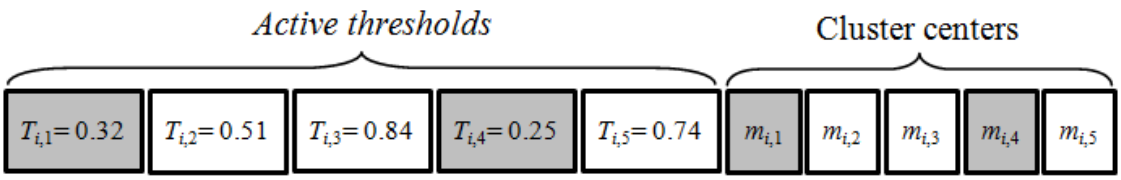

Figure 3-2: Active thresholds and their corresponding cluster centroids in vector i (the white and grey centroids are active and inactive, respectively).

activation thresholds. In this model, if $T_{i,j} \geq 0.5$, the jth cluster center in the ith vector will be used for clustering of the associated data. In contrast, if $T_{i,j} \leq 0.5$, the corresponding jth cluster center will not be considered in the partitioning process. In other words, $T_{i,j}$'s are used as selection rules in each vector controlling the activation of cluster centers. The second part of vectors contains K_{max} D-dimensional centroids. Figure 3.2 shows a vector containing of five centroids and their corresponding activation thresholds. As it can be seen, only three of those centroids are active (have activation thresholds more than 0.5) in this vector.

In this scheme, when a new vector is constructed, the T values are used to active the cancroids of clusters. If in a vector all $T_{i,j}$'s are smaller than 0.5, two of the thresholds will be selected randomly and their values will be reinitialized between 0.5 and 1.0 which means the minimum number of clusters in a vector is 2.

In OBAPI, each clustering vector is considered as a hunting site. Ants are moving on the search space and can take or drop centroids according to the behavioral rules of the algorithm. Then, the nest is brought closer to the proper hunting sites and ants go back to new fruitful sites to try another pick up. To compare the performance of our proposed algorithm with the performance of other reported algorithms Das et al. (2008), we also applied the CS measure and DB measure introduced in Section 3.2. Therefore, two fitness functions are constructed as:

$$\begin{cases} f_1 = \frac{1}{CS_i(K)} \\ f_2 = \frac{1}{DB_i(K)} \end{cases} \tag{3.10}$$

where CS_i and DB_i are the clustering indexes defined in Equations 3.4 and 3.6. These indexes evaluate the quality of the clusters delivered by vector i. Since all selected centroids and their opposites are always built inside the boundary of the dataset, there is no probability of a division by zero while computing the DB and/or CS measures.

3.6 Experimental Results and Discussion

In this work, five real world clustering problems from the UCI database Blake and Merz (1998), which is a well-known database repository for machine learning, are used to evaluate the performance of the Opposition-Based API (OBAPI) algorithm. The datasets are briefly summarized as (Here, n is the number of data points, d is the number of features, and K is the number of clusters):

1) Iris ($n = 150$, $d = 4$, $K = 3$): This dataset with 150 random samples of flowers from the iris species setosa, versicolor, and virginica consists 50 observations for sepal length, sepal width, petal length, and petal width in cm.

2) Wine ($n = 178$, $d = 13$, $K = 3$): This dataset is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 instances with 13 numeric attributes in the wine dataset. All attributes are continuous and there is no missing attributes.

3) Wisconsin breast cancer ($n = 683$, $d = 9$, $K = 2$): The Wisconsin breast cancer database has 9 relevant features: clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The dataset has two types: benign (239 objects) or malignant (444 objects) tumors.

4) Vowel ($n = 871$, $d = 3$, $K = 6$): This dataset consists of 871 Indian Telugu vowel sounds. The dataset has 3 features which are the first, second, and third vowel frequencies, and 6 overlapping classes named d (72 objects), a (89 objects), i (172 objects), u (151 objects), e (207 objects), and o (180 objects).

5) Glass ($n = 214$, $d = 9$, $K = 6$): This dataset presents 6 different glass types called building windows float processed (70 objects), building windows nonfloat processed (76 objects), vehicle windows float processed (17 objects), containers (13 objects), tableware (9 objects), and headlamps (29 objects), respectively. Each of these types has 9 features: refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron.

The performance of the OBAPI algorithm is compared with three recently proposed partitional clustering algorithms called automatic clustering using an improved deferential evolution (ACDE) Das et al. (2008), genetic clustering with an unknown number of clusters K (GCUK) Bandyopadhyay and Maulik (2002), and dynamic clustering particle swarm optimization (DCPSO) Omran et al. (2005). The improvement effects of our modified algorithm with normal API have been also investigated dealing with similar clustering problems. We used the default parameter settings, selected in Monmarché et al. (2000), for all conducted experiments:

1- Number of ants, Na = 20.

2- Number of iterations (explorations performed by each ant between two nest moves), T =50.

3- Number of hunting sites, $p = 2$.

4- Search number (number of times ant a_i cannot catch any prey in a hunting site which has been explored successively), $t_{local}(a_i) = 50$, $i = 1, ..., N_a$.

For API and OBAPI, the hunting sites (cluster centroids) are selected randomly between the minimum and maximum numerical values of any feature of the datasets. Parameter O_{rand} generates a uniformly distributed random point within those intervals. Parameter O_{explo} is also used to create new hunting site $s' = (x'_1, ..., x'_D)$ from $s = (x_1, ..., x_D)$ as follows:

$$x'_i = x_i + U \times A \times (b_i - a_i), \forall i \in [1, ..., D] \tag{3.11}$$

where $x_{ij} \in [a_i, b_i] \forall i \in \{1, 2, ..., D\}$, U is a uniformly distributed value within [-0.5,+0.5] and A is the maximum amplitude of the move introduced in Equation 3.7.

The maximum and minimum number of clusters, K_{max} and K_{min}, are set to 20 and 2, respectively.

In this study, a comprehensive comparison between the results of the API and OBAPI algorithms and the results of the ACDE, GCUK, and DCPSO reported in Das et al. (2008) has been made to verify the performance of our proposed approach. We compare the convergence speed of all the algorithms by measuring the number of function calls (NFCs) which is most commonly and fair used metric in optimization literature. The quality of obtained solutions, determined by the CS and DB measures, and ability of the algorithms to find the optimal number of clusters have been also considered as two other evaluation metrics. In order to minimize the effect of the stochastic nature of API and OBAPI on the metrics, our reported results for each clustering problem is the average over 40 independent trials which is equal to the number of independent the algorithms' runs reported in Das et al. (2008). The results of two sets of experiments are presented by utilizing the five evolutionary clustering algorithms (API, OBAPI, ACDE, GCUK, and DCPSO) while CS and DB measures are separately considered as their fitness functions. For a detailed discussion on the parameter settings and simulation strategy of the ACDE, GCUK, and DCPSO algorithms please refer to Das et al. (2008); Bandyopadhyay and Maulik (2002); Omran et al. (2005). We implemented both the API and OBAPI algorithms in Python 2.7.6 on a Intel Core i7, with 2.4 GHz, 8 GB RAM in Ubuntu 14.04 environment.

In order to compare the accuracy of OBAPI and API with ACDE, DCPSO, and GCUK, maximum NFCs is set to 10^6 and considered as the termination criterion for each clustering algorithm. Afterwards, final solutions are considered as the number of clusters found, final value of fitness function, and two other metrics called inter-cluster and intra-cluster distances. The inter-cluster distance shows the average of distances among centroids of the obtained clusters and the intra-cluster distance presents the average of distances among data vectors inside a cluster. To achieve crisp and compact clusters, the clustering algorithms try to maximize the inter-cluster distance and minimize intra-cluster distance, simultaneously. Table 3.1 shows the average number of found clusters, the final CS values (Equation 3.6), and the inter-cluster and intra-

Table 3.1: Mean and standard deviation values of final solutions of the clustering algorithms over 40 independent trials using CS measure (maximum number of function calls, NFCs = 10^6, is set as the termination criterion).

Data	Algorithm	Clust. Num.	CS value	intra- dist.	inter- dist.
Iris	OBAPI	**3.11±0.05214**	**0.6122±0.053**	**2.8736±1.542**	**2.7211±0.362**
	API	3.42±0.02451	0.6812±0.142	3.2232±0.324	2.4516±0.024
	ACDE	3.25±0.0382	0.6643±0.097	3.1164±0.033	2.5931±0.027
	DCPSO	2.23±0.0443	0.7361±0.671	3.6516±1.195	2.2104±0.773
	GCUK	2.35±0.0985	0.7282±2.003	3.5673±2.792	2.5058±1.409
Wine	OBAPI	3.16±0.0874	**0.9622±0.047**	**4.005±0.004**	**3.6411±0.324**
	API	3.21±0.0456	0.9132±0.0514	4.096±0.041	3.1123±0.745
	ACDE	3.25±0.0391	0.9249±0.032	4.046±0.002	3.1483±0.078
	DCPSO	**3.05±0.0352**	1.8721±0.037	4.851±0.184	2.6113±1.637
	GCUK	2.95±0.0112	1.5842±0.328	4.163±1.929	2.8058±1.365
Breast	OBAPI	**2.00±0.00**	0.4726±0.015	4.3232±0.214	3.2114±0.526
	API	2.15±0.0496	0.4869±0.637	4.4568±0.0354	3.0412±2.324
	ACDE	**2.00±0.00**	**0.4532±0.034**	**4.2439±0.143**	**3.2577±0.138**
	DCPSO	2.25±0.0632	0.4854±0.009	4.8511±0.373	2.3613±0.021
	GCUK	2.00±0.0083	0.6089±0.016	4.9944±0.904	2.3944±1.744
Vowel	OBAPI	**6.13±0.0421**	**0.9011±0.624**	**1406.32±9.324**	**2796.67±0.547**
	API	5.77±0.0645	0.9232±0.224	1434.85±0.457	2732.11±0.213
	ACDE	5.75±0.0751	0.9089±0.051	1412.63±0.792	2724.85±0.124
	DCPSO	7.25±0.0183	1.1827±0.431	1482.51±3.973	1923.93±1.154
	GCUK	5.05±0.0075	1.9978±0.966	1495.13±12.334	1944.38±0.747
Glass	OBAPI	**6.00±0.00**	**0.3112±0.647**	**521.278±65.23**	**896.31±6.123**
	API	6.11±0.0324	0.4236±0.278	550.217±14.52	871.35±3.662
	ACDE	6.05±0.0148	0.3324±0.487	563.247±134.2	853.62±9.044
	DCPSO	5.96±0.0093	0.7642±0.073	599.535±10.34	889.32±4.233
	GCUK	5.85±0.0346	1.4743±0.236	594.67±1.789	

Table 3.2: Clustering error mean and standard deviation values over 40 independent trials using CS measure (NFCs = 10^6).

Data	clust. error				
	OBAPI	API	ACDE	DCPSO	GCUK
Iris	**2.14±0.00**	2.22±0.01	2.35±0.00	4.15±0.00	5.00±0.00
Wine	**34.21±2.00**	37.23±2.30	36.65±0.00	99.4±1.09	100.24±1.05
Breast	21.87±0.47	**26.63±0.04**	22.25±0.28	27.01±1.25	29.00±1.55
Vowel	**401.62±4.12**	425.89±0.08	418.75±3.10	453.58±6.61	476.42±6.92
Glass	**87.65±0.47**	95.24±0.05	92.55±0.19	102.1±0.68	98.21±0.08

Table 3.3: Mean and standard deviation values of NFCs required by clustering algorithms to reach the defined cutoff thresholds (using CS measure and over 40 independent trials).

Data	Algorithm	Cutoff value	Ave. NFCs	intra- dist.	inter- dist.
Iris	OBAPI	0.95	**284567.23±24.36**	**3.3145±0.471**	**2.8674±0.547**
	API		432578.36±84.65	3.9124±0.841	2.0456±0..875
	ACDE		459888.95±20.50	3.7836±0.509	2.0758±0.239
	DCPSO		679023.85±31.75	3.9637±1.666	2.0093±0.795
	GCUK		707723.70±120.21	3.9992±2.390	1.9243±1.843
Wine	OBAPI	1.90	**42311.84±77.12**	**3.9165±0.874**	**3.5211±0.0774**
	API		66251.32±87.59	4.6232±0.547	2.8765±0.145
	ACDE		67384.25±56.45	4.9872±0.148	3.1275±0.0357
	DCPSO		700473.35 ±31.42	4.0743±0.093	1.9967±1.828
	GCUK		785333.05±21.75	5.9870±1.349	2.1323±1.334
Breast	OBAPI	1.10	**165278.32±15.36**	5.1221±0.132	2.8011±0.411
	API		273111.67±14.56	5.43266±0.025	2.832±0.741
	ACDE		292102.50±29.73	**4.9744±0.105**	**3.0096±0.246**
	DCPSO		587832.50±7.34	5.6546±0.241	2.1173±0.452
	GCUK		914033.85±24.83	8.0442±0.435	2.0542±1.664
Vowel	OBAPI	2.50	**292487.32±14.36**	**1475.32±0.852**	**2932.64±1.459**
	API		405524.65±32.11	1482.65±0.741	2687.57±0.573
	ACDE		437533.35±51.73	1494.12±0.378	2739.85±0.163
	DCPSO		500493.15±35.47	1575.51±3.786	1923.93±1.154
	GCUK		498354.10±74.60	1593.72±1.789	2633.45±1.213
Glass	OBAPI	1.80	**288524.62±74.32**	**572.326±65.78**	**861.56±0.901**
	API		408975.41±98.32	600.985±42.32	852.11±0.324
	ACDE		443233.30±47.65	590.572±34.24	853.62±0.44
	DCPSO		566335.80±25.73	619.980±15.98	846.67±0.804
	GCUK		574938.65±82.64	615.88±20.95	857.34±1.465

cluster distances obtained by OBAPI and API and the other three algorithms. The mean values and standard deviations of misclassified data are presented in Table 3.2. These values have been obtained based on the fact that the benchmark datasets have known nominal partitions and the objects that were assigned to clusters other than the nominal ones cause some misclassification errors. Then, we need to compare the different algorithms in term of convergence speed. For each dataset, a cutoff value of CS fitness function is selected as a threshold. This values is somewhat larger than the minimum CS fitness function amount obtained by each algorithm in Table 3.3. The NFCs that each algorithm takes to achieve the cutoff CS fitness function value is given in Table 3.3. Exactly similar experiments are conducted again over the benchmark datasets using a DB measure fitness function (Equation 3.4) and the similar entries are presented in Tables 3.4 to 3.6. Best obtained values are sown in boldface in all the tables.

It is demonstrated in Tables 3.1 and 3.4 that for the iris dataset the OBAPI has gained the lowest values of the final CS and DB measures and the best values of mean intra- and inter-cluster distances. As discussed in Das et al. (2008), the considerable overlap between two clusters (virginica and versicolor) in the iris dataset has caused GCUK and DCPSO to gain only two clusters on average while OBAPI, API, and ACDE were successful in finding about three clusters and among them OBAPI has yielded the closest value to the real number of iris clusters. For the wine dataset, all the algorithms have been outperformed by DCPSO in term of number of clusters. However, all the five algorithms have obtained comparable accuracies. Again, OBAPI has achieved the best average values of fitness functions, and intra- and inter-cluster distances.

It is also observed in Tables 3.1 and 3.4 that for the breast cancer dataset, despite the fact that OBAPI, ACDE, and GCUK were competitively successful to yield high accurate vales of the number of clusters, ACDE has outperformed the other algorithms in terms of the other metrics. This challenge may happen due to substantial increase in the number of both data vectors and features of the current dataset in comparison to other ones that had some bad effects on the performance of the OBAPI algorithm.

Table 3.4: Mean and standard deviation values of the clustering algorithms over 40 independent trials using DB measure fitness function (maximum number of function calls, NFCs = 10^6, is set as the termination criterion)

Dataset	Algorithm	Clust. Num	DB value	intra- dist.	inter- dist.
Iris	OBAPI	**3.01±0.0124**	**0.4011±0.014**	**2.9911±0.745**	**2.8965±0.475**
	API	3.15±0.0851	0.4565±0.087	3.1845±0.047	2.3574±0.012
	ACDE	3.05±0.0712	0.4645±0.022	3.1633±0.076	2.8387±0.658
	DCPSO	2.25±0.0593	0.6899±0.008	3.8536±0.122	2.2544±0.039
	GCUK	2.30±0.0738	0.7377±0.065	3.8436±0.076	2.1438±0.022
Wine	OBAPI	3.10±0.054	2.9614±0.047	4.2156±0.469	3.3641±6.457
	API	3.15±0.0741	3.2652±0.412	4.6689±0.0485	2.9611±5.648
	ACDE	3.25±0.0931	3.0432±0.021	4.4212±0.096	3.1029±0.047
	DCPSO	3.05±0.0024	4.3432±0.232	4.8668±0.154	2.6113±1.635
	GCUK	2.95±0.0173	5.3424±0.343	5.1312±1.342	2.7565±2.128
Breast Cancer	OBAPI	2.05±0.0845	0.5315±0.241	4.6415±0.214	3.0524±0.078
	API	2.46±0.0785	0.5801±0.325	4.6213±0.075	3.0065±0.045
	ACDE	**2.05±0.0563**	**0.5813±0.006**	**4.5463±0.023**	**3.1002±0.064**
	DCPSO	2.50±0.0621	0.5754±0.073	4.9232±0.373	2.2684±0.063
	GCUK	2.50±0.0352	0.6328±0.002	6.5541±0.433	1.8032±0.016
Vowel	OBAPI	**5.80±0.542**	**0.9200±0.247**	**1440.17±0.321**	**2311.22±0.784**
	API	5.68±0.0745	1.0013±0.214	1451.13±0.123	2300.69±0.145
	ACDE	5.75±0.0241	0.9224±0.334	1449.12±0.834	2289.85±0.163
	DCPSO	7.25±0.0652	1.2821±0.009	1500.57±3.748	1747.76±1.764
	GCUK	5.05±0.0561	2.9482±0.028	1573.23±4.675	2271.89±1.222
Glass	OBAPI	**6.02±0.149**	**1.00±0.014**	**501.268±3.8**	**898.11±4.30**
	API	6.25±0.0312	1.0423±0.021	505.621±0.36	895.63±4.25
	ACDE	6.05±0.0248	1.0092±0.083	501.757±4.3	893.46±3.32
	DCPSO	5.95±0.0193	1.5152±0.073	514.554±9.5	856.00±8.07
	GCUK	5.85±0.0346	1.8371±0.034	518.903±2.9	852.32±5.43

Table 3.5: Clustering error mean and standard deviation values over 40 independent trials using DB measure (NFCs = 10^6).

Data	clust. error				
	OBAPI	API	ACDE	DCPSO	GCUK
Iris	**1.92±0.01**	2.34±0.04	2.22±0.00	2.79±0.55	2.75±0.08
Wine	**31.23±0.00**	36.11±0.32	40.15±0.00	112.5±2.50	118.45±1.77
Breast	28.58±0.36	29.65±0.11	26.75±0.25	30.23±0.46	**26.50±0.80**
Vowel	**410.98±3.10**	420.25±6.41	418.35±7.50	435.00±3.75	473.46±3.57
Glass	**6.23±0.54**	7.65±0.26	8.86±0.42	14.35±0.26	17.98±0.67

Table 3.6: Mean and standard deviation values of NFCs required by clustering algorithms to reach the defined cutoff thresholds (using DB measure and over 40 independent trials).

Data	Algorithm	Cutoff value	Ave. NFCs	intra- dist.	inter- dist.
Iris	OBAPI	0.80	**335614.21±13.54**	**3.5147±0.014**	**2.6385±0.574**
	API		484175.32±85.62	3.8657±0.0487	2.1152±0.398
	ACDE		504783.45±12.65	3.9928±0.029	2.1029±0.842
	DCPSO		679084.75±16.57	3.7852±1.842	1.7641±0.439
	GCUK		790865.90±10.21	4.4587±3.782	1.9383±1.307
Wine	OBAPI	6.00	**315268.26±6.32**	**4.2589±0.048**	**3.6015±0.184**
	API		479523.14±4.57	4.7612±0.541	3.1511±0.415
	ACDE		464653.35±5.50	4.8292±0.732	3.0219±0.069
	DCPSO		486885.85±2.85	5.1472±0.472	2.1161±1.623
	GCUK		598743.35±8.09	4.9383±1.722	2.9121±0.353
Breast	OBAPI	0.90	**293142.26±4.62**	5.6516±0.745	2.8641±0.689
	API		446213.62±9.78	5.9863±0.851	2.5876±0.459
	ACDE		424732.30±8.93	5.4489±0.342	**3.0234±0.683**
	DCPSO		467854.60±10.12	**5.2885±0.552**	2.0124±1.596
	GCUK		678874.90±7.82	6.8832±0.733	2.1637±1.458
Vowel	OBAPI	3.00	**291454.25±1.25**	**1362.11±1.98**	**2315.65±0.475**
	API		463211.65±3.87	1684.28±1.85	1896.45±0.847
	ACDE		435743.05±2.65	1544.92±0.834	2081.31±0.679
	DCPSO		556865.00 ±4.26	1652.58±2.341	1264.87±3.069
	GCUK		575854.65±1.29	1582.55±7.332	1989.38±7.734
Glass	OBAPI	2.00	**324825.32 ±14.67**	**128.475±16.3**	**14.42±Âś1.54**
	API		486425.41±14.52	146.574±34.62	13.24±4.21
	ACDE		506754.00±12.27	132.757±15.8	13.46±2.54
	DCPSO		569787.95±10.83	155.856±24.7	10.42±4.69
	GCUK		687678.75±10.97	178.809±30.3	10.21±1.09

However, as it can be seen the difference between the final solutions of the two best algorithms (ACDE and OBAPI) is not significant. Tables 3.1 and 3.4 also show that the OBAPI algorithm has provided better results than the other four algorithms dealing with vowel and glass datasets which consist of large number of data vectors as well as six overlapping clusters.

The clustering errors reported in Tables 3.2 and 3.5 imply that, despite of the acceptable performance over clustering of the benchmark datasets, all the five algorithms contain some misclassification with respect to the nominal clusters. It is explained in Das et al. (2008) that this clustering error is not only caused by the optimization algorithms' performance. But, some other factors such as definition assumptions of the fitness functions, error in collecting data, outliers in the datasets, and errors made by human in the nominal data might be more significant reasons for this type of errors. As it can be seen, except the breast cancer dataset, OBAPI achieved the least amount of clustering error among the five clustering algorithms for both CS and DB measure fitness functions.

Tables 3.3 and 3.6 clearly illustrates the effectiveness of the proposed OBAPI algorithm dealing with clustering of the benchmarks. As it is shown, a significantly lower NFCs is needed by our algorithm to reduce both CS and DB fitness function values to the cutoff thresholds in all cases. After OBAPI, ACDE, API, DCPSO, and GCUK have needed lesser NFCs to achieve cutoff threshold values, respectively. Moreover, OBAPI has yielded the best amount of mean intra- and inter-cluster distances over most datasets.

To conclude, the obtained results indicate that OBAPI surpass normal API on the clustering of all the benchmarks. The OBL method applied to the API led to accuracy improvements in most clustering problems and convergence speed-ups reaching about 33%. It is interesting to see that improvements of the convergence speed were relatively similar for all benchmark datasets. In contrast, OBAPI was not as successful as ACDE dealing with the breast cancer dataset in term of accuracy. In general, it seems that OBL performs well with the more difficult problems, as it helps the learning process. These results are very encouraging, as they demonstrate that

opposition can help improve performance. However, it is important to consider here that OBAPI performs better than normal API according to the current comparison strategies as well.

4

Multiobjective Clustering Analysis using Particle Swarm Optimization

In this chapter, partitional clustering is defined as a multiobjective optimization problem. The aim is to obtain well-separated, connected, and compact clusters and for this purpose, two objective functions are defined based on the data connectivity and cohesion concepts. In addition, an efficient multiobjective particle swarm optimization algorithm is applied to automatic grouping of large unlabeled datasets. A comprehensive experimental study is conducted and the obtained results are compared with the results of four other classical clustering techniques. It is shown that the proposed algorithm can achieve the optimal number of clusters, is robust and, in most cases, outperforms the other methods on several benchmark datasets in term of accuracy.

Introduction

In this chapter, a multiobjective clustering particle swarm optimization (MCPSO) framework is proposed to obtain well-separated, connected, and compact clusters in any unlabeled datasets with different dimensions and cluster characteristics. MCPSO also aims to determine the optimal number of clusters, automatically. To achieve this objectives, two contradictory objective functions are defined based on the concepts of *connectivity* and *cohesion* and MCPSO is used to find a set of non-dominated

clustering solutions as a pareto front. Finally, we utilize a simple decision maker to select the best solution along the obtained pareto solutions. A comprehensive comparison of the results of MCPSO with four conventional clustering approaches is investigated. The accuracy [1]measured on the results of final clustering, together with computational time, are used as the performance metrics in the comparative analyses.

The rest of this chapter is organized as follows. In Section 4.2, the clustering problem, similarity measures, and clustering validity measures are defined in a formal language. The proposed MCPSO algorithm and the clustering objective functions are introduced in detail in Section 4.3. A comprehensive set of experimental results are provided in Section 4.4. Finally, the work is concluded in Section 4.5.

4.1 Clustering Problem

The clustering problem consists of dividing a set of data into different groups, based on one or more features of the data (Jain et al., 1999). This tool explores the data structure and attempt to group objects into clusters such that the objects in the same clusters are similar and objects from different clusters are dissimilar. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a set of d-dimensional n vectors in the given search space S. The ith vector $\mathbf{x}_i$ corresponds to the ith object in S and each element $x_{i,j}$ characterizes the jth value of the ith vector where $i = 1, ..., n$ and $j = 1, ..., d$. Given the set of vectors X, the aim of a clustering algorithm is to find the optimal set of K clusters $C^* = \{C_1, C_2, ..., C_K\}$, $C_p \cap C_q = \emptyset$ where $p, q \in \{1, 2, ..., K\}$ and $p \neq q$, in such a way that the objects inside a cluster are very similar, whereas the objects located in distinct clusters are very different based on a given similarity measure function.

As it is mentioned, clustering is the method of grouping objects of a dataset into distinct partitions based on some similarity measures. It is shown that usually the similarity between two different vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in a given feature space S is related to the amount of distance between them Jain et al. (1999). As a general method the

[1]Selected datasets are in fact labeled. Hence, we have been able to measure the average 'accuracy' on clusters, assuming that each of them is actually related to a label.

distance between objects of a given d-dimensional vector space can be found using the Minowsky metric given by Hamerly and Elkan (2003):

$$D_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{t=1}^{d} \left(x_{i,t} - x_{j,t} \right)^p \right)^{1/p} \tag{4.1}$$

The Euclidean and Manhattan distance measures are two special cases of the Minowsky metric when $p = 2$ and $p = 1$, respectively Jain et al. (1999). It is shown in Hamerly and Elkan (2003) that the distances between vectors increase dramatically with the growth of feature space dimensions. So the Minowsky metric is not efficient enough dealing with high dimensional clustering problems. As an alternative, the cosine metric can be used due to its vector normalization over a common range:

$$D_{cosine}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{t=1}^{d} x_{i,t} x_{j,t}}{\| \mathbf{x}_i \| \| \mathbf{x}_j \|} \tag{4.2}$$

The Mahalanobis metric is another distance measure which is defined by Jain et al. (1999):

$$D_m^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j) \sum^{-1} (\mathbf{x}_i - \mathbf{x}_j)^T \tag{4.3}$$

where $\sum^{-1}$ is the covariance matrix of the vectors and T stands for the transpose operation. This metric takes into account the correlations of the dataset and in this way easily considers different associations between features and thus is a scale-invariant measure.

There is always a question for clustering analysis which is how to evaluate the goodness of the results of a clustering algorithm. In order to answer to this question, many validity indexes are introduced in terms of statistical and mathematical functions Halkidi and Vazirgiannis (2001). In some cases, these measures can be used to determine the number of clusters. The clustering validity functions are defined to ideally provide three aspects of clustering:

(1) *cohesion*: The objects within a cluster should be as similar to each other as possible.

(2) *separation*: The objects inside different clusters should be as dissimilar to each other as possible.

(3) *connectivity*: The neighboring objects in the search space should belong to the same cluster.

However, the concept of *separation* is related to and opposite of *cohesion* of clusters. There are two main groups of validity functions which are named *internal* and *external* criteria.The *internal* validity criteria attempt to evaluate the quality of the results of data clustering without having access to any external information. Sum of Squares Error (SSE) (Halkidi and Vazirgiannis, 2001), Chou-Su (CS) measure Chou et al. (2004), Davies-Bouldin (DB) measure Davies and Bouldin (1979b), and Silhoutte Coefficient (CS) are some of the well-known *internal* validity indexes in the literature. The *external* criteria evaluate the goodness of the clustering results based on some known information which are sometimes available in terms of data labels in the clustering problems. Rand index Rand (1971) and Jaccard index Halkidi and Vazirgiannis (2001) are two measures in the category of the *external* indexes which need a reference clustering to evaluate the validity of solutions.

4.2 Multiobjective Clustering with Particle Swarm Optimization

In this section, we propose a clustering method based on particle swarm optimization algorithm Kennedy and Eberhart (2001) in a multiobjective framework (MCPSO). MCPSO consists of two main phases named optimization and decision making. Two conflicting objective functions are defined based on *connectivity* and *cohesion* with the aim of obtaining well-separated, compact, and connected clusters. The optimization phase results in a set of optimal clusterings, called pareto solutions Kasprzak and Lewis (2001), which represent compromises among the conflicting objectives. Each pareto solution is a trade-off partitioning with different number of clusters. Therefore, MCPSO is also able to determine the optimal number of clusters, automatically. As

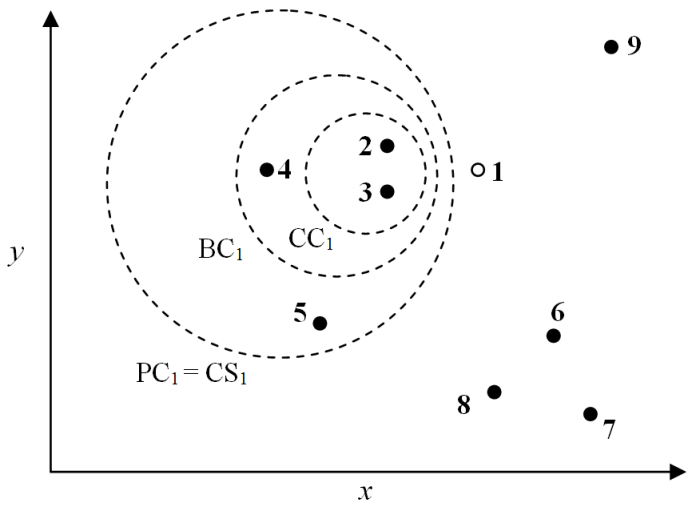

Figure 4-1: Example dataset for the NC algorithm. CC_1, BC_1, PC_1, and CS_1 include the *core neighbors*, *density connected neighbors*, *extended neighbors*, and *final neighbors* of point 1, respectively.

anyone of the pareto solutions can be an acceptable clustering and considered optimum in some respects, we apply a simple decision maker to select the best clustering solution along the pareto solutions based on a compromise on two objectives.

In the clustering analysis, *connectivity* is a local concept that measures the degree to which neighboring data points in a dataset share the same cluster Handl and Knowles (2007). Single linkage agglomerative clustering Voorhees (1985) and other density-based clustering methods Ester et al. (1996) mainly use this concept and are proper to find clusters with random shapes. k-nearest neighbors (KNN), ϵ-neighborhood Ester et al. (1996), and the Neighborhood Construction (NC) algorithm İnkaya and Özdemirel (2013) are some of the popular proposed neighborhood construction algorithms. It is stated in İnkaya and Özdemirel (2013) that the NC algorithm shows better performance in comparison with KNN and ϵ-neighborhood dealing with clusters with arbitrary shapes and different densities. Moreover, NC is capable of finding neighbors of a single data point and can generate sub-clusters by merging the data points having common neighbors. NC performs four steps, which are briefly described here:

Step 1: For each point i in a dataset D, a list L_i is generated, which contains all

other points listed in increasing order of their distance to point i. Be T_i the set of all points in D but i and $T_i(j)$ the jth member of the ordered set T_i, $T_i(j)$'s density, say $density_i(j)$, is defined as the number of points in D lying inside the sphere passing through points i and $T_i(j)$ with diameter $d_{iT_i(j)}$. Points with density values equal to and more than 0 are considered as the points with direct and indirect connections to point i, respectively. Points in T_i that are closer to point i than the nearest point with indirect connection are named *core neighbors* of point i and included in set CC_i. The core neighbors set of point 1, $\mathrm{CC}_1 = \{2, 3\}$, is shown in Figure 1. $T_1 = \{2, 3, 4, 5, 6, 7, 8, 9\}$ and $density_1 = \{0, 0, 2, 1, 0, 2, 3, 0\}$ are the increasing order and density sets of point 1.

Step 2: The first point in T_i at which the density starts to decrease is considered as the break point, say $break_i$. A break point can be interpreted as the beginning of a region with different density in D. Points in T_i that are closer to point i than $break_i$ are listed in *density connected neighbors* set BC_i. $\mathrm{BC}_1 = \{2, 3, 4\}$ is presented for point 1 in Figure 4.1.

Step 3: Point i and $T_i(break_i)$ are indirectly connected if $\mathrm{BC}_i \cap \mathrm{BC}_{T_i(break_i)} \neq \emptyset$. Therefore, $break_i$ and its subsequent points will be considered as *extended neighbors* until the next break point cannot satisfy this condition. In Figure 1, the first break point for point 1 is data point 5. The intersection of $\mathrm{BC}_1 = \{2, 3, 4\}$ and $\mathrm{BC}_5 = \{3, 1, 2\}$ is nonempty. Hence, point 5 added to *extended neighbors* set for point 1 (PC_1). Following the ordering in T_1, another break point is found (point 6). As $\mathrm{BC}_1 \cap \mathrm{BC}_6 = \emptyset$, the *extended neighbors* set PC_1 for point 1 becomes $\mathrm{PC}_1 = \{2, 3, 4, 5\}$ (Figure 4.1).

Step 4: In the last step, first *final neighbors* set CS_i of point i is initialized as PC_i. Then, a mutual connectivity test is conducted between point i and all the members of CS_i. The mutual connectivity test is explained in detail in (İnkaya and Özdemirel, 2013). Let $\mathrm{CS}_i(j)$ be the jth member of the set CS_i. If point i and point $\mathrm{CS}_i(j)$ pass the mutuality test, $\mathrm{CS}_i(j)$ will be considered as the *final neighbors* of point i. Otherwise, $\mathrm{CS}_i(j)$ and the points coming after it will be removed from CS_i. After conducting step 1 through step 4, $\mathrm{CS}_1 = \{2, 3, 4, 5\}$ is obtained for point 1 (Figure 4.1).

Finally, sub-cluster $M_1 = \{1, 2, 3, 4, 5\}$ is created by adding point 1 itself to CS_1.
As the obtained sub-clusters of all data points can be considered as a foundation of a clustering solution, we applied this method to evaluate the *connectivity* objective function value. It is worth nothing that the NC procedure is precomputed only once in the initialization phase of the algorithm. Let K and N be the number of clusters and data points, respectively. Connectivity of cluster C_i with respect to the all sub-clusters can be defined as:

$$Con_i = \frac{1}{N} \sum_{j=1}^{N} \frac{\mid C_i \cap M_j \mid}{\mid M_j \mid} \tag{4.4}$$

In this equation, $\frac{|C_i \cap M_j|}{|M_j|}$ takes a value of one if cluster C_i and sub-cluster M_j fully overlap. If sub-cluster M_j is part of a cluster other than C_i, then $C_i \cap M_j = \emptyset$ and consequently $\frac{|C_i \cap M_j|}{|M_j|}$ will take a value of zero. In other cases, this value would be in between zero and one which means sub-cluster M_j is divided among cluster C_i and one or more other clusters. We define the *connectivity* objective function for all K clusters as:

$$f_1 = \frac{1}{K} \sum_{i=1}^{K} Con_i \tag{4.5}$$

The value of f_1 would be in the interval $[0, 1]$. A value near to zero indicates that neighboring data points are divided in the different clusters whereas a value close to one shows that neighboring data points are mostly assigned to the same clusters. Therefore, as an objective, f_1 should be maximized.
In order to express cluster *cohesion*, we compute sum of the maximum within-cluster distances between data points:

$$f_2 = \frac{1}{K} \sum_{i=1}^{K} \left[\frac{1}{N_i} \sum_{\mathbf{x}_p \in C_i} \max_{\mathbf{x}_q \in C_i} \left\{ D(\mathbf{x}_p, \mathbf{x}_q) \right\} \right] \tag{4.6}$$

where $D(.,.)$ is the *cohesion* distance function (such as Euclidean distance) and N_i is the number of data points in cluster C_i. As an objective, f_2 should be minimized to obtain maximum similarity among data points assigned to each cluster. This function

has been applied as the numerator of CS measure in Chou et al. (2004) and proved to be more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures.

These two objective functions are able to balance each other's inclination towards increasing or decreasing the number of clusters. When the number of clusters increases, f_1 worsens (decreases) and f_2 improves (decreases). As it is mentioned earlier, the concept of *separation* can be perceived as the opposite of *cohesion*, so we only considered *connectivity* and *cohesion* to define the objective functions.

Particle Swarm Optimization (PSO) Kennedy and Eberhart (2001) is inspired by the social and cognitive behavior of birds in a flock or fish in a school adapting to their environments to find a source of food. PSO leads the population of particles (swarm) toward the best area of the search space to find the global optimal solution. In PSO, a velocity vector is used to update the current position of each particle in the swarm. The velocity vector is updated based on the memory gained by each particle as well as the knowledge gained by the swarm as a whole. In other words, particles adapt themselves to the environment using both their own memory and the knowledge gained by the swarm. The position $\mathbf{x}$ of particle i at iteration k is updated by the following equation:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1}\Delta t \tag{4.7}$$

where $\mathbf{v}_i^{k+1}$ is the corresponding velocity vector, and Δt is the time step value Shi and Eberhart (1998a). The velocity vector of each particle is calculated as:

$$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + c_1 r_1 \frac{\left(\mathbf{p}_i^k - \mathbf{x}_i^k\right)}{\Delta t} + c_2 r_2 \frac{\left(\mathbf{p}_g^k - \mathbf{x}_i^k\right)}{\Delta t} \tag{4.8}$$

where $\mathbf{v}_i^k$ is the velocity of particle i at iteration k, r_1 and r_2 are random numbers between 0 and 1, $\mathbf{p}_i^k$ shows the best position of particle i which is obtained so far (personal best), and $\mathbf{p}_g^k$ corresponds to the global best position in the swarm at iteration k (global best). Three other terms are problem-dependent parameters. c_1 and c_2 represent trust parameters, indicating how much confidence the current particle has in itself (c_1 or cognitive parameter) and how much it has in the swarm (c_2 or

social parameter). Complete mathematical analysis of PSO is beyond the scope of this study; however, knowing $r_1, r_2 \in [0, 1]$, it is demonstrated in Oliveira et al. (2002) that the necessary and sufficient conditions for convergence of PSO can be derived as:

$$\begin{cases} 0 < (c_1 r_1 + c_2 r_2) < 4 \\ \dfrac{(c_1 r_1 + c_2 r_2)}{2} - 1 < w < 1 \end{cases} \tag{4.9}$$

where w is the inertia weight factor which plays an important role to control exploration and exploitation in the search space and convergence of PSO. A large amount of the inertia weight factor increases the ability of global search, while a small amount of the inertia weight factor facilitates a local search. The algorithm has a tendency to search more globally at the beginning and more locally at the end of the run course by reducing w from a relatively large value to a small value throughout the whole PSO process. It is also demonstrated that, compared with all fixed inertia weight factors, a dynamic inertia weight starting with a value close to 1 and dropping to 0.1 during the run course will give the best performance to PSO Shi and Eberhart (1998a).

The original form of PSO does not have essential capabilities for handling multiobjective (MO) optimization problems. But, recently, some different approaches have been proposed that used the basic concept of PSO to solve MO problems (MOPSO) Reyes-Sierra and Coello (2006). A MO optimization problem can be defined as the problem of finding a set of n vectors $\mathbf{X} = [X_1, X_2, ..., X_n]$ which satisfies m equality constraints $h_i(\mathbf{X}) = 0,\ i = 1, ..., m$ and p inequality constraints $g_j(\mathbf{X}) < 0,\ j = 1, ..., p$ and optimizes (minimize or maximize) a vector of k objective functions $F(\mathbf{X}) = [f_1(\mathbf{X}), f_1(\mathbf{X}), ..., f_k(\mathbf{X})]^T$, simultaneously. For MO problems, each objective function achieves its optimum at different points. Thus, the concept of pareto optimality is used to consider this type of problems (Kasprzak and Lewis, 2001). Considering a minimization problem, a decision vector $\mathbf{X}^* \in \mathbf{X}$ is called pareto optimal (non-dominated) solution if and only if there exists no $\mathbf{X}' \in \mathbf{X}$ such that $f_i(\mathbf{X}') \leq f_i(\mathbf{X}^*)$, for $i = 1, ..., k$, and with $f_i(\mathbf{X}') < f_i(\mathbf{X}^*)$ for at least one i.

The recent MOPSO works applied the concept of pareto optimality to select non-dominated particles in a swarm as leaders to converge the solutions to the true pareto front Reyes-Sierra and Coello (2006). In the case of MO problems, each particle could have a set of different leaders from which just one must be selected in order to update its position. The set of leaders (non-dominated solutions) found during the optimization process is usually stored in a different repository from the swarm. These stored non-dominated solutions are used as the leaders when the particles' positions have to be updated in the search space. Moreover, the repository's contents are usually considered as the final pareto optimal solutions of the algorithm.

We applied the locus-based adjacency genetic scheme proposed in Park and Song (1998) to construct particles in a swarm. In this graph-based scheme, shown in Figure 2, each particle is presented as a vector consisting of N elements where N is the number of data points. These elements can take values in the range $\{1, 2, ..., K\}$, where K is the number of clusters. Let a be the value in the connections vector that is assigned to the data point b. This assignment is interpreted as a link between data points a and b which means they belong to the same cluster in the clustering solution. All connected data points are then placed inside the same cluster and are assigned to their cluster number in the particle vector (Figure 4.2). The main advantage of using this scheme is that the number of clusters can be determined automatically for each particle. This particle is indeed representative of a candidate clustering solution. Therefore, it is possible for the algorithm to compare particles as clustering solutions with different number of clusters and lead them toward global optimum in just one run.

As defined in the previous section, a discrete particle presentation is used in this work to set up the clustering methodology. Therefore, it is not possible to use PSO in its original (continuous) form and we use the extension of PSO algorithm introduced in Jarboui et al. (2007) within our MO framework. This extended version is able to deal with a combinatorial representation of PSO by adding only one more parameter to continuous PSO. Here we introduce the algorithm briefly, but the comprehensive combinatorial PSO is defined in details in Jarboui et al. (2007). Let particle i be

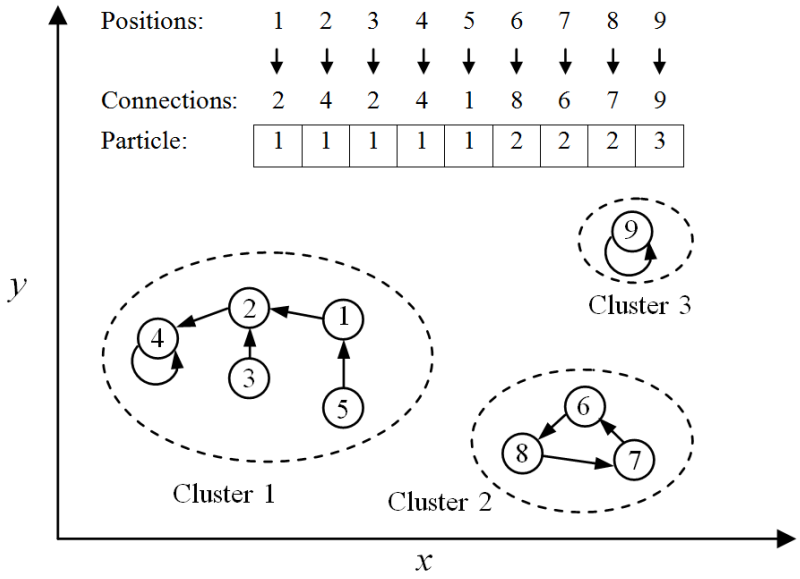

Figure 4-2: The locus-based adjacency method used to transform nine data points to a particle vector which represents a clustering solution consisting of three clusters.

shown as a candidate clustering solution vector $\mathbf{x}_i^k = \left\{x_{i1}^k, x_{i2}^k, ..., x_{iN}^k\right\}$ at iteration k. Another vector $\mathbf{y}_i^k = \left\{y_{i1}^k, y_{i2}^k, ..., y_{iN}^t\right\}$ is assigned to vector $\mathbf{x}$ that can take values in $\{-1, 0, 1\}$. Vector $\mathbf{y}_i^k$ transforms the discrete and continuous representations of particle i to each other and its jth element is defined by:

$$y_{ij}^k = \begin{cases} 1 & \text{if} \quad x_{ij}^k = p_{ij}^k, \\ -1 & \text{if} \quad x_{ij}^k = p_{jg}^k, \\ 1 \text{ or } -1 \text{ randomly} & \text{if} \quad x_{ij}^k = p_{ij}^k = p_{jg}^k, \\ 0 & \text{if otherwise.} \end{cases} \tag{4.10}$$

where p_{ij}^k and p_{jg}^k are the jth elements of personal best vector $\mathbf{p}_i^k = \left\{p_{i1}^k, p_{i2}^k, ..., p_{iN}^k\right\}$ and global best vector $\mathbf{p}_g^k = \left\{p_{1g}^k, p_{2g}^k, ...p_{Ng}^k\right\}$, respectively. Elements of the particle's velocity vector $\mathbf{v}_i^{k+1} = \left\{v_{i1}^{k+1}, v_{i2}^{k+1}, ..., v_{iN}^{k+1}\right\}$ are updated by:

$$v_{ij}^{k+1} = wv_{ij}^k + c_1 r_1 \left(-1 - y_{ij}^k\right) + c_2 r_2 \left(1 - y_{ij}^k\right) \tag{4.11}$$

Then the update solution y_{ij} is calculated by:

$$y_{ij}^{k+1} = \begin{cases} +1 & \text{if} \quad \lambda_{ij}^{k+1} > \alpha, \\ -1 & \text{if} \quad \lambda_{ij}^{k+1} < -\alpha, \\ 0 & \text{if} \quad \text{otherwise.} \end{cases} \tag{4.12}$$

where $\lambda_{ij}^{k+1} = y_{ij}^{k} + v_{ij}^{k+1}$ and α is the parameter that adjusts intensification and diversification of the algorithm. Finally, following rules result in elements of the new particle vector at iteration $k+1$:

$$x_{ij}^{k+1} = \begin{cases} p_{jg}^{k} & \text{if} \quad y_{ij}^{k+1} = 1, \\ p_{ij}^{k} & \text{if} \quad y_{ij}^{k+1} = -1, \\ \text{random value in } \{1, 2, ..., K\} & \text{if} \quad \text{otherwise.} \end{cases} \tag{4.13}$$

where K is the number of clusters. The small and large values of α incline the PSO toward intensification (setting x_{ij}^{k+1} to p_{jg}^{k} or p_{ij}^{k}) and diversification (setting x_{ij}^{k+1} to other than values).

The proposed Multiobjective Clustering Particle Swarm Optimization (MCPSO) algorithm is discussed in this section:

1. Initialization: A random distribution of initial swarm is generated by using the K-means algorithm with different number of clusters and an initial set of random velocities is assigned to them. The personal best for each particle is also initialized to the starting location of that particle.

2. Evaluations: The objective functionsâĂŹ (Equation 4.5 and Equation 4.6) values are evaluated for the given input vector of particles.

3. Analysis: The concept of MaxiMin strategy Simon (1958) is applied here to determine pareto optimal solutions. This method has some valuable advantages with regard to MO optimization problems. For instance, MaxiMin strategy needs no nich-

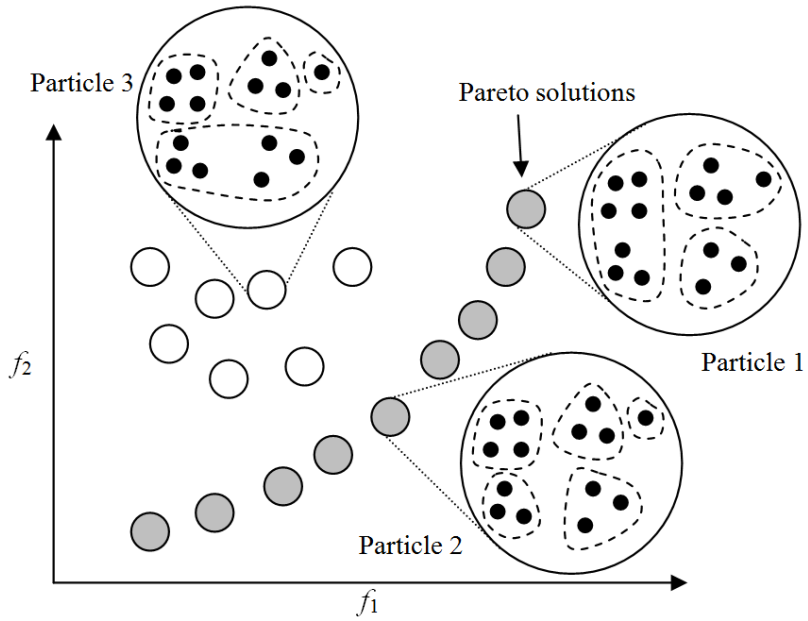

Figure 4-3: Three particles of the multiobjective clustering problem based on *connectivity*, f_1, and *cohesion*, f_2, as the objective functions. Particle 1 and particle 2 dominate particle 3 and can be selected as the candidate clustering (pareto) solutions considering the trade-off between two objective functions (f_1 should be maximized and f_2 should be minimized).

ing or clustering technique for preserving the diversity of optimal solutions along the pareto front. This property significantly decreases the computational cost of the optimizer. The applied strategy is described in detail in Li (2004) This method improves the convergence and diversity of the pareto optimal solutions. As mentioned before, the set of found non-dominated particles is stored in a repository different from the swarm.

4. Personal best selection: Selection of personal bests is straightforward. If the current particle vector $\mathbf{x}_i^k$ dominates its previous personal best particle, $\mathbf{p}_i^k$ at iteration $k+1$, then $\mathbf{p}_i^{k+1}$ is set to the current particle i vector.

5. Leader selection: As mentioned before, the leader selection is an important step of the MOPSO algorithm. A very simple approach is to randomly select one of the non-dominated particles as a new leader. In this way, each non-dominated solution can be considered as a leader, $\mathbf{p}_g^k$, in Equation 4.10. Here, we choose randomly a leader from the top portion of the best particles at each iteration.

6. Updating velocity vectors: The new velocity vector for each particle is calculated using Equation 11. In this study, the inertia weight factor, w_k, is dynamically adjusted throughout the optimization process Hart and Vlahopoulos (2010):

$$w_k = (w_{\max} - w_{\min}) \left(\frac{k_{\max} - k}{k_{\max}} \right) + w_{\min} \tag{4.14}$$

where $k_{\max}$ is the maximum number of iteration. Equation 4.14 contains three parameters which are defined to control the magnitude of the velocity vector during the optimization. When a bound constraint is violated, the algorithm sets the value of w_k to 0. Thus, the particle only uses its cognitive and social experiences to update its position and comes back into the feasible space.

7. Updating the particle vectors: The particle vectors in the swarm are updated using Equation 12 and Equation 13. The selected leader in step 5 is used as $\mathbf{p}_g^k$ in this equation.

8. Termination criterion: We applied a maximum number of iteration k_{max} to terminate the algorithm.

Figure 4.3 illustrates a set of clustering solutions that correspond to a tradeoff between two objective functions proposed. Pareto solutions are depicted as gray circles which dominate other clustering candidates shown as white circles. The solutions to the top right of the pareto front, such as particle 1, correspond to the particles that made effort to achieve solutions with higher number of neighboring data points in the same cluster (maximizing f_1 with lower number of clusters). In contrast, the solutions close to the bottom left of the pareto front, such as particle 2, represent particles that were more successful in creating compact clusters (minimizing f_2 with higher number of clusters). The other dominated particles, such as particle 3, were not successful to obtain well connected and compact clustering solutions in comparison with the pareto ones.

Pareto optimal solutions represent a set of solutions in the sense that improving the value in one objective function leads to a degradation in at least one other objective function. Therefore, a decision maker is required to make a tradeoff decision when

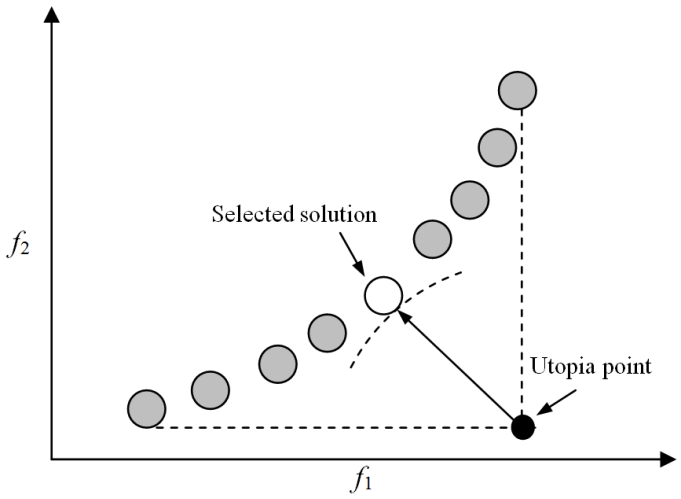

Figure 4-4: Distance technique to find final solution as the closest solution in the pareto set to the utopia location.

presented with a large finite number of pareto solutions. A decision maker usually chooses only one point or a few points based on some predefined criteria. Several methods exist to determine which member of pareto set should be selected as the final solution Kasprzak and Lewis (2001). Here, we use a distance technique that finds the solution in the pareto set which has the minimum distance from an ideal solution called utopia point. We define utopia location as the intersection point of the lines passed through the top right and bottom left solutions of the pareto front in the area of possible outcomes (Fig 4). In other words, coordinates of the utopia point is the best values obtained for each objective function during the optimization process.

4.3 Experimental Results and Discussion

In this section, we empirically evaluate the performance of MCPSO. The test datasets and clustering validity criteria are presented. Then, we perform a set of experiments in order to set the parameters of MCPSO by using some pilot datasets. Finally, the performance of our proposed algorithm is compared with other clustering algorithms. We implemented MCPSO in Python 2.7.6 on a Intel Core i7, with 2.4 GHz, 8 GB

Table 4.1: The MCPSO algorithm parameters

Parameter	Description	Value
Δt	Time step value	1
c_1	Cognitive parameter	1.42
c_2	Social parameter	1.63
r_1	Random value	$[0, 1]$
r_2	Random value	$[0, 1]$
$w_{\max}$	Maximum value of inertia weight factor	0.9
$w_{\min}$	Minimum value of inertia weight factor	0.4
α	Decision parameter	0.5
$k_{\max}$	Number of iterations	1000
$K_{\max}$	Maximum number of clusters	$150 \sim 10$
N	Number of particles	$2 \times K_{\max}$
$P_{\max}$	Maximum number of non-dominated solutions	150

RAM in Ubuntu 14.04 environment.

In this study, we use 27 different experimental datasets, collected from repositories in (Bache and Lichman, 2013; Franti, 2015), to test our method. These are 2- and higher-dimensional datasets with outliers and intra- and inter-clusters variations, which consist of different shapes of clusters (such as spiral, circular, elongated). Figure 4.5 presents some example of 2-dimensional datasets.

Rand (R) index is used to compare the performance of MCPSO with other clustering algorithms. The R index is a measure of the similarity between the obtained clusters and the known correct clusters Rand (1971):

$$\mathrm{R} = \frac{tp + tn}{tp + fp + tn + fn} \tag{4.15}$$

When the decision is to assign two data points to the same cluster, tp and fp stand for the number of correct and incorrect assignments, respectively, and when the decision is to assign two data points to different clusters, tn and fn are the number of correct and incorrect assignments, respectively. The R index thus has a value in the interval [0,1] and its value closer to 1 indicates better quality of the obtained clusters comparing with the true clusters.

In this study, a comprehensive experimental design is conducted in order to determine the best settings for the parameters of MCPSO. We selected 10 datasets

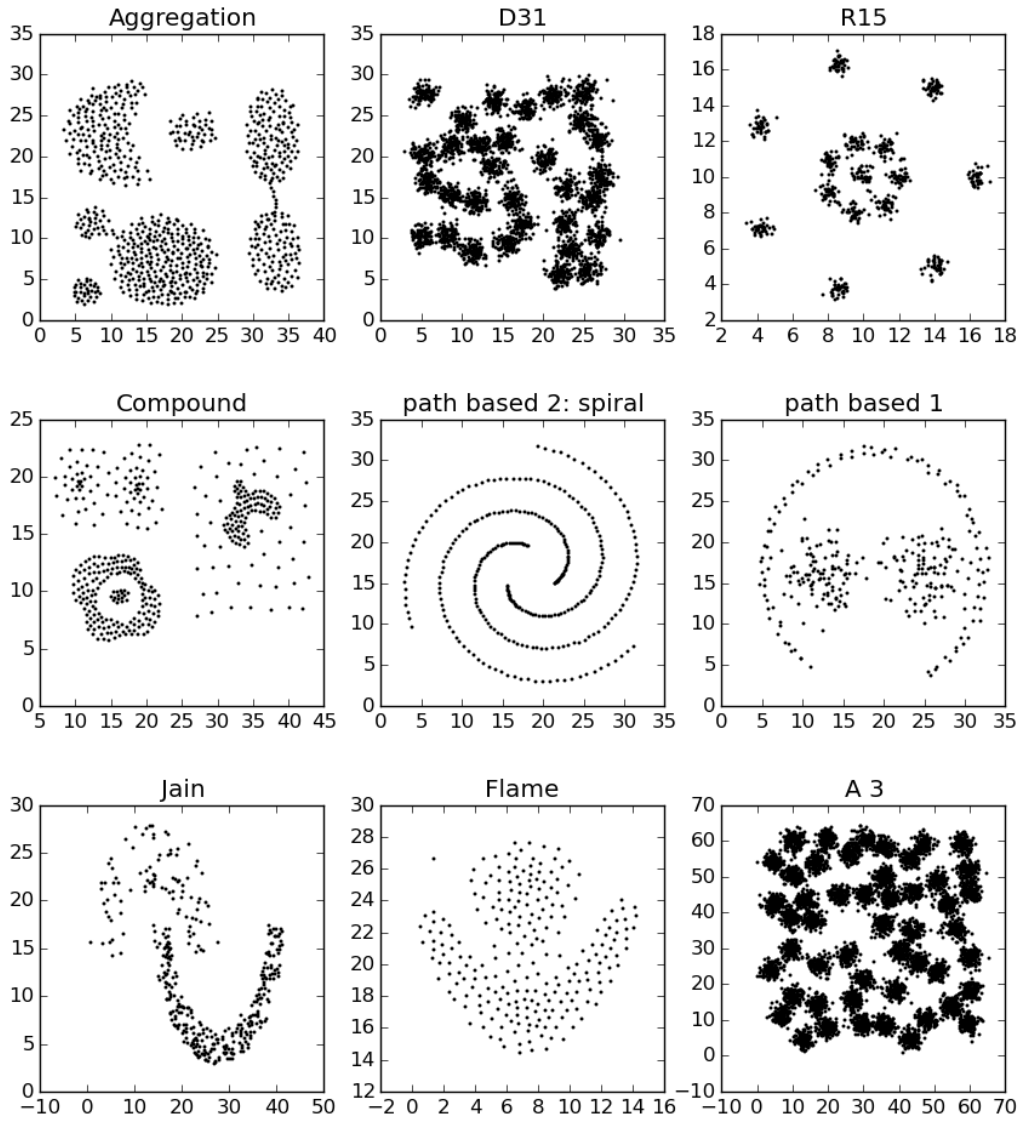

Figure 4-5: Exmple of 2-dimensional datasets with different shapes of clusters.

with different properties of all datasets and carried out 30 independent experiments. Parameter $K_{\max}$ is the maximum number of clusters expected in the dataset. We select 150 as a default which is quite high large value. When the dataset has few number of clusters, $K_{\max}$ can be set to smaller values, such as 10, to construct more accurate clustering solutions. We also set the number of particles, N, to twice $K_{\max}$, and recorded the iteration number in which the obtained non-dominated solutions had not been changed for 50 iterations. We experimentally assessed that this value is less than 1000 in all experiments and, therefore, we set it as the maximum iteration number $k_{\max}$. In all experiments, the final number of non-dominated clustering solutions are less than 100. Therefore, we selected $P_{\max} = 150$ as the size limit of number of pareto solutions which is more than sufficient to store all non-dominated solutions.

Investigating different random values of the cognitive and social parameters, we found that constant $c_1 = 1.42$ and $c_2 = 1.63$ resulted in better quality solutions in term of R. The same values are used for all the datasets. As discussed, Equation 4.14 adaptively controls parameter w_k. Thus, only its limits have to be chosen by the user. The maximum inertia weight factor, $w_{\max}$, is typically 0.9, as it allows to quickly find a global optimum. At each iteration, this value is repeatedly decreased (until $w_{\min} = 0.4$) to control the exploratory and the explotative nature of the algorithm during the optimization process Shi and Eberhart (1998b).

As in Jarboui et al. (2007), the authors do not describe how the decision parameter α is set, here we selected $\alpha = 0.5$ to keep the balance between intensification and diversification of the algorithm. A time step unit $\Delta t = 1$ is used throughout this study. All parameter settings used for MCPSO in our experimental study are given in Table 4.1.

In this section, the performance of the MCPSO algorithm is compared with the results of K-means, signle-linkage, DBSCAN Ester et al. (1996), and NC-closures İnkaya and Özdemirel (2013). K-means and single-linkage are the representatives of the partitional and hierarchical clustering algorithms, respectively. The density-based clustering is performed by DBSCAN and NC-closures is the method that uses the concept of neighborhood construction to create clusters by merging the obtained

Table 4.2: Mean and standard deviation of the Rand index (R) measured one the outputs of MCPSO (over 40 independent runs), K-means, single-linkage, DBSCAN, and NC-closures.

Dataset	K	D	**MCPSO**	**K-means**	**single-linkage**	**DBSCAN**	**NC-closures**
Jain	2	2	0.95 ± 0.008	0.89 ± 0.012	0.75 ± 0.004	**0.96±0.002**	0.90 ± 0.006
Flame	2	2	**0.91±0.021**	0.83 ± 0.006	0.85 ± 0.003	0.89 ± 0.051	0.87 ± 0.042
Thyoid	2	5	**0.90±0.015**	0.82 ± 0.007	0.80 ± 0.017	0.78 ± 0.002	0.81 ± 0.012
Wdbc	2	32	0.92 ± 0.009	0.77 ± 0.045	**0.93±0.004**	0.86 ± 0.014	0.84 ± 0.027
Pathbased	3	2	**0.94±0.006**	0.84 ± 0.005	0.85 ± 0.037	0.90 ± 0.005	0.89 ± 0.007
Spiral	3	2	0.89 ± 0.003	0.79 ± 0.084	0.87 ± 0.002	**0.93±0.020**	0.81 ± 0.007
Compound	6	2	**0.92±0.075**	0.81 ± 0.003	0.75 ± 0.052	0.80 ± 0.016	0.79 ± 0.014
Aggregation	7	2	**0.97±0.028**	0.94 ± 0.003	0.89 ± 0.014	0.88 ± 0.023	0.85 ± 0.017
Glass	7	9	0.95 ± 0.007	**0.96±0.014**	0.90 ± 0.004	0.92 ± 0.006	0.90 ± 0.016
Unbalance	8	2	**0.97±0.002**	0.95 ± 0.032	0.88 ± 0.062	0.91 ± 0.088	0.90 ± 0.023
Yeast	10	8	**0.91±0.035**	0.85 ± 0.005	0.783 ± 0.037	0.73 ± 0.058	0.77 ± 0.014
R15	15	2	**0.94±0.042**	0.91 ± 0.037	0.87 ± 0.009	0.89 ± 0.014	0.90 ± 0.007
S 1	15	2	**0.85±0.005**	0.79 ± 0.014	0.64 ± 0.014	0.58 ± 0.036	0.76 ± 0.004
S 2	15	2	**0.80±0.014**	0.64 ± 0.075	0.48 ± 0.003	0.41 ± 0.032	0.56 ± 0.042
S 3	15	2	**0.76±0.027**	0.57 ± 0.034	0.39 ± 0.098	0.32 ± 0.063	0.49 ± 0.007
S 4	15	2	**0.79±0.018**	0.50 ± 0.031	0.33 ± 0.005	0.37 ± 0.011	0.43 ± 0.003
Dim032	16	32	0.90 ± 0.001	**0.92±0.006**	0.86 ± 0.003	0.90 ± 0.007	0.83 ± 0.13
Dim064	16	64	**0.90±0.018**	0.89 ± 0.023	0.85 ± 0.033	0.89 ± 0.017	0.84 ± 0.074
Dim128	16	128	**0.88±0.038**	0.87 ± 0.009	0.81 ± 0.000	0.85 ± 0.017	0.83 ± 0.009
Dim256	16	256	**0.83±0.001**	0.80 ± 0.044	0.78 ± 0.006	0.81 ± 0.005	0.75 ± 0.033
Dim512	16	512	**0.80±0.022**	0.76 ± 0.000	0.70 ± 0.007	0.73 ± 0.027	0.70 ± 0.063
A 1	20	2	0.92 ± 0.033	**0.94±0.035**	0.88 ± 0.017	0.92 ± 0.011	0.90 ± 0.003
A 2	35	2	0.90 ± 0.000	0.91 ± 0.017	0.83 ± 0.046	**0.90±0.098**	0.86 ± 0.000
A 3	50	2	**0.88±0.002**	0.86 ± 0.040	0.79 ± 0.066	0.86 ± 0.011	0.81 ± 0.000
Birch 1	100	2	**0.82±0.042**	0.53 ± 0.037	0.30 ± 0.067	0.36 ± 0.001	0.49 ± 0.003
Birch 2	100	2	**0.76±0.087**	0.59 ± 0.034	0.32 ± 0.067	0.41 ± 0.000	0.61 ± 0.003
Birch 3	100	2	**0.72±0.018**	0.51 ± 0.040	0.31 ± 0.007	0.44 ± 0.098	0.45 ± 0.023

Table 4.3: Mean and standard deviation of the average number of clusters (Av. K) determined by MCPSO (over 40 independent runs), K-means, single-linkage, DBSCAN, and NC-closures.

Dataset	K	D	**MCPSO**	**K-means**	**single-linkage**	**DBSCAN**	**NC-closures**
Jain	2	2	2.13 ± 0.001	2.27 ± 0.007	5.35 ± 0.001	**2.10±0.000**	2.20 ± 0.003
Flame	2	2	**2.14±0.003**	3.14 ± 0.006	4.21 ± 0.011	2.41 ± 0.005	2.26 ± 0.002
Thyoid	2	5	**2.56±0.008**	3.36 ± 0.003	9.63 ± 0.010	13.68 ± 0.012	5.11 ± 0.002
Wdbc	2	32	2.50 ± 0.005	10.42 ± 0.005	**2.47±0.001**	4.89 ± 0.009	3.65 ± 0.007
Pathbased	3	2	**3.26±0.003**	8.36 ± 0.022	8.45 ± 0.003	6.12 ± 0.003	5.44 ± 0.006
Spiral	3	2	3.63 ± 0.002	10.98 ± 0.008	14.32 ± 0.022	**3.43±0.006**	7.42 ± 0.001
Compound	6	2	**6.78±0.009**	12.63 ± 0.003	9.32 ± 0.005	35.12 ± 0.006	24.62 ± 0.004
Aggregation	7	2	**7.22±0.007**	7.86 ± 0.004	15.63 ± 0.006	19.37 ± 0.003	10.42 ± 0.001
Glass	7	7	7.64 ± 0.004	**7.37±0.008**	15.74 ± 0.023	9.63 ± 0.001	17.49 ± 0.006
Unbalance	8	2	**8.31±0.008**	9.11 ± 0.006	0.88 ± 0.006	12.84 ± 0.007	10.42 ± 0.066
Yeast	10	8	**11.62±0.007**	15.32 ± 0.003	27.32 ± 0.004	32.14 ± 0.013	25.13 ± 0.003
R15	15	2	**15.96±0.006**	17.68 ± 0.004	35.24 ± 0.017	24.32 ± 0.004	18.34 ± 0.002
S 1	15	2	**19.34±0.002**	25.34 ± 0.004	32.13 ± 0.004	39.63 ± 0.007	29.42 ± 0.002
S 2	15	2	**21.67±0.035**	31.46 ± 0.007	44.12 ± 0.004	49.13 ± 0.007	34.34 ± 0.004
S 3	15	2	**27.13±0.007**	41.63 ± 0.007	47.63 ± 0.009	37.36 ± 0.007	30.14 ± 0.004
S 4	15	2	**30.11±0.005**	51.63 ± 0.007	87.16 ± 0.001	76.61 ± 0.011	64.38 ± 0.002
Dim032	16	32	20.14 ± 0.003	**18.34±0.003**	28.46 ± 0.074	25.74 ± 0.004	25.46 ± 0.007
Dim064	16	64	**22.46±0.006**	37.63 ± 0.009	35.64 ± 0.006	41.63 ± 0.037	39.14 ± 0.003
Dim128	16	128	**23.63±0.007**	28.34 ± 0.003	35.14 ± 0.002	31.02 ± 0.008	26.14 ± 0.002
Dim256	16	256	**24.63±0.004**	29.13 ± 0.002	51.33 ± 0.002	34.63 ± 0.002	30.11 ± 0.006
Dim512	16	512	**29.34±0.004**	35.41 ± 0.011	43.45 ± 0.023	54.78 ± 0.032	31.06 ± 0.047
A 1	20	2	22.14 ± 0.006	**21.36±0.046**	42.16 ± 0.074	24.16 ± 0.009	29.75 ± 0.037
A 2	35	2	39.63 ± 0.004	47.13 ± 0.063	57.14 ± 0.008	**42.14±0.013**	40.16 ± 0.006
A 3	50	2	**57.32±0.005**	64.35 ± 0.046	86.47 ± 0.046	60.19 ± 0.037	63.18 ± 0.007
Birch 1	100	2	**114.63±0.007**	135.32 ± 0.008	151.42 ± 0.041	172.63 ± 0.084	169.41 ± 0.033
Birch 2	100	2	**125.74±0.087**	163.47 ± 0.011	206.14 ± 0.078	187.96 ± 0.033	154.74 ± 0.063
Birch 3	100	2	**136.45±0.016**	179.84 ± 0.047	230.67 ± 0.011	241.63 ± 0.003	200.41 ± 0.067

closures. K-means and single-linkage are run with different number of clusters as an input. We vary this number between 2-10% of the number of points in the datasets with increments of 1, and the best R index value is selected for each algorithm. The number of clusters found and quality of clustering solutions of DBSCAN are affected by a parameter named *MinPts* (within a range of 1 to 15) Dunn (1974). Therefore, we run DBSCAN with all possible *MinPts* values and again select the best R value as the final result. In order to minimize the effect of the stochastic nature of MCPSO on the R index and on number of clusters, 40 independent trails are ran and (as explained) the decision making process is implemented to select the final clustering from the obtained non-dominated solutions in each run. Finally, the solution corresponding to the best R value is selected as the final clustering.

The results over 27 datasets are summarized in Table 4.2 and Table 4.3. It can be seen that single-linkage, K-means, and DBSCAN have better performance on only a few datasets (Wdbc, Glass, Dim032, A 1, A2, Jain , and Spiral) in terms of R and the average number of clusters (Av. K). There are elongated shape and spatially well-separated clusters in Wbdc dataset which is the cluster model assumed by the single-linkage method. Therefore, this algorithm shows a good performance on Wbdc. However, MCPSO has obtained the second best values of R and K among the other algorithms dealing with this dataset. Glass, Dim032, and A 1 contain spherical Gaussian clusters which can be assumed as the proper clustering shapes for K-means. However, it is shown that by increasing the dimensions and/or the actual number of clusters of the Dim (Dim064, Dim128, Dim256, and Dim512) and A (A 2 and A 3) problem sets, MCPSO slightly surpasses K-means. DBSCAN is the algorithm which shows a good performance dealing with arbitrary shape clusters. Even if a cluster is completely surrounded (but not connected) by a different cluster, this method is capable of obtaining promising results. Jain, Spiral, and A 2 are the representative of such clustering problems. Again, it is presented that MCPSO resulted in R and Av. K values which are very close to the results of DBSCAN on Jain, Spiral, and A 2.

High dimensional Dim sets, Birch and S series, R15, and some other clustering

Table 4.4: Average running time of MCPSO, K-means, single-linkage, DBSCAN, and NC-closures implemented in Python 2.7.6 on a Intel Core i7, with 2.4 GHz, 8 GB RAM in Ubuntu 14.04 environment. Average characteristic values in all datasets are shown to provide an overview of the number of data points (N), dimensions (D), and clusters (K) considered in our experimental study.

Average characteristics			Average Running Time (sec)				
N	D	K	MCPSO	K-means	single-linkage	DBSCAN	NC-closures
5183.85	40	22.11	510.12	12.34	21.36	16.14	246.15

problems in Table 4.2 and Table 4.3 can be considered as difficult clustering problems which contain clusters with large differences in densities, overlaps, and many outliers. K-means and DBSCAN tend to subdivide elongated cluster shapes and single-linkage tends to isolate outliers on datasets with overlaps. The average representative running times for the five algorithms in our experiments are presented in Table 4.4. The average values of the number of data points, clusters, and dimensions are calculated to show the characteristics of the all datasets. In general, experimental results point out that the MCPSO algorithm is much more time consuming than K-means, DBSCAN, and single-linkage. This is basically due to its stochastic nature and to the random behavior of particles. Therefore, the computational time can be considered as the main limitation of the MCPSO algorithm which requires improvement.

Low values of R and the significant increase in K (average) depend on the drawbacks of K-means, single-linkage, and DBSCAN dealing with difficult clustering problems. On the other hand, the results for MCPSO show the strong performance and robustness expected from the simultaneous optimization of the two objectives. While the NC procedure takes care cluster *connectivity* within the first objective function, the second objective function helps MCPSO to achieve well-separated clusters. The results obtained by NC-closures display that this method shows a more robust behavior, in comparison with K-means, single-linkage, and DBSCAN, in the task of dealing with outliers in the datasets. However, as the separation is not independently taken into account as an objective, NC-closures is not successful to overcome all other methods in any problem.

5

Conclusions and open issues

Different swarm based clustering methodologies have been addressed in this thesis. They represent improvements with respect to the state-of-the-art and conventional algorithms. In the first part of the thesis, including Chapter 1, the state-of-the-art swarm intelligence clustering algorithms have been presented.

In Chapter 2, it is shown that despite The K-means algorithm is a simple and efficient clustering method that has been applied to many engineering problems, it suffers from several drawbacks due to its choice of initializations. This chapter has developed a combined algorithm for solving the clustering problem which is based on the combination of Artificial Bee Colony algorithm and K-means technique. The algorithm has been implemented and tested on several well known real datasets and preliminary computational experience is very encouraging. In other word it has been proved that the ABC+K-means algorithm will definitely converge to optimal solution in almost runs. The ABC+K-means clustering algorithm developed in this paper can be applied when the number of clusters is known a prior.

The main motivation for Chapter 3 was utilizing the notion of opposition values to accelerate an ant-based algorithm called API (after the name of Pachycondyla API-calis ants) for crisp clustering of real-world datasets. The performance of the proposed algorithm is studied by comparing it with three different state-of-the-art clustering algorithms and original version of API. The obtained results over five benchmark datasets show that the enhanced API algorithm, called OBAPI, is able to outperform

four other algorithms over a majority of the datasets. The proposed method can significantly decrease the number of function evaluations while improving the quality of solutions in most cases without adding any new parameter to the original API. Moreover, OBAPI is able to automatically find the optimal number of clusters and does not need to know them in advance. It is also important to note that the results discussed in this work are only examined and valid for the clustering problems used here. In other words, the proposed technique makes a heuristic method which is only studied for clustering datasets with average number of features. As a part of our future work, we plan to enhance and apply the OBAPI algorithm in bi- or multi-clustering of some gene expression datasets which consist of high dimensional data.

In Chapter 4, a multiobjective particle swarm optimization algorithm for partitional clustering of different datasets is proposed. In order to achieve this goal, two objective functions are defined to consider the concepts of *connectivity* and *cohesion*. Each particle represents a possible clustering solution in the swarm and a set of non-domintaed particles are obtained after each run of the algorithm. Finally, a simple decision maker selects the best solution giving equal credits to each objective. The performance of the proposed algorithm is studied in comparison with four different conventional clustering algorithms. The obtained results over 27 benchmark datasets show that the proposed method, called MCPSO, is able to outperform the other algorithms in terms of precision and robustness over a majority of the datasets. Moreover, MCPSO does not need to know the number of clusters in advance, as it is able to automatically find it. As a part of our future work, we plan to enhance and apply the MCPSO algorithm to some real-world datasets which consist of more high dimensional data.

Bibliography

I. E. Evangelou, D. G. Hadjimitsis, A. A. Lazakidou, and C. Clayton, "Data mining and knowledge discovery in complex image data using artificial neural networks," in *in Proc. Workshop Complex Reason. Geogr. Data, Paphos*. Citeseer, 2001.

R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.

K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.

P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical programming*, vol. 79, no. 1-3, pp. 191–215, 1997.

T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote sensing and image interpretation*. John Wiley & Sons, 2014.

M. Rao, "Cluster analysis and mathematical programming," *Journal of the American statistical association*, vol. 66, no. 335, pp. 622–626, 1971.

E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *Computers, IEEE Transactions on*, vol. 100, no. 1, pp. 68–86, 1971.

T. M. Mitchell *et al.*, "Machine learning. wcb," 1997.

E. Falkenauer, *Genetic algorithms and grouping problems*. John Wiley & Sons, Inc., 1998.

S. Paterlini and T. Minerva, "Evolutionary approaches for cluster analysis," in *Soft Computing Applications*. Springer, 2003, pp. 165–176.

O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer, 2005, vol. 2.

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

S. Mitra, S. K. Pal, and P. Mitra, "Data mining in soft computing framework: a survey," *Neural Networks, IEEE Transactions on*, vol. 13, no. 1, pp. 3–14, 2002.

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.

K. Alsabti, S. Ranka, and V. Singh, "An efficient k-means clustering algorithm," 1997.

C. D. Nguyen and K. J. Cios, "Gakrem: a novel hybrid clustering algorithm," *Information Sciences*, vol. 178, no. 22, pp. 4205–4227, 2008.

T. Niknam, B. B. Firouzi, and M. Nayeripour, "An efficient hybrid evolutionary algorithm for cluster analysis," in *World Applied Sciences Journal*. Citeseer, 2008.

T. Niknam, J. Olamaie, and B. Amiri, "A hybrid evolutionary algorithm based on aco and sa for cluster analysis," *Journal of Applied Science*, vol. 8, no. 15, pp. 2695–2702, 2008.

T. Niknam, B. Amiri, J. Olamaei, and A. Arefi, "An efficient hybrid evolutionary optimization algorithm based on pso and sa for clustering," *Journal of Zhejiang University Science A*, vol. 10, no. 4, pp. 512–519, 2009.

F. S. Chahine, *A genetic algorithm that exchanges neighboring centers for fuzzy c-means clustering*. Nova Southeastern University, 2012.

M. Fathian, B. Amiri, and A. Maroosi, "Application of honey-bee mating optimization algorithm on clustering," *Applied Mathematics and Computation*, vol. 190, no. 2, pp. 1502–1513, 2007.

K. Krishna and M. N. Murty, "Genetic k-means algorithm," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 3, pp. 433–439, 1999.

U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern recognition*, vol. 33, no. 9, pp. 1455–1465, 2000.

P. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187–195, 2004.

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. Basturk, "Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing*. Springer, 2007, pp. 789–798.

——, "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.

A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem," *Swarm Intelligence: Focus on Ant and particle swarm optimization*, pp. 113–144, 2007.

M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. H. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Information Sciences*, vol. 181, no. 16, pp. 3459–3475, 2011.

S. Z. Selim and M. A. Ismail, "K-means-type algorithms: a generalized convergence theorem and characterization of local optimality," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 1, pp. 81–87, 1984.

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

R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

M. Forina *et al.*, "Arvus-an extendible package for data exploration, classification and correlation," *Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno*, vol. 16147, 1991.

T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

A. Dalli, "Adaptation of the f-measure to cluster based lexicon quality evaluation," in *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?* Association for Computational Linguistics, 2003, pp. 51–56.

J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Elsevier, 2011.

H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 450–465, 1999.

Y. Leung, J.-S. Zhang, and Z.-B. Xu, "Clustering by scale-space filtering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 12, pp. 1396–1410, 2000.

O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, 2004.

A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern recognition*, vol. 35, no. 6, pp. 1197–1208, 2002.

S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 38, no. 1, pp. 218–237, 2008.

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

M. G. Omran, A. P. Engelbrecht, and A. Salman, "Dynamic clustering using particle swarm optimization with application in unsupervised image classification," in *Proceedings of world academy of science, engineering and technology*, vol. 9, 2005.

M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.

E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.

N. Monmarché, G. Venturini, and M. Slimane, "On how pachycondyla apicalis ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 937–946, 2000.

I. Ciornei and E. Kyriakides, "Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 1, pp. 234–245, 2012.

S. Aupetit, N. Monmarché, M. Slimane, and P. Liardet, "An exponential representation in the api algorithm for hidden markov models training," in *Artificial Evolution*. Springer, 2005, pp. 61–72.

N. Monmarché, M. Slimane, and G. Venturini, "On improving clustering in numerical databases with artificial ants," in *Advances in Artificial Life*. Springer, 1999, pp. 626–635.

E. D. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*. MIT Press, 1994, pp. 501–508.

M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, vol. 5, no. 2, pp. 137–172, 1999.

N. Monmarché, M. Slimane, and G. Venturini, "Antclass: discovery of clusters in numeric data by an hybridization of an ant colony with the kmeans algorithm," 1999.

V. Ramos and J. J. Merelo, "Self-organized stigmergic document maps: Environment as a mechanism for context learning," *arXiv preprint cs/0412075*, 2004.

N. Labroche, N. Monmarche, and G. Venturini, "A new clustering algorithm based on the chemical recognition system of ants," in *ECAI*, 2002, pp. 345–349.

P. M. Kanade and L. O. Hall, "Fuzzy ants as a clustering concept," in *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*. IEEE, 2003, pp. 227–232.

J. Handl, J. D. Knowles, and M. Dorigo, "On the performance of ant-based clustering." in *HIS*, 2003, pp. 204–213.

J. Handl and B. Meyer, "Improved ant-based clustering and sorting in a document retrieval interface," in *Parallel Problem Solving from NatureâĂŤPPSN VII*. Springer, 2002, pp. 913–923.

N. Labroche, N. Monmarché, and G. Venturini, "Antclust: ant clustering and web usage mining," in *Genetic and Evolutionary ComputationâĂŤGECCO 2003*. Springer, 2003, pp. 25–36.

H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "Anttree: a new model for clustering with artificial ants," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4. IEEE, 2003, pp. 2642–2647.

C.-F. Tsai, C.-W. Tsai, H.-C. Wu, and T. Yang, "Acodf: a novel data clustering approach for data mining in large databases," *Journal of Systems and Software*, vol. 73, no. 1, pp. 133–145, 2004.

V. Hartmann, "Evolving agent swarms for clustering and sorting," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 217–224.

L. Admane, K. Benatchba, M. Koudil, L. Siad, and S. Maziz, "Antpart: an algorithm for the unsupervised classification problem using ants," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 16–28, 2006.

X. Huang, Y. Yang, and X. Niu, "Towards improving ant-based clustering-an chaotic ant clustering algorithm," in *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*. IEEE, 2007, pp. 421–424.

Y. Wang, R.-W. Li, B. Li, P.-J. Zhang, and Y.-H. Li, "Research on an ant colony isodata algorithm for clustering analysis in real time computer simulation," in *Digital Media and its Application in Museum & Heritages, Second Workshop on*. IEEE, 2007, pp. 223–229.

Z. Tao, L. Xiaodong, and Z. Zaixu, "An improved clustering algorithm based on ant colony approach," in *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*. IEEE, 2007, pp. 437–440.

B. Boryczka, "Ant clustering algorithm, intelligent information systems," 2008.

A. Ghosh, A. Halder, M. Kothari, and S. Ghosh, "Aggregation pheromone density based data clustering," *Information Sciences*, vol. 178, no. 13, pp. 2816–2831, 2008.

Z. Sadeghi, M. Teshnehlab, and M. M. Pedram, "K-ants clustering-a ew strategy based on ant clustering," in *Scope of the Symposium*, 2008, p. 45.

Q. Chen and J. Mo, "Optimizing the ant clustering model based on k-means algorithm," in *Computer Science and Information Engineering, 2009 WRI World Congress on*, vol. 3. IEEE, 2009, pp. 699–702.

Z. Weili, "An improved entropy-based ant clustering algorithm," in *Information Engineering, 2009. ICIE'09. WASE International Conference on*, vol. 2. IEEE, 2009, pp. 41–44.

H. R. Tizhoosh, "Opposition-based reinforcement learning." *JACIII*, vol. 10, no. 4, pp. 578–585, 2006.

D. Craigen, S. Gerhart, and T. Ralston, "An international survey of industrial applications of formal methods," in *Z User Workshop, London 1992*. Springer, 1993, pp. 1–5.

M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 187–194.

D. L. Davies and D. W. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 224–227, 1979.

C.-H. Chou, M.-C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," *Pattern Analysis and Applications*, vol. 7, no. 2, pp. 205–220, 2004.

C. Blake and C. J. Merz, "{UCI} repository of machine learning databases," 1998.

J. Han and M. Kamber, *Data mining: concepts and techniques, the Morgan Kaufmann Series in data management systems*. Morgan Kaufmann, 2000.

J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. Morgan Kaufman, 2001.

P. K. Bharne, V. Gulhane, and S. K. Yewale, "Data clustering algorithms based on swarm intelligence," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 4. IEEE, 2011, pp. 407–411.

C. Grosan, A. Abraham, and M. Chis, "Swarm intelligence in data mining," in *Studies in Computational Intelligence*. Springer, 2006, vol. 34, pp. 1–20.

A. Abraham, S. Das, and S. Roy, "Swarm intelligence algorithms for data clustering," in *Soft Computing for Knowledge Discovery and Data Mining*. Springer, 2008, pp. 279–313.

M. G. Omran, A. Salman, and A. P. Engelbrecht, "Dynamic clustering using particle swarm optimization with application in image segmentation," *Pattern Analysis and Applications*, vol. 8, no. 4, pp. 332–344, 2006.

R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

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.

N. I. Ghali, N. El-Dessouki, A. Mervat, and L. Bakrawi, "Exponential particle swarm optimization approach for improving data clustering," *International Journal of Electrical, Computer, and Systems Engineering*, vol. 3, no. 4, pp. 208–212, 2009.

N. A. Latiff, C. Tsimenidis, B. S. Sharif, and C. Ladha, "Dynamic clustering using binary multi-objective particle swarm optimization for wireless sensor networks," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*. IEEE, 2008, pp. 1–5.

S. Janson and D. Merkle, *A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking*. Springer, 2005.

A. Abubaker, A. Baharum, and M. Alrefaei, "Automatic clustering using multi-objective particle swarm and simulated annealing," *PloS one*, vol. 10, no. 7, pp. 205–220, 2015.

S. Sarkar, A. Roy, and B. S. Purkayastha, "Application of particle swarm optimization in data clustering: A survey," *International Journal of Computer Applications*, vol. 65, no. 25, pp. 38–46, 2013.

E. M. Kasprzak and K. E. Lewis, "Pareto analysis in multiobjective optimization using the collinearity theorem and scaling method," *Structural and Multidisciplinary Optimization*, vol. 22, no. 3, pp. 208–218, 2001.

G. Hamerly and C. Elkan, "Learning the k in k-means," in *Neural Information Processing Systems*. MIT Press, 2003.

D. L. Davies and D. W. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 1, no. 2, pp. 224–227, 1979.

W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 1, pp. 56–76, 2007.

E. M. Voorhees, "The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval," Cornell University, Tech. Rep., 1985.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceeding the 2nd International Conference on Knowledge Discovery and Data Mining*, vol. 96, 1996, pp. 226–231.

K. İnkaya and Özdemirel, "A neighborhood construction algorithm for the clustering problem," Middle East Technical University, Ankara, Turkey, Tech. Rep., 2013.

Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on.* IEEE, 1998, pp. 69–73.

P. Oliveira, J. Cunha, and J. Coelho, "Design of pid controllers using the particle swarm algorithm," in *Twenty-first IASTED international conference: modelling, identification, and control (MIC 2002), Innsbruck, Austria*, 2002.

M. Reyes-Sierra and C. C. Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.

Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proceedings of the third annual conference on genetic programming*, 1998, pp. 568–575.

B. Jarboui, M. Cheikh, P. Siarry, and A. Rebai, "Combinatorial particle swarm optimization (cpso) for partitional clustering problem," *Applied Mathematics and Computation*, vol. 192, no. 2, pp. 337–345, 2007.

H. Simon, "Games and decisions-introduction and critical survey-luce, rd, raiffa, h," 1958.

X. Li, "Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function," in *Genetic and Evolutionary Computation–GECCO 2004*. Springer, 2004, pp. 117–128.

C. G. Hart and N. Vlahopoulos, "An integrated multidisciplinary particle swarm optimization approach to conceptual ship design," *Structural and Multidisciplinary Optimization*, vol. 41, no. 3, pp. 481–494, 2010.

K. Bache and M. Lichman, "UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA," 2013.

P. Franti, "Speech and Image Processing Unit, clustering datasets. School of Computing, University of Eastern Finland," 2015.

Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary programming VII*. Springer, 1998, pp. 591–600.

J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

M. Zareh, C. Seatzu, and M. Franceschelli, "Consensus of second-order multi-agent systems with time delays and slow switching topology," in *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on*, 2013, pp. 269–275.

——, "Consensus on the average in arbitrary directed network topologies with time-delays," in *4th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2013, pp. 342–347.

M. Zareh, D. V. Dimarogonas, M. Franceschelli, K. H. Johansson, and C. Seatzu, "Consensus in multi-agent systems with non-periodic sampled-data exchange and uncertain network topology," in *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on*. IEEE, 2014, pp. 411–416.

——, "Consensus in multi-agent systems with second-order dynamics and non-periodic sampled-data exchange," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–8.

List of Publications Related to the Thesis

Published papers

- G. Armano, M.R. Farmani, *Multiobjective Clustering Analysis using Particle Swarm Optimization*, Expert Systems with Applications, in press, 2016.
- M.R Farmani, G. Armano, *Clustering Analysis using Opposition-Based API Algorithm*, IJCCI, the 7th International Joint Conference on Computational Intelligence, Lisbon 2015.
- G. Armano, M.R. Farmani, *Clustering Analysis with Combination of Artificial Bee Colony Algorithm and k-means Technique*, 6th International Conference on Computer Science and Information Technology (ICCSIT), Paris 2013.