AN INTERACTIVE PREFERENCE BASED MULTIOBJECTIVE EVOLUTIONARY ALGORITHM FOR THE CLUSTERING PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KEREM DEMİRTAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

MAY 2011

Approval of the thesis:

AN INTERACTIVE PREFERENCE BASED EVOLUTIONARY ALGORITHM FOR THE CLUSTERING PROBLEM

submitted by **KEREM DEMİRTAŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Sinan Kayalıgil _________________
Head of Department, **Industrial Engineering**

Prof. Dr. Nur Evin Özdemirel _________________
Supervisor, **Industrial Engineering Dept., METU**

Assoc. Prof. Dr. Esra Karasakal _________________
Co-Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Assist. Prof. Dr. Cem İyigün _________________
Industrial Engineering Dept., METU

Prof. Dr. Nur Evin Özdemirel _________________
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Esra Karasakal _________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Sinan Gürel _________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Tuğba Taşkaya Temizel _________________
Information Systems Dept., METU

Date: 26/05/2011

ABSTRACT

AN INTERACTIVE PREFERENCE BASED EVOLUTIONARY ALGORITHM FOR THE CLUSTERING PROBLEM

Demirtaş, Kerem

M.Sc., Department of Industrial Engineering

Supervisor: Prof. Dr. Nur Evin Özdemirel

Co-Supervisor: Assoc. Prof. Dr. Esra Karasakal

May 2011, 67 Pages

We propose an interactive preference-based evolutionary algorithm for the clustering problem. The problem is highly combinatorial and referred to as NP-Hard in the literature. The goal of the problem is putting similar items in the same cluster and dissimilar items into different clusters according to a certain similarity measure, while maintaining some internal objectives such as compactness, connectivity or spatial separation. However, using one of these objectives is often not sufficient to detect different underlying structures in different data sets with clusters having arbitrary shapes and density variations. Thus, the current trend in the clustering literature is growing into the use of multiple objectives as the inadequacy of using a single objective is understood better. The problem is also difficult because the optimal solution is not well defined. To the best of our knowledge, all the multiobjective evolutionary algorithms for the clustering problem try to generate the whole Pareto optimal set. This may not be very useful since majority of the solutions in this set may be uninteresting when presented to the decision maker. In this study, we incorporate the preferences of the decision maker into a well known multiobjective evolutionary algorithm, namely SPEA-2, in the optimization process using reference points and achievement scalarizing functions to find the target clusters.

Keywords: Clustering, Multiobjective Optimization, Metaheuristics, Evolutionary Algorithms, SPEA2, Preference Based

ÖZ

KÜMELEME PROBLEMİ İÇİN ETKİLEŞİMLİ TERCİH TABANLI BİR ÇOK AMAÇLI EVRİMSEL ALGORİTMA

Demirtaş, Kerem

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Nur Evin Özdemirel

Ortak Tez Yöneticisi: Doç. Dr. Esra Karasakal

Mayıs 2011, 67 Sayfa

Bu çalışmada kümeleme problem için etkileşimli, tercih tabanlı, çok amaçlı bir evrimsel algoritma önermekteyiz. Kümeleme problemi gayet kombinasyonal olup literatürde NP-Hard olarak geçmektedir. Problemin amacı belirli bir benzerlik ölçüsüne göre benzer öğeleri aynı kümelere, benzeşmeyen öğeleri farklı kümelere koyarken sıkılık, bağlanabilirlik ve ayrışma gibi içsel amaçları sağlamaktır. Ancak bahsi geçen amaçlardan bir tanesinin kullanımı, rastgele şekilleri olan veya yoğunluk farkı bulunan kümeleri içeren farklı veri kümelerinin altında yatan yapıları ortaya çıkarmakta yeterli olmamaktadır. Tek amaç kullanımının yetersizliği daha iyi anlaşılmakla birlikte, literatürdeki güncel eğilim çok amaçlı yaklaşımların kullanılması yönündedir. Problemin bir zorluğu da optimal çözümün iyi tanımlanamamasından kaynaklanmaktadır. Bildiğimiz kadarıyla, kümeleme problemi için önerilmiş bütün çok amaçlı algoritmalar tüm Pareto optimal çözümlerin oluşturduğu kümeyi oluşturmayı hedeflemektedir. Bu kümedeki çoğu çözümün bir karar verici tarafından değerlendirildiğinde ilgi çekici bulunmaması bu yaklaşımın pek faydalı olmadığını gösterebilir. Bu çalışmada, hedef kümeleri bulmak için, karar vericinin tercihlerini iyi bilinen bir çok amaçlı algoritma olan SPEA2'nin içerisine, referans noktaları ve başarı skalarlaştırma fonksiyonları kullanarak dahil etmekteyiz.

Anahtar Kelimeler: Kümeleme, Çok Amaçlı Optimizasyon, Sezgisel Yaklaşım, Evrimsel Algoritma, SPEA2, Tercih Tabanlı

To my mom

ACKNOWLEDGMENTS

I have received valuable support and motivation by many people throughout this masters thesis study.

First of all, I would like to express my sincere gratitudes to my academic supervisors, Dr. Nur Evin Özdemirel and Dr. Esra Karasakal for their endless interest in my work, valuable advices they have given and the support they have shown. I could not imagine a better guidance.

I would also like to express my gratitudes to the examining committee members Dr. Cem İyigün, Dr. Sinan Gürel and Dr. Tuğba Taşkaya Temizel for their valuable comments and reviews.

I would like to thank my dear friends, Büşra Atamer, Özlem Karsu, Çınar Kılcıoğlu, Aykut Bulut, Volkan Gümüşkaya, Erdem Çolak, Bilge Çelik, Banu Lokman and Tülin İnkaya for their support and motivation. They were always there, even at the hardest moments, easing the stress. I would also like to thank Tülin for her opinions, feedback and advices she has given in our precious corridor chats.

I wish to express my deepest thanks to my dearest wife, Ayşegül Demirtaş, for her never ending patience and support. Her presence was enough to keep me motivated, and, even seeing her around was enought to keep my face smiling.

Last but not least, I would like to thank my mother for her endless love and support during my whole life. I thank her for raising me to who I am, and without her, I would have never made it up this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Increasing capability of computers makes it easier to store vast amounts of data in digital media. This technological development together with the steady preference for relational databases triggers an untractably rapid and immense growth in data accumulation in modern enterprises. This brings about the problem of extracting knowledge and useful information from this mass of data, where the importance of data mining comes forward.

Data mining, which mainly aims to extract useful information from large data sets, is a growing field that often intersects with the field of operations research, and receives great contribution from it through formulation and solution of numerous data mining problems as optimization problems. Moreover, many operations research applications can be addressed using data mining methods. (Olafsson et al. 2008)

The clustering problem, which we deal with in this study, is one of the data mining problems that can be formulated as an optimization problem. In this chapter, we will define and explain the clustering problem in Section 1.1. In Section 1.2, we will state the challenging issues about the problem and give our motivation.

1.1 Problem Definition

The clustering problem is addressed by many disciplines from several fields, so the terminology used in different fields may vary. Its applications can be found in several fields such as machine learning, pattern recognition, artificial intelligence, web mining, text mining, image classification, genetics, biology, microbiology, paleontology, psychiatry, pathology, geography, and geology (Abraham et al. 2006). Here we start by giving the terminology that we use throughout this study.

A data set consists of individual data points, which are also referred to as data items, observations, records, feature vectors, or patterns in the open literature of different fields. A data point is defined by its attributes in a multidimensional space. The number and/or the characteristics of attributes that define a data point may vary with different data sets. An attribute may be continuous, binary or categorical. For example, a point in a 2-dimensional space is defined by its two continuous attributes, namely x and y coordinates. To illustrate further, consider a data point which represents a car whose color, brand, mileage and condition of being second hand are of interest. This data point is defined by four attributes with different characteristics, which are color (categorical), brand (categorical), mileage (continuous), and condition of being second hand (binary, i.e., 1 if second hand, 0 if first purchase). In this study, we focus on data sets that consist of data points having continuous attributes in two dimensions.

To give a more formal definition of the clustering problem, consider *N* data points $X_1 \dots X_N$, the union of which comprises the whole data set *D*. Let any data point X_i be defined in an *m*-dimensional space. So, each X_i is defined by its *m* attributes, i.e., $X_i = \{X_{i1}, X_{i2}, \dots, X_{im}\}, i = 1, \dots, N$. Then, the aim of the clustering problem trying to partition *D* into *k* clusters is to form *k* disjoint sets $C_1, \dots, C_k$, such that the union of these sets gives the whole data set *D*.

The assumption of the clustering problem is that the data set includes hidden patterns of data points and the goal is to reveal those patterns as clusters by an appropriate approach. Unlike classification, which is also a data mining problem, no information about the class labels is available on any representative data point which can be used for learning and testing purposes. In clustering, the aim of the problem is to form meaningful groups from data points with unknown labels, so it is in general referred to as unsupervised learning.

The goal of the clustering problem is to put similar items in the same cluster and dissimilar items into different clusters according to a certain similarity measure, while maintaining some internal objectives such as compactness, connectivity and spatial separation. In other words, data points in the same cluster are desired to be at close similarity levels (compactness) while an acceptable level of dissimilarity is maintained between different clusters (separation). Also, similar or spatially connected data points should be in the same cluster (connectivity).

1.2 Challenging Issues and Motivation

There are some challenging issues, such as the unknown number of clusters, arbitrary shaped clusters, outliers, inter-cluster density differences, and density variation within a cluster, which make the basic clustering problem even harder. An illustrative example of a data set including a combination of these challenging issues can be found in Figure 1.1.

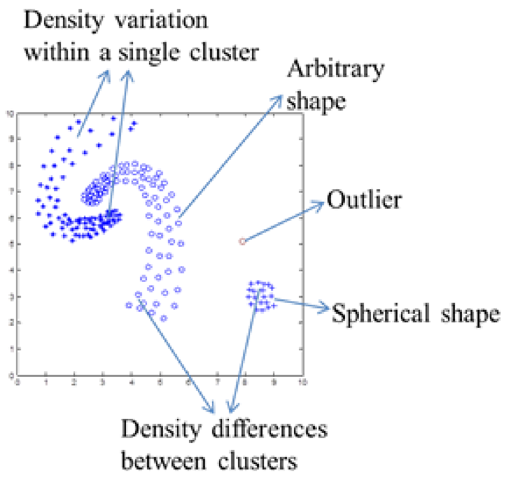

Figure 1.1 An example illustrating the challenging issues in clustering problems

In this section, we will explain the aforementioned challenging issues in more detail and give our motivation.

1.2.1 The Number of Clusters

The number of clusters a given data set will be partitioned to is one of the basic, yet difficult to handle issues in clustering. Many traditional clustering algorithms assume that the number of clusters is known prior to the execution of the algorithm, and partition the data set into that exact number of clusters. It is obvious that, a priori knowledge on the number of clusters improves the performance of a clustering algorithm dramatically. However, such an information may not be easily available for all data sets. It may require expert knowledge which may be hard to obtain or it may simply be unavailable. Yet, even with its presence, the expert knowledge on the number of clusters may be inaccurate for huge data sets.

As the sizes of the data sets of interest increase, it gets harder to extract a priori information on the number of clusters present in a data set. Moreover, the number of clusters in a data set may be subjective and may differ as the opinions of the decision makers differ. An idea can be to run an algorithm, which assumes the number of clusters to be known a priori, several times each time for a different number of clusters, and to apply a selection mechanism to all the generated solutions. However, it would be quite time consuming to try such an approach for a large range of the number of clusters. Therefore, it would be better for a clustering algorithm to be capable of discovering the number of clusters rather than using a given exact number during its execution time.

In this study, we assume that the number of clusters in a data set is unknown, and the solutions we generate in our search can have different number of clusters.

1.2.2 Arbitrary Shapes

For data sets in two or three dimensions with data points having only continuous attributes, the shapes of the clusters can be identified visually once a plot of the data set is obtained. There are several shapes a cluster may have, most common of which is spherical. In spherical shaped clusters, the dissimilarity or distance of data points from the data point in the center of their cluster becomes important, and we naturally want the data points in a cluster to be close to its center. However, considering the distances from a cluster center may not always yield the clusters that lie hidden within a data set. The clusters may have arbitrary shapes, such as elongated or spiral, as opposed to spherical shapes. Furthermore, in a data set, a combination of clusters with different shapes may exist.

Our aim is not to limit our algorithm for finding clusters of specific shapes, and we claim to detect clusters with different shapes in a single data set.

1.2.3 Outliers

Similar to its definition in statistics, an outlier is called to a data point that is dissimilar to all of the clusters in the clustering problem. Outliers are often considered as separate clusters consisting of single data points. Their existence deteriorates the performance of most traditional clustering algorithms.

Handling outliers is a problematic issue in clustering, since the outliers distort the patterns of data points in the data sets. Therefore, clustering algorithms have to be robust for the cases where outliers exist. For outlier detection purposes, even special mechanisms are proposed (Jain et al. 1999). However, it would be better to handle the presence of outliers by the structures of the algorithm rather than applying an external outlier detection mechanism in terms of maintaining a global perspective.

In this study we treat the outliers as a part of the whole data set, and do not apply an additional mechanism for their detection. Yet, we still claim to detect the outliers present in the data sets by taking appropriate measures in our algorithm, which we describe in Chapter 3.

1.2.4 Inter-cluster Density Differences and Intra-cluster Density Variation

Density is another problematic issue in clustering problems, which may occur in two ways. Inter-cluster density differences is the different densities in different clusters. Two different clusters may be homogenous when they are considered solely. However, the densities of those clusters may be different from each other, which we call the inter-cluster density difference.

Intra-cluster density variation is related with the density of data points in a specific single cluster. Most common traditional clustering algorithms assume that the density of a specific cluster remains homogenous in all of its regions. However, inside a single cluster, more sparse and dense regions may exist without the need to split that cluster into smaller clusters according to the regional densities.

In this study, we do not assume any limitation on the densities of the clusters, and aim to extract clusters with density differences and variations.

1.2.5 Fuzzy vs. Crisp Clustering

There are two different ways of clustering in terms of the assignment of data points to clusters. Crisp clustering assumes that the clusters are well separated, and by crisp clustering methods, each data point is assigned to only one cluster. However, in fuzzy clustering, the clusters may be overlapping which makes it hard to decide which cluster a data point lying in the overlapping region belongs to. Therefore, in fuzzy clustering, each data point has a degree of membership in each of the clusters, which may also be considered as the probability of being in that cluster.

In this study, we are not interested in fuzzy clustering. We assume that the clusters are well separated and we do not deal with any overlapping clusters.

1.2.6 Multiple Objectives and Decision Maker Preferences

There are specific methods developed for dealing with the challenging issues mentioned in this section. Yet, a method performing well with a data set having a certain challenging issue may fail if a different challenging issue occurs. Such methods often try to optimize a single objective which can reveal only a specific type of hidden information in the data set. For example, a method trying to minimize the overall deviation of within cluster distances is limited to detecting clusters of only spherical shapes.

In an attempt to find compact and well-separated clusters, combining compactness and separation measures into a single measure, such as taking the separation-to-compactness ratio, results in loss of information. Hence, in most of the cases, using a single objective is not sufficient to detect different underlying structures in the data sets. Thus, the current trend in the clustering literature is growing into the use of multiple objectives as the inadequacy of using a single objective is being understood better. By using multiple objectives concurrently, it is possible to account for a variety of tradeoffs between different objectives, which can perform better for different challenging issues if used separately.

By a multiobjective approach, a good representation of the Pareto optimal set of solutions corresponding to the objectives used can be obtained. However, this Pareto optimal set may include partitions that can be uninteresting when presented to a decision maker. For example, using two objectives one of which tries to merge and the other tries to split the data points, a solution that includes all the data points in a single cluster can be nondominated in a multiobjective sense as well as a solution with all data points in different singleton clusters. Such uninteresting solutions can be eliminated during the execution of the algorithm if any useful information can be gathered from the decision maker.

In this study, we propose an interactive multiobjective approach that can exploit the responses of a decision maker by simple questions, such as his/her opinion about two data points being in the same cluster, in order to direct the search trying to generate solutions that can be preferred by him/her.

1.3 Scope and Contribution

In this study we propose a new algorithm for clustering data points in sets having the following properties.

- Number of clusters is unknown a priori to the execution of the algorithm.
- Clusters may have arbitrary shapes.
- There may be density variation within a cluster.
- There may be density differences between clusters.
- Outliers may exist.
- The clusters are well separated (crisp).

The algorithm we develop is based on a well known multiobjective evolutionary algorithm, namely SPEA2 (Zitzler et al. 2002), which can be applied to any multiobjective problem with proper adjustments. The main structures of the evolutionary algorithm, such as the general population, archive and mating pool are borrowed from the original SPEA2. By defining our own fitness functions and introducing an original way of including decision maker's preferences throughout the execution, we offer an interactive preference based multiobjective evolutionary algorithm for the clustering problem.

CHAPTER 2

LITERATURE REVIEW

Clustering is a problem addressed by a variety of disciplines, making it hard to maintain a common terminology in the vast literature of algorithms developed. Stating the difficulty of constructing a truly comprehensive survey due to the sheer mass of literature, Jain et al. (1999) review the concepts and techniques of clustering related with statistics and decision theory. A similar perspective and perception is adopted in the outstanding surveys of Berkhin (2001) and Xu and Wunsch (2005). Following the classification schemes and ideas in these surveys, we will describe traditional clustering algorithms in Section 2.1. Then, we will give some metaheuristic applications to the clustering problem in Section 2.2. In Section 2.3, we will focus on multiobjective clustering, specifically multiobjective evolutionary algorithms, which aim to provide the whole set of nondominated partitions, or the best partition determined according to a certain selection scheme among that set. Finally, in Section 2.4, we will briefly mention preference based multiobjective evolutionary algorithms.

2.1 Traditional Clustering Algorithms

Jain et al. (1999), Berkhin (2001), and Xu and Wunsch (2005) provide extensive surveys and reviews in the area of clustering algorithms. In a broad view, they split the techniques used for clustering into two, namely hierarchical and partitional according to the output provided by the technique. Partitional clustering techniques yield a single partition or a set of different partitions of the data set. On the other hand, hierarchical clustering techniques end up with a tree-like clustering structure occasionally referred to as a dendrogram, which represents the nested grouping of data points at different similarity levels. The root node of the dendrogram is the whole data set and each leaf is a data point. The intermediate nodes can be interpreted as connections between data points having a certain similarity. Then, different partitions from a dendrogram can be obtained by slicing it at different similarity levels. Hierarchical methods can be classified as either agglomerative or divisive according to the way how the output is constructed. In agglomerative methods, each data point starts as a singleton cluster. They are merged according to a chosen measure until whole data points are in the same cluster. Divisive methods behave in the opposite way, starting from a single cluster that contains all data points. Clusters are successively divided into smaller clusters until all data points become singleton clusters. Most hierarchical agglomerative clustering algorithms use the idea of linkage metrics which can be referred to as inter-cluster distances. Single-link, complete-link and average-link are the most common linkage metrics (Murtagh 1985, Olson 1995). Examples of usages of these metrics in hierarchical clustering algorithms can be found in SLINK (Sibson 1973), Voorhees' method (Voorhees 1986) and CLINK (Defays 1977). Among the hierarchical agglomerative clustering algorithms, the single-link (SL) algorithms are most common. In SL algorithms, clusters are merged iteratively by their closest points in a similar way to minimum spanning tree construction.

Classical hierarchical clustering algorithms are disadvantageous for lacking robustness and being sensitive to noise and outliers. Clusters formed by these algorithms that use the notion of linkage metrics are known to be spherical, thus clusters of arbitrary shapes are not caught. These algorithms are also criticized for being irrevocable in the sense that it is not possible to correct errors made in previous iterations. Once mistakenly placed in a wrong cluster, a data point remains there throughout the run of the algorithm. Excessive memory requirement is another disadvantage of classical hierarchical clustering algorithms. (Xu and Wunsch 2005, Berkhin 2001)

BIRCH (Zhang et al. 1996, 1997) is another important hierarchical clustering algorithm with its strength in detecting outliers and dealing with large data sets. CURE, proposed by Guha et al. (1998), utilizes the representatives' aggregate linkage metric instead of the single-link and average-link metrics. By choosing the cluster representatives spread among the boundaries of the clusters rather than the centroids, CURE allows clusters of arbitrary shapes and different sizes to be revealed. It is also known to be insensitive to outliers and can handle large data sets. Guha et al. (1999) extend their existing algorithm CURE to cases where categorical attributes also exist and propose ROCK. Karypis et al. (1999a) propose CHAMELEON which relies on graph partitioning and involves the idea of a connectivity graph corresponding to the k-nearest neighbor subgraph. The algorithm is proven to detect clusters of arbitrary shapes, different sizes and densities.

Among the partitional clustering techniques the k-means algorithm (Hartigan 1975, Hartigan and Wong 1979) is the most popular with its applications in both industry and science. The k-means algorithm is a special case of k-medoid methods where the cluster is represented by one of its points. In the k-means algorithm, the clusters are represented by their centroids and data points are iteratively assigned to their nearest clusters with the aim of optimizing a certain criterion, usually chosen as the sum of squares of errors. Since squares of errors are computed with respect to the cluster centroids, the k-means algorithm tends to provide spherical clusters. Moreover, the original k-means algorithm starts with the initialization of k clusters, either randomly or by a special procedure. This makes the final output very dependent to the initial clusters generated, which can be stated as another disadvantage of the algorithm in addition to its lack of capability of detecting arbitrary shaped clusters. Another problematic issue is about the number of clusters. Usually, the determination of the number of clusters, k, requires expert knowledge, which is highly unlikely to exist for most data sets. In such cases, the algorithm is run many times for different values of k, which is computationally inefficient. Also, the algorithm is known to be incapable of detecting outliers. (Berkhin 2001)

Density based clustering methods are developed from the concepts of density, connectivity and boundary which are closely related with a data point’s nearest neighbors (Berkhin 2001). Thus, it is possible to find clusters of arbitrary shapes using density based algorithms. Moreover, due to the nature of the algorithms, outliers can be detected easily. However, density variation within clusters still remains problematic although density differences between clusters can be detected. Well known density based algorithms include DBSCAN (Ester et al. 1996), GDBSCAN (Sander et al. 1998), OPTICS (Ankerst et al. 1999), DBCLASD (Xu et al. 1998) and DENCLUE (Hinneburg and Keim 1998).

All the mentioned traditional clustering algorithms have strong points and weaknesses depending on both the underlying assumptions of the algorithm and the structure of the data set. To the best of our knowledge, no algorithm has been proven to be successful on all types of data sets.

2.2 Metaheuristic Clustering

Clustering problem can be regarded as a category of combinatorial optimization problems. Even for small data sets, it is computationally expensive to consider all possible assignments. The search space grows exponentially as the size of the data set increases, so, simple heuristic methods can easily get stuck in local optima. Therefore, metaheuristics have been widely applied to the clustering problem with the purpose of exploring the solution space more efficiently and finding optimal or near optimal partitions. In the literature, there are examples of both direct metaheuristic approaches to clustering and hybrid approaches of metaheuristics with traditional algorithms. Several applications of metaheuristics on clustering can be found in the comprehensive reviews of Rayward-Smith (2005) and Das et al. (2009).

Babu and Murty (1993) try to improve the performance of the original k-means algorithm by finding near optimal seeds at the initialization step, since the final output is highly dependent on the generated initial cluster centroids. They also apply simulated annealing for the same purpose in Babu and Murty (1994). For small data sets, Selim and Al-Sultan (1991) propose a simulated annealing algorithm and Al-Sultan (1995) proposes a tabu search algorithm. A genetic algorithm (Maulik and Bandyopadhyay 2000) and ant colony optimization (Kanade and Hall 2004) have been applied to find cluster centroids. There are other ant colony optimization applications for clustering in Lumer and Faieta (1994), Monmarche (1999), Chu et al. (2004) and Handl et al. (2003). In Handl et al. (2003), the number of clusters is automatically determined within the algorithm. Krishna and Murty (1999) and Bandyopadhyay and Maulik (2002) both use hybrid genetic algorithms with k-means for clustering. Results show that genetic algorithms have been very useful for improving the performance of k-means algorithms (Xu and Wunsch 2005, Berkhin 2001).

Among all metaheuristics, evolutionary algorithms are the most common ones used for clustering problems. In a recent survey, Hrushcka et al. (2009) provide a taxonomy of evolutionary algorithms for clustering in terms of important aspects of the problem and the specifications of the algorithm components. Mainly, the algorithms are categorized by their capability of handling variable number of clusters, evolutionary operators, solution representations, fitness functions and initialization procedures.

2.3 Multiobjective Clustering

Without proper modification for a given specific data set, current metaheuristic applications that use single objectives have limited capabilities. As the objective to be minimized, most of them use within cluster variance, and they take Euclidean distance as the dissimilarity measure. As a consequence, they are unable to capture arbitrary shaped clusters in the data sets and result in spherical clusters. Also, possible density variations within clusters are ignored, and the resulting clusters are homogeneous. In addition to that, usually, the number of clusters are assumed to be known a priori and taken as fixed, ignoring partitions with different number of clusters. Since the characteristics of data sets vary, robust results cannot be obtained from these algorithms. In addition, a data set may be challenging in more than one aspect, i.e., it may include arbitrary shaped clusters with different densities, and the density within a cluster may also vary. Each challenging issue favors one algorithm over another. Thus, such situations are impossible to be handled using a single objective.

In clustering ensemble methods, good parts of the partitions obtained by running different algorithms are tried to be combined in a final partition that is better than the original partitions in terms of a defined validity criterion (Hruschka et al. 2009). In other words, clustering ensembles need an initial set of partitions provided by several runs of an algorithm or set of algorithms, to be combined into a single partition (Handl and Knowles 2007). Such methods ignore the tradeoffs between the objectives of the individual algorithms, since each algorithm tries to optimize its own objective. However, the decision maker should be provided with a set of solutions that reflects the tradeoffs between the objectives, where multiobjective clustering comes forward.

2.3.1 Objectives, Validity Indices

Traditional algorithms and metaheuristics try to find the best possible partitioning of a given data set according to a performance measure that is hoped to define the quality of a given partition. Generally, these performance measures can be referred to as validity indices. Validity indices can be categorized into two: internal indices and external indices. External indices use a reference or a prespecified partition to validate a given clustering solution. Rand, Adjusted-Rand (Rand 1971) and Jaccard indices can be counted in this category. On the other hand, internal indices do not need a reference partition and are directly driven from the partition at hand and the structure of the data set (Xu and Wunsch 2005). Such measures may also behave as the objectives of an algorithm when the algorithm generates a partition trying to optimize the value of these measures.

Some widely used objectives are compactness, connectivity and separation. As far as compactness is concerned, less variation between data points in the same cluster is desirable. There are several validity indices, and other measures suggested to quantify the compactness of a clustering solution. Average distance between data points of a cluster and cluster center (DB index, Davies and Bouldin, 1979), total variance of distances between data points and cluster center (CH index, Calinski and Harabasz, 1974), total distance between data points and cluster center (I Index) are well known compactness measures that tend to provide spherical clusters. In order to detect arbitrary shaped clusters, graph theoretic measures are used as the compactness objective. Yousri et al. (2008) propose to use the standard deviation of the edges of a minimum spanning tree constructed in a cluster. Similarly, Pal and Biswas (1999) suggest taking the maximum edge in a minimum spanning tree, relative neighborhood graph or a Gabriel graph constructed for a cluster. Such measures are known to be successful as the construction of the corresponding structures is closely parallel with the concept of clustering.

Connectivity is related with placing neighboring points into the same cluster. Thus, clusters of arbitrary and mostly elongated shapes are easily detected with this objective. In their multiobjective approach, Handl and Knowles (2007) use connectivity as one of the objectives. Their aim is to penalize the neighboring points put in different clusters by minimizing the connectivity function they define as follows.

$$Conn(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{H} x_{i,nn_{ij}} \right)$$
$$\text{where } x_{i,nn_{ij}} = \begin{cases} \frac{1}{j} & \text{if } i \text{ and } nn_{ij} \text{ are in different clusters} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In the above function, *N* is the total number of data points, *H* is a parameter defining the maximum neighborhood size, and nn_{ij} indicates the j^{th} nearest neighbor of point *i*.

The goal of separation is to keep an acceptable dissimilarity level between any two different clusters. Most common separation measures are single link (minimum distance between data points from different clusters), complete link (maximum distance between data points from different clusters) and average link (average distance between data points from different clusters) separation.

2.3.2 Multiobjective Evolutionary Clustering Algorithms

Most of the multiobjective approaches to the clustering problem make use of evolutionary algorithms. Ripon et al. (2006) use Variable-length Real Jumping Genes Genetic Algorithm (VRJGGA) to identify the non-dominated solutions. They use cosine symmetry for intra-cluster entropy and Euclidean distance for inter-cluster distance. As the first objective, they maximize overall intra-cluster entropy which can be referred to as a compactness measure. The second objective is a separation objective that minimizes average separated distance between all cluster centers. They evaluate nondominated solutions by classification accuracy, generalized Dunn's index, overall deviation as well as the values of the objectives used in optimization. The algorithm does not need the number of clusters to be known a priori.

Handl and Knowles (2004a) propose VIENNA (Voronoi Initialised Evolutionary Nearest-Neighbor Algorithm) which uses the elitist multiobjective evolutionary algorithm PESA-II to find a set of Pareto optimal solutions for a predetermined number of clusters. They use Euclidean distance for synthetic data sets, cosine symmetry for real data sets. As the compactness objective, they minimize overall intra-cluster variance as in the k-means algorithms. In order to detect clusters of arbitrary shape, they introduce their second objective which is a measure of connectivity as given in equation (1). Solutions that are Pareto optimal at the end of the run are evaluated according to the F-measure, a measure used to define the quality of partitions. However, their algorithm needs the number of clusters to be given a priori.

Handl and Knowles (2004b, 2007) extend their work in Handl and Knowles (2004a) and propose Multi-Objective Clustering with automatic k determination (MOCK) so that the number of clusters can vary throughout the run. The result of their work is a

Pareto optimal set consisting of solutions corresponding to different tradeoffs as well as different number of clusters. They also propose a way of selecting a single solution from the resulting Pareto optimal set using control fronts, an idea inspired by Gap Statistic (Tibshirani et al. 2001). The objectives used in these studies are the same as those in Handl and Knowles (2004a), which are minimizing compactness defined as overall intra-cluster variance and minimizing connectivity defined as in equation (1). The resulting Pareto optimal set and the solution selected among the solutions in that set are evaluated using the adjusted Rand Index.

Chen and Wang (2005) use NSGA-II to optimize two objectives: minimizing compactness measured as overall intra-cluster deviation and maximizing connectivity measured similar to equation (1). Instead of penalizing neighboring data points put in different clusters, Chen and Wang (2005) propose to award the neighboring data points put in the same cluster. To evaluate the solutions, they use the F-measure. Their algorithm does not need an a priori determined number of clusters.

Won et al. (2008), Du et al. (2005), Korkmaz et al. (2006) and Özyer et al. (2004) use overall intra-cluster deviation as a compactness objective to be minimized. As a second objective they propose to minimize the number of clusters, k. Won et al. (2008) use a hybrid strategy, including a k-means algorithm for fine tuning at the end. Therefore, their partitions are limited to clusters of spherical shape. Du et al. (2005) propose Niched Pareto Genetic Algorithm, NPGA (Horn et al. 1994), with linked-list based chromosome encoding. Their solutions are again limited to partitions consisting of spherical clusters. They propose to evaluate the final solutions by the amount of leap in overall intra-cluster variance with respect to the change in k to select a single solution. Korkmaz et al. (2006) extend the work in Du et al. (2005) by using an improved linked-list based chromosome encoding to obtain a one-to-one mapping between the chromosome representation and the corresponding actual partitioning. In this way, they reduce the redundancy issues, which results in more efficient exploration of the solution space. Özyer et al. (2004)

apply multiobjective genetic k-means algorithm (MOKGA), which combines the original k-means algorithm and NPGA to gene expression data.

Law et al. (2004) propose a two-step process which includes detection of clusters according to a set of candidate objectives in the first step, and the integration of objectives in a goodness function to give the final partitioning using re-sampling techniques in the second step. Their approach lacks the competency to detect the tradeoff between the objectives in the first step. Yet, they try to consider the tradeoff between objectives in the second step.

A summary of the algorithms mentioned here is presented in Table 2.1. Most frequently used objectives are compactness (COM) measured as overall intra-cluster distance variation, connectivity (CON) as defined in equation (1) or similar to equation (1), and the number of clusters (NUMC). For the number of clusters column, "K" indicates that the number of clusters is known a priori and is fixed. On the other hand, algorithms having a "U" in this column are able to generate partitions having an unknown number of clusters throughout the run. "A" in the shape column is an abbreviation for arbitrary, which indicates the algorithm's capability to detect clusters of arbitrary shape. So, algorithms having an only "S" in the shape column are restricted to finding partitions having only spherical clusters.

Table 2.1 A review of multiobjective evolutionary algorithms for clustering

Author(s) and Year	Objectives		# of Clusters	Shape	Approach
	1	2			
Ripon et al. (2006)	*Max* overall intra-cluster entropy	*Min* average distance between cluster centers	U	S	VRJGGA
Handl and Knowles (2004a)	*Min* COM	*Min* CON	K	S, A	VIENNA
Handl and Knowles (2004b, 2007)	*Min* COM	*Min* CON	U	S, A	MOCK
Chen and Wang (2005)	*Min* COM	*Max* CON	U	S, A	NSGA-II
Won et al. (2008)	*Min* COM	*Min* NUMC	U	S	Hybrid strategy with k-means for fine tuning
Du et al. (2005)	*Min* COM	*Min* NUMC	U	S	NPGA with linked-list based chromosome encoding
Korkmaz et al. (2006)	*Min* COM	*Min* NUMC	U	S	NPGA with an improved linked-list based chromosome encoding
Özyer et al. (2004)	*Min* COM	*Min* NUMC	U	S	MOKGA

All the algorithms mentioned here generate the whole Pareto optimal set. However this may not be a good idea when the solutions in this set are considered as partitions. First of all, it is computationally hard to generate all the nondominated solutions. Secondly, a majority of the solutions in this set may be uninteresting when presented to a decision maker. A few algorithms propose a selection mechanism, yet they still apply this mechanism to the whole set of nondominated solutions they found. It would be a better idea to incorporate the preferences of a decision maker in the optimization process so that the nondominated solutions favored by him/her can be generated instead of the whole set of nondominated solutions. To the best of our knowledge, until this time, there has been no application of a preference based multiobjective evolutionary algorithm for the clustering problem.

2.4 Preference Based Multiobjective Evolutionary Algorithms

Multiobjective evolutionary algorithms aim to generate a good representation of the whole Pareto optimal set by generating and keeping a well distributed set of nondominated solutions, construction of which is not straightforward. Special mechanisms are needed to direct the search while assuring both convergence and diversity. However, many solutions in this set may not be desired by the decision maker although they are equally important in a pure multiobjective sense, since they are all nondominated. It also requires more execution time to keep the diversity and spread among the representative Pareto optimal set. Therefore it may not be necessary to generate the whole Pareto optimal set, especially when the decision maker can provide information about his/her preferences.

An early survey on handling decision maker preferences can be found in Coello (2000). The instant a decision maker provides his/her preference information is one important aspect of preference based evolutionary algorithms. Depending on when the preference information is obtained, the ways a decision maker can express his/her preferences is classified into three: a priori, a posteriori and interactively. Another classification can be made according to the characteristics of the preference information provided by the decision maker. He/she can give a full preference information such as the desired objective function levels. Also he can implicitly provide an insight about his/her preferences i.e., by comparing two solutions, where the preference information obtained in this case would be partial (Branke, 2008).

Whether full or partial, the preference information provided by the decision maker should be incorporated into the optimization process by an appropriate technique. Branke (2008) provides an invaluable review for such techniques and also provides a classification based on the methods used in these techniques. Krettek et al. (2009) propose an interactive way of incorporating decision maker preferences into a multiobjective evolutionary algorithm. At each interaction, the decision maker is asked to compare a pair of two solutions in terms of comparability and quality, and his/her answers are used to induce a ranking on the population. Deb (1999) introduces bias among the solutions by using weights in a sharing function so that solutions from the preferred regions of the Pareto optimal set are generated and carried through the generations. Branke et al. (2001) alter the region dominated by a solution by changing the angle of domination from a right angle to a wider angle using tradeoff functions. In this way, according to the preference information obtained from the decision maker, originally nondominated solutions may become dominated. However, this approach is limited to problems with two objectives, and it cannot be easily generalized to problems with more objectives. Deb et al. (2006), Deb and Kumar (2007a), and Deb and Kumar (2007b) use reference points and achievement scalarizing functions, which are ideas originated from the work by Wierzbicki (1980) to direct the evolutionary search into regions preferred by the decision maker. Molina et al. (2009) modify the definition of domination and propose g-dominance by which an originally dominated solution can actually be preferred to the solution that dominates it. Thiele et al. (2007) modify the $\varepsilon-$indicator in Indicator Based Evolutionary Algorithm (IBEA) proposed by Zitzler and Kuenzli (2004). They weight it with an achievement scalarizing function and a reference point given by the decision maker, and propose the Preference Based Evolutionary Algorithm, PBEA.

From the approaches used in the aforementioned algorithms, it is obvious that reference points and achievement scalarizing functions play a crucial role in incorporating decision maker preferences in multiobjective evolutionary algorithms, as they also do in our proposed algorithm. Therefore, more detailed explanation about them will be given in Section 3, before describing our algorithm.

CHAPTER 3

PROPOSED ALGORITHM

In this chapter, we will begin by introducing a theoretical background that forms the backbone of our algorithm in Section 3.1. In Section 3.2, we will give the notation used for the clustering problem and define our objectives that we use in our algorithm. After giving the notation used in the algorithm in Section 3.3, we will give an overview of our proposed algorithm in Section 3.4, and explain it in detail in Section 3.5.

3.1 Theoretical Background

Our algorithm is based on the Strength Pareto Evolutionary Algorithm (SPEA2) (Zitzler et al. 2002). In this section, we first discuss the important aspects of the original SPEA2 algorithm. Then, we explain the idea of reference points and achievement scalarizing functions.

3.1.1 The Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 (Zitzler et al. 2002) is an improved version of the Strength Pareto Evolutionary Algorithm (SPEA) proposed by Zitzler and Thiele (1999). SPEA2 operates with a regular population and an external population often referred to as the archive. In any generation, individuals in the current regular population and the archive from the previous generation are evaluated together. Then a new archive is selected from the union of individuals in the current population and the archive of the previous generation. In contrast to SPEA, archive size is fixed in SPEA2, so dominated solutions may also enter the archive. If the number of nondominated solutions candidate to enter the archive exceeds the archive size, then a truncation method is applied to reduce this number to the size of the archive. This method also prevents the removal of boundary solutions from the archive so that a good spread of nondominated solutions can be maintained. On the other hand, if the number of nondominated solutions entering the archive is less than the archive size, the remaining portion of the archive is filled with dominated individuals according to their fitness values. From the archive selected, a mating pool is constructed and the offsprings generated from this mating pool are copied to the next population by replacing the old population.

When compared to SPEA, fitness assignment scheme is improved in SPEA2 by taking into account the number of individuals an individual dominates or is dominated by. Fitness of an individual is the sum of two parts, namely the raw fitness and the density. Raw fitness of an individual is determined by the strengths of its dominators, where the strength of an individual is defined as the number of individuals it dominates from the union of the population at that generation and the archive of the previous generation. So, the raw fitness value of any nondominated solution is equal to zero. Density of an individual is calculated by a decreasing function of its distance to its k^{th} nearest neighbor, where k is a parameter fixed before the execution of the algorithm. Density is arranged in a way that it is always greater than zero and less than one.

3.1.2 Reference Points and Achievement Scalarizing Functions

Achievement scalarizing functions (ASF) are first introduced by Wierzbicki (1980). They are mostly used in reference point methods. In such methods, there exists the concept of a reference point, which can be interpreted as the levels in the objectives that seem desirable for the decision maker. Once the objective values of the reference point are specified by the decision maker, ASFs can be used to evaluate the feasible solutions at hand according to their proximities to the given reference point. For a given reference point, an ASF simply tries to find the maximum weighted difference among all the corresponding pairs of objective values between the reference point and any feasible solution. To illustrate, let *g* be the reference point having an objective value of g_i for the i^{th} objective. Assuming that all objectives are to be minimized and using a weight of w_i for scalarizing objective *i*, the achievement scalarizing function value asf_p of a solution *p* with objective value of f_i for the i^{th} objective is defined as $asf_p = \max_i \{ w_i (f_i - g_i) \}$. Then, the solution having the minimum *asf* value among all the solutions in the feasible region will be nondominated and it can be referred to as the closest solution on the efficient frontier to the reference point with an importance of w_i tied for objective *i*. Thus, solving the achievement scalarizing problem which aims to find the minimum of these *asf* values among the set of all feasible solutions at hand can be interpreted as finding the reflection of the reference point on that set, which can be visualized in Figure 3.1.

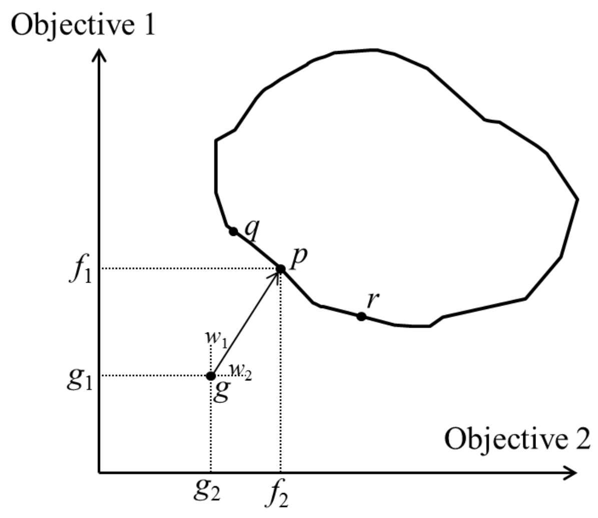

Figure 3.1 Visualization of achievement scalarizing functions

In Figure 3.1, solution *p* is the reflection of the reference point *g* on the efficient frontier for weights of w_1 and w_2. Note that, either by using different weights or using another reference point with the same weights, solutions *q* and *r* can also be the reflections on the efficient frontier.

3.2 Notation and Objectives for the Clustering Problem

We use the following notation to characterize a clustering problem.

D	Set of all data points.
N	Number of data points in set *D*.
i, j	Indices for data points, $i, j = 1, \ldots, N$.
d_{ij}	Euclidean distance between data points *i* and *j*.
NN_i	Ordered set of nearest neighbors of data point *i*.
H	Number of nearest neighbors in NN_i, given for all data points.
den_i	Density of data point *i* defined as the length of the longest edge in the minimum spanning tree constructed for the set NN_i for a given *H*.
K	Number of clusters found in a solution.
k, l	Indices for clusters, $k, l = 1, \ldots, K$.
C_k	Set of data points in cluster *k*, $k = 1, \ldots, K$.
com_k	Compactness of cluster *k*, which is the length of the longest edge in the minimum spanning tree constructed for cluster *k*.
sep_{kl}	Single link separation of clusters *k* and *l* defined as follows. Let *i* and *j* be two data points from clusters *k* and *l*, respectively, such that they are the closest points to each other in these two clusters. Then, $sep_{kl} = d_{ij}$.

Among the above, only the neighborhood size *H* is a problem parameter that needs to be determined externally.

In our multiobjective approach, we use two objectives defined as below for the clustering problem.

1. Minimize compactness *fcom*, where $fcom = \max_k \{com_k\}$.

2. Maximize separation for which two different measures are used.

2.1. From the decision maker's perspective:

$$fsep = \min_{k<l} \{sep_{kl}\}$$

2.2. In the algorithm:

Let *i* and *j* be the two data points from clusters *k* and *l*, respectively, that satisfy the *fsep* definition given in 2.1 above.

$$fasep = \begin{cases} d_{ij} & \text{if } den_i > d_{ij} \text{ and } den_j > d_{ij} \\ d_{ij} \dfrac{d_{ij}}{den_i} \dfrac{d_{ij}}{den_j} & \text{if } den_i < d_{ij} \text{ and } den_j < d_{ij} \\ d_{ij} \dfrac{d_{ij}}{\min\{den_i, den_j\}} & \text{if } den_i < d_{ij} \text{ or } den_j < d_{ij} \end{cases}$$

This "adjusted" separation measure is used to be able to detect the density differences in the data set and to "inflate" the separation measure, when it is defined between particularly dense regions of the two clusters.

3.3 Notation and Definitions for the Evolutionary Algorithm

We use the following notation in describing our proposed evolutionary algorithm.

P_t	Set of solutions in the population in generation *t*.
P^{size}	Number of solutions in the population .
A_t	Set of solutions in the archive in generation *t*.
A^{size}	Number of solutions in the archive.
A^{mult}	Multiplier to determine the portion of the archive that will be allocated to solutions preferred by the decision maker.
I_t	The incumbent solution in generation *t*.
IS_t	The set of interesting solutions in generation *t*.
MP_t	Set of solutions in the mating pool in generation *t*.
MP^{size}	Number of solutions in the mating pool.

p, q	Indices for solutions.
$fcom_\mathrm{p}$	Compactness objective value for solution p.
$fasep_\mathrm{p}$	Adjusted separation objective value for solution p.
$gcom$	Compactness objective value of the given reference point.
$gasep$	Adjusted separation objective value of the given reference point.
w^{com}	Weight used in scalarizing the compactness objective.
w^{asep}	Weight used in scalarizing the adjusted separation objective.
asf_p	Achievement scalarazing function value of solution p. $asf_p = \max\{w^{com}(fcom_p - gcom), w^{asep}(gasep - fasep_p)\} + \varepsilon(fcom_p - gcom) + \varepsilon(gasep - fasep_p)$, where $\varepsilon = \min\left(\frac{1}{d_{\max}}, 0.001\right)$, and $d_{\max}$ is the maximum distance between all data point pairs.
S_p	Strength of solution p defined as the number of solutions dominated by solution p.
FR_p	Raw fitness of solution p (the smaller the better). $FR_p = \sum_{\substack{q \succ p \\ q \in P \cup A}} S_q$, where $q \succ p$ indicates that solution q dominates solution p.
O_p	Minimum rank of solution p in terms of $fcom_p$ and $fsep_p$ in the population. Solutions in set P are once sorted in nondecreasing order of $fcom_p$ and once in nonincreasing order of $fasep_p$. Between the two ranks of solution p in these sortings, the smaller one is taken as O_p.
FO_p	Objective fitness (the smaller the better) of solution p. $FO_p = FR_p + \frac{O_p}{P^{size} + A^{size} + 1}$.
C_p	Crowding distance of solution p. $C_p = \frac{fcom_{(p+1)} - fcom_{(p-1)}}{fcom_{\max} - fcom_{\min}} + \frac{fasep_{(p-1)} - fasep_{(p+1)}}{fasep_{\max} - fasep_{\min}}$, where subscripts in parentheses indicate that compactness values of solutions in set P are sorted in nondecreasing order, separation values in nonincreasing order.

Terms in the denominator are the maximum and the minimum compactness and separation values in the population.

FD_p

FD_p Crowding distance fitness (the smaller the better) of solution *p* defined as

$$FD_p = FR_p + \frac{1}{C_p + 1}.$$

FA_p Achievement scalarizing fitness (the smaller the better) of solution *p* defined as

$FA_p = |\{q : q \in P \cup A \text{ and } asf_q < asf_p\}|$, where $|\cdot|$ indicates the cardinality of the set.

mprob Mutation probability for each gene of each offspring.

3.4 Overview of the Proposed Algorithm: Interactive Preference Based Multiobjective Evolutionary Clustering (IP-MOEC)

We modify some parts of the original SPEA2 by introducing a reference point approach together with the notion of achievement scalarizing function, as suggested in Wierzbicki (1980). Besides, as another addition to the original SPEA2, we incorporate the preferences of the decision maker in the optimization process interactively in order to direct our search toward solutions preferred by him/her, rather than generating the whole Pareto optimal set. Also, the crowding distance in NSGA II (Deb et al. 2002) is used as proposed in Karasakal and Silav (2010) for truncation of the archive.

Below is a brief overview of our proposed evolutionary algorithm. In the algorithm, there is a regular internal population used for general purposes and an external archive that helps to maintain elitism. Preferences of the decision maker are incorporated into the algorithm directly in the interaction phase and implicitly in the selection phase.

Phase 0. “Initialization”

Initial population generation and creation of the initial empty archive. Initialization of the reference point by assigning worst possible values to its objectives.

Phase 1. “Union and Evaluation”

Union of the population and the archive and evaluation of the individuals in the union.

Phase 2. “Interaction”

Interaction with the decision maker if certain conditions are satisfied and making the necessary changes.

Phase 3. “Selection”

Archive selection from the union.

Phase 4. “Evolution”

Construction of the the mating pool from the new archive and application of genetic operators to the mating pool to generate the offspring. Setting the new population and restarting Phase 1.

The overview of the algorithm presented above is simple and can be explained under four main headings, namely Union and Evaluation, Interaction, Selection and Evolution with an additional step of Initialization at the beginning. Here we will give a more thorough explanation of the algorithm by providing the details of these phases.

```
Step 0. Generate the initial population of size $P^{size}$ randomly and put in set $P_1$.
        Set $A^{size} = P^{size}/2$. Set $A^{mult} = 1$.
        Create empty archive set $A_0$ of size $A^{size}$.
        Set $gcom = 0$, $gasep = d_{max}$, where $d_{max}$ is the longest distance between all pairs of points in set $D$.
        Set generation count $t = 1$.

Step 1. Calculate $asf_p$, $S_p$, $FR_p$, and $FA_p$ for each solution $p$ in set $P_t \cup A_{t-1}$. Find the solution $q$ having the smallest $asf$ value. If no information is available about the data point pairs that determine the separation and compactness of solution $q$, interact with the decision maker. According to his/her responses, update the reference point, the incumbent solution and the set of interesting solutions if necessary. If the reference point is changed for the first time, set $A^{mult} = 0.5$.

Step 2. Find the number of non-dominated solutions $\#nondominated$ in $P_t \cup A_{t-1}$.
IF $\#nondominated < A^{size} \times A^{mult}$
        Calculate $FO_p$ for each solution $p$ in $P_t \cup A_{t-1}$. Select the best $A^{size} \times A^{mult}$ solutions in terms of $FO_p$ values and put them in $A_t$. (All nondominated solutions are included in $A_t$).
ELSE ($\#nondominated \geq A^{size} \times A^{mult}$)
        Calculate $FD_p$ for each solution $p$ in $P_t \cup A_{t-1}$. Select the best $A^{size} \times A^{mult}$ solutions in terms of $FD_p$ values and put them in $A_t$. Find the solution $q$ having the smallest $FD$ value. If no information is available about the data point pairs that determine the separation and compactness of solution $q$, interact with the decision maker. According to his/her responses, update the reference point, the incumbent solution and the set of interesting solutions if necessary. If the reference point is changed for the first time, set $A^{mult} = 0.5$.
END IF
```

```
IF $A_t$ has not reached size $A^{size}$
    From the solutions in $P_t \cup A_{t-1} - A_t$, select the best solutions in terms of $FA_p$ values and put them in $A_t$ until $A_t$ has size $A^{size}$.
END IF
```

Step 3. Duplicate each solution in A_t once and construct the mating pool MP_t of size MP^{size}. Perform uniform crossover on the mating pool, apply mutation to the generated offspring and set them as the new population P_{t+1}. If the stopping condition is not satisfied, then set $t = t + 1$ and go to Step 1.

3.5 Details of the Algorithm

In this section, the procedures mentioned in the overview of the algorithm and additional structures used in the algorithm will be explained further in detail.

3.5.1 Solution Representation

Solution representation is an important issue in multiobjective evolutionary algorithms for clustering problems. Several different coding schemes are proposed in the literature. Hruschka et al. (2009) give a classification of the coding schemes in the open literature in terms of their various aspects and provide a comprehensive review stating their strengths and weaknesses.

In our algorithm, edge based encoding is used to represent the solutions. Each solution is represented as a chromosome of N genes. Each gene indicates a link between two data points represented by its value and its index, implying that those two data points are in the same cluster in that particular solution. To illustrate, consider the following example with seven data points and an arbitrary solution given in Figure 3.2.

1	3	7	4	6	5	1

Figure 3.2 Chromosome representation of an arbitrary solution

In this solution, the first gene indicates a link from data point 1 to itself. Similarly data point 2 is linked with 3, 3 is linked with 7, and so on. Links between data points can be revealed with a simple decoding procedure as proposed in Handl and Knowles (2007), and it can be completed in linear time. Once all the links are revealed, actual clusters can be observed. The visualization of the partitioning for this example can be found in Figure 3.3.

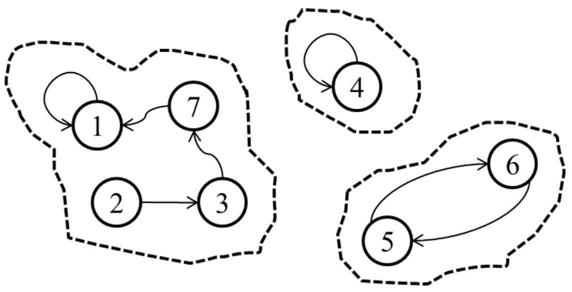

Figure 3.3 Actual partitioning of the example representation

It can be seen from the figure that the solution represented by the provided example chromosome actually corresponds to a clustering solution with K = 3, and the three clusters are $C_1 = \{1,2,4,7\}, C_2 = \{3,5\}, C_3 = \{6\}$.

3.5.2 Initialization

Generation of the initial population is another aspect that has been widely studied in evolutionary algorithms for clustering. The success of the technique used to generate the individuals in the initial population depends on the selected representation scheme and how the objectives are measured. In our algorithm, no special technique is used for initial population generation. The individuals in the initial population are generated randomly, simply by assigning a random value for each gene in every individual.

The initialization of the reference point is done by assigning the worst possible values for each objective. At the beginning of the algorithm, we do not have any information about the preferences of the decision maker, so the reference point cannot reflect his/her preferences. Because of this reason, A^{mult} is initialized to 1, meaning that all the archive is filled according to *FO* and *FD* values and no portion is allocated to solutions having good *FA* values. According to the responses of the decision maker in the interaction steps, the reference point is approximated and it starts affecting the direction of the search.

3.5.3 Evaluation

In the union and evaluation phase of the algorithm, the population at that generation is united with the archive from the previous generation first, and then, the solutions in the union are evaluated together. The evaluation is mainly the calculation of the strengths, achievement scalarizing function values and the necessary fitness values of the solutions.

Strength and raw fitness definitions are directly taken from the original SPEA2 algorithm. The strength, S_{p}, of solution *p* in the union is found by counting the number of solutions in the union dominated by *p*. Hence, a solution's strength increases as it dominates more solutions. On the other hand, raw fitness FR_{p} of a solution *p* depends on the sum of the strengths of its dominators. Therefore FR_{p} may increase rapidly if the strength values of its dominators are large, and unlike the strength, it is better to have a smaller *FR* value. For nondominated solutions, *FR* values are simply zero.

Before the calculation of other fitness values, the number of nondominated solutions, #nondominated, in the union is found, since reserved separate portions of the archive which will be filled according to different fitness types depend on this number. If #nondominated is greater than $A^{size} \times A^{mult}$, it means that a truncation is needed to reduce this number to available space in the archive, which is $A^{size} \times A^{mult}$. Density fitness *FD*, that uses the crowding distance idea in NSGA II (Deb et al.

2002) is used in truncation as proposed in Karasakal and Silav (2010). By using *FD*, the nondominated solutions in relatively less dense regions of the Pareto optimal front are promoted. On the other hand, if #nondominated is less than $A^{size} \times A^{mult}$, then the available space in the archive is filled according to the objective fitness, *FO* as proposed in Karasakal and Silav (2010). *FO* tries to favor solutions lying on the boundaries to introduce diversity in the next generations and widen the evolutionary search.

After filling a reserved portion of the archive according to either *FD* or *FO*, the remaining space is filled according to the achievement scalarizing fitness, *FA*. In order to calculate FA_p, for solution *p*, first, the achievement scalarizing function values of the solutions in the union should be calculated. In calculating the achievement scalarizing function value, asf_p of any solution *p*, the terms multiplied by ε 's are added in order to ensure that weakly efficient solutions have worse *asf* values than their corresponding strictly efficient solutions. Solutions entering the archive according to the *FA* are expected to be close to the target solution considering that the target solution can be represented by the reference point, which is updated according to the decision maker's responses in the interactions phase.

3.5.4 Interaction with the decision maker

There are a few widely used ways of interacting with the decision maker in preference based multiobjective algorithms. He/she may be asked to compare two solutions if he prefers one over the other. This comparison is mostly in terms of the objective function values. He/she may give a total preference, or may state the amount of loss in an objective to gain some for another objective, which can indicate information about the tradeoff between those objectives. The decision maker may be asked to express desirable levels for each objective, which can be treated as the objective values of a reference point. However, the aforementioned methods to extract decision maker preferences may not be realistic in our problem. The decision maker may not initially have a concrete idea about the value of the compactness objective of the clustering problem. It gets even more unrealistic for the decision maker to express a desirable value about the separation objective of the problem especially for the cases where the separation value of the target solution is adjusted. On the other hand, presenting two solutions to the decision maker is also a problematic task. For problems of two or three dimensions, the clusters of each solution can be plotted and visually presented to the decision maker so that he/she can provide his/her preference. However, it would be time consuming to do so, and such an approach may not be received well by the decision maker. Moreover, it is not possible to plot the clusters in a visually recognizable way for data sets with higher than three dimensions.

In our problem, the interaction with the decision maker is simple and straightforward. In each generation, two solutions, which are the best solutions in terms of *FA* and *FD* separately, are found. The idea is that, the solution best in *FA* is the closest solution on the Pareto optimal front to the current reference point and the solution best in *FD* is from the least dense region of the Pareto optimal front and could also be interesting. Once these two solutions are found, the data point pairs that determine the compactness and separation of the solutions are detected and presented to the decision maker to learn if they are in the same cluster or not according to him/her. According to the response of the decision maker, certain updates for the incumbent solution, set of interesting solutions and/or the reference point are made if the need arises. For example, if the decision maker thinks that points i and j should be in the same cluster, then the compactness value of the clustering problem is updated according to the definition of compactness objective if necessary. Similarly, if the two points should be in different clusters, then the algorithm considers updating the adjusted separation value of the clustering problem if necessary.

The procedure of interaction with the decision maker is given below in an algorithmic fashion.

```
Let solution $a$ be the best solution in terms of $FA$ (or $FD$), $i$ and $j$ be the two data points that determine the compactness value for solution $a$. Ask the decision maker if $i$ and $j$ are in the same cluster.
IF "YES"
    IF "$gcom$ has not changed yet"
        Set $gcom = fcom_a$, $A^{mult} = 0.5$.
    ELSE
        Set $gcom = \max\{gcom, fcom_a\}$.
    END IF
ELSE ("NO")
    Calculate the adjusted distance $ad_{ij}$.
    $$ad_{ij} = \begin{cases} d_{ij} & \text{if } den_i > d_{ij} \text{ and } den_j > d_{ij} \\ d_{ij}\frac{d_{ij}}{den_i}\frac{d_{ij}}{den_j} & \text{if } den_i < d_{ij} \text{ and } den_j < d_{ij} \\ d_{ij}\frac{d_{ij}}{\min\{den_i, den_j\}} & \text{if } den_i < d_{ij} \text{ or } den_j < d_{ij} \end{cases}$$
    Set $gasep = \min\{ad_{ij}, gasep\}$.
END IF
Let $i$ and $j$ be the two data points that determine the separation value for solution $a$. Ask the decision maker if $i$ and $j$ are in the same cluster.
IF "NO"
    IF "$gasep$ has not changed yet"
        Set $gasep = fasep_a$, $A^{mult} = 0.5$.
    ELSE
        Set $gasep = \min\{fasep_a, gasep\}$.
    END IF
ELSE ("YES")
    Set $gcom = \max\{gcom, d_{ij}\}$.
END IF
```

At the end of interaction with the decision maker, any solution that was able to update the reference point is added to the set of interesting solutions. The condition for an update in the incumbent solution is stricter. For a solution to be the new incumbent solution, its compactness points should be in the same cluster, separation points should be in different clusters and it should at the same time be able to update both *gcom* and *gasep*.

3.5.5 Evolutionary Operators

Uniform crossover and bitwise mutation are the operators used for evolutionary purposes. In a single crossover event, two offspring are generated from two parents. For each pair of parents, a uniform mask, which is a binary array having the same size of an individual’s chromosome, is constructed randomly. Then using this mask, genes of the first offspring whose corresponding values are 1 in the uniform mask are copied from the first parent. Similarly, the genes whose corresponding values are 0 in the uniform mask are copied from the second parent. Second offspring is constructed by reversing the parents. In Figure 3.4 an example of a crossover event can be found. With the crossover operator, merging and splitting of clusters is performed which is an effective way of exploring the solution space.

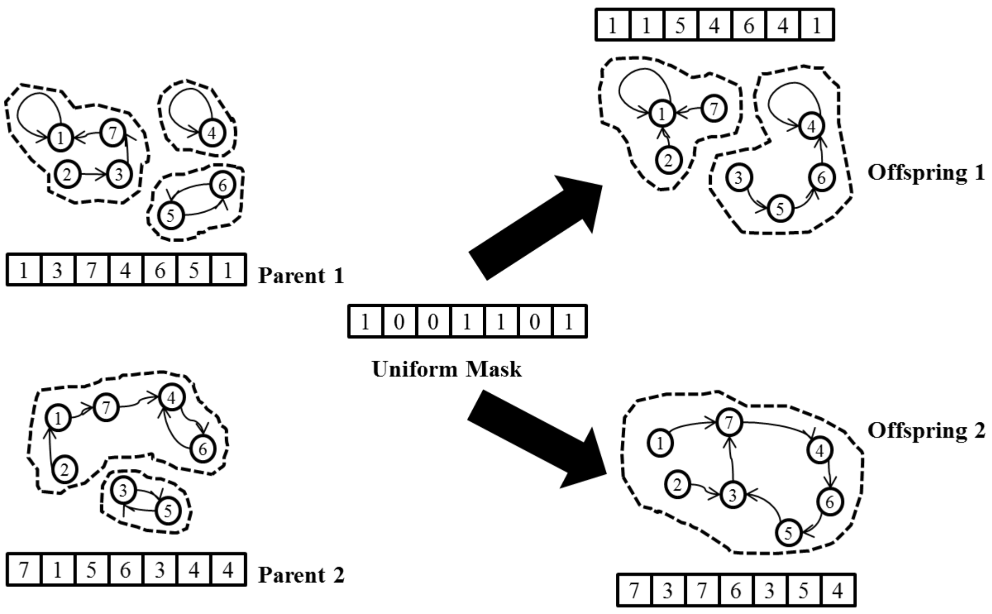

Figure 3.4 A crossover example

Every offspring constructed by the crossover is then subjected to a mutation event. In the algorithm, neighborhood biased mutation, which is proposed by Handl and Knowles (2007), is used. It is a kind of bitwise mutation, where each gene can be mutated with a certain probability, *mprob*. The mutation is done by changing the current value of the mutating gene by randomly assigning a different value for it. Since assigning a value for a gene means constructing a link between two data points represented by the index of the gene and the value of the gene, it may not be a good idea to construct such links between distant data points. Therefore, in the neighborhood biased mutation we use, the set of values that can randomly be assigned to a mutating gene is limited to its H nearest neighbors, where H is a parameter determined before the execution of the algorithm. A mutation operator on an arbitrary solution and visualization of its effect on the clustering is illustrated in Figure 3.5. Similar to the crossover operator, merging and splitting of clusters is performed by the mutation operator.

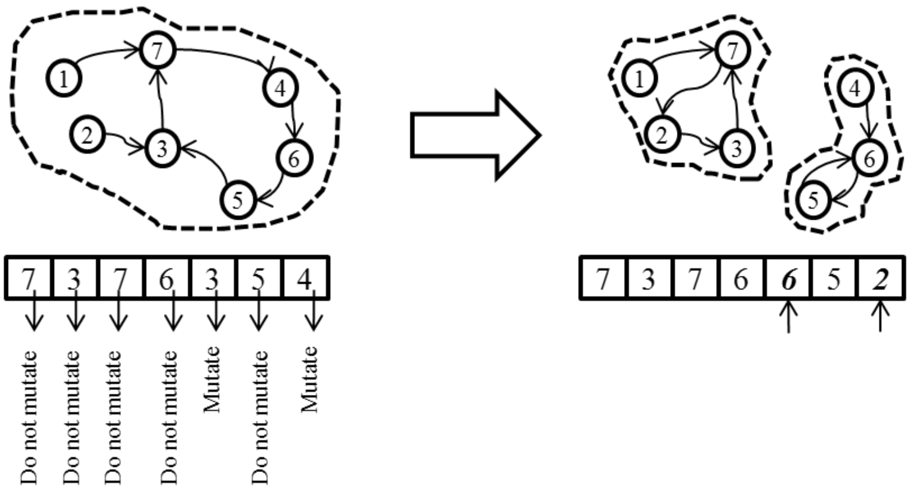

Figure 3.5 A mutation example

Note that, allowing a data point to connect to itself by mutation helps constructing singleton clusters. Therefore, we allow such links to be formed by our mutation operator to be able to detect outliers.

CHAPTER 4

COMPUTATIONAL EXPERIMENTS

In this chapter, we report the empirical results obtained by experimenting with our algorithm. In Section 4.1, we describe the data sets used in this study. After setting the parameters of the algorithm by the results of the pilot runs given in Section 4.2, we give the computational results corresponding to this parameter setting in Section 4.3.

4.1 Data Sets

There are many well-known data sets that can be used for the clustering problem, most of which can be obtained from UC Irvine Machine Learning Repository[1]. However a majority of the data sets provided there include categorical attributes which are not included in our scope. In addition, for some of the data sets, the dimension of the data points may be higher than two, which makes the data set undesirable for us. In this study, we focus on two dimensional data sets with numerical attributes, and the data sets we use are taken from the open literature (Sourina 2011). Table 4.1 provides a list of the data sets used in this study. In Table 4.1, N is the number of data points in the data set, and K indicates the number of clusters in the target solution for that data set, including the number of outliers.

[1] http://archive.ics.uci.edu/ml/

Table 4.1 Data sets used

Set #	**Data Set Name**	***N***	***K***	**# of outliers**	**Data Set Properties***
1	data_60	60	3	0	Elongated, Box
2	data_66	66	4	0	Elongated, Box
3	data-c-cc-nu-n	289	7	4	Spherical, Elongated, Ring, DDBC, DVWC, Outlier
4	data-c-cc-nu-n_v2	285	3	0	Spherical, Elongated, Ring, DDBC, DVWC
5	data-c-cc-nu-n2	195	6	3	Spherical, Elongated, Ring, DDBC, DVWC, Outlier
6	data-c-cc-nu-n2_v2	192	3	0	Spherical, Elongated, Ring, DDBC, DVWC
7	data-c-cv-nu-n	76	6	3	Spherical, Elongated, DDBC, DVWC, Outlier
8	data-c-cv-nu-n_v2	73	3	0	Spherical, Elongated, DDBC, DVWC
9	data-c-cv-u-n	81	5	3	Spherical, DDBC, DVWC, Outlier
10	data-uc-cc-nu-n	191	6	3	Spiral, Spherical, DDBC, DVWC, Outlier
11	data-uc-cc-nu-n_v2	188	3	0	Spiral, Spherical, DDBC, DVWC
12	data-uc-cv-nu-n	127	6	3	Spherical, Elongated, DDBC, DVWC, Outlier
13	dataX	202	4	2	Butterfly, DVWC, Outlier
14	dataX_v2	200	2	0	Butterfly, DVWC
15	data-oo	144	6	4	Spherical, Ring, DDBC, DVWC, Outlier
16	data-oo_v2	140	2	0	Spherical, Ring, DDBC, DVWC
17	train1	307	6	2	Spherical, DDBC, DVWC, Outlier
18	train1_v1	306	5	1	Spherical, DDBC, DVWC, Outlier
19	train2	287	4	0	Spherical, Elongated, DDBC, DVWC
20	train3	397	6	30	Elongated, Snake, DDBC, DVWC, Noise
21	train3_v1	361	5	0	Elongated, Snake, DDBC, DVWC

*DDBC: Density differences between clusters
DVWC: Density variation within a cluster

Besides different number of data points, outliers and target number of clusters, each data set given in Table 4.1 includes a combination of the challenging issues, which are arbitrary shaped clusters, density variation within a cluster and density differences between clusters, as introduced in Chapter 1. The plots of these data sets can be found in Appendix A.

4.2 Parameter Settings and Performance Measures

We have a few parameters that need to be set before the execution of the algorithm. In Table 4.2, we present the final values of the parameters that we choose as a result of our pilot runs. Here, P^{size}, A^{size} and MP^{size} settings are borrowed from SPEA2. We choose equal weights in w^{com} and w^{asep} not to favor one objective over the other.

Table 4.2 Parameter Setting

Parameter	**Value**
H	$\sqrt{N}$
P^{size}	100
A^{size}	50
MP^{size}	100
w^{com}	0.5
w^{asep}	0.5
mprob	0.005
Number of generations	10,000

Taking the neighborhood size H as $\sqrt{N}$ is commonly seen in literature. For *mprob* we have also tried 0.01 for five of the data sets, namely 2, 4, 7, 9 and 11, and seen that 0.005 yields better results. Since we have several types of fitness values in our algorithm, it is difficult to determine a termination criterion that depends on the improvements in the best of these fitness values. To understand its convergence behavior, we ran our algorithm for a long time, i.e. 10,000 generations.

The solutions are evaluated in terms of Jaccard and Rand Indices, which are external validity indices that measure the closeness of our solution to the target solution.

To illustrate, for an arbitrary solution *s*, let

a: the number of data point pairs in the same cluster both in the target solution and *s*,

b: the number of data point pairs in the same cluster in the target solution, but in different clusters in *s*,

c: the number of data point pairs in different clusters in the target solution, but in the same cluster in *s*,

d: the number of data point pairs in different clusters in both *s* and the target solution.

Then the Jaccard Index (*JI*) is calculated as $JI = \frac{a}{a+b+c}$, and the Rand Index (*RI*) is calculated as $RI = \frac{a+d}{a+b+c+d}$. *JI* focuses only on the data point pairs that are assigned to the same cluster correctly in our solution according to the clusters in the target solution. On the other hand, *RI* also gives importance to the data point pairs that are in different clusters in the target solution which are assigned to different clusters in our solution.

The algorithm is coded in C using open source CodeBlocks development environment, and the pilot runs are made on a PC with 3GB RAM, and Intel Core 2 Duo 2.4 GHz processor running Ubuntu 10.04. The results of the pilot runs are given in Table 4.3.

Table 4.3 Pilot Results

		Target				**Best in *I* and *IS***				
Set #	**Data set name**	***K***	***fcom***	***fasep***	**Q**	***K***	***fcom***	***fasep***	***JI***	**g***
1	data_60	3	1.00	12.39	29	3	1.00	12.39	1.00	1,600
2	data_66	4	1.00	1.27	10	4	1.00	1.27	1.00	5,550
3	data-c-cc-nu-n	7	0.77	0.46	19	6	0.77	3.01	0.99	6,000
4	data-c-cc-nu-n_v2	3	0.77	3.01	22	3	0.77	3.01	1.00	2,650
5	data-c-cc-nu-n2	6	0.55	2.60	17	6	0.55	2.60	1.00	3,900
6	data-c-cc-nu-n2_v2	3	0.55	37.54	8	3	0.55	37.54	1.00	2,000
7	data-c-cv-nu-n	6	0.78	6.53	19	6	0.78	6.53	1.00	2,350
8	data-c-cv-nu-n_v2	3	0.78	6.53	19	2	0.71	0.72	0.89	1,050
9	data-c-cv-u-n	5	0.65	5.73	16	5	0.65	5.73	1.00	2,900
10	data-uc-cc-nu-n	6	0.68	4.41	31	5	0.68	2.32	0.50	6,450
11	data-uc-cc-nu-n_v2	6	0.68	4.41	14	2	0.68	94.33	0.96	4,600
12	data-uc-cv-nu-n	6	0.67	2.03	25	5	0.67	9.36	0.98	1,750
13	dataX	4	0.90	51.92	23	4	0.90	51.92	1.00	1,600
14	dataX_v2	2	0.90	51.92	23	2	0.90	51.92	1.00	2,050
15	data-oo	6	0.55	1.27	26	5	0.55	18.18	0.50	450
16	data-oo_v2	2	0.55	1.27	13	1	0.55	0.50	0.50	-
17	train1	6	0.03	0.18	15	6	0.03	0.18	1.00	2,500
18	train1_v1	5	0.03	0.18	23	5	0.03	0.18	1.00	2,800
19	train2	4	0.03	0.26	27	4	0.03	0.26	1.00	4,000
20	train3	36	0.74	0.02	8	3	0.15	0.13	0.31	-
21	train3_v1	5	0.05	0.35	22	5	0.05	0.35	1.00	4,800

In Table 4.3, g* stands for the generation number in which the best solution in terms of its closeness to the target solution with respect to the Jaccard Index is seen in the archive. Q is the number of questions asked to the decision maker throughout the run. Reported solutions are the best among the final incumbent solution (*I*), the set of interesting solutions (*IS*) and the archive (*A*). According to Table 4.3, 2,500-3,000 generations are enough for most of the data sets; however, for some of them at least 6,000-7,000 generations are needed for the best solution to appear in the archive. Note that, the 20th data set, namely train, 3 includes noise, which is different from outliers. Since noise is not within our scope, this data set is discarded from the final runs.

4.3 Computational Results without Decision Maker Preference

From the results of the pilot runs given in Table 4.3, we see that the parameter settings given in Table 4.2 are reasonable. Therefore, we set our parameters as in Table 4.2 in our final runs. Also, in order to observe the effect of the interaction with the decision maker, we first run our algorithm (IP-MOEC) as a purely multiobjective evolutionary clustering algorithm (MOEC), by discarding the interaction phase and decision maker's preferences.

Since our algorithm includes randomness, we have made five replications for each data set. We report the number of distinct solutions found in the final archive in Table 4.4.

Table 4.4 Characteristics of the replications in terms of distinct solutions in the final archive

Set #	**Data Set Name**	**Number of distinct solutions***		
		min	***avg***	***max***
1	data_60	1	1	1
2	data_66	3	3	3
3	data-c-cc-nu-n	4	4	4
4	data-c-cc-nu-n_v2	2	3.6	7
5	data-c-cc-nu-n2	3	3.6	5
6	data-c-cc-nu-n2_v2	2	2.2	3
7	data-c-cv-nu-n	5	13.8	19
8	data-c-cv-nu-n_v2	11	14.4	18
9	data-c-cv-u-n	7	9.4	12
10	data-uc-cc-nu-n	3	5.6	11
11	data-uc-cc-nu-n_v2	2	3.6	6
12	data-uc-cv-nu-n	8	10.2	12
13	dataX	8	8.2	9
14	dataX_v2	8	8.2	9
15	data-oo	6	9.2	10
16	data-oo_v2	6	6	6
17	train1	3	4	5
18	train1_v1	3	4.2	5
19	train2	6	6.8	8
20	train3_v1	4	5.8	8

* Minimum, average and maximum of five replications

The number of distinct solutions in the final archive depends on the characteristics of the data set rather than its size. For example, 20th data set has the greatest number of data points, but the average number of distinct solutions reported for this data set is almost half of the average number of distinct solutions reported for the 7th data set which contains only half the number of data points as data set 20. For the first two data sets, the possible number of nondominated solutions is low, so the number of distinct solutions is also low since the archive is totally filled with nondominated solutions.

For each data set, from the distinct solutions in the final archive, we report the best solution in terms of *JI* and *RI* indices in each replication. The results of five replications are summarized in Table 4.5.

Table 4.5 Performance of proposed algorithm without decision maker interaction (MOEC) in terms of solution quality

Set #	**Data Set Name**	***JI****			***RI****		
		min	***avg***	***max***	***min***	***avg***	***max***
1	data_60	1.00	1.00	1.00	1.00	1.00	1.00
2	data_66	1.00	1.00	1.00	1.00	1.00	1.00
3	data-c-cc-nu-n	0.99	0.99	0.99	0.99	0.99	0.99
4	data-c-cc-nu-n_v2	1.00	1.00	1.00	1.00	1.00	1.00
5	data-c-cc-nu-n2	1.00	1.00	1.00	1.00	1.00	1.00
6	data-c-cc-nu-n2_v2	1.00	1.00	1.00	1.00	1.00	1.00
7	data-c-cv-nu-n	1.00	1.00	1.00	1.00	1.00	1.00
8	data-c-cv-nu-n_v2	1.00	1.00	1.00	1.00	1.00	1.00
9	data-c-cv-u-n	1.00	1.00	1.00	1.00	1.00	1.00
10	data-uc-cc-nu-n	0.45	0.69	0.80	0.61	0.84	0.92
11	data-uc-cc-nu-n_v2	0.45	0.70	0.80	0.60	0.84	0.92
12	data-uc-cv-nu-n	0.98	0.98	0.98	0.99	0.99	0.99
13	dataX	1.00	1.00	1.00	1.00	1.00	1.00
14	dataX_v2	1.00	1.00	1.00	1.00	1.00	1.00
15	data-oo	0.50	0.50	0.50	0.53	0.53	0.53
16	data-oo_v2	0.50	0.50	0.50	0.50	0.50	0.50
17	train1	1.00	1.00	1.00	1.00	1.00	1.00
18	train1_v1	1.00	1.00	1.00	1.00	1.00	1.00
19	train2	1.00	1.00	1.00	1.00	1.00	1.00
20	train3_v1	1.00	1.00	1.00	1.00	1.00	1.00

* Minimum, average and maximum of five replications

Results show that our algorithm performs well for almost all of the data sets. However for data sets 15 and 16, in which two clusters are separated from each other with a distance that is very close to the density of the less dense cluster, the inflation of the separation approach cannot adjust the separation distance properly. Therefore, for these data sets, the target solution remains dominated considering our objectives, and cannot enter the archive.

4.4 Computational Results with Decision Maker Preference

Since our algorithm includes randomness, we have made five replications for each data set. We report the number of distinct solutions found in the union of the final incumbent solution, the set of interesting solutions and the archive in Table 4.6. In Table 4.6, we also report the results about the number of questions asked to the decision maker.

Table 4.6 Characteristics of the replications

Set #	**Data Set Name**	**Number of distinct solutions***			**Number of questions asked***		
		min	***avg***	***max***	***min***	***avg***	***max***
1	data_60	2	3	4	7	17.6	29
2	data_66	3	4	6	10	25.8	45
3	data-c-cc-nu-n	5	5.4	6	14	18.2	22
4	data-c-cc-nu-n_v2	3	4.6	6	12	15.4	22
5	data-c-cc-nu-n2	6	6.4	7	14	16.8	20
6	data-c-cc-nu-n2_v2	2	3.2	4	8	12.2	21
7	data-c-cv-nu-n	10	10.2	11	19	20.8	24
8	data-c-cv-nu-n_v2	6	7.8	11	12	15.2	19
9	data-c-cv-u-n	6	7.8	10	16	20.6	25
10	data-uc-cc-nu-n	5	6.2	8	13	22.2	31
11	data-uc-cc-nu-n_v2	3	4.2	5	14	15.6	18
12	data-uc-cv-nu-n	8	9.2	11	18	22	25
13	dataX	9	10.2	12	23	25.6	32
14	dataX_v2	8	9.2	10	22	23.6	26
15	data-oo	6	8.8	10	14	20.2	26
16	data-oo_v2	5	6	7	14	18.2	23
17	train1	4	5.4	6	15	17.2	19
18	train1_v1	4	5.2	6	12	16.2	23
19	train2	6	6.8	7	15	19.6	27
20	train3_v1	5	6.2	8	16	20.2	26

* Minimum, average and maximum of five replications,

Note that, the size of the archive is taken as 50, and the average number of distinct interesting solutions observed in a replication for all the data sets is approximately 5. Yet, the results presented in Table 4.4 show that the number of distinct solutions is quite less than the maximum possible number of distinct solutions that can occur in the union of the final incumbent solution, the set of interesting solutions and the archive. In addition, the average number of distinct solutions with the preference based approach is expected to be higher than the approach that discards the decision maker preferences, since the number of distinct nondominated solutions is not more than the archive size in any of the data sets. Moreover, on the average, it is expected to observe more distinct solutions with the preference based approach, since with this approach; dominated solutions may also enter the archive. However, the averages in Tables 4.4 and 4.6 do not reflect such a big difference. These show that the preference based approach that lies within our algorithm works well.

Also, the number of questions asked to the decision maker is reasonable and much smaller than the total number of point pairs in a data set. Besides, the number of questions does not increase with the data set size and varies depending on the properties of the data sets. For example, a maximum of 45 questions asked for the 2nd data set is mainly because most of the distances are the same with each other and limited information can be deducted by asking different pairs of points since the objectives determined by those points are equal.

For each data set, we report the best solution from the set of distinct solutions in the final incumbent solution, the set of interesting solutions and the archive in terms of *JI* and *RI* indices. The results are summarized in Table 4.7

Table 4.7 The performance of IP-MOEC in terms of solution quality

Set #	**Data Set Name**	***JI****			***RI****		
		min	***avg***	***max***	***min***	***avg***	***max***
1	data_60	1.00	1.00	1.00	1.00	1.00	1.00
2	data_66	1.00	1.00	1.00	1.00	1.00	1.00
3	data-c-cc-nu-n	0.99	0.99	0.99	0.99	0.99	0.99
4	data-c-cc-nu-n_v2	1.00	1.00	1.00	1.00	1.00	1.00
5	data-c-cc-nu-n2	1.00	1.00	1.00	1.00	1.00	1.00
6	data-c-cc-nu-n2_v2	1.00	1.00	1.00	1.00	1.00	1.00
7	data-c-cv-nu-n	1.00	1.00	1.00	1.00	1.00	1.00
8	data-c-cv-nu-n_v2	1.00	1.00	1.00	1.00	1.00	1.00
9	data-c-cv-u-n	1.00	1.00	1.00	1.00	1.00	1.00
10	data-uc-cc-nu-n	0.50	0.66	0.80	0.61	0.78	0.92
11	data-uc-cc-nu-n_v2	0.50	0.86	0.98	0.60	0.90	0.99
12	data-uc-cv-nu-n	0.98	0.98	0.98	0.99	0.99	0.99
13	dataX	1.00	1.00	1.00	1.00	1.00	1.00
14	dataX_v2	1.00	1.00	1.00	1.00	1.00	1.00
15	data-oo	0.50	0.50	0.50	0.53	0.53	0.53
16	data-oo_v2	0.50	0.50	0.50	0.50	0.50	0.50
17	train1	1.00	1.00	1.00	1.00	1.00	1.00
18	train1_v1	1.00	1.00	1.00	1.00	1.00	1.00
19	train2	1.00	1.00	1.00	1.00	1.00	1.00
20	train3_v1	1.00	1.00	1.00	1.00	1.00	1.00

* Minimum, average and maximum of five replications

Table 4.7 shows that we succeed in finding the target partitioning in 14 data sets out of 20. Moreover, we find the target partitioning for those data sets in all of the five replications, which shows the robustness of our algorithm.

For data sets 3 and 12, our algorithm performs well in all of the five replications, but still fails to find the target partitioning. Each of these data sets includes three outliers, one of which our algorithm fails to detect. Yet our algorithm is still capable of detecting the remaining outliers even though there is no specific mechanism used for this purpose in the algorithm.

Data sets 10 and 11 include both density variation within a cluster and density differences between clusters, and therefore are quite challenging. Data set 10 also includes outliers which makes it even more difficult. However, our algorithm performs relatively worse in only one of the five replications for these data sets. For these data sets, the separation distance occurs between two data points located in particularly dense regions of their clusters, which may be hard to detect and adjust. Yet, in the remaining four replications, our algorithm seems to be capable of adjusting the separation properly and its performance can be considered reasonable.

Our algorithm consistently performs badly for data sets 15 and 16. Unlike data sets 10 and 11, the density difference between clusters in the separation region occurs smoothly for data sets 15 and 16. In these data sets, the separation distance is similar to the density of one of the end points in its corresponding cluster. Therefore, we cannot adjust that separation value properly and our algorithm fails to find the target partitioning for these data sets.

In Table 4.8, we report the performance of our algorithm in terms of the execution times. The maximum observed execution time is 1553 seconds (25 minutes) for data set 20, which has a total of 361 data points.

Table 4.8 The performance of IP-MOEC in terms of execution times

Set #	**Data Set Name**	**Execution times***		
		min	***avg***	***max***
1	data_60	75	78	81
2	data_66	85	87	90
3	data-c-cc-nu-n	1061	1088	1126
4	data-c-cc-nu-n_v2	994	1120	1428
5	data-c-cc-nu-n2	487	496	507
6	data-c-cc-nu-n2_v2	467	519	644
7	data-c-cv-nu-n	105	106	106
8	data-c-cv-nu-n_v2	98	100	102
9	data-c-cv-u-n	122	124	125
10	data-uc-cc-nu-n	459	492	528
11	data-uc-cc-nu-n_v2	455	540	649
12	data-uc-cv-nu-n	237	239	241
13	dataX	533	543	548
14	dataX_v2	528	576	668
15	data-oo	347	348	349
16	data-oo_v2	326	333	349
17	train1	1050	1166	1427
18	train1_v1	1047	1072	1090
19	train2	912	1016	1207
20	train3_v1	1512	1529	1553

* Minimum, average and maximum of five replications (in seconds)

The performance of our algorithm is compared to three well known classical clustering algorithms, namely k-means, SL (single linkage) and DBSCAN, as well as the version of our algorithm that does not account for the preferences of the decision maker.

The results of the classical algorithms are taken from İnkaya et al. (2010). In İnkaya et al. (2010), the k-means algorithm is run for several values of *k* (i.e. the number of clusters in the k-means algorithm) in the range between 2% and 10% of the points in the data set with increments of 1, and the best solution in terms of *JI* is reported in order to be fair in comparison. To illustrate, for a data set having 100 data points, the k-means algorithm is run nine times, each time with a different *k* value in the range [2, 11], which is assured to include the number of clusters in the target solution. Runs of SL are made in the same manner. Similarly, DBSCAN is run for several values of *MinPts* (2-10), which is the only parameter of the algorithm, and the solution having the best *JI* value is taken (İnkaya et al, 2010). The results are summarized in Table 4.9.

Since all three algorithms have shorter execution times than ours, we make multiple runs of those for a fair comparison. However, it should be noted that the number of clusters is given in each run k-means and SL, whereas our algorithm tries to determine this number during its single execution.

Table 4.9 Comparison of IP-MOEC with other clustering algorithms

Set #	**Data set name**	**k-means**[1]		**SL**[2]		**DBSCAN**[3]		**MOEC**[4]		**IP-MOEC**[5]	
		JI	***RI***	***JI***	***RI***	***JI***	***RI***	***JI***	***RI***	***JI***	***RI***
1	data_60	0.79	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	data_66	0.66	0.83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
3	data-c-cc-nu-n	0.78	0.86	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
4	data-c-cc-nu-n_v2	0.80	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
5	data-c-cc-nu-n2	0.28	0.64	1.00	1.00	0.99	0.99	1.00	1.00	1.00	1.00
6	data-c-cc-nu-n2_v2	0.29	0.63	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
7	data-c-cv-nu-n	0.59	0.83	1.00	1.00	0.63	0.68	1.00	1.00	1.00	1.00
8	data-c-cv-nu-n_v2	0.61	0.84	1.00	1.00	0.66	0.86	1.00	1.00	1.00	1.00
9	data-c-cv-u-n	0.93	0.97	1.00	1.00	1.00^{Δ}	1.00^{Θ}	1.00	1.00	1.00	1.00
10	data-uc-cc-nu-n	0.59	0.73	0.48	0.62	0.50	0.83	0.69	0.84	0.66	0.78
11	data-uc-cc-nu-n_v2	0.34	0.73	0.45	0.60	0.59	0.83	0.70	0.84	0.86	0.90
12	data-uc-cv-nu-n	0.62	0.83	0.98	0.99	0.98	0.99	0.98	0.99	0.98	0.99
13	dataX	0.98	0.99	1.00	1.00	1.00^{Δ}	1.00^{Θ}	1.00	1.00	1.00	1.00
14	dataX_v2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
15	data-oo	0.49	0.75	0.50	0.53	0.50	0.53	0.50	0.53	0.50	0.53
16	data-oo_v2	0.52	0.76	0.89	0.95	0.95	0.98	0.50	0.50	0.50	0.50
17	train1	0.99	1.00	1.00	1.00	1.00^{Θ}	1.00	1.00	1.00	1.00	1.00
18	train1_v1	1.00^{ψ}	1.00^{Θ}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
19	train2	0.78	0.91	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
20	train3_v1	0.39	0.75	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

ψ this value is 0.997. Δ this value is 0.998. Θ this value is 0.999.
1 k-means: Best of 0.1*N*-0.02*N* replications, where *N* is the number of data points in the data set
2 SL: Best of 0.1*N*-0.02*N* replications, where *N* is the number of data points in the data set
3 DBSCAN: Best of nine replications
4 MOEC: Average of five replications
5 IP-MOEC: Average of five replications

The performance of MOEC is very close to IP-MOEC, where decision maker's preferences are included. For data set 10, MOEC performs better, and for data set

11, IP-MOEC outperforms MOEC. They perform equally well for the remaining data sets.

Among the algorithms compared, k-means performs worst since it tends to provide only spherical clusters.

All the algorithms perform badly for data set 15. For data set 16, which is a simplified version of data set 15 by removing outliers, SL and DBSCAN perform better than IP-MOEC and MOEC. As stated before, the main reason for IP-MOEC and MOEC to perform badly for data sets 15 and 16 is due to the characteristics of these sets. The clusters are separated by distances close to the density of the less dense cluster, which makes it harder to detect by the approach that aims to inflate the separation in such situations. SL and DBSCAN perform relatively bad for data sets 10 and 11, in which both inter-cluster density differences and intra-cluster density variation is present. For data set 10, IP-MOEC performs better than SL in terms of both *JI* and *RI*. Again, for that data set, the performance of IP-MOEC is better than DBSCAN in terms of *JI* but slightly worse in terms of *RI*. For data set 11, IP-MOEC performs better than both SL and DBSCAN in terms of both *JI* and *RI*. In these data sets, the separation between two clusters occurs between the dense regions of both clusters; therefore this separation can be adjusted properly by MOEC and IP-MOEC.

In general, for all the data sets that we perform well, the performance of our algorithm is not worse than any other algorithm in terms of both *JI* and *RI*.

In Table 4-10, we give a comparison of the algorithms in terms of execution times.

Table 4.10 Comparison of clustering algorithms in terms of execution times

Set #	Data set name	k-means[1]	SL[2]	DBSCAN[3]	MOEC[4]	IP-MOEC[5]
1	data_60	0.21	0.52	0.07	89.43	77.81
2	data_66	0.07	0.38	0.03	98.31	87.39
3	data-c-cc-nu-n	0.24	0.66	0.26	1361.59	1087.74
4	data-c-cc-nu-n_v2	0.05	0.87	0.05	1464.42	1120.33
5	data-c-cc-nu-n2	0.11	0.49	0.19	672.68	496.44
6	data-c-cc-nu-n2_v2	0.32	0.49	0.38	786.75	519.43
7	data-c-cv-nu-n	0.05	0.38	0.04	145.79	105.68
8	data-c-cv-nu-n_v2	0.19	0.39	0.06	132.07	100.36
9	data-c-cv-u-n	0.11	0.38	0.24	162.82	124.26
10	data-uc-cc-nu-n	0.10	0.48	0.12	595.90	491.65
11	data-uc-cc-nu-n_v2	0.09	0.48	0.13	720.90	539.99
12	data-uc-cv-nu-n	0.07	0.42	0.13	394.50	238.73
13	dataX	0.10	0.51	0.13	676.87	543.09
14	dataX_v2	0.83	0.50	2.34	665.86	575.53
15	data-oo	0.11	0.43	0.23	437.81	347.78
16	data-oo_v2	0.73	0.43	1.95	578.46	332.95
17	train1	0.07	0.69	0.13	1311.65	1166.42
18	train1_v1	0.08	0.72	0.11	1319.83	1071.62
19	train2	0.10	0.65	0.11	1177.13	1016.38
20	train3_v1	0.07	0.80	0.12	1803.64	1528.83

1 k-means: Total of 0.1*N*-0.02*N* replications, where *N* is the number of data points in the data set
2 SL: Total of 0.1*N*-0.02*N* replications, where *N* is the number of data points in the data set
3 DBSCAN: Total of nine replications
4 MOEC: Average of five replications
5 IP-MOEC: Average of five replications

Traditional algorithms are simple algorithms and run quite fast. However, they need the number of clusters to be known before the execution, which is not required in MOEC and IP-MOEC. Results in Table 4.10 show that MOEC requires more computation time than its interactive preference based version IP-MOEC. This is because of the difference between convergence behaviors of these two algorithms. IP-MOEC converges faster with the help of the information gathered from the decision maker. With decision maker's preferences, IP-MOEC does not need to search some portion of the solution space that includes solutions with different number of clusters. Since both algorithms construct minimum spanning trees for each cluster of each solution, converging to a front with solutions having the target number of clusters decreases the execution time in later generations.

CHAPTER 5

CONCLUSIONS

The clustering problem with its importance in exploratory data analysis is one of the active research areas under data mining. It has several challenging issues, such as the unknown number of clusters, arbitrary shapes, outliers, inter-cluster density differences and intra-cluster density variations, which are introduced in more detail in Chapter 1. There are many algorithms developed for dealing with the aforementioned challenging issues. However, an algorithm performing well for a data set with a certain challenging issue, occasionally fails when a different challenging issue is introduced. To the best of our knowledge, no algorithm is proven to be successful with all data sets. Our scope in this study includes clustering problems with all of the challenging issues mentioned above.

For the clustering problem, we developed and implemented an interactive multiobjective evolutionary algorithm which also considers the preferences of a decision maker during its execution. Decision maker preferences are extracted from his/her responses to simple questions such as two data points being in the same cluster or not. These preferences are implicitly used in the algorithm to generate a small portion of the Pareto optimal front desired by the decision maker, rather than generating a well spread representation of the whole Pareto optimal front. As reported in Chapter 4, the small number of distinct solutions generated and the reasonable number of solutions asked to the decision maker indicates that our preference based approach works fine.

We used 20 data sets from open literature (Sourina, 2011) to test the performance of our algorithm, and observed that the results are quite reasonable. The performance of our algorithm is compared with three well-known clustering algorithms, namely k-means, SL, and DBSCAN and is proven to provide at least as good results as the compared algorithms in all data sets except one, namely data-oo_v2. Moreover, in most of the data sets, the performance of our algorithm is better than the performances of the compared algorithms, which shows its strength and robustness for data sets with different characteristics.

From the results, we observe that our algorithm performs consistently badly for two data sets, namely data-oo and data-oo_v2. As a future work, the underlying reason may be investigated further to adapt the algorithm in finding the target clusters in such data sets.

Our algorithm can detect outliers, but in some cases outliers very close to a cluster may be missed. Moreover, we do not deal with data sets having noise. As another future work, a preprocessing mechanism may be developed to detect outliers and remove the noise before the execution of the algorithm. By this way, the size of the data set is expected to decrease, and the algorithm is expected to yield better results.

In the algorithm, the responses of the decision maker are implicitly used to include his/her preferences in the search by altering the reference point if possible. However, the information provided by the decision maker is more valuable than that since he/she directly tells if two data points belong to the same cluster or different clusters. In addition to affecting the reference point, user information can be converted into cannot link and must link constraints. Then such constraints may be enforced to the evolutionary operators in order to reduce the search space, so that a more efficient search can be made. However, such an improvement is not straightforward considering the structures of the algorithm; especially the edge based solution representation, and is therefore aimed to be done as future work.

The weights used in achievement scalarizing functions are taken as equal not to favor one objective over the other. However, with proper information from the decision maker, these weights may be adjusted to incorporate the decision maker preferences in a different way.

The maximum execution time observed in all the replications is less than half an hour, which is not quite high and can be considered reasonable. However, we believe that it can be decreased even further by a better coding scheme. This would make the algorithm more scalable and allow its use for larger data sets.

We use a fixed neighborhood size for all data points in a data set, and it is equal to the square root of the total number of data points in that set. This neighborhood size should be supported with an upper bound for data sets having higher number of data points. Even better, with a special procedure, point specific neighborhood sizes can be found by preprocessing the data set, which is may improve the performance of our algorithm as well as saving us from setting a parameter.

We focus on two dimensional data sets in this study. The adaptation of the algorithm for higher dimensional data sets may also be done in the future.

In conclusion, we suggest our algorithm for relatively small spatial data sets with outliers which also include density differences between clusters and density variation within a cluster, and the solution quality is of higher importance than the execution time.

REFERENCES

Abraham, A., Grosan, C., and Chis, M., “Swarm intelligence in data mining”, *Swarm Intelligence in Data Mining, Studies in Computational Intelligence*, p.1-20, Abraham, A., Grosan, C., Ramos, V. (Eds.), 2006.

Al-Sultan, K., “A tabu search approach to the clustering problem”, *Pattern Recognition*, 28(9):1443-1451, 1995.

Ankerst, M., Breunig, M., Kriegel, H.-P., and Sander, J., “OPTICS: Ordering points to identify clustering structure”, *In Proceedings of the ACM SIGMOD Conference*, 49-60, Philadelphia, PA., 1999.

Babu, G. P., and Murty, M. N., “A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm”, *Pattern Recognition Letters*, 14(10):763-769, 1993.

Babu, G. P., and Murty, M. N., “Simulated annealing for optimal initial seed selection in k-means algorithm” *Indian Journal of Pure and Applied Mathematics*, 25:85- 94, 1994.

Bandyopadhyay, S., and Maulik, U., “An evolutionary technique based on k-means algorithm for optimal clustering”, *in Rn. Information Science Applications: An International Journal*, 146(1-4):221-237, 2002.

Bandyopadhyay, S., Maulik, U., and Mukhopadhyay, A., “Multiobjective Genetic Clustering for Pixel Classification in Remote Sensing Imagery” *Transactions On Geoscience And Remote Sensing* , Volume: 45, Issue: 5, Pages: 1506- 1511, 2007.

Berkhin, P., “A survey of clustering data mining techniques, Grouping Multidimensional Data”, *Springer Berlin Heidelberg, Kogan, J., Nicholas, C., and Teboulle M. (Eds.),* p.25-71, 2006.

Branke, J., “Consideration of Partial User Preferences in Evolutionary Multiobjective Optimization”, *Springer, Berlin,* pp. 157–178, 2008.

Branke, J., Kaußler, T., and Schmeck, H., “Guidance in evolutionary multi-objective optimization”, *Adv. Eng. Software,* vol. 32, no. 6, pp. 499–507, 2001.

Calinski, R. B., & Harabasz, J. (1974). “A dendrite method for cluster analysis”, *Communications in Statistics, 3,* 1-27.

Chen, E., Wang, F., “Dynamic Clustering Using Multi-objective Evolutionary Algorithm”, *CIS 2005, Part I, LNAI 3801, pp. 73 – 80,* 2005.

Chu, S.C., Roddick, J. F., Su, C.J., and Pan, J.S., “Constrained ant colony optimization for data clustering” *In PRICAI,* pages 534-543, 2004.

Coello, C. A. C., “Handling Preferences in Evolutionary Multiobjective Optimization”, *A Survey, in: Proceedings of the 2000 Congress on Evolutionary Computation,* IEEE Service Center, Piscataway, NJ, 30-37, 2000.

Das, S., Abraham, A. and Konar, A., “Metaheuristic Clustering”, *Studies in Computational Intelligence*, Springer-Verlag Berlin Heidelberg, 2009.

Davies, D. L., & Bouldin, D. W. (1979). “A cluster separation measure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1, 224-227.*

Deb K, Pratap A, Agarwal S., “A fast and elitist multi-objective genetic algorithm: NSGA-II” *IEEE Transactions on Evolutionary Computation,* 6 (2); 182-197, 2002.

Deb, K., “Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions”, *Indian Inst. Technol., Kanpur, India, KanGAL* Rep. 99002, 1999.

Deb, K., and Kumar, A., “Interactive evolutionary multi-objective optimization and decision-making using reference direction method”, *Technical Report KanGAL report number 2007001, Kanpur Genetic Algorithms Laboratory, Department of Mechanical engineering, Indian Institue of Technology Kanpur, India*, 2007.

Deb, K., Kumar, A., “Light beam search based multi-objective optimization using evolutionary algorithms”, *in: Proceedings of the Congress on Evolutionary Computation (CEC-07),* pp. 2125–2132, 2007.

Deb, K., Sundar, J., Rao, U.B., Chaudhuri, S., “Reference point based multi-objective optimization using evolutionary algorithms”, *International Journal of Computational Intelligence Research,* 2(3) (2006) 273–286.

Defays, D., “An efficient algorithm for a complete link method”, *The Computer Journal,* 20, 364-366, 1977.

Du, J., Korkmaz, E. E., Alhajj, R., Barker, K., “Alternative Clustering by Utilizing Multi-objective Genetic Algorithm with Linked-List Based Chromosome Encoding”, *MLDM 2005, LNAI 3587, pp. 346–355,* 2005.

Ester, M., Kriegel, H-P., Sander, J. and Xu, X. “A density-based algorithm for discovering clusters in large spatial databases with noise”. *In Proceedings of the 2nd ACM SIGKDD,* 226-231, Portland, Oregon, 1996.

Guha, S., Rastogi, R., and Shim, K., “CURE: An efficient clustering algorithm for large databases”, *In Proceedings of the ACM SIGMOD Conference, 73-84, Seattle, WA,* 1998.

Guha, S., Rastogi, R., and Shim, K., “ROCK: A robust clustering algorithm for categorical attributes”, *In Proceedings of the 15th ICDE, 512-521, Sydney, Australia,* 1999.

Handl, J. and Knowles, J., “Evolutionary Multiobjective Clustering”, *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII),* pages 1081-1091, 2004a.

Handl, J. and Knowles, J., “Multiobjective clustering with automatic determination of the number of clusters”, *Technical Report TR-COMPSYSBIO-2004-02. UMIST, Manchester, UK,* 2004b.

Handl, J., and Knowles, J., “An evolutionary approach to multiobjective clustering,” *IEEE Trans. Evol. Comput.,* vol. 11, no. 1, pp. 56–76, Feb. 2007.

Handl, J., Knowles, J. D., and Dorigo, M., “On the performance of ant-based clustering”, *In A. Abraham, M. Koppen, and K. Franke, editors, Design and Application of Hybrid Intelligent Systems, Proc. Third International Conference on Hybrid Intelligent Systems, volume 105 of Frontiers in Artificial Intelligence and Applications,* pages 204-213. IOS Press, 2003.

Hartigan, J., “Clustering Algorithms”, *John Wiley & Sons, New York, NY,* 1975.

Hartigan, J., and Wong, M., “Algorithm AS136: A k-means clustering algorithm”, *Applied Statistics,* 28, 100-108, 1979.

Hinneburg, A., and Keim, D., “An efficient approach to clustering large multimedia databases with noise”, *In Proceedings of the 4th ACM SIGKDD, 58-65, New York, NY,* 1998.

Hruschka, E. R., Campello, R. J. G. B., Freitas, A., Carvalho, P. L. F., “A Survey of Evolutionary Algorithms for Clustering”, “*Transactions On Systems Man And Cybernetics Part C-Applications And Reviews*”, Volume: 39, Issue: 2, Pages: 133-155, 2009

Horn, J., Nafpliotis, N., and Goldberg, D. E., “A Niched Pareto Genetic Algorithm for Multiobjective Optimization”, *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence,* volume 1, pages 82–87, 1994.

İnkaya T., Kayalıgil S., Özdemirel N.E., “A new density-based clustering approach in graph theoretic context”, *International Journal of Computer Science and Information Technology,* 5(2), 117-135, 2010.

Jain, A., Murty, M., and Flynn, P., "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

Kanade, P. M., and Hall, L. 0., "Fuzzy ants clustering with centroids" *In Proceedings of the International Conference on Fuzzy Systems*, 2004.

Karasakal, E., and A. Silav, "A Multi-Objective Genetic Algorithm for a Bi-Objective Facility Location Problem with Partial Coverage", *10-05, IE, METU,* December, 2010.

Karypis, G., Han, E.H., and Kumar, V., "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling", *COMPUTER*, 32, 68-75, 1999a.

Korkmaz, E., E., Du, J., Alhajj, R., Barker, K., "Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering", Intelligent Data Analysis, Volume: 10, Issue: 2, Pages: 163-182, 2006.

Krettek, J., Braun, J., Hoffmann, F., and Bertram, T., "Interactive Incorporation of User Preferences in Multiobjective Evolutionary Algorithms", *Applications of Soft Computing Advances in Intelligent and Soft Computing*, Volume 58/2009, 379-388, 2009.

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

Law, M. H. C., Topchy, A. P., Jain, A. K., "Multiobjective Data Clustering," *cvpr, vol. 2, pp.424-430, 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)* - Volume 2, 2004

Lumer, E. D., and Faieta, B., "Diversity and adaptation in populations of clustering ants", *In From Animals to Animats 3: Proc. of the 3th Int. Conf. on the Simulation of Adaptive Behaviour,* pages 501-508, Cambridge, MA, MIT Press, 1994.

Maulik, U., and Bandyopadhyay, S., "Genetic algorithm based clustering technique", *Pattern Recognition,* 33:1455-1465, 2000.

Molina, J., Santana, L.V., Hernandez-Diaz, A.G., Coello Coello, C.A., Caballero, R., “g-dominance: Reference point based dominance”, *European Journal of Operational Research*, 2009.

Monmarche, N., “On data clustering with artificial ants”, *In A. A. Freitas, editor, Data Mining with Evolutionary Algorithms: Research Directions,* pages 23-26, Orlando, Florida, AAAI Press, 1999.

Murtagh, F., “Multidimensional Clustering Algorithms”, *Physica-Verlag, Vienna,* 1985.

Olafsson, S., Li, X., Wu, S., “Operations research and data mining”, *European Journal of Operational Research* Volume 187, Issue 3, 16, Pages 1429-1448, June 2008.

Olson, C., “Parallel algorithms for hierarchical clustering”, *Parallel Computing*, 21, 1313-1325, 1995.

Özyer, T., Liu, Y., Alhajj, R., Barker, K., “Multi-objective Genetic Algorithm Based Clustering Approach and Its Application to Gene Expression Data”, Advances In Information Systems, Proceedings, Lecture Notes In Computer Science, Volume: 3261, Pages: 451-461, 2004.

Pal and Biswas (1997). “Cluster Validation Using Graph Theoretic Concepts”, *Pattern Recognition, Vol. 30, No. 6, pp. 847-857*

Rand, W. “Objective criteria for the evaluation of clustering methods”, *J. Amer. Statist. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.

Rayward-Smith, V. J., “Metaheuristics for Clustering in KDD”, *Proceedings of the IEEE Congress on Evolutionary Computing*, 3, p.2380- 2387, 2005.

Ripon, K. S.N., Tsang, C., Kwong, S., Ip, M., “Multi-Objective Evolutionary Clustering using Variable-Length Real Jumping Genes Genetic Algorithm”, *Proceedings of the 18th International Conference on Pattern Recognition*, 2006

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X., "Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications", *In Data Mining and Knowledge Discovery*, 2, 2, 169-194, 1998.

Selim, S. Z., and Al-Sultan, K., "A simulated annealing algorithm for the clustering problem", *Pattern Recognition,* 24(10):1003-1008, 1991.

Sibson, R., "SLINK: An optimally efficient algorithm for the single link cluster method", *Computer Journal*, 16, 30-34, 1973.

Sourina, O., 2011. "Current Projects in the Homepage of Olga Sourina", (http://www.ntu.edu.sg/home/eosourina/projects.html, last accessed on March 2, 2011).

Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J. "A preference-based interactive evolutionary algorithm for multiobjective optimization", *Technical Report W-412, Helsinki School of Economics, Helsinki, Finland*, 2007.

Tibshirani, R., Walther, G., and Hastie, T., "Estimating the number of clusters in a dataset via the Gap statistic" *J. Royal Statist. Soc.: Series B (Statistical Methodology),* vol. 63, no. 2, pp. 411–423, 2001.

Voorhees, E.M., "Implementing agglomerative hierarchical clustering algorithms for use in document retrieval", *Information Processing and Management*, 22, 6, 465-476, 1986.

Wierzbicki, A. P., "The use of reference objectives in multiobjective optimization", In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making Theory and Applications,* pages 468–486. Berlin: Springer-Verlag, 1980.

Won, J., Ullah, S., Karray, F., "Data Clustering Using Multi-Objective Hybrid Evolutionary Algorithm", *International Conference on Control, Automation and Systems*, 2008.

Xu, R., and Wunsch, D., "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

Xu, X., Ester, M., Kriegel, H.-P., and Sander, J., “A distribution-based clustering algorithm for mining in large spatial databases”, *In Proceedings of the 14th ICDE,* 324-331, Orlando, FL, 1998.

Yousri et al. (2008). “A Novel Validity Measure for Clusters of Arbitrary Shapes and Densities” *International Conference on Pattern Recognition, 3454-3457*

Zhang, T., Ramakrishnan, R. and Livny, M., “BIRCH: an efficient data clustering method for very large databases”, *In Proceedings of the ACM SIGMOD Conference,* 103-114, Montreal, Canada, 1996..

Zhang, T., Ramakrishnan, R., and Livny, M., “BIRCH: A new data clustering algorithm and its applications”, *Journal of Data Mining and Knowledge Discovery,* 1, 2, 141-182, 1997.

Zitzler E., and Kuenzli S., “Indicator-based Selection in Multiobjective Search”, *in:, Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings, Springer-Verlag, Berlin,* 832-842, 2004.

Zitzler, E., and Thiele, L., “Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach”, *IEEE Transactions on Evolutionary Computation 3*(4), 257–271, 1999.

Zitzler, E., Laumanns, M., and Thiele, L., “SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization”, *In K. Giannakoglou et al., editors, EUROGEN 2001, International Center for Numerical Methods in Engineering (CIMNE),* pages 95–100, 2002.

APPENDIX A

THE PLOTS OF THE DATA SETS USED IN THE STUDY

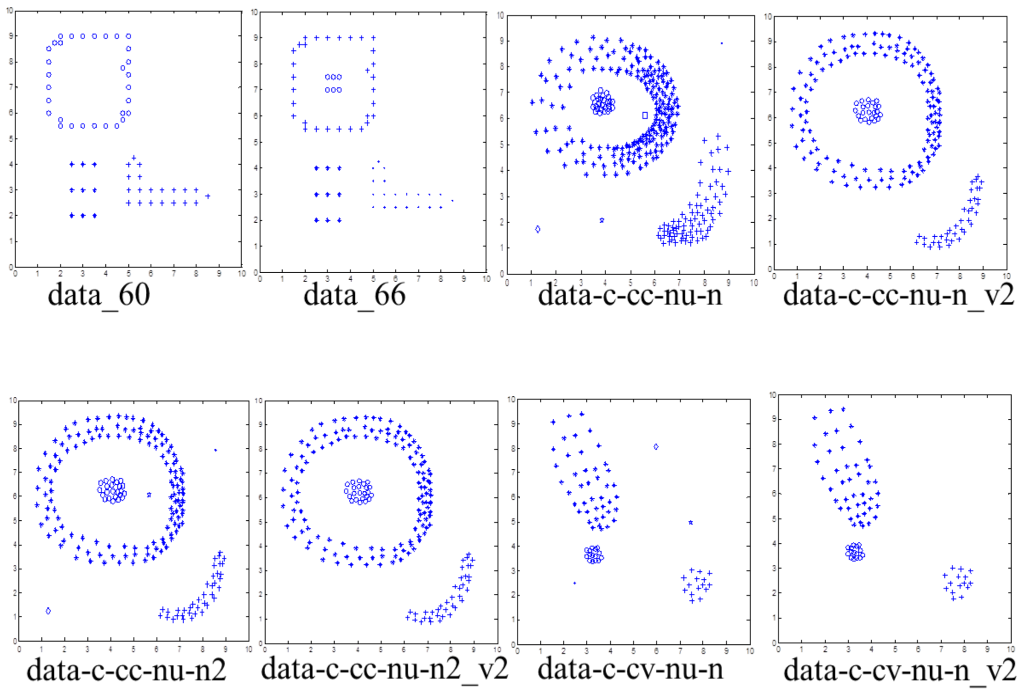

Figure A.1 Plots of data sets (1-8)

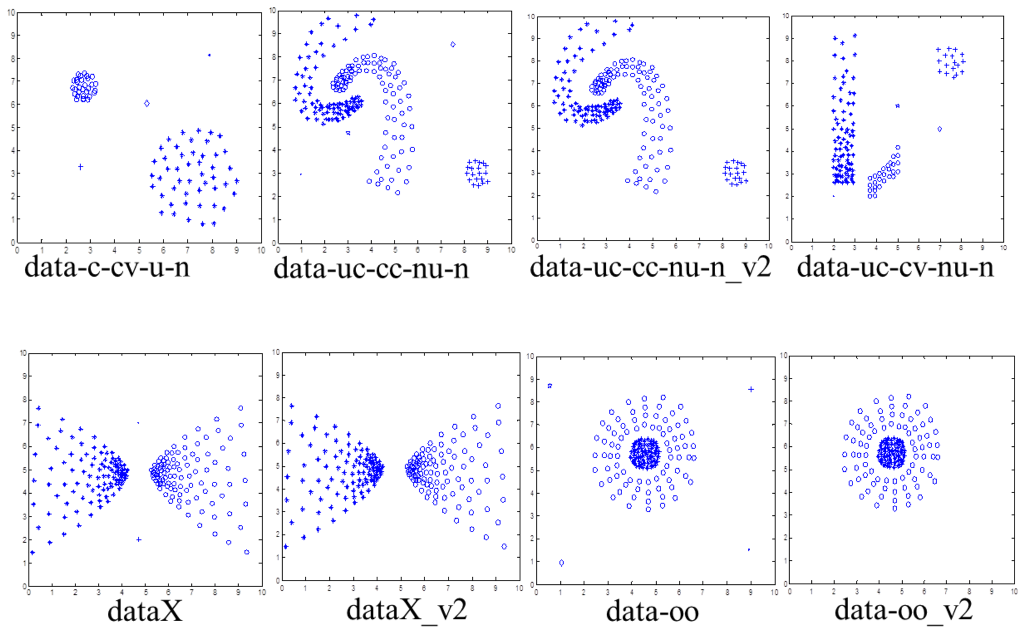

Figure A.2 Plots of data sets (9-16)

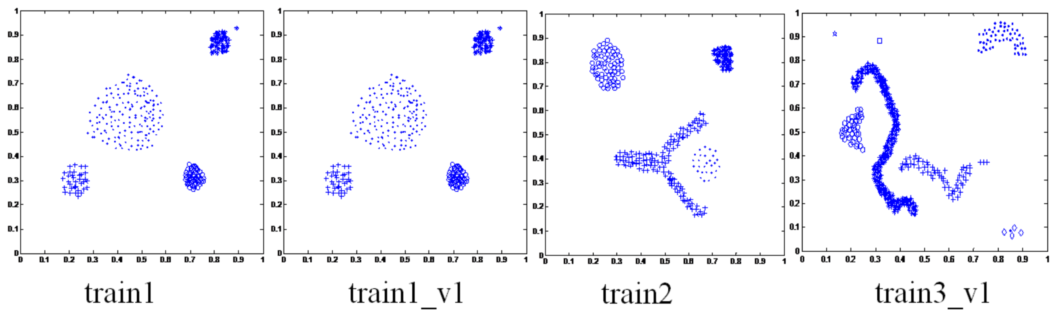

Figure A.3 Plots of data sets (17-20)