A METHODOLOGY OF SWARM INTELLIGENCE APPLICATION IN CLUSTERING BASED ON NEIGHBORHOOD CONSTRUCTION

TÜLİN İNKAYA

MAY 2011

A METHODOLOGY OF SWARM INTELLIGENCE APPLICATION IN CLUSTERING BASED ON NEIGHBORHOOD CONSTRUCTION

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

BY

TÜLİN İNKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

MAY 2011

Approval of the thesis:

A METHODOLOGY OF SWARM INTELLIGENCE APPLICATION IN CLUSTERING BASED ON NEIGHBORHOOD CONSTRUCTION

submitted by **TÜLİN İNKAYA** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy 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. Sinan Kayalıgil _____________________
Supervisor, **Industrial Engineering Dept., METU**

Prof. Dr. Nur Evin Özdemirel _____________________
Co-Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

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

Prof. Dr. Sinan Kayalıgil _____________________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Tuğba Taşkaya Temizel _____________________
Informatics Institute, METU

Assist. Prof. Dr. Pınar Şenkul _____________________
Computer Engineering Dept., METU

Assoc. Prof. Dr. Murat Caner Testik _____________________
Industrial Engineering Dept., Hacettepe University

Date: 06.05.2011

ABSTRACT

A METHODOLOGY OF SWARM INTELLIGENCE APPLICATION IN CLUSTERING BASED ON NEIGHBORHOOD CONSTRUCTION

İnkaya, Tülin
Ph.D., Industrial Engineering Department
Supervisor : Prof. Dr. Sinan Kayalıgil
Co-Supervisor : Prof. Dr. Nur Evin Özdemirel

May 2011, 303 pages

In this dissertation, we consider the clustering problem in data sets with unknown number of clusters having arbitrary shapes, intracluster and intercluster density variations.

We introduce a clustering methodology which is composed of three methods that ensures extraction of local density and connectivity properties, data set reduction, and clustering. The first method constructs a unique neighborhood for each data point using the connectivity and density relations among the points based upon the graph theoretical concepts, mainly Gabriel Graphs. Neighborhoods subsequently connected form subclusters (closures) which constitute the skeleton of the clusters. In the second method, the external shape concept in computational geometry is adapted for data set reduction and cluster visualization. This method extracts the external shape of a non-convex n-dimensional data set using Delaunay triangulation. In the third method, we inquire the applicability of Swarm Intelligence to clustering using Ant Colony Optimization (ACO). Ants explore the data set so that the clusters are detected using density break-offs, connectivity and distance information. The proposed ACO-based algorithm uses the outputs of the neighborhood construction (NC) and the external shape formation. In addition, we propose a three-phase clustering algorithm that consists of NC, outlier detection and merging phases.

We test the strengths and the weaknesses of the proposed approaches by extensive experimentation with data sets borrowed from literature and generated in a controlled manner. NC is found to be effective for arbitrary shaped clusters, intracluster and intercluster density variations. The external shape formation algorithm achieves significant reductions for convex clusters. The ACO-based and the three-phase clustering algorithms have promising results for the data sets having well-separated clusters.

Keywords: Density-based Clustering, Spatial Data Sets, Neighborhood Construction, Ant Colony Optimization, Connectivity

ÖZ

KÜMELEMEDE KOMŞULUK KURMAYA DAYALI SÜRÜ ZEKASI UYGULAMA METODOLOJİSİ

İnkaya, Tülin
Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Sinan Kayalıgil
Ortak Tez Yöneticisi : Prof. Dr. Nur Evin Özdemirel

Mayıs 2011, 303 sayfa

Bu tezde, küme sayısının bilinmediği, değişik şekilde kümeler ve yoğunluk farklılıkları içeren veriler için kümeleme problemini ele aldık.

Yerel yoğunluk ve bağlantı özelliklerini ortaya çıkaran, veri indirgeme ve kümelemeyi sağlayan üç metottan oluşan bir kümeleme metodolojisi önerdik. İlk metot, başta Gabriel çizgeleri olmak üzere çizge kuramı kavramlarını temel alarak ve noktalar arası bağlantı ve yoğunluk ilişkilerini kullanarak her veri noktası için kendine özgü bir komşuluk tanımlar. Sonrasında birbirlerine bağlı komşuluklar altkümeleri (örtüm) oluşturur. Bu altkümeler kümelerin temel iskeletini meydana getirir. İkinci metotta, hesaplamalı geometrideki dış şekil kavramı, veri indirgeme ve küme görselleştirme açalı uyarlanmıştır. Bu metot, dışbükey olmayan *n*-boyutlu verilerin dış şeklini Delaunay üçgenlemeyi kullanarak bulmaktadır. Üçüncü metotta, karınca kolonisi sezgiseli (KKS) kullanarak sürü zekasının kümeleme problemine uygulanabilirliği araştırılmıştır. Karıncalar verileri incelerler; böylece yoğunluk kırılmaları, bağlantılar ve uzaklık bilgileri kullanılarak kümeler tanımlanır. Önerilen

KKS tabanlı algoritma, komşuluk kurma (KK) ve dış şekil oluşturma yöntemlerinin çıktılarını kullanır. Ayrıca, YK, aykırı değer bulma ve birleştirme aşamalarından oluşan, üç-aşamalı kümeleme algoritması önerilmiştir.

Önerilen yaklaşımların güçlü ve zayıf yönleri, literatürden alınan veriler ile kontrollü bir şekilde meydana getirilen veriler kullanılarak kapsamlı deneyler ile test edilmiştir. KK, değişik şekilde kümeler ile küme içi ve küme dışı yoğunluk farklılıkları için etkili bulunmuştur. Dış şekil oluşturma algoritması, dışbükey kümelerde anlamlı azalmalar sağlamıştır. KKS tabanlı algoritma ve üç-aşamalı kümeleme algoritması iyi ayrılmış kümelerde umut verici sonuçlar vermiştir.

Anahtar Kelimeler: Yoğunluk Tabanlı Kümeleme, Uzamsal Veri, Komşuluk Oluşturma, Karınca Kolonisi Sezgiseli, Bağlantı

To Mom, Dad, Tuğba, Auntie

and

to the beloved memory of Gülsüm Batır in particular

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my supervisors, Prof. Dr. Sinan Kayalıgil and Prof. Dr. Nur Evin Özdemirel for their patient guidance, support, encouragements and insight. Working with them was a wonderful challenge, made up of hard work, enthusiasm and fun at the same time. I deem myself fortunate to have worked with such two excellent academicians. I will miss our meetings very much.

I would like to express my special thanks to my Thesis Monitoring Committee Members, Assist. Prof. Dr. Cem İyigün and Assist. Prof. Dr. Tuğba Taşkaya Temizel for their precious guidance and contributions throughout the study. I also would like to thank to Assist. Prof. Dr. Pınar Şenkul and Assoc. Prof. Dr. Murat Caner Testik for their valuable contributions and comments in my thesis defense jury.

I would like to express my sincere gratitude to Prof. Dr. Erdal Emel and Assist. Prof. Dr. Mehmet Akansel for their encouragement and faith that had initiated and yielded this work.

I feel myself as a part of a big family during my teaching and research assistantship years in the Industrial Engineering Department of Middle East Technical University. I am grateful to all my professors, my colleagues, my students and the staff of the department for their contributions and smiling faces.

Particularly, I would like to thank Prof. Dr. Meral Azizoğlu, Assist. Prof. Dr. Sedef Meral, Prof. Dr. Levent Kandiller, Prof. Dr. Erol Sayın, and Assist. Prof. Dr. Seçil Savaşaneril who provided me encouragement and guidance from the beginning of my PhD study. I also acknowledge Şule Çimen for her help, everlasting patience and kindness.

My special thanks go to my beloved friend Banu Lokman for her brilliant ideas, joy and contributions. I am greatly indebted to Pelin Bayındır, Mustafa Gökçe Baydoğan, Baykal Hafızoğlu and Bora Kat who provided every mean of support and made me smile in the hardest times of this dissertation. I also owe thanks to Büşra Atamer, Sakine Batun, Aykut Bulut, Bilge Çelik, Erdem Çolak, Ayşegül Demirtaş, Kerem Demirtaş, Özlem Karabulut, Çınar Kılcıoğlu, Melih Özlen, Selin Özpeynirci, Oğuz Solyalı and Banu Soylu for everything they have shared with me.

I am grateful to my dormitory friends Tülay Akbey, Nuray Ateş, Berna Hasçakır, Güliz Karaarslan and Şule Okuroğlu for their continuous motivation and cheerful talks. I also would like to thank to my dear friends Hacer Aslan, Işıl Kirkizoğlu, Zeynep Kirkizoğlu and Güneş Kartaltepe for their support and presence that enriched my life.

Last and the most, special thanks go to my parents Nursen and Kadir İnkaya, my aunt Firuzan Batır, and my sister Tuğba İnkaya. During this long journey, I always felt their unconditional love, endless support and faith. Their presence is the most precious blessing in my life, and every success I gain is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Data mining has emerged as an interdisciplinary field to extract the valid, interesting and potentially useful patterns in a data set. Data mining has an analogy with gold mining. Knowledge, as precious as gold, is to be mined from a vast pile of data. During the last few decades, the developments in data storage and processing have increased the electronic data production rate immensely. Hence penetration of data mining into several application areas has taken place quite fast, bringing about the necessity of new approaches in the field. Just as Rutherford D. Roger, a librarian in Yale University, stated (Hastie et al. 2009).

"We are drowning in information and starving for knowledge."

Data mining (DM) has been applied successfully in several real-life problems such as Customer Relationship Management (CRM), fraud detection, astronomy, geography, and manufacturing. For example, Wal-Mart used customer transaction databases from its approximately 2900 stores in six countries to extract the customer-buying patterns (market basket analysis) as input to CRM systems (Spinello 1997). Another DM application is the detection and prevention of money laundering activities by the U.S. Treasury Financial Crimes Enforcement Network. In a two-year period the reported laundered funds were approximately $1 billion (Senator et al. 1995). DM also leads automation in astronomical discoveries. For example Jet Propulsion Laboratory and Palomar discovered 22 quasars using DM (Han and Kamber 2001). Three European airlines used a system called CASSIOPEE developed by General Electric and SNECMA (a French engine manufacturer) for the prediction and diagnosis of faults in Boeing 737 airplanes. CASSIOPEE won the European first prize for innovative applications (Piatetsky-Shapiro et al. 1996).

Why is data mining important? First, it is a valuable tool to provide a deeper understanding of a system through analyzing a data set. That is, it helps to model the system, discover the relationships, make predictions and generalizations. The insights gained from this process contribute to decision making. DM is also used for visualization purposes. Hence, DM attracts a wide variety of fields for application such as science, business, engineering, and so on.

Background of Clustering

Main data mining tasks are classification, clustering, association rule mining and regression. In this dissertation we focus on clustering. Basically, clustering forms groups such that similar objects are assigned to the same group whereas dissimilar ones are in different groups. It is an unsupervised learning method where the natural groupings are explored without any guidance or use of external information.

The goals of a clustering application can be summarized as follows.

(1) An exploratory tool to discover the hidden and the potentially useful patterns

- Categorization and typology
- Conceptualization of the properties
- Hypothesis generation for exploratory purposes
- Hypothesis testing for confirmation purposes
- Generalization

(2) Simplification and data compression through generalizations

(3) Visualization

The exploratory property of clustering ensures to broaden the understanding of the system of interest and provides invaluable insight to the domain experts. For example, gene clustering in biology facilitates tracing the evolutionary process throughout the ages, clusters in market segmentation help to identify the consumer patterns and target markets, and city planners use clusters formed by buildings with similar properties (size, style, etc.) to improve the existing environment of the community.

A real-world clustering example for exploratory purposes emerged from the cooperation of NASA and Jet Propulsion Laboratory in California Institute of Technology. In years 1990 through 1994 Magellan spacecraft collected a huge amount of data from the surface of planet Venus. Burl et al. (1998) applied clustering to this image data of Venus in order to recognize the characteristics of the volcanoes and to understand the geological evolution of the planet. This project helped geologists to discover the volcanic properties, and it was one of the leading real-world DM applications in the early ages of DM. Currently, Venusian image data set is a classical test data set for benchmarks, and it is available in UCI Machine Learning Repository (Frank and Asuncion 2010).

Another real-world exploratory clustering application is in the domain of public health. BIOMED (Biomedicine and Health Program affiliated with European Commission) initiated a European collaborative project about childhood leukemia across 17 countries in Europe, namely EUROCLUS (Alexander et al. 1998). EUROCLUS explored the relation between the residence locations and the cancer diagnosis. Besides, the spatial clustering results were analyzed in terms of environmental factors. The results facilitated to identify the environmental causes of childhood leukemia. The work triggered several research studies on cancer diagnosis field and biotechnology (Dockerty et al. 1999, Birch et al. 2000, Hoffman et al. 2007).

Clustering can also be used for simplification and data compression through generalizations. That is, instead of the entire data set, representatives of the clusters, can be used for scalability purposes. However, this brings about losing some details in the data set. For example, in marketing research, representatives of consumer clusters can be used when the size of the data set is too big or there exists too much detail.

Another use of clustering is visualization. Visualization provides a deeper understanding and insight. It is especially useful for pattern recognition, image processing and geographical information systems.

Clustering has been a natural habit of human beings all along. We use clustering to discriminate objects, organisms and even abstract things, such as culture, thoughts, and so on. In fact, clustering is a kind of learning tool using the characteristics of the things in a comparative manner. For example, a basic grouping example in animal life is birds and reptiles. They form two different clusters in terms of their movement characteristics. This kind of clustering also helps to specify the properties of these animals and make generalizations. Today, the increasing growth in the computer world provides automation of the clustering task and deployment of clustering to a multitude of applications such as biology, marketing, city planning, pattern recognition, health, social sciences, and so on.

Clustering Approaches

Clustering is a hard problem due to its challenging characteristics such as arbitrary shaped clusters, multidimensionality, unknown number of clusters, scalability, mixed data types, and clustering evaluation function. As a natural result of this, clustering has emerged from the combination of many fields, i.e. statistics, pattern recognition, management information systems, optimization, and artificial intelligence.

Research on clustering has its roots in 1960s (MacQueen 1967), and it is still one of the most popular topics in the DM field. In a macro perspective, the clustering approaches in the literature can be classified as hierarchical, partitional, probabilistic, density-based, graph-based, metaheuristics, fuzzy clustering algorithms and artificial neural networks.

Every clustering approach defines the clustering problem from a different perspective. For example, partitional-based and hierarchical approaches and most of the metaheuristics define clustering as the division of a data set into disjoint subsets so that intracluster similarity and intercluster dissimilarity are maximized. Generally they do not take into account the connectivity i.e. the chaining of similarities through some form of transitivity or density relations among the data points i.e. intensity of the texture in certain regions explicitly. On the other hand, in density-based approaches a cluster is defined as a group of objects in a dense region surrounded by less dense regions. Graph-based approaches are built upon connectivity idea in graphs and a cluster corresponds to a set of connected components. Probabilistic approaches assume that clusters originate from certain probability distributions and try to estimate the associative parameters.

Another distinction is crisp versus fuzzy clustering. That is, each point is assigned to exactly one cluster in crisp clustering approaches whereas fuzzy clustering explores the degree of membership of a point in each cluster. The above clustering approaches are in general applicable to both crisp and fuzzy clustering. In addition to these, some clustering algorithms are developed specific to fuzzy clustering. The important discriminative property between probabilities in probabilistic clustering and fuzziness in fuzzy clustering is that the probability of assigning a data point to a cluster is related to a chance event whereas fuzziness represents the indifference of the cluster membership of a data point to an extent.

In order to apply clustering properly, the clustering goal, the application area and the area specific clustering requirements need to be clarified explicitly. Then, the clustering requirements are matched with capabilities and assumptions of the clustering approaches to select the appropriate clustering approach.

Spatial Clustering

In this dissertation, we focus on clustering problems typically exemplified in spatial data sets. In a spatial data set location of the objects and their geometric relations are of significance. That is, topology, proximity and connectivity become the key issues in clustering. Spatial data sets are particularly seen in geographical information systems, point based graphics and biomedical systems. Consider the city planning application we have discussed above. In order to understand the cultural heritage of a city, regions that include similar building types are formed. Thus, each region is composed of adjacent buildings having similar properties such as area and height.

An application from biomedical field is processing and classifying the image data from the medical imaging devices such as PET and MR. For example, in PET, the image segmentation is applied so that clusters represent the locations in the images in which tissue time-activity curves have similarities in terms of shape and magnitude. Thus, different types of tissue, bone, and blood can be identified for diagnostic purposes.

The great leap in computer world also necessitates clustering task in the point based graphics. In point based graphics a set of points is supplied via scanning and the aim is to extract the appearance of complex real-world objects in computer aided manufacturing, visualization of point clouds, and virtual reality applications.

As a real life example, European Topic Center on Air and Climate Change study on the air quality and greenhouse gas monitoring analysis and related legislation can be given (Nosovskiy et al. 2008). For the air quality database, representative stations throughout Europe are selected so that all the stations are covered and the relationships among the stations are discovered. In Figure 1.1 all the stations used for data collection are shown. Some of the stations are crowded in certain regions, e.g. in Germany, whereas some other stations are distributed in some regions in a sparse manner, e.g. in Portugal. Hence, there exist density variations in the distribution of stations across Europe.

Another real-life spatial clustering example is provided for the automated cancer diagnosis in the breast tissues (Bilgin et al. 2007). The digital images of the pathological tissue samples extracted in the biopsy are segmented (clustered) in graph theoretical context. An example breast tissue that is taken from the archive of Mount Sinai School of Medicine Pathology Department in New York is depicted in Figure 1.2. The tissue image includes arbitrary shaped patterns.

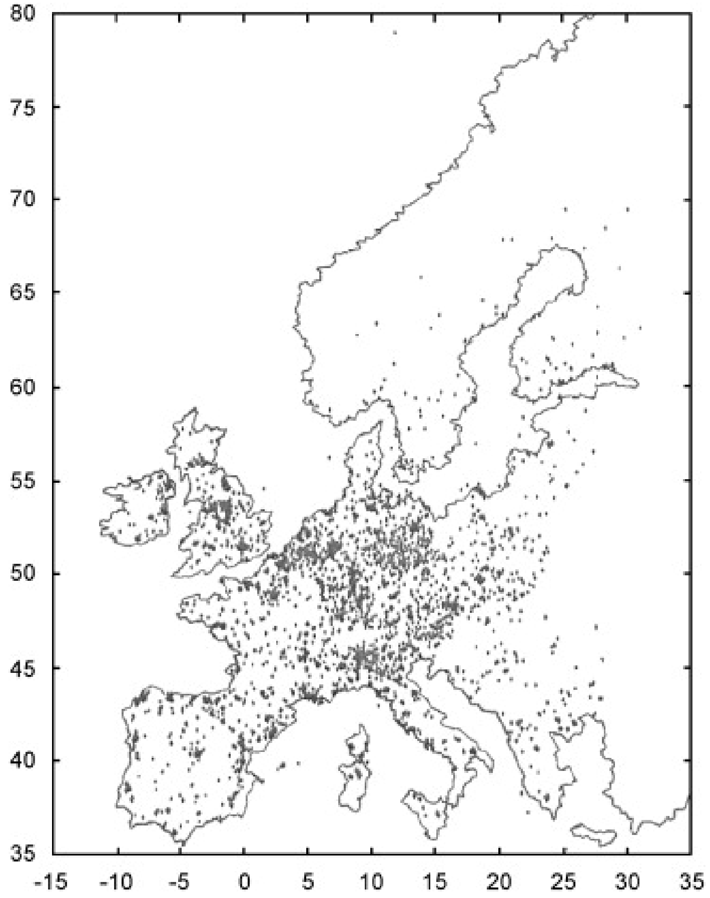


Figure 1.1 Station locations in Europe (Nosovskiy et al. 2008)

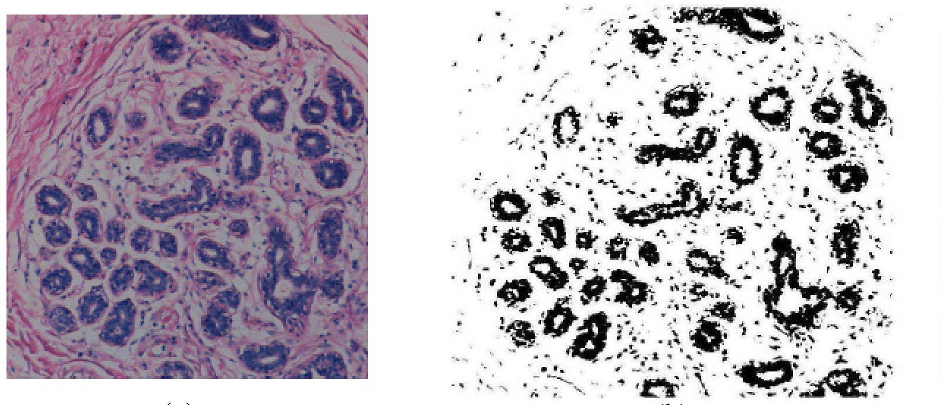

(a) (b)

Figure 1.2 Breast cancer tissue (a) Original breast tissue. (b) The result of clustering. (Bilgin et al. 2007)

In all these applications a wide variety of cluster characteristics particularly arbitrary shapes, density variations, and unknown number of clusters, are observed. In addition to the cluster extraction, it is sometimes useful to determine the cluster boundaries for visualization purposes. For example, the sexual offenses in year 1997 around Queensland in Australia given in Figure 1.3 (a) are clustered, and the boundaries of crime regions are extracted to understand the potential associations between the regions (Lee and Estivill-Castro 2006). In Figure 1.3 (b) the large cluster corresponds to a region where colleges and universities are located. The other crime regions (clusters) match an urban area where most of the streets, railways and highways run across, and a tourist attraction region near the coastline. The circular cluster with a hole shows a suburban area.

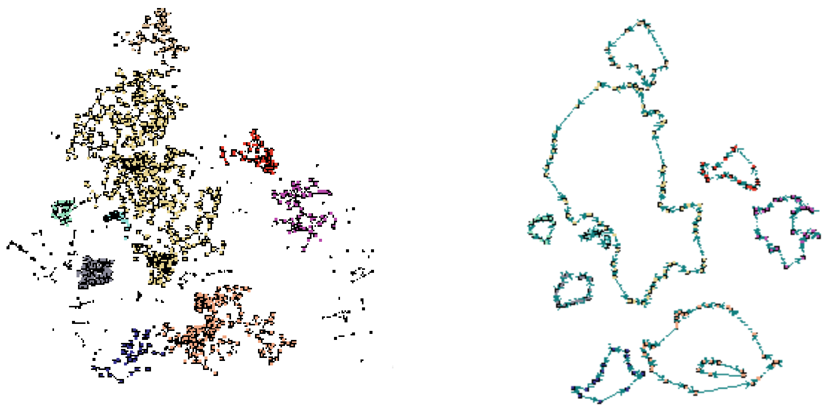


Figure 1.3 Real data set for sexual offenses around Queensland in Australia. (a) Real data set. (b) Cluster boundaries. (Lee and Estivill-Castro 2006)

Motivation and Scope

In this dissertation, we consider the clustering problem having the following characteristics.

(1) The number of clusters is unknown.

(2) Points and clusters are characterized (located, populated, separated) spatially.

(3) The relatedness (similarity or dissimilarity) between pairs of data points is measured by the Euclidean distance.

(4) A cluster is composed of connected data points in a dense region that is surrounded by low density regions or the vice versa, sparse area circumscribed by dense regions.

(5) Clusters may have arbitrary shapes.

(6) There may be density variations within the clusters as long as it follows a rough pattern (i.e. consistency along a direction).

(7) Different clusters may have distinct densities.

In order to handle a clustering problem with these specifications, the key issues become connectivity, density and proximity. Partitional algorithms have difficulty in finding the clusters with arbitrary shapes and density variations, so we particularly focus on density-based and graph-based clustering approaches. However, the performance of these clustering approaches is notably affected from the parameter settings, and clusters with density variations may not be extracted. Moreover, the determination of the proper parameters needs additional effort and expert knowledge.

There exist parameter-free approaches in the literature. Among them, AMOEBA (Estivill-Castro and Lee 2000) and AUTOCLUST (Estivill-Castro, V. and Lee, I., 2001) are graph-based methods that classify the edges of Delaunay triangulation according to their statistical properties, i.e. mean and standard deviation of edge lengths. This classification helps to identify the local and global properties inherent in a data set. Different from AMOEBA, AUTOCLUST can identify the clusters connected by bridges. However, neither of them can detect the clusters having intracluster density variations. ASCDT (Deng et al. 2011) is an extension of AUTOCLUST, and it integrates the direction information with the statistical properties of the edges in the Delaunay triangulation. Although it uses parameters during execution, it is able to extract the clusters with intracluster density variations. These approaches take into account proximity and distance using Delaunay triangulation, but they barely take advantage of the two important concepts in spatial clustering, namely density and connectivity. Moreover, typically, these algorithms are designed for 2-dimensional data sets.

In this dissertation, we aim to develop parameter-free clustering approaches for 2- and higher dimensional data sets, using the main concepts in spatial clustering in their entirety, i.e. connectivity, density, proximity, and distance.

Although metaheuristic applications to clustering are mostly limited to combinatorial approaches, the flexibility in the design of metaheuristics makes them a promising tool for spatial data sets. Particularly, swarm intelligence (SI) is a newly emerging field among metaheuristics, and it has an analogy with the clustering problem in spatial data sets. That is, a swarm is composed of simple agents having in some well-defined space, and it can perform complex tasks through sharing information and experience among the swarm members and with their space (i.e. environment). For example, consider an ant colony. The communication among the ants is ensured by the pheromone substance released by the ants. The swarm is directed towards the food sources or shelter attraction regions (both are spatial features) with the help of this substance. In the spatial data sets, density and connectivity form attraction regions to join the relevant data points, and such attracted data points form the cores of the clusters. There are (may) multiple cores, multiple swarms moving in different ways. Thus, agents can collectively discover the attractive regions, i.e. potential core regions of clusters through sharing density and connectivity information. This is the main motivation that makes us inquire the applicability of SI to the clustering problem in this dissertation.

There exist swarm intelligence applications to the clustering problem in the literature. Grosan et al. (2006) and Abraham et al. (2008) compile the previous work and introduce comprehensive reviews. The SI applications to clustering differ according to the properties of the clustering problem under consideration and the specifics of the SI application. Most of the SI applications address the clustering problem as a combinatorial optimization problem and search for a clustering solution that minimizes the total within cluster variation or distance.

In order to handle clusters with arbitrary shapes and density variations, density and connectivity issues should be integrated in SI. The difference among individual members is in their random interpretation of a collective information. The swarm intelligence integrates the information (attractiveness of linking data) gathered by individual swarm members. However, the SI based work in this context is limited, and most of these do not use the full benefits of a swarm. That is, the guidance of the swarm is not provided by the emergent knowledge gathered from the interaction among the swarm members and with the environment (i.e. collectivity). Instead, every swarm member moves individually. Besides, the performance of these approaches depends highly on the parameter settings. We primarily propose a new SI based clustering methodology for the spatial data sets, which takes advantage of the collective choices (to reflect interdependencies) with as few parameters as possible.

The outline of the dissertation is as follows.

Chapter 2 explains the general clustering problem and discusses the challenging issues in clustering. A vast amount of work about clustering has accumulated up to now due to its broad application areas. In this chapter, we classify the clustering algorithms in the literature. We explain the clustering goals, the perspectives and the underlying assumptions of each approach briefly, and discuss strengths, limitations and complications of these approaches.

Neighborhood concept is crucial to identify the local properties in a clustering problem, and a purified neighborhood, which is composed of only the "similar" points, needs to be constructed. Particularly SI needs a neighborhood definition in order to construct a solution and exploit the search space in spatial data sets. Thus, in Chapter 3 we propose a parameter free neighborhood construction algorithm for data sets having clusters with arbitrary shapes, intracluster and intercluster density variations. Solely distance and density information may not be sufficient to construct purified neighborhoods. Hence, the proposed algorithm considers connectivity, proximity and density information extracted from one of the well-known proximity graphs, namely the Gabriel Graph.

There are two outputs of the neighborhood construction algorithm: a unique neighborhood for each point and subclusters (closures) formed by the union of the neighborhoods having common points. The capabilities of the proposed approach are tested with various data sets. The comparison with other neighborhood based algorithms, namely distance and density based approaches, indicates the strengths and the limitations of the proposed algorithm.

In Chapter 4 a three-phase clustering algorithm for spatial data sets is introduced. It works based on locality, connectivity and density information. The first phase extracts the local properties using the neighborhood construction algorithm described in Chapter 3. Subclusters (closures) that are formed at the end of the neighborhood construction algorithm are refined in the two subsequent phases. The second phase is dedicated to outlier detection, and the third phase performs hierarchical agglomeration by merging the subclusters. In the second phase the key issue is the consistency of a point with its neighborhood, and a point is classified as an outlier if it is different from the points in its neighborhood in terms of either distance or density. The first two phases only focus on the local properties, so the third phase tries to bring in the global perspective inherent in the clustering problem. Hence, the improvement in the compactness and the separation values that are calculated relative to the neighborhoods are checked for merging. The proposed algorithm is tested empirically and it is compared with the well-known competing approaches, i.e. agglomerative, partitional and density based clustering algorithms.

Given a finite set of data points and a partitioning of the data set, Chapter 5 concentrates on forming the boundaries of the groupings. In fact, boundary formation corresponds to external shape generation in computational geometry. We adapt this concept to clustering for scalability and visualization purposes as spatial data allows for boundary definition. Using the insights gained from the local perspective, i.e. the neighborhood construction algorithm, we can infer that the interior points of the closures are already connected. Therefore, it is sufficient to consider only the boundary points for outlier detection and merging purposes. This results in removal of the interior points from further consideration, and hence reduction of the data set. The second function of our boundary formation is visualization of cluster boundaries. This capability has use in many point based graphics applications such as computer aided design and manufacturing. As we deal with clusters with arbitrary shapes, the external shape of a cluster may be non-convex. Thus, our concern becomes generation of the non-convex hull of a set of points.

We propose two external shape generation algorithms based on Delaunay triangulation in Chapter 5. The former one is restricted to 2-dimensional space, whereas the latter is designed to work in higher dimensional spaces as well. For both algorithms the accuracy and the amount of data set reduction are analyzed thoroughly in numerical experiments.

After forming an appropriate background for SI in Chapters 3 and 5, Chapter 6 is dedicated to the SI based clustering algorithm. First, we study the design steps of a SI based algorithm. In the design stage agent representation is one of the key issues in determining the capabilities of a SI algorithm, so we propose a classification for the SI based clustering algorithms based on the agent representation. We review the advantages and the shortcomings for each agent representation.

In natural life ants live in colonies, and they have a natural tendency of clustering for the food search, brood feeding and corpse sorting activities. Ant Colony Optimization (ACO) is chosen for the clustering problem among SI approaches due to this analogy between an ant colony and clustering.

The inputs of the proposed ACO based clustering are the neighborhoods and the closures from Chapter 3, and the external shapes of the subclusters (closures) from Chapter 5. Hence, locality and scalability issues are taken into account in ACO. Ants move in the solution space and insert edges between the data points. In so doing they propose such data points to be in the same cluster. Ant representation allows finding the initially unknown number of clusters in the data set, as well as the clusters with arbitrary shapes and density variations.

ACO handles the clustering problem by an improvement based optimization approach. Experience gained from evaluation of a clustering solution guide the ant colony to the attractive regions. This implies that target clustering is the optimal solution. However, there is not a generally accepted evaluation function that favors the "target" number of clusters in the data set. The evaluation becomes particularly more complicated when a data set includes clusters with arbitrary shapes and density variations. Thus, we propose a new clustering evaluation scheme in which compactness and separation are measured relative to the neighborhoods. When these two measures are combined in a single measure via mathematical operators such as summation, division, and so on, the trade-offs between compactness and separation may not be observed. Therefore while comparing the performance of clustering solutions, compactness and separation measures are evaluated in a bicriteria manner. The performance of ACO applied to clustering is tested and compared using various data sets, and its capabilities are explored.

Our clustering methodology can be applicable to real-life spatial data sets that require minimum amount of a priori domain knowledge. Particularly, the methodology can yield effective results in the clustering problems in which accuracy is the main concern. For example, in medical image segmentation and satellite imagery data sets, the aim is to identify the homogeneous and connected regions that have similar density and texture properties. The segmentation of the medical image data sets ensures the delineation of anatomical structures and other regions of interest, diagnosis and localization of pathology, so the accuracy of the clustering results is vital. The satellite imagery data provides to understand up-to-date information in a global manner in several applications such as land use and rainfall patterns, epidemic and traffic network.

Contributions

The main contributions of this dissertation are as follows.

(1) A novel parameter free neighborhood construction algorithm is proposed for the clustering problem. Connectivity, density and proximity are the key issues for the proposed approach. In fact, it is possible to use the proposed approach in classification and clustering validity index calculation purposes.

(2) A new clustering algorithm in graph theoretic context is developed for spatial data sets. We combine ideas inherent in the density and the graph based clustering approaches. Local properties from the neighborhood are used in cluster formation and outlier detection.

(3) The external shape concept in computational geometry is adapted to clustering for data reduction and cluster visualization purposes. A novel algorithm to extract the external shape of a non-convex multi-dimensional data set is proposed. The proposed algorithm can be used in computational geometry as well.

(4) We develop a new ACO based clustering algorithm to examine applicability of Swarm Intelligence. The proposed algorithm is entirely inspired by Swarm Intelligence. A new agent representation, which is capable of finding arbitrary shaped clusters with density variations and the unknown number of clusters, and a new pheromone update mechanism for bicriteria are introduced.

(5) In compactness and separation calculations, it is proposed that edges in a clustering solution should be evaluated with respect to the neighborhoods of its end points so that clustering solution that includes arbitrary shaped clusters and density variations is evaluated by local properties and more effectively.

(6) Use of parameters is avoided in the proposed approaches. The need for parameters is substituted by problem specific properties gathered through computations.

(7) In the overall, the dissertation proposes a clustering methodology which is composed of three major methods: extraction of local density, proximity and connectivity properties, visualization and data set reduction, and collective information based intelligence.

CHAPTER 2

PROBLEM DEFINITION AND LITERATURE REVIEW

In this chapter we introduce the clustering problem and briefly review the clustering procedures. We then discuss the characteristics and classes of the clustering problem and inquire why clustering is a challenging problem. In the clustering literature there are distinct points of view and assumptions. Their properties and capabilities are reviewed in this chapter. The motivation for this work will be substantiated.

2.1. The Clustering Problem

A data point represents an observation in a system, and it is a d-dimensional vector in which each dimension represents a physical or an abstract attribute of the system, $\mathrm{x} = (x_1,..,x_j,..,x_d)$. An attribute can be either qualitative (i.e. nominal or ordinal) or quantitative (i.e. continuous, discrete or interval). Typically proximity/similarity between a pair of data points is measured by a distance function (e.g. Manhattan, Euclidean, Mahalanobis, Minkowski) for quantitative attributes whereas similarity measures (e.g. simple matching coefficient, Jaccard coefficient) are used for qualitative attributes. The details of these measures are available in Xu and Wunsch (2005).

Technically speaking, given that a data set D which is composed of n data points with d attributes, $\mathrm{D} = \{\mathrm{x}_1,..,\mathrm{x}_i,..,\mathrm{x}_n\}$, clustering forms c subsets (clusters) in D so that data points in the same cluster are similar/close to each other, and data points in different cluster are dissimilar/far away. Conceptually cluster analysis involves grouping data points based on similarity/proximity information extracted from the data set.

Definition of a cluster is vague, and there may not be a unique clustering in a data set due to properties of the data set, the application field and the goal of the clustering. An example data set with 14 data points and 2 attributes is provided in Figure 2.1 (a). In the figure each ball represents a data point, and x- and y-axes denote the two attributes. In the macro perspective one can form two clusters as in Figure 2.1 (b), whereas it is possible to cluster the data points into three and four as in Figures 2.1 (c) and (d).

A wide variety of application areas, including pattern recognition, marketing, biology, geology and web analysis, make use of clustering for exploratory purposes. For example, using loyalty card transactions, a retailer can form consumer profiles (clusters) that include consumers having similar interests, income level and habits. Thus, the retailer can determine effective sales strategies based on these consumer profiles, and increase its profit. Another example is web search engines. Document clustering helps to browse and reach to web pages related to a query in a fast and effective manner. In biology it is assumed that genes and proteins with similar functions have sequences and gene expression data in common. Hence, instead of making expensive and time-consuming experiments, gene clustering explores the roles of the genes in human genome sequence by forming clusters of similar genes. In geology, oil companies use geological features of the oil exploration sites obtained during the drilling process to discover promising reserves for oil exploration and production. In this case, clustering helps to identify homogeneous sites showing similar geological features. This systematic analysis increases the probability of success in oil drilling process.

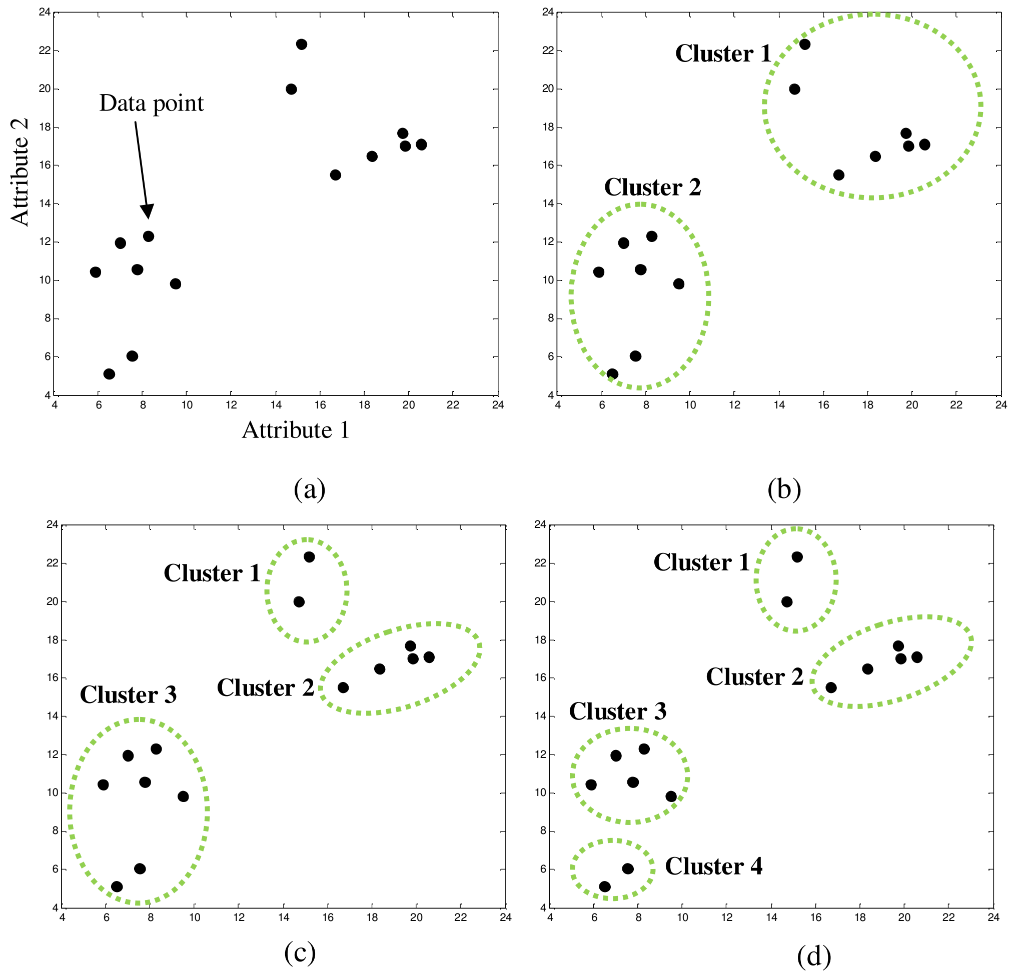


Figure 2.1 A clustering example. (a) The data set. (b) The clustering result with two clusters. (c) The clustering result with three clusters. (d) The clustering result with four clusters.

The second goal implies that the data set is represented by relatively fewer clusters. Although simplification causes loss of details, simplicity is still sought. For example, the vast of amount of consumer data in market segmentation can be simplified and reduced through using the representatives of the consumer clusters.

The third goal ensures visualization of clusters so that analysts identify the characteristics of groups. For example, visualization of the clustering results for gene expression data provides insight to understand the biologically meaningful genes for cancer diagnosis.

The clustering procedures are in general composed of the steps such as: attribute selection and extraction, clustering the data (including definition of similarity/dissimilarity measure and clustering objective), validation, and interpretation of results. Attribute selection determines the most effective attributes whereas attribute extraction makes transformations on the attributes in order to improve the performance of clustering analysis. For example, intrusion detection systems examine network traffic using a large number of attributes such as protocol type, duration, service etc. Attribute selection becomes an important step to determine clusters with malicious activities in a fast and correct manner. Next, taking into account the domain and characteristics of the data set, an appropriate similarity/dissimilarity measure is defined and the clustering objective is determined. There are quite a number of clustering algorithms in the literature. Given a data set either one of them is adapted for the data set, or a new clustering method is developed considering the characteristics of the data set and the purpose.

A partitioning of the data set is obtained as a result of any clustering algorithm. Thus, validation is necessary to show that resulting clusters are not obtained by chance, or they are not obtained incidentally by the clustering algorithm. Clustering validation indices, i.e. internal, external and relative validity indices, or expert knowledge are considered for validation purposes. Although expert knowledge is a subjective evaluation scheme, validity indices quantify the clustering performance in a so-called objective manner. The last but not the least, domain experts interpret clustering results so that meaningful insights are gathered from this knowledge discovery process through clustering.

2.2. Characteristics of the Clustering Problem

The general clustering problem is NP-hard (Garey and Johnson 1979). Given a data set D with n data points and a given number of clusters, c, there are $\frac{1}{c!}\sum_{i=0}^{c}(-1)^{c-i}\binom{c}{i}i^n$ alternative clustering solutions (Anderberg 1973). It is not possible to enumerate all possible solutions even for a small-sized data set. In addition to the vast amount of clustering solutions, the nature of the clustering problem brings about some challenging issues such as unknown number of clusters, handling arbitrary shaped clusters, density variations, mixed data types, clustering objectives, additional constraints, and scalability.

Clustering may be classified as an unsupervised learning scheme, so the number of clusters in the data set is not known a priori generally. Besides, a data set may have various patterns such as arbitrary shaped clusters and density variations, and these patterns complicate both calculation and optimization of compactness, separation and connectivity objectives. Attributes of a data set may include various types of data such as numerical, ordinal, nominal, and this affects the calculation of similarity/dissimilarity calculations between pairs of data points. Conceptually clustering aims to obtain compact, well-separated and connected clusters. However, it is difficult to quantify and combine these objectives, and come up with a scalar to be optimized so that target clusters are achieved ultimately. In some clustering applications domain specific constraints some of which contradicts with the natural clustering tendency of the data set are forced. The need of clustering in large scale data sets brings about some problems such as long processing times, large memory requirements, etc. These challenging issues are discussed in detail as follows.

Clustering Objectives and Density Variations

Clustering tries to partition the data set into groups so that points in the same cluster are similar (i.e. compactness) whereas points in different clusters are dissimilar (i.e. separation), and, at the same time, a point and its close neighbors are assigned to the same cluster (i.e. connectivity). A clustering algorithm should take into account these three objectives simultaneously. Although it is possible to conceptualize the objective of the clustering problem, it is difficult to come up with measures to quantify and combine these objectives for the data sets in various fields concurrently in all these aspects.

Throughout the text the terms “similarity/dissimilarity measure” and “distance” are used interchangeably. This is mainly because our focus is on spatial data that is in pure numerical form.

Compactness is the consistency of data points in a cluster in terms of similarity and density. The compactness objective ensures intracluster similarity, that is, the points in the same cluster should be similar/close to each other. Compactness measures in the literature can be mainly classified into two: representative point based and edge based measures. As the name implies, representative point based ones focus on minimization of the total distance/dissimilarity measure between cluster members and a point that represents the cluster (center, medoid, etc.). In these cases, generally, the number of representative points identical to the number of clusters should be set a priori. Clustering methods that simply optimize an objective based on such a measure are limited with spherical shapes, that is, elongated or spiral shaped clusters can hardly be obtained by these algorithms. (Ester et al. 1996, Guha et al. 1998)

The edge based methods consider the intracluster distances/dissimilarities such as the sum of distances between pairs of data points in a cluster or the maximum edge length in a connected graph of the cluster. They are more powerful than representative point based methods in handling arbitrary shapes. However, they do not succeed in dealing with intracluster variations in density which is defined as the number of data points within unit volume. (Ester et al. 1996, Liu et al. 2008)

Separation is the intervening space or the occurrence of a density change between the clusters. Thus, the separation objective provides the intercluster dissimilarity. That is, the data points in different clusters should be dissimilar. Linkage metrics (i.e. single-link, average-link, and complete-link), total intercluster distance variation and distance between cluster representatives are some common separation measures in the literature. Linkage metrics are dissimilarity measures for clusters. Single-link calculates the dissimilarity (distance) between a pair of clusters as the closest (most similar) points between them whereas complete-link takes into account the most distant (most dissimilar) points between a pair of clusters. In average-link the average distance between pairs of points in two clusters is used.

The evaluation of both compactness and separation in an absolute manner is not sufficient in the data sets having density variations. An example is seen in Figure 2.2. The distance between any two points in the less dense upper region of cluster 1 is larger than the minimum distance between two points in clusters 1 and 2.

Optimization of an objective based on absolute distances cannot result in the target clusters seen in Figure 2.2. Either clusters 1 and 2 are merged or cluster 1 is divided into more than one clusters.

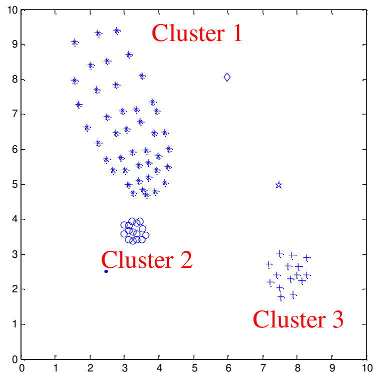


Figure 2.2 An example data set with intercluster density difference and intracluster density variation

Connectivity is the linkage between the data points in close proximity and having similarities, and data points in a certain vicinity of a given data point constitute the neighbors of the associated point. Hence, the connectivity measure focuses on the proximity relations among the data points so that a point and its close neighbors are assigned to the same cluster. For example, Handl and Knowles (2007) calculate the connectivity objective as the degree of the neighboring data points placed in the same cluster for *k*-nearest neighbors.

The neighborhood definition is crucial in finding the target clusters with this objective. Neighborhood of a point should only include points from the same target cluster (a pure neighborhood). Otherwise, target clusters cannot be found by optimization of the connectivity objective. To the best of our knowledge, there is not a generally accepted method to define a pure neighborhood. Consequently, use of connectivity objective has also complications.

To sum up, a generic clustering objective or measure that fits every data set is not available in the literature. The data set characteristics (types of attributes, properties of clusters, etc.) and the application field affect the determination of the objectives to be used in clustering and the measures for these objectives. In addition, there exist trade-offs among clustering objectives. For example, as the compactness objective improves (or stays the same) when the number of clusters in the data set increases, the separation and connectivity objectives worsen (or do not improve). The integration of the objectives via basic arithmetic operations such as summation, division or multiplication undermines the details inherent in each objective and causes information loss. Thus, it is not possible to obtain a high-quality clustering solution just by optimizing one of these objectives or a combination of them. As a remedy to this problem, multiobjective clustering accounts for the information in each objective and ensures analysis of the trade-offs between objectives.

Arbitrary Shaped Clusters

Data sets can be composed of clusters of any size, density and shape such as ellipsoids, elongated structures, and concentric shapes. Figure 2.3 shows some examples of clusters with arbitrary shapes. Particularly, arbitrary shaped clusters are seen in geospatial data sets in geographical information systems, geology and earth sciences, and image segmentation (Ester et al. 1996, Jain et al. 1999). Extraction of arbitrary shaped clusters depends on both the clustering objective and the similarity/distance function used. For example, it is not possible to discover cluster 1 in Figure 2.2 (an elongated cluster with intracluster density variation) using either a representative point based or an edge based compactness measure. Proximity relations and connectivity issues become vital to discover the arbitrary shapes in addition to the compactness and separation objectives.

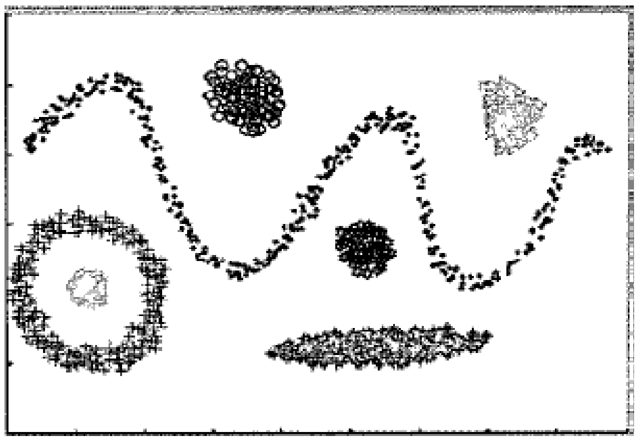


Figure 2.3 Examples for arbitrary shaped clusters (He and Chen 2003)

The Number of Clusters

Clustering may be classified as unsupervised learning because there are no given class labels as in classification. In addition the number of clusters may also be unknown. The difficulty comes especially from the points distributed in a heterogeneous manner in the data set. This is due to the inability of applying one general function (or measure) across all the data set.

Some of the algorithms in the literature assume a fixed number of clusters given a priori like in facility location and market segmentation. For example, the number of facilities (clusters) to be built is determined in the strategic plans, and one needs to determine the location of the facilities and the assignment of customers to facilities in the tactical level. However, many real world clustering tasks come without this prior knowledge like in image segmentation, bioinformatics (e.g. gene clustering) and geographic information systems. Instead, one should extract it from the data set as a major piece of knowledge. Some approaches use a validity index to find the number of clusters. They run the clustering algorithm with several values for the number of clusters, and select the one with the "best" validity index. Here, "best" usually denotes a knee point for the validity index. However, determination of this knee point is a subjective decision, and there might be ambiguous cases.

Data Types

Data sets might include two types of attributes: qualitative (i.e. binary, nominal or ordinal) or quantitative (i.e. continuous, discrete or interval). This variety particularly affects the calculation of the similarity/dissimilarity measure. This measure quantifies the degree of similarity/dissimilarity between two data points in terms of their attribute values.

Dissimilarity is measured by distance functions such as Manhattan, Euclidean, Mahalanobis, and Minkowski distances, when all attributes are quantitative (e.g. height, weight). In order to calculate the similarities in qualitative data (e.g. color, rank), some well-known similarity measures are simple matching coefficient, Jaccard and Hamming distances. If there exist both quantitative and qualitative attributes in a data set, i.e. mixed case, similarities/dissimilarities among the attributes are not comparable. As a remedy, the general similarity coefficient by Gower (1971) and the generalized Minkowski distance by Ichino and Yaguchi (1994) are used in mixed data sets. (Han and Kamber 2001, Jain et al. 1999, Xu and Wunsch 2005)

The contribution of each attribute to the computation of the dissimilarity measure is affected from the range of its values. In order to ensure comparability between the attributes, attributes are normalized by various scaling methods such as min-max, z-score and decimal normalization. This scaling becomes problematic when the data set include mixed types of attributes, as binary data usually dominate the numerical data (Jain et al. 1999). Hence, attribute types and dissimilarity measure used have important effects on the development of a clustering algorithm.

Constraints

Constrained clustering is regarded as semi-supervised learning in which additional conditions are satisfied while grouping similar points into clusters. Constraints can be interpreted as a priori domain knowledge, so they improve the clustering accuracy and efficiency. However, additional mechanisms are needed in a clustering algorithm to handle the constraints.

Real world clustering problems have various types of constraints. Constraints are often seen in the practical cases where the decision maker (expert) has some domain knowledge. Constraints can be classified into three categories: cluster-level, attribute-level and instance-level constraints (Basu and Davidson 2011).

a) Cluster-level constraints

- Capacity constraint
- δ-constraint (minimum cluster separation)
- ε-constraint (ε-distance neighbor within cluster)
- Maximum/minimum cluster diameter

b) Attribute-level constraints

- Attribute order preferences
- Constraints on attribute values

c) Instance-level constraints (relational)

- Partial labels
- User feedback
- Pairwise relationships
    i) Must-link constraint
    ii) Cannot-link constraint
    iii) Entailed instance-level constraint

Among cluster level constraints, capacity constraint limits the size of a cluster. δ-constraint implies that the distance between points in any two clusters must be at least δ. ε-constraint enforces that each point in a cluster must have another point in the same cluster such that the distance between these two points is smaller than ε. Maximum (minimum) cluster diameter defines a threshold for the maximum (minimum) distance between points in the same cluster.

Attribute-level constraints guide the clustering assignments based on the values of an attribute, e.g. in clustering census data, each cluster is expected to include individuals having with the same gender in order to extract the gender distinctions (Wagstaff 2002). Among instance-level constraints, partial labels denote that some points are necessarily assigned to certain clusters. User feedback implies interaction with the decision maker, and this feedback directs the clustering algorithm to the preferred clustering patterns. Pairwise relationships include must-link (ML) constraints where two instances must be in the same cluster, cannot-link (CL) constraints where two instances must not be in the same cluster, and entailed instance level constraints which represent the transitive properties between ML and CL constraints.

Scalability

Improvements in the data storage field enable to work with large amounts of data sets, hence analysis of these data becomes crucial and inevitable. As the number of data points and the number of attributes increase, data analysis consumes more memory and computing time. For example, traditional hierarchical clustering algorithms have a time complexity of O(n^2), and their memory requirements are O(n^2). This quadratic structure complicates clustering for very large-scale data sets. Although k-means, a well-known partitional algorithm, has approximately linear time complexity, O(ncd), it has disadvantages like sensitivity to outliers and initialization of parameters, shapes of clusters. In order to overcome these, random sampling (e.g. CURE by Guha et al. (1998)), randomized search (e.g. CLARANS by Ng and Han (2002)), summarization of the original data set (e.g. BIRCH by Zhang et al. (1996)), and parallelization (e.g. Dahlhaus (2000)) are applied.

Clustering algorithms that are capable of handling large data sets are necessary and efficient scalable mechanisms need to be developed.

2.3. Clustering Algorithms in the Literature

The wide applicability of the clustering problem brings about a vast variety of clustering algorithms in the literature. This variety is the consequence of different points of view and assumptions in solving the clustering problem. In this section we review the ideas behind the clustering algorithms and summarize their capabilities.

Recent comprehensive reviews for the clustering algorithms are provided by Xu and Wunsch (2005), Berkhin (2006) and Jain et al. (1999). They give a perspective on the state-of-the-art clustering approaches. In addition, they discuss the strengths and weaknesses of the existing clustering algorithms.

In fact, there is not a clear cut classification for the clustering algorithms and overlaps between classes exist. In this work, taking into account the main perspectives, we group the clustering algorithms into hierarchical, partitional, probabilistic, density-based, graph-based, metaheuristic, and fuzzy clustering and artificial neural network algorithms. In this section we first review each of these classes briefly. We then focus on the swarm intelligence algorithms as they are the main concern in this work.

2.3.1. Hierarchical Clustering Algorithms

A tree-like cluster hierarchy, namely dendrogram, is built in hierarchical clustering algorithms. These algorithms can be categorized into two according to their hierarchical structuring: (Han and Kamber 2001)

- *Agglomerative (bottom-up):* It assigns each point to a singleton cluster. Then, clusters are successively merged according to a criterion until a single cluster including all the points is formed. Single-link, average-link and complete-link are the criteria commonly used for merging operations.
- *Divisive (top-down):* It moves in just the opposite direction of the agglomerative clustering. It starts with a single cluster including all the points, and a cluster is split in each step successively until each point is assigned to a singleton cluster.

Well-known algorithms in this area are BIRCH (Zhang et al. 1996), CURE (Guha et al. 1998) and ROCK (Guha et al. 2000). Hierarchical methods are applicable to several data types and similarity (distance) functions. Besides, the number of clusters does not need to be given a priori. The resulting dendrogram provides a visual representation for the hierarchy of clusters. However, extra mechanisms are required in order to make a decision about the number of clusters. Furthermore, hierarchical approaches are greedy and the merging/splitting decisions could not be changed throughout the iterations. Thus, they might come up with suboptimal solutions. (Xu and Wunsch 2005)

2.3.2. Partitional Clustering Algorithms

Partitional clustering algorithms consider the clustering problem as a combinatorial optimization problem. They directly decompose the data set into a set of disjoint clusters optimizing an objective function (e.g. the total distance/variation among the data points and the cluster representative points). Most of the previous work assume that the number of clusters is given a priori. The major advantage with the partitional clustering algorithms is their scalability. That is, they are capable of generating clusters in large data sets in reasonable times. The shapes of the resulting clusters depend on the objective function. Minimization of the total distance among the data points and the cluster representative points is the most commonly used objective function. However, this objective yields only spherical shaped clusters, thus it hardly identify arbitrary shaped clusters. The other deficiencies of the partitional clustering algorithms are the dependency of the performance on the initialization of representative points, the sensitivity to the outliers and the limited use for the data sets having numerical attributes. Well-known algorithms in this category are *k*-means (MacQueen 1967), and PAM (Ng and Han 1994).

2.3.3. Probabilistic Clustering Algorithms

Probabilistic clustering algorithms assume that clusters are drawn from certain distributions and their aim is to estimate the parameters of these distributions. The performance of these algorithms depends on the initialization of parameters and the estimation procedure might yield suboptimal solutions. Besides, they have strong assumptions regarding the distribution of the data, and their computational time can be quite high. Among them Expectation-Maximization (Mitchell 1997) is the most famous one. It estimates the distribution parameters using maximum likelihood estimation, and algorithm continues until likelihood convergence is achieved.

2.3.4. Density-based Clustering Algorithms

Density-based clustering algorithms require a metric space and they are based on density, boundary and connectivity concepts. They assume that a cluster is a connected and dense region, and it extends in the direction of density increase. Clusters are separated with regions having relatively lower density. This idea facilitates extraction of clusters with arbitrary shapes.

Density-based approaches form two main groups with respect to density calculation: density connectivity and density functions. The former group classifies the points as boundary or core points according to density, connectivity and proximity relations. DBSCAN (Ester et al. 1996), GDBSCAN (Sander et al. 1998) and OPTICS (Ankerst et al. 1999) are examples of this type of clustering algorithms. The latter group focuses on the calculation of some density functions over the attribute space. DENCLUE (Hinneburg and Kiem 1998) is a well-known algorithm based on the density function.

The main advantage of density-based clustering algorithms is their ability to find arbitrary shaped clusters. Besides they are capable of handling outliers and noise with reasonable scalability. Despite these strong capabilities, the performance of density-based approaches is affected by several parameters to be decided a priori, mainly the minimum number of points to define a neighborhood or the neighborhood size. Correct setting of these parameters is crucial and needs extra effort. Besides, the early works on these methods have limitations in extracting clusters with intracluster density variations. In addition, the interpretability of the clustering results is more difficult.

Grid-based approaches are a combination of density-based and hierarchical clustering algorithms. Like density-based algorithms, grid-based clustering also uses the density and connectivity ideas. However, density measurements are conducted in the grids obtained by space partitioning. Clusters are generated by using the density information in the grids in a hierarchical manner, and arbitrary shapes with density variations can be handled. Different from the density-based algorithms, they can handle any type of data. However, they are based on strong assumptions, and fine tuning of the parameters is needed. Some example algorithms with grid-based approaches are CLIQUE (Agrawal et al. 1998), WaveCluster (Sheikholeslami et al. 1998) and STING (Wang et al. 1997). Generally grid-based algorithms take advantage in large scale data sets due to their computational complexity. For example, WaveCluster has a time complexity of O(n).

2.3.5. Graph-based Clustering Methods

Connectivity issue in the clustering problem can be handled in a graph theoretic context. Data points represent the nodes, and the edges between nodes show the similarity/proximity relations between corresponding pairs of data points. Proximity graphs such as k-nearest neighbor (k-NN), minimum spanning tree (MST), and Delaunay triangulation (DT) are the main tools used to ensure the connectivity relations among the data points (Karypis et al. 1999, Zahn 1971, Estivill-Castro and Lee 2000). Note that hierarchical approaches have a direct relationship with the graph theory, that is, single-link and complete-link are equivalent to MST and maximal complete subgraph, respectively. In addition to these, the clustering problem can be formulated as finding the minimum cut, the maximum clique, and so on in a given graph. (Sharan and Shamir 2000, Ben-Dor et al. 1999).

Graph-based approaches are generally combined with other clustering algorithms such as hierarchical and density-based, but the main discriminative property of graph theoretic approach is its reference to connectivity. Main limitation of this approach is the time complexity regarding the construction of the graphs. For example, construction of proximity graphs such as k-NN and MST has a time complexity of O(n^2), and DT needs O(n log$n$) for $d \leq 3$ and O($n^{\lfloor d/2 \rfloor} / \lfloor d/2 \rfloor !$) for $d \geq$ 3.

2.3.6. Metaheuristics

Metaheuristic approaches view clustering as a combinatorial problem in which an objective function is to be optimized. In the traditional heuristics such as *k*-means there is the possibility of getting stuck at a local optimum. Stochastic search strengthens the explorative property of metaheuristics and help to reach the global optimum. In the literature there are several metaheuristic implementations for the clustering problem including the Evolutionary Algorithms (EA), Simulated Annealing (SA), Tabu Search (TS), and Swarm Intelligence (SI). A brief survey about metaheuristics by Rayward-Smith (2005) focuses on the direct applications of metaheuristics as well as hybrid algorithms.

EA, which is inspired from natural evolution, is widely used in clustering. The first EA study on clustering is proposed by Raghavan and Birchand (1979). Several representation schemes, crossover and mutation operators are introduced in the later work. Hybrid EA approaches with other clustering algorithms such as partitional (Babu and Murty 1993) and hierarchical (Lozano and Larranaga 1999) approaches are also proposed. As EA is a population based approach, it also has effective applications in multiobjective clustering (Handl and Knowles 2004a, Handl and Knowles 2007). A comprehensive literature review on application of EA to clustering is performed by Hruschka et al. (2009). Their work includes all EA applications on clustering covering multi-objective and cluster-ensemble clustering.

The major problem with EAs is their high computational cost. Besides, the performance is sensitive to the parameters such as population size, crossover and mutation probabilities. Although some guidelines for parameter selection are proposed, they are not effective in all problems.

SA, inspired from the annealing process in the metallurgical processes, is a global search method. Klein and Dubes (1989) are the first to apply SA to clustering, in which non-improving solutions are accepted in a stochastic manner. Selim and Al-Sultan (1991) study the effects of parameters on SA. Although Aarts and Korst (1989) statistically show that SA algorithms can reach the global optimal solution, the temperature decrease must be done very slowly, and this causes a computational burden for SA.

The main idea in TS is solution space restriction using a tabu list. Al-Sultan (1995) proposed a TS based clustering algorithm. Extra mechanisms such as packing/releasing and secondary tabu list are developed in order to increase the efficiency of TS (Sung and Jin 2000). However, tabu definitions and corresponding memory requirements limit the size of data sets to be solved.

A newly emerging type of metaheuristic is SI and its variants Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). They are inspired from the collective behavior of the swarm, and the analogy (stemming from collective nature to be explored) between SI and the clustering problem arises an interest in this field. The details of SI and its applications will be further discussed in Section 2.4.

A comparison of clustering metaheuristics is conducted by Trejos et al. (2006). They compared *k*-means, SA, TS, Genetic Algorithms (GA), PSO, and ACO for different types of numerical data sets. They state that these metaheuristics help to improve the results of traditional clustering methods in challenging data sets with numerical data. In addition, *k*-means has deficiencies in almost all data sets except the easiest ones. In particular, population-based metaheuristics, i.e. GA, ACO and PSO, perform well even for the challenging data sets.

To sum up, current metaheuristic applications have limited success in clustering. They yield global optimal clustering solutions for small clustering problems. However, they suffer from scalability problems. In addition, most of them take the fitness value as the minimization of the within cluster variance, and they use Euclidean distance as the dissimilarity measure. These choices result in spherical and compact clusters. Besides, they usually assume a fixed number of clusters. Another complication with the metaheuristic applications is the need for a priori parameter setting.

2.3.7. Fuzzy Clustering and Artificial Neural Networks

Clustering algorithms described up to now result in hard (crisp) clusters, that is, a data point is either assigned to a cluster or not, and clusters are disjoint. However, for the clustering problems in which cluster boundaries are not clear cut, the assignment of points to clusters is not definite. In such cases, fuzzy clustering which introduces a membership function to associate each point with a cluster can handle uncertainty in cluster assignments. Fuzzy versions of partitional, hierarchical and metaheuristics clustering approaches are proposed in the literature. Among them Fuzzy C-Means (FCM) (Bezdeck et al., 1984), which is the fuzzy version of classical *k*-means, is a renown approach due to its simplicity and effectiveness. Several variants of FCM (Dave 1992, Höppner et al. 1999, Eschrich et al. 2003) are developed in order to handle various data types.

Artificial neural network (ANN) has also been a legitimate approach in clustering. Well-known clustering algorithms based on ANN are self-organizing map (SOM) and adaptive resonance theory (ART) models. SOM (Kohonen 1990) maps high-dimensional data to 2-dimensional lattice using prototype vectors as cluster representatives. Thus, visualization of the clusters is ensured. ART is a learning theory based on resonance in neural circuits by adapting weights between attributes and clusters simultaneously so that speed and stability is ensured. Main limitations with ANN is the parameter setting, that is, the number of clusters (nodes in the output layer) and initial weight parameters are required a priori.

2.4. Swarm Intelligence for Clustering

SI is inspired from the collectivity property emerging from a swarm. A swarm of agents such as ants, birds, fish, and insects, can accomplish complex tasks that a single agent cannot do alone, e.g. finding food sources, migrating and protection from enemies. Central control mechanisms commanding the members are not seen in swarms. On the contrary, agents interact with each other and with their environment, and collective behavior is ensured. Thus, SI depends on the self organized and decentralized collective behavior of the swarm.

SI is regarded as one of an Evolutionary Computation (EC) algorithm. Like other EC algorithms such as GA and other evolutionary algorithms, SI is a population based approach, and the population is guided by the objective function value. However, the information sharing mechanisms are different. For example, individuals in a GA share their chromosomes for reproduction, and a competition take place among all the individuals in order to survive. However, SI is based on cooperation and information sharing. That is, swarm members share their experiences and reveal collective intelligence so that swarm steers towards to the promising regions.

This phenomenon was applied to hard computational problems due to its simplicity, scalability and robustness, and promising results were obtained.

Main properties of the collective behavior are as follows (Grosan et al. 2006):

- Homogeneity: Members in the swarm have the same behavior model. That is, swarm moves without a leader although temporary leaders might appear.
- Velocity matching: Swarm members try to match their velocity with the nearby swarm members.
- Locality: Motion of a swarm member is affected from the nearest swarm members.
- Collision avoidance: Swarm members avoid crashing with nearby swarm members.
- Flock centering: Swarm tries to move towards the perceived center of the flock.

A review article about SI in data mining is introduced by Grosan et al. (2006). They report particle swarm and ant colony as the two primary approaches of SI in this field. Reviews about SI that specifically stand for clustering are compiled by Abraham et al. (2008) and Handl and Meyer (2007). Like Grosan et al. (2006), Abraham et al. (2008) also classify clustering algorithms as ACO based and PSO based. Handl and Meyer (2007) mainly focus on ant-based clustering algorithms.

Their classification depends on natural inspiration and they identify two categories: methods directly mimicking the real ant colonies and ant-based general purpose optimization algorithms (methods less inspired from the nature). They emphasize that some algorithms fall into both categories.

Previous classifications do not set clear-cut boundaries between categories, that is, an algorithm might fall into different categories.

In this thesis a classification for SI based clustering algorithms that maintain three disjoint categories is proposed: PSO, ACO and other SI based methods. Properties of these three categories are identified in Table 2.1, and each work in the corresponding category is discussed in the light of these properties. Note that some of these properties serve the same functions in the algorithm, whereas some properties are unique for that category making the difference between categories.

Table 2.1 Properties of SI based algorithms

Particle Swarm Optimization (PSO)	**Ant Colony Optimization (ACO)**	**Other Swarm Intelligence Based (OSIB) Metaheuristics**
▪ Particle representation	▪ Ant representation	▪ Agent representation
▪ Homogeneity	▪ Neighborhood	▪ Neighborhood
▪ Velocity matching	▪ Pheromone update	▪ Solution construction
▪ Locality	▪ Decision rule for solution construction	▪ Exploration mechanism
▪ Collision avoidance	▪ Exploration mechanism	▪ Exploitation mechanism
▪ Flock centering	▪ Exploitation mechanism	

2.4.1. Particle Swarm Optimization (PSO)

PSO has its roots in social psychology. That is, social physiologists show that individuals' behaviors, beliefs and memories are affected from the other individuals in the environment, and cognition is a social process. Thus, PSO emerges from the integration of sociocognitive perspective with problem solving mechanisms. In PSO particles in the swarm fly in the search space with velocities updated according to the experiences they gained throughout the search. The aim is to discover better places in the search space. The interested reader may refer to Poli et al. (2007) and Eberhart and Shi (2001).

PSO applications in clustering are given in Table A.1 in Appendix A. In these applications clustering problem is considered as an optimization problem. That is, an objective function, such as a validity index (like total within cluster similarity or variation) is optimized by PSO approach. In most of the previous work the number of clusters is given a priori, and the objective function is the sum of squared distances or dissimilarities of the points from the cluster representatives (Omran et al. 2002, Omran et al. 2005, Jarboui et al. 2007, Kao et al. 2008, Ahmadi et al. 2010). Thus, the resulting clusters are limited with spherical shapes. There are only a few attempts trying to find the number of clusters inherent in the data set, but they also generate spherical clusters as the objective function maximizes the total within cluster similarity (Picarougne et al. 2007, Veenhuis and Köppen 2006).

Others such as Omran et al. (2006) and Ahmadi et al. (2010) optimize a clustering validity index as the objective function. The important assumption in these studies is that the solution that minimizes or maximizes such a validity index is equivalent to the target clustering solution. However, to the best of our knowledge, the optimization of known validity indices does not guarantee finding the target clustering. Besides, most of the clustering validity indices depend on “the highest” marginal benefit gained from the clustering solution, and they select the clustering solution forming a knee point in the validity index as the “best clustering solution”. Thus, their minimum or maximum values may not give the target clusters.

To sum up, the PSO work in the literature focuses on the deficiencies of *k*-means and its variants specifically in exploration, and tries to get closer to the global optimum as much as possible. In order to increase the effectiveness and efficiency of PSO the proposed algorithms are usually combined with some local search algorithms such as *k*-means (Kao et al. 2008, Ahmadi et al. 2010).

2.4.2. Ant Colony Optimization (ACO)

ACO tries to imitate the behavior of real ants. Ants search for food on the ground and, during their search, they deposit a substance called pheromone. Other ants follow the pheromone trail and this directs their search. Promising paths get more pheromone and more ants are directed along these promising paths. As a result, ants can find the food sources by indirect communication between them.

The general framework for an ACO algorithm is presented in Figure 2.4. ACO has three main functions. These three functions are explained briefly as follows.

```
ACO Algorithm
Set parameters and initialize pheromone values
While termination conditions not met do
    Solution construction
    Local search
    Pheromone update
end while
```

Figure 2.4 The general framework for ACO

Solution Construction: Each ant in the colony constructs a solution. Suppose C = $\{c_1, c_2,..,c_n\}$ is the finite set of solution components. An ant starts solution with an empty sequence, s, and in each construction step, a feasible solution component is selected from set $\mathrm{N}(s) \subseteq \mathrm{C} \backslash s$ according to the decision rule, where $\mathrm{N}(s)$ denotes the set of neighboring feasible solution components (neighborhood) for the partial solution s. Decision rule has a stochastic choice mechanism which depends on the pheromone amount on the solution components. A basic version of the choosing probability conditioned on the current solutions and a solution component c_1 is provided as follows.

$$p(c_i / s) = \frac{(\tau_i)^{\alpha} (\eta(c_i))^{\beta}}{\sum_{c_j \in \mathrm{N}(s)} (\tau_j)^{\alpha} (\eta(c_j))^{\beta}} \quad \forall c_i \in \mathrm{N}(s) \qquad (2.1)$$

where $\eta(c_i)$ denotes the heuristic information function for solution component c_i, and α and β control the relative importance of pheromone value versus heuristic information. Then, the selected solution component is appended to partial solution *s*. The solution construction steps continue until N(*s*) is empty.

Local Search: This part is optional. In this function solutions constructed by the ants can be improved by problem specific local search mechanisms. Thus, it strengthens the exploitation property of ACO.

Pheromone Update: The aim of the pheromone update is to ensure both exploration and exploitation in the search space. This is achieved by the two components in the pheromone model: (1) pheromone evaporation (forgetting) for exploration, (2) pheromone release for exploitation. That is, pheromone evaporation prevents rapid convergence of the algorithm to a suboptimal solution by decreasing all the pheromone values, and good solution components are favored by releasing pheromone. Although there are variants of pheromone update mechanisms, we present a general version for solution component *i* as follows.

$$\tau_i = (1-\rho)\tau_i + \rho \sum_{s\in S_u | c_i \in s} w_s F(s) \tag{2.2}$$

where $\rho \in (0,1]$ denotes the evaporation rate and S_u is the set of solutions used for update. Set S_u can be all the solutions generated in the current solutions, the best solution in the current iteration or the best-so-far solution. Furthermore, *F*(*s*) is the objective function value for solution *s*, and w_s is the weight associated with solution *s*.

There are several variants of ACO algorithms which mainly differ from the pheromone update mechanism. The reader can refer to Dorigo and Blum (2005) and Blum (2005) for the details of ACO.

ACO applications in clustering are presented in Table A.2 in Appendix A. The common property of these articles is the objective function, i.e. minimization of within cluster distance or variance. In Ho and Ewe (2005), Runkler (2005), Chen and Chen (2006), Prabhaharan et al. (2005), and Wang and Wei (2009) the number of clusters is given a priori, and the solution component of ACO is defined as the cluster assignment of a data point. Thus, when an ant completes its path, it represents the cluster assignments of data points. Pheromone values are calculated for these point-to-cluster assignments.

As the cluster labels are not static in this representation, the solution components (i.e. point-to-cluster assignments) do not show the quality of the clustering properly. Instead, the points assigned to the same cluster become connected, and a pattern can be observed to indicate the solution quality. Thus, pheromone update of point-to-cluster assignments does not perfectly reflect the quality of a clustering solution, and it is necessary to take into account the connected patterns in the clusters during the design of the pheromone update mechanisms.

As a remedy, in Tsai et al. (2004) and Sinha et al. (2007) the solution component in ACO is defined as the edge between two data points, that is, if there exists an edge between two points, then these two points are assigned to the same cluster. In this last group of ACO applications each ant visits several data points in every iteration, and pheromone amount is related with the inverse of the tour length of the ant. Clusters are not formed during the iterations, instead clustering solution is constructed after the termination of the algorithm by merging the edges having dense pheromone values. The edges of higher pheromone amounts are identified using a prespecified threshold. With this approach, it is possible to obtain arbitrary-shaped clusters, however intracluster density variations could not be handled.

2.4.3. Other Swarm Intelligence Based (OSIB) Metaheuristics

Algorithms classified as others match the exact properties of neither PSO nor ACO. This group can be analyzed in two categories. First category denotes the algorithms that display the general SI properties. Emergent behavior of the swarm can be seen in this group. Ant-based clustering algorithms are significant examples for this category. It is particular to this group that ants use the stigmergy property which ensures the communication between agents in an indirect manner using the environment (Handl and Meyer 2007). They move in the search space split into grids, and they form clusters from similar points by picking up or dropping off a point on the grids according to its similarity with its neighborhood. These operations are known as corpse clustering, brood sorting and nest building in the literature (Deneubourg et al. 1991, Lumer and Faieta 1994, Handl and Meyer 2007).

The articles in this group are presented in Table A.3 in Appendix A. Martin et al. (2002), Kao and Fu (2006), Yang and Kamel (2006), Handl et al. (2006), and Boryczka (2009) propose ant-based clustering approaches that are based on Deneubourg et al.'s (1991) model. In this model the data points are assigned to grids, and ants move the data points through the grids so that similar points fall into adjacent grids. The moves between the grids are performed according to a probability function reflecting similarity of the neighboring points. Clusters are formed from the points falling into adjacent grids. A mechanism for the interaction of ants such as pheromone is not used. An interesting application in ant-based clustering is due to Azzag et al. (2007). They build a hierarchical tree using ants so that branches are sufficiently dissimilar whereas their roots are similar to each other.

Second group of algorithms does not benefit from SI in a direct manner, and they can be defined better as evolutionary, simulated annealing or tabu search algorithms. Mainly, they lack the collective behavior of a swarm.

The other SI applications include wasp swarm optimization (WSO) (Runkler 2008), honey-bee mating (HBM) (Fathian et al. 2007), and cat swarm optimization (CSO) (Santosa and Ningrum 2009). In WSO limited resources are shared between wasps according to the social status of the wasp. HBM imitates the marriage of real honey-bee. Division of labor between the honey-bees (i.e. queen, workers and broods) ensure to develop high quality generations. CSO is inspired from the behaviors of cats. That is, a cat is either resting in alert-looking way or tracing the targets. However, CSO does not benefit from the emergent property of a group of cats properly. In these studies the objective function is minimization of the total within cluster variance or distance, and the agents represent the representative points of the clusters. The shapes of the resulting clusters are limited to spherical.

2.4.4. Comparison of Swarm Intelligence Based Algorithms

The three categories under SI have the following common points.

- Locality in PSO is related with the neighborhood concept in ACO and OSIB. Locality assets that agents are influenced from the nearest agents. Neighborhood also defines the nearby region that an agent can be primarily affected from.
- Pheromone update in ACO serves a purpose similar to velocity matching and flock centering in PSO. Pheromone update helps both exploration and exploitation. Higher pheromone concentration means higher probability to be selected as a solution component, whereas evaporation of pheromone gives a chance to explore the unpopular paths. Velocity matching functions in a similar manner. It helps the agent to move towards the promising regions, the global and local bests, whereas randomness ensures exploration. Flock centering also functions as an exploitation mechanism by making the swarm members move to the perceived center of the flock (global best and local bests).
- Homogeneity in PSO has commonalities with the decision rule for solution construction in ACO and OSIB. Homogeneity implies that agents use the same behavioral models, that is, agents determine their next position and velocity according to the same rules. Similarly, agents in ACO and OSIB algorithms construct the solutions using the same decision rules.

Besides these resemblances, there are unique properties of these categories. In fact, these also point out the differences in capabilities of SI based algorithms.

- Collision avoidance in PSO prevents the swarm members to collide with the nearby agents. However, such an explicit mechanism is not available in ACO and OSIB, and identical solutions can be constructed during the execution of the algorithm. In fact, ACO is expected to explore the search space thoroughly in order to find the optimal solution, so identical solutions are not desired in the beginning of ACO. On the other hand, thru the end of ACO the existence of identical solutions is an indicator of convergence for ACO.

- Velocity matching in PSO helps the swarm members to move together. Hence, direct interaction among agents can be maintained. However, agents in ACO and OSIB algorithms benefit from the stigmergy property which implies that interaction among agents is ensured by changes they induce in the environment (indirect interaction). Exchange of information by pheromone is an example for this. Hence, collectivity is emphasized less yielding a higher exploration potential.
- Flock centering ensures connectivity among agents, that is, agents move with a smooth density and fewer break-ups. It is an emergent behavior in PSO and an explicit leader is not available in the swarm. Such an implicit effect of the collective behavior is not available in ACO and OSIB. Hence breakups (gaps) can naturally occur yielding distinct and peculiar courses of events to help ramp up exploration.

Work on PSO satisfies most of the principles of SI based algorithms. Homogeneity is ensured by the similar behavior models of the agents. Velocity updates of agents are affected from their own past and the global best. Locality is ensured by various definitions of the neighborhood. Flock centering is usually provided by the tendency to move towards the global best. On the other hand, collision avoidance is not emphasized in a direct manner.

2.5. Clustering Validity Indices

Clustering validity is the assessment of the clustering results obtained by a clustering approach. Basically, a valid clustering solution is expected to be obtained neither by chance nor by a deficiency of a clustering algorithm, so it is a crucial step of the clustering process (Jain et al. 1999). Some important functions of clustering validity are as follows (Tan et al. 2005).

- The existence of clustering tendency in the data set, i.e. data set includes a clustering pattern.

- Determination of the target number of clusters.
- Assessment of the clustering results without a priori knowledge.
- Comparison of the clustering results with target cluster labels given as a priori knowledge.
- Comparison of the two clustering results.

There are three quantitative approaches for measuring the clustering validity.

(1) External criteria: Clustering results are evaluated with respect to a predefined external structure. There are two ways for evaluation in external criteria: comparison of the clustering results with a predefined clustering structure and comparison of the proximity matrix to the clustering result. In the former one, the predefined clustering structure reflects the intuitive information about the clustering, and the degree of correspondence between the clustering solution and the predefined target cluster labels measures the quality of the clustering. Validity indices in this group use either classification based measures such as entropy, purity, and F-measure, and similarity based measures such as Jaccard index, Rand index, and Folkes and Mallows index. In the latter one, clustering results are transformed into a matrix, and this matrix is compared with the predefined proximity matrix using Γ or normalized Γ measures.

(2) Internal criteria: Clustering result is evaluated in terms of the properties inherent in the data set only, and external information is not supplied. These criteria include compactness (i.e. total within cluster distances) and separation (i.e. total intercluster distances) based terms or a combination of these two terms (i.e. Silhouette coefficient). Particularly, these measures are effective in the evaluation of partitional and graph-based clustering algorithms. Besides, for the validation of hierarchical clustering results, the degree of proximity in which two data points fall into the same cluster is compared with the proximity matrix (e.g. Cophenetic correlation coefficient). Besides, single clustering scheme can be analyzed using the proximity matrix as well.

(3) Relative indices: A set of clustering solutions are relatively evaluated using these indices. Given a clustering algorithm, these indices provide to determine the best clustering solution considering the parameters and the assumptions of the algorithm. Some well-known validity indices for hard clustering are the modified Hubert Γ statistic, the Dunn indices, the Davies-Bouldin index, root-mean-square standard deviation, and R-squared. The number of clusters in a data set can be determined using these validity indices.

Both external and internal criteria are statistical basis. That is, external and internal criteria test the null hypothesis about the random structure of the data set. In order to decrease the computational efficiency, they apply Monte-Carlo simulation techniques. The aim of these techniques is the calculation of the probability density function of the validity indices by generating sufficient number of test data sets. Different from external and internal criteria, relative criteria do not use a prespecified information during clustering evaluation. They help to compare the clustering results obtained by different parameter settings.

The reader can refer to Halkidi et al. (2002a) and (2002b) for comprehensive studies on validity indices. The details of these validity indices are provided as well.

Despite the vast literature on clustering validity, challenging issues still exist in this field.

- Verification of the clustering validity methods is easy to perform in 2 and 3-dimensional spatial data sets. However, generalization to higher dimensions and different types of data is limited.
- The absolute value of a clustering validity index may not indicate proper evidences about the clustering quality. Thus, a validity index may need to be interpreted in a relative manner. For example, a validity index of 2 does not show the quality of the clustering. For a proper comparison, the statistical distribution of the validity index can be analyzed.
- It is difficult to apply and interpret the complex validity indices.
- Clustering validity has a subjective point of view in terms of application domain and the characteristics of the clustering algorithm used. For example, spherical shaped clusters tend to occur in facility location, so the total sum of distances between the cluster representatives and the points can be used as a validity index. When a density-based clustering algorithm is applied to a data

set with arbitrary shaped clusters, the total sum of distances between the cluster representatives and the points may not be a proper measure for the clustering quality.

2.6. Scope and Motivation

In this dissertation, we focus on the development and the validation of a new clustering methodology where effective attributes for clustering are provided a priori. We study the clustering problems in which (1) the number of clusters is unknown, (2) clusters may have intracluster density variations and intercluster density differences, and (3) clusters may have arbitrary shapes. Our scope includes handling data sets having spatial properties and numerical attributes. Thus, it is assumed that points are dispersed and clusters are validated in Euclidean space. Data sets having such characteristics are particularly seen in geology, geographical information systems, city planning, and image segmentation.

The proposed clustering methodology includes neighborhood construction, data set reduction and clustering. In the cluster analysis of the spatial data sets, particularly connectivity, proximity and density concepts are crucial. Thus, the proposed clustering methodology is developed based on these concepts.

The swarm concept provides the self-organization and the emergence of collective behavior induced by population dynamics. These properties point at SI as a promising solution approach to the clustering problems. Thus, in this dissertation, we explore the applicability of Ant Colony Optimization (ACO) as one of the SI algorithms to the clustering problem.

In ACO, the neighborhood definition is crucial in ants' moves and it affects the performance and the capabilities of ACO. In this context, we propose a hierarchy of neighborhoods (i.e. top-down hierarchy includes clusters, closures, candidate set, break point set, core sets, data points) for the general clustering problem. This top-down hierarchy reveals the relationships between the data points using the geometrical (topology) properties and the density. That is, the lower levels of the hierarchy resemble kinship, whereas upper levels reflect the citizenship.

The scalability issue is taken into consideration during the development of ACO algorithm. We propose data set reduction mechanisms through boundary formation for the general clustering problem. Given a set of subclusters, the number of data points is reduced by extracting the discriminative properties of subclusters and representing the subclusters by their boundaries. It is based on the adjacency and connectivity information between the data points. As a side product, constructed boundaries can be used for visualization of clusters.

The proposed ACO-based clustering algorithm starts with the neighborhoods and the reduced form of the original data set, and its aim is to find a set of non-dominated solutions that includes the target clustering in terms of separation and compactness. Ants explore the data set so that attractive regions (i.e. clusters) are detected using density break-offs, connectivity and distance information. Hence, a new agent (ant) representation scheme is introduced, and the traditional pheromone update mechanism is adapted for the clustering problem.

In addition to the ACO-based clustering, a three-phase clustering algorithm that integrates the density-based, the graph-based the hierarchical clustering is introduced. The first phase corresponds to the neighborhood construction algorithm. After formation of subclusters (closures) in the first phase, the second phase is dedicated to the outlier detection. Finally, in the third phase the subclusters are merged according to the improvement in the separation and compactness.

CHAPTER 3

A NEIGHBORHOOD CONSTRUCTION ALGORITHM FOR THE CLUSTERING PROBLEM

In this chapter a cluster is assumed to include connected data points in a dense region that is surrounded by low density regions. Thus, data set properties should be considered in both local and global context in clustering. Especially local properties such as proximity, density and connectivity are crucial as they constitute the basis of clustering. In this context neighborhood definition plays a key role in the design of clustering algorithms that depend on search mechanisms in a certain locality. Swarm Intelligence is among this type of algorithms where the agent representation, solution construction, exploration and exploitation mechanisms are all affected by the neighborhood concept. Thus, in this chapter, we consider the extraction of local properties through neighborhood definition as an input to a clustering algorithm.

This chapter is organized as follows. The neighborhood construction algorithms used in the clustering literature are reviewed in Section 3.1. Section 3.2 addresses the strengths and weaknesses of the existing neighborhood construction approaches. The proposed NC algorithm, which is characteristically based on proximity, connectivity and density information, is described in Section 3.3. Typical outputs and time complexity of the NC algorithm are also presented in this section. Section 3.4 includes experimental results with the NC application. We compare NC with some other well-known neighborhood algorithms in terms of effectiveness and homogeneity. Finally, we criticize the performance of NC in Section 3.5.

3.1. Literature Review on the Neighborhood Construction for the Clustering Problem

Neighborhood of a point is defined as the local area or region that includes points in the proximity of the corresponding point (O'Callaghan 1975, Chaudhuri 1996). There are different ways of identifying the proximity relations among points. A simple approach defines the neighborhood of a point according to the distance, that is, points with the smallest distance to the corresponding point are its neighbors. However, this may cause mixing of points from other density regions or clusters.

An early work by O'Callaghan (1975) incorporates direction with the distance. The angle between neighboring points is restricted with a given parameter so that homogeneous density regions are discovered. However, O'Callaghan's neighborhood definition depends on two user-specified parameters and it does not give an ordering of the neighbors.

Chaudhuri (1996) proposes nearest centroid neighborhood which satisfies two conditions: (1) neighbors of a point must be close to that point, and (2) center or centroid of the neighbors of a point must be close to that point. The approach ensures that the neighbors are distributed symmetrically around each point. It especially works on choosing a representative subset from a set of data and detection of border and interior points. Although it is introduced as a parameter-free approach, the size of the neighborhood, *k*, must be given a priori. Besides, the symmetric distribution of the points constitutes a special neighborhood definition, which may be appropriate for the points in the interior of a data set. However, the points on the boundary are crowded in only a certain direction and neighbor mixes with other density regions cannot be avoided.

Graph-based neighborhood definitions consider the proximity and the density structure around a point. One common and simple approach is the *k*-nearest neighbor (KNN) graphs. It is especially used in agglomerative hierarchical clustering methods as it helps to decompose the data set into small connected sets of points, namely subclusters. In Lu and Fu (1978) a point is assigned to the cluster of its nearest neighbor (NN) if the distance between them is smaller than a given threshold. In an extension of NN, cluster assignment of a point is decided with respect to its KNN.

Another agglomerative approach is CHAMELEON (Karypis et al. 1999). Its first phase uses the KNN graph to represent the data set with subclusters as it helps to capture the disconnections and the density. In the second phase, these subclusters are hierarchically merged according to relative interconnectivity and relative closeness properties.

In addition to these hierarchical approaches, some clustering algorithms use the KNN graph to estimate the density as in Wong and Lane (1983). The trade-offs between local and global properties are explored in multiobjective clustering (Handl and Knowles 2005; Handl and Knowles 2007). Connectedness of data points represents the local property, and it is measured as the proportion of the neighboring points in the KNN graph that are in the same cluster. As the objective becomes the maximization of the total connectedness measure, it is implicitly assumed that KNN neighborhood of each data point includes similar points from the same cluster. However, verification of this assumption is not provided in these studies.

In addition to NN and KNN, proximity graphs such as minimum spanning tree (MST) and Delaunay triangulation (DT) help to extract the adjacency information. In order to find the inconsistent edges with respect to their neighborhood, Zahn (1971) compares the edges in the MST with the average length of the edges in the neighboring subtree of depth *d*. Hence, *d* is the user-specified parameter in this approach. AMOEBA (Estivill-Castro and Lee 2000) uses adjacent points in DT as the neighborhood of a point. DT reflects the proximity relations among all the points in the data set, however the points on the boundary of clusters may include neighbors from entirely different density regions or clusters.

A distance-based neighborhood concept was introduced in density-based clustering. In DBSCAN (Ester et al. 1996) ε-neighborhood is defined as the circle with a given radius (ε). If there are at least *MinPts* points falling into the circular neighborhood of a point, the corresponding point is classified as a core point. Points having fewer points in their neighborhood than the threshold (*MinPts*) are either border or noise points. Using these point types, density-connected points are determined in defining the clusters.

OPTICS (Ankerst et al. 1999) extends the neighborhood definition of DBSCAN for varying local densities by working with a spectrum of radius parameters, which are smaller than a given generating radius (ε). Density-based clustering structure is then found by creating an ordering of the data set.

GDBSCAN (Sander et al. 1998) generalizes DBSCAN for spatial data sets. In order to handle spatial data, distance-based neighborhood is extended based on a binary predicate so that intersection and meeting operators can be used for polygons. Besides counting of points in a given neighborhood, density measures are extended to non-spatial features such as taking the average or sum of a particular feature.

Ertoz et al. (2003) combine a density calculation scheme in an ε-neighborhood (distance-based neighborhood) with the KNN (graph-based neighborhood). They can address categorical data as well, however the number of parameters to be determined increases to three, namely k, *MinPts* and ε.

In fact, local properties are also used in other data mining methods such as classification and outlier analysis. For example, in classification the class label of a point can be determined using the class labels of the points restricted to the point's own neighborhood. Density in a certain region helps to distinguish local and global outliers in outlier analysis. In addition to the aforementioned neighborhood definitions, there are neighborhoods defined based on other proximity graphs such as relative neighborhood graph and Gabriel graph and their variations. A comprehensive study about the use of proximity graphs in classification is provided by Toussaint (2002).

3.2. Shortcomings of the Previous Neighborhood Construction Approaches

In most of the neighborhood definitions, the neighborhood size is determined by one or more parameters such as k, d, ε, *MinPts*. There is not a well-defined method to set these parameters properly. Thus, clustering methods that use these neighborhood definitions are sensitive to these parameters.

One of the graph-based neighborhood approaches, the KNN graph, is easy to use. However the parameter k is very critical in extracting the local properties around a point. An inappropriate setting of k might cause errors in the neighborhood, such as mixing points from different density zones or clusters. Besides, KNN does not take into account the connectivity of points. Let us consider the clustering problem given in Figure 3.1. For $k \leq 3$ the neighborhood of point i in Cluster 2 includes the points from the same cluster, namely points r, m and n. For $k = 4$, although points i and p are connected over point m, the neighborhood does not include point p because the connectivity is ignored during neighborhood construction. Instead point o from Cluster 1 is added to the neighborhood. Note that for $k \geq 4$, the neighborhood of point i always includes points from Cluster 1.

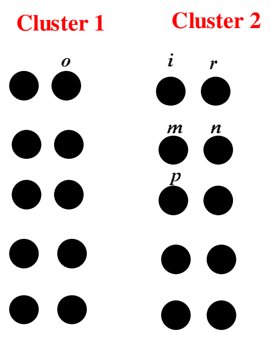


Figure 3.1 Example for the KNN

In the distance-based neighborhood the use of a global neighborhood parameter (ε) fails when there are clusters having density variation. An example is shown in Figure 3.2. Radius ε is set using all the points in the data set, and this setting assures that neighborhood of point j includes points from the same cluster. However, neighborhood of point i is drawn empty, although it is not an outlier. When we increase the ε value, point i is no longer an outlier. However, this increases the neighborhood size of point j and gives rise to neighbor mixes from Cluster 3. In the extensions of distance-based neighborhood, the ε parameter setting is specific for each neighborhood but there are still parameters to be set a priori (Ankerst et al. 1999).

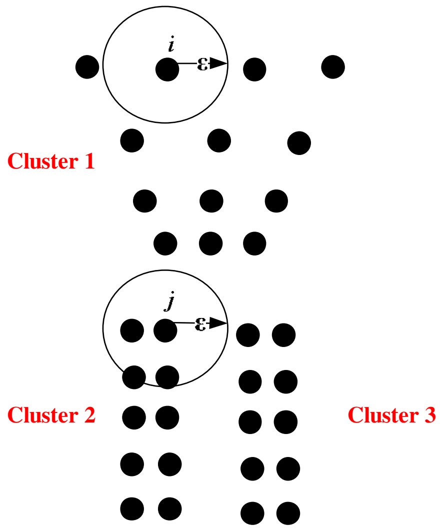


Figure 3.2 Example for distance-based neighborhood

There is a lack of a robust neighborhood construction algorithm, which takes into account proximity, connectivity and density in a local region simultaneously. Zahn (1971) claims that graph-theoretical approaches are powerful in detection of patterns inherent in the data sets. Proximity graphs in the graph theory extract the influence and relevance of the neighboring nodes in a graph and represent proximity information of the nodes. Proximity between any pair of nodes is determined by the distance between the nodes and the existence of other intermediary nodes. Toussaint (1980) states that proximity graphs defined according to a region of influence, such as relative neighborhood graph (RNG) and Gabriel graph (GG), are effective means in pattern recognition. He also claims that both of these graphs are less sensitive to the position of the points, i.e. the angle between connecting arcs, and no restrictions are implied during graph construction unlike in MST.

In NC we determine the proximity and connectivity using GG, which is proposed by Gabriel and Sokal (1969) in order to handle connectedness and contiguity in geographic variation data sets. We use the influence region (a circle in 2-dimensional data sets and a hyperball in higher dimensional data sets) of GG as a reference to calculate the density. The proposed approach is parameter-free and it defines a unique (case sensitive) neighborhood for each data point.

3.3. Neighborhood Construction (NC) Algorithm

We conceive of two basic requirements for a neighborhood definition in a clustering problem: First, neighbors of a point should be determined according to proximity, connectivity and density information processed simultaneously. In order to group similar data points into clusters, solely proximity or distance is not sufficient to construct the neighborhood, especially for the data sets having density variations. Incorporation of density and connectivity information helps to avoid premature break-ups in the neighborhoods and mixes from other density regions. The second issue is the homogeneity of the neighborhood. All neighboring points should be similar, and similarity is to weaken if points from other neighborhoods are forced in.

Considering that proximity graphs are effective in pattern recognition problems (Jaromczyk and Toussaint 1992), we propose a Neighborhood Construction (NC) algorithm based on the Gabriel graph, which is one of the fundamental proximity graphs (İnkaya et al. 2010a and 2010b). Our scope is two or higher dimensional data sets having numerical attributes, and we use the Euclidean distance as the dissimilarity measure. We assume that the number of clusters is unknown. Neighborhood structures constructed by NC work well where (1) clusters have arbitrary shapes, (2) different clusters have different densities, and (3) density varies within a cluster. The major advantage of NC is that it is a parameter-free algorithm using mutual connectivity and density information. NC tries to avoid the generalizations about the neighborhoods of points (such as a minimum number of points within the close vicinity or a predefined distance) and it produces a neighborhood unique to each data point.

The details of the NC algorithm are explained as follows.

3.3.1. Notation and Definitions in NC

We use the notation given below in the discussion to follow.

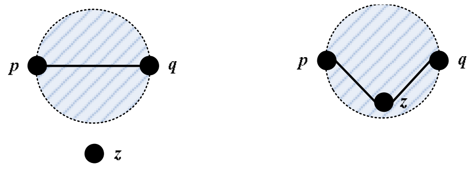
D	set of data points (nodes of the graph)

i, j, p, q	indices for data points
d_{ij}	Euclidean distance between points i and j
CC_i	core candidate set of point i
BC_i	break point candidate set of point i
PC_i	potential candidate set of point i
CS_i	final candidate set (neighborhood) of point i
Cl_m	set of points in closure m

Before proceeding with the description of the NC algorithm, we give the definitions related with the GG. Let B(p, r) denote the set of points included in an open ball centered at point p with radius r, i.e. $\mathrm{B}(p, r) = \{q: d_{pq} < r, q \neq p\}$.

The (p, q) edge is inserted in the GG if and only if $\mathrm{B}(s, d_{pq} / 2) \cap \mathrm{D} = \varnothing$, where s is the midpoint on the line connecting points p and q. Equivalently, the (p, q) edge is inserted if and only if $d_{pq} \leq \min_i \left\{ \sqrt{d_{pi}^2 + d_{iq}^2} : i \in \mathrm{D} \right\}$.

Two nodes p and q are *directly connected* by an edge of the GG if and only if the hyperball having diameter d_{pq} and passing through these two points does not contain any other node of D in its interior. Direct connection makes all connected nodes reachable. Two nodes p and q are *indirectly connected* if the ball with diameter d_{pq} contains at least one other node of D in its interior. This implies that there exists at least one path between the two nodes whose maximum edge length is shorter than d_{pq}. Density between nodes p and q is measured by the number of nodes lying in the ball with diameter d_{pq}. Figure 3.3 shows an example of direct and indirect connections, and the density calculation.



(a) (b)

Figure 3.3 Examples for direct and indirect connection. (a) Direct connection: density between nodes p and q is 0. (b) Indirect connection: density between nodes p and q is 1.

3.3.2. Steps of the NC Algorithm

NC algorithm is composed of five steps. A layer (an enlarging set of points) around a point is defined in each step of NC.

Step 1. Core Candidate Set construction by direct connectivity

In this step, we classify the neighbors of each data point by considering the (direct or indirect) connectivity and density information. Given a point i as the base point, all remaining points in D are listed in non-decreasing order of their distance to point i, and the ordered set T_i is formed. Following the GG construction, the nearest point having an indirect connection to point i is identified as point j. Then, d_{ij} is the distance of the first indirect connection to point i. Data points having a distance to point i shorter than d_{ij} are directly connected to point i with density 0. We call these data points *core (neighbor) points* of point i and include them in CC_i. Indirect connections to other points will be established via these core points.

An example of step 1 is presented in Figure 3.4. First, neighbors of data point 1 are ranked and T_1 is formed. Points 2 and 3 are directly connected to point 1 and the first point subject to indirect connection is point 4, so points 2 and 3 having shorter distance than d_{14} form CC_1.

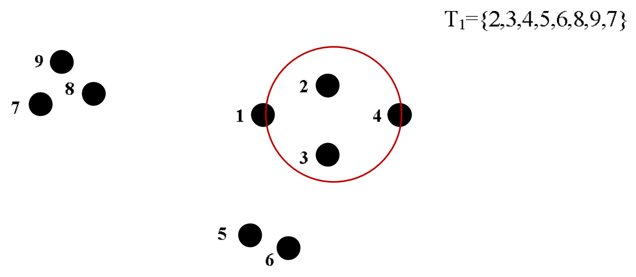


Figure 3.4 Construction of CC_1 = {2, 3}

Step 2. Break Point Candidate Set construction by density tracking

Detection of a density change (if any) is performed next. As one moves to the next member of T$_i$, the density is expected to stay the same or to increase for close neighbors of point *i*. The first data point in T$_i$ at which the density starts to decrease is identified and called the *break point*. This point may be the sign of a density change (a different cluster). The points that are closer to point *i* than the break point form BC$_i$. BC$_i$ is a superset of CC$_i$, and includes points with indirect connections as well.

Figure 3.5 shows an example for step 2. Density values of points in T$_1$ = {2, 3, 4, 5, 6, 8, 9, 7} are 0, 0, 2, 0, 2, 0, 1, 2, respectively. The first density decrease occurs at point 5, hence point 5 becomes a break point. Points 2, 3 and 4 having a shorter distance than d_{15} are included in BC$_1$.

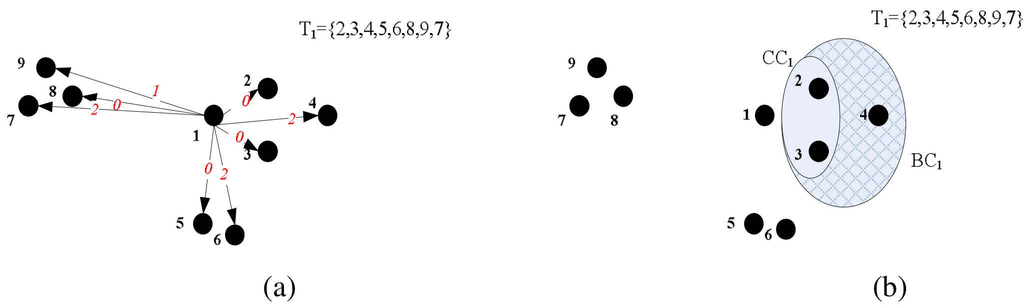


Figure 3.5 Construction of BC$_1$. (a) Density values of points in T$_1$, (b) Construction of BC$_1$ = {2, 3, 4}

Step 3. Potential Candidate Set construction by indirect connectivity checks

The break point marked in step 2 may indicate either a new density region (a different cluster) or simply a dramatic change of direction away from the core of the currently defined neighboring points. Premature set wrapping at a break point may cause falling short in defining the neighborhood of a data point. As a remedy, BC$_i$ is extended by checking the connectivity of points. Let *k* be the first break point of point *i*. If the intersection of sets BC$_i$ and BC$_k$ is nonempty, then there exists at least one point ensuring an indirect connection between points *i* and *k*. Following the ordering in T_i, this check is conducted for every subsequent break point until the first empty intersection of a pair of break point sets (BC_i's) is found. Points collected up to the first empty intersection form PC_i.

Let us consider extension of BC_1 in Figure 5. With data point 5 found as the first break point, we check if there exists a direction change or beginning of a new density region around it. The set $\{1\} \cup BC_1$ is compared with BC_5 to check the existence of a point that induces connectivity between points 1 and 5. BC_5 is {3, 1, 2, 4} and the intersection is {1, 2, 3, 4}. So point 5 is added to PC_1. The density increases to 2 for the next member of T_1 (point 6). This implies that we are moving along a similar density region (as points 3 and 5 are within reach now). So point 6 is also in PC_1. The next density decrease occurs at point 8. Again the set $\{1\} \cup BC_1$ is compared with BC_8. This time, the intersection set is empty as BC_8 is {7, 9}. Hence, potential candidate set construction for point 1 ends with PC_1 = {1, 2, 3, 4, 5, 6}. Looking at Figure 3.5 (b) one can see the separate nature of PC_1 members from {7, 8, 9}.

Step 4. Candidate Set construction by mutuality tests

PC_i includes potential neighboring points of data point i, nevertheless mutual connectivity among the neighboring data points is neglected. Thus, final decision about a neighboring point is made by a mutual connectivity test. Through this operation, PC_i is shrunk to CS_i so that mutuality is provided through at least one direct link in the constructed neighborhoods.

Let point j be any point in PC_i. If point j is in CC_i, then CC_i and CS_j are compared for mutuality. If the intersection of these sets is nonempty, points i and j are mutual (nearest) neighbors. So point j is added to CS_i. If the intersection is empty, then these points are not likely to share the same neighborhood.

In order to remove point j and the subsequent points in the ordered set PC_i there is a final test. If point j is not in CC_i, but the intersection of CS_i and CS_j is nonempty, then point j still passes the mutuality test and gets added to CS_i. These mutuality tests are conducted for each point in successive rounds until no change occurs in any of the CS_i sets.

As an example, let us consider point 7 in Figure 3.5. When we apply steps 1 through 3, we come up with $BC_7 = PC_7 = \{8, 9, 1, 2, 3, 4, 5, 6\}$. BC_7 and PC_7 are identical in this case and both sets include points 1-6 from another cluster revealed by inspection. The reason is that moving away from point 7 the density does not fall due to the far position of point 7. However, mutual connectivity is not satisfied for points 7 and 1 in either of the tests above. Step 4 eliminates point 1 and the points further away from point 1 with respect to point 7 in constructing CS_7. Hence, the eventual set is $CS_7 = \{8, 9\}$.

Step 5. Formation of closures (subclusters) by coverage

Points with common neighbors in the final candidate sets imply that they are connected. Thus, closure sets are formed by taking the union of final candidate (neighborhood) sets that share some data points. Closures constitute the skeleton of the target clustering solution. For example, in Figure 3.6, $CS_1 \cap CS_2$, $CS_2 \cap CS_3$, $CS_3 \cap CS_4$, $CS_4 \cap CS_7$, $CS_7 \cap CS_8$ are all nonempty. Therefore $Cl_1 = CS_1 \cup CS_2 \cup CS_3 \cup CS_4 \cup CS_7 \cup CS_8$. Note here that although $d_{67} < d_{78}$, data points 7 and 8 occur in the same closure, whereas data points 6 and 7 are in different closures. This is simply due to data points 5 and 6 “attracting” each other as an isolated couple.

The pseudocode of the NC algorithm is provided in Section C.1 in Appendix C.

3.3.3. Output of NC

Two main outputs of NC are the neighborhood of each point, CS_i, and the closures, Cl_m. Two complications may arise in the neighborhoods constructed.

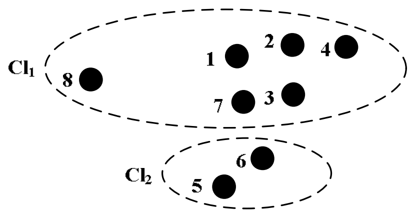
(1) Outlier mixing: If there exist more than one core point for an outlier and if these core points are mutual core neighbors, then outlier mixing is unavoidable. Two or more distinct outliers fall into a cluster in this case.

(2) Divided clusters: Because NC lacks a global view of the data set, some local density drops are taken to mark different density regions. This cuts off data points from the same target clusters and closures tend to be subclusters.

An example is shown in Figure 3.6. In this data set there are 8 points and 2 target clusters, namely, $C_1 = \{1, 2, 3, 4, 5, 6, 7\}$ and $C_2 = \{8\}$. Candidate sets of all points are given in Figure 3.6. Note that points 1 and 7 are mutual core neighbors and both of them are core neighbors of point 8. Thus, intersection of CS_1, CS_7 and CS_8 is nonempty and these points are in the same closure although point 8 is an outlier. This exemplifies the first complication.

When point 7 is the base point, density drops by the move towards point 6. Moreover point 7 constitutes the first break point for point 6. So there is no mutual connectivity between points 6 and 7 and they are not in the same closure.

As a result, we end up with a divided cluster and a mixed outlier in two closures, $Cl_1 = \{1, 2, 3, 4, 7, 8\}$ and $Cl_2 = \{5, 6\}$. This is as case with the second complication.



$CS_1 = \{2, 3, 4, 7\}$
$CS_2 = \{1, 3, 4, 7\}$
$CS_3 = \{1, 2, 4, 7\}$
$CS_4 = \{1, 2, 3, 7\}$
$CS_5 = \{6\}$
$CS_6 = \{5\}$
$CS_7 = \{1, 2, 3\}$
$CS_8 = \{1, 2, 3, 4, 7\}$

$Cl_1 = \{1, 2, 3, 4, 7, 8\}$
$Cl_2 = \{5, 6\}$

Figure 3.6 NC applied to an 8-point data set

Properties of the constructed neighborhoods are as follows.

1. Core points in the neighborhood of a base point have direct connections to each other. They also have shorter distances to the base point than the minimum indirect distance in the candidate set of that point.
2. Mutual nearest neighbors are in the core candidate set of each other.
3. The first break point has the shortest indirect connection to the base point.
4. Final candidate set of a point includes the points with direct and indirect connections.
5. A closure is the union of the candidate sets with some common neighbors.

3.3.4. Computational Complexity of NC

In step 1, for every point in the data set GG is constructed between the associated point and the remaining data points. GG construction has a time complexity of O(n^2) with n nodes. Thus, the time complexity of step 1 of NC is O(n^3). Break point candidate set construction in step 2 is performed for every point in the data set, and for a given point the remaining points in the data set are checked until the first density decrease is detected. Hence, the worst case time complexity of step 2 is O(n^2). Potential candidate set extension in step 3 checks the connectivity between a point and the remaining points for which there exist density decreases. As this is repeated for every point in the data set, the worst case time complexity is O(n^2). Step 4 checks the mutual connectivity of data points with their neighborhoods in O(n^2) time, and this is repeated until no change occurs in the neighborhoods. In Step 5, first, neighborhood of each point is extended by the points both having direct and indirect connection to the associated point. The closures become the union of these extended neighborhoods which have non-empty intersection. Thus, we can infer that step 5 forms closures in O(n^2) time. As a result, the overall time complexity of the algorithm is MO(n^3) where M denotes the number of repetitions for the repeat-until loop.

3.4. Experimental Results of NC

The performance of NC algorithm is tested empirically. In this section, data sets, performance measures and competing approaches are introduced. The experimental results are discussed as well.

3.4.1 Data sets

In our experiments we used three groups of data sets. The first group is composed of 2- and higher dimensional data sets compiled from several sources (Frank and Asuncion 2010, Sourina 2008, İyigün 2008). The properties and plots of group 1 data sets are shown in Table B.2 in Appendix B. These include various shapes of clusters (circular, elongated, spiral, etc.), display intracluster and intercluster density variations, and contain outliers. Some example data sets for group 1 are presented in Figure 3.7. For example, train2 in Figure 3.7 (a) includes both circular and arbitrary shaped clusters with density variation among clusters, there is no outlier and no intracluster density change. data-c-cc-nu-n in Figure 3.7 (b) has circular and elongated clusters with both intracluster and intercluster density differences and contains outliers.

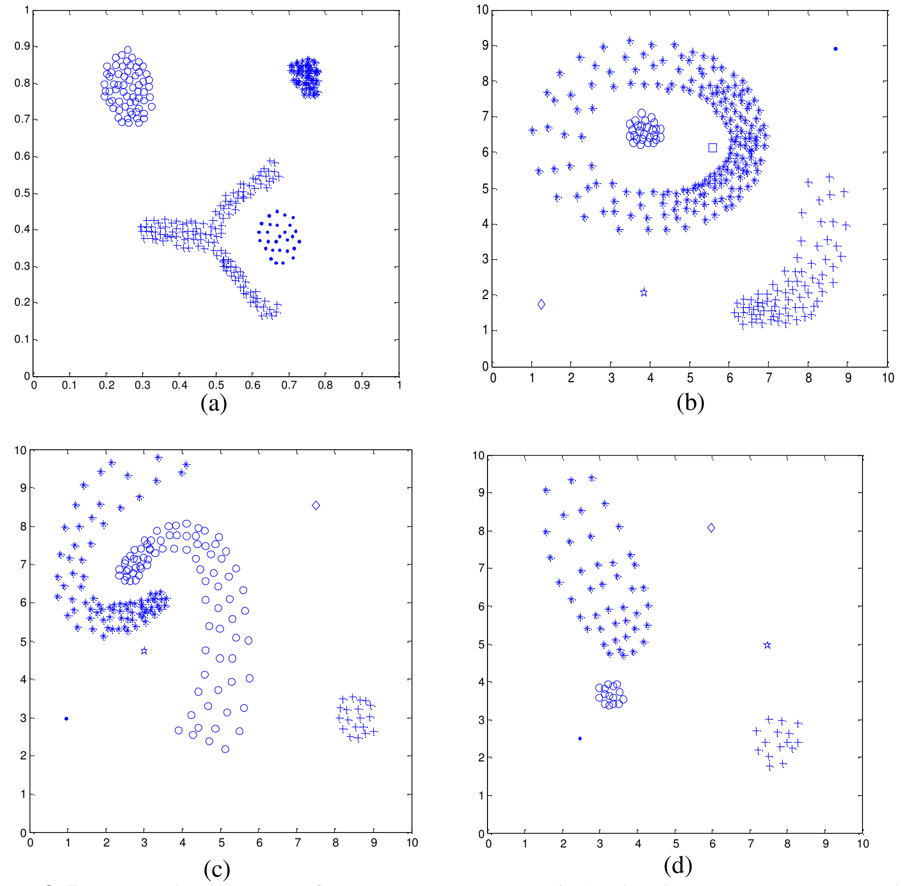
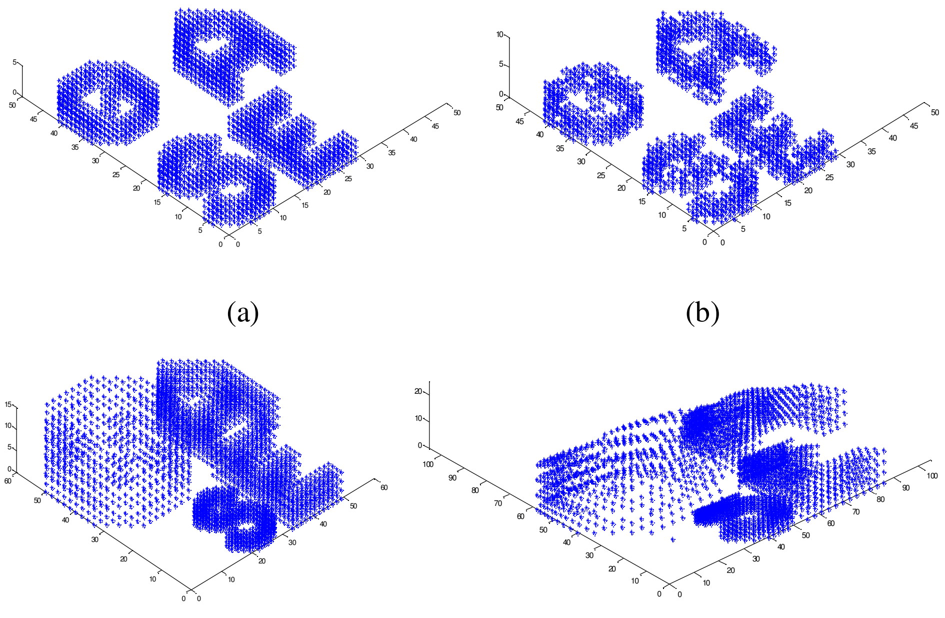


Figure 3.7 Example data sets from group 1. (a) train2, (b) data-c-cc-nu-n, (c) data-uc-cc-nu-n, (d) data-c-cv-nu-n

For the second and third groups, we generated 3-dimensional data sets in order to test the strengths and weaknesses of the proposed approach in a controlled experiment. The data set generation method for both groups is explained in detail in Section B.1 in Appendix B. Tables B.3 and B.4 in Appendix B includes the properties and the plots of groups 2 and 3 as well. Example data sets from groups 2 and 3 are shown in Figure 3.8.



(a) (b) (c) (d)

Figure 3.8 Example data sets from group 2. (a) D_0000: no intercluster density difference, no intracluster density variation, distant clusters, no outlier, (b) D_0100: no intercluster density difference, random intracluster density variation, distant clusters, no outlier, (c) D_1010: clusters with intercluster density difference, no intracluster density variation, close clusters, without outliers, (d) D_1211: clusters with intercluster density difference, smooth intracluster density variation, close clusters, with outliers

3.4.2. Performance Criteria and Comparison

We compared the neighborhoods constructed by KNN, ε-neighborhood and NC in terms of effectivity and homogeneity. Effectivity is the correct contribution of the neighborhood to the solution of the clustering problem. That is, the points that have common neighbors imply the potential for being connected. The closures obtained by taking the union of such neighborhoods are expected to form subclusters. The closer these subclusters are to the target clusters, the higher the accuracy of the neighborhoods. Hence, as an effectivity (accuracy) measure, the closures formed by merging the neighborhoods with shared points are compared with the target clusters.

We use four measures (similarity coefficients) for this comparison: Jaccard index (JI), Rand index (RI), quasi-Jaccard index (QJI) and fraction of points having a pure neighborhood (PPN). We define these measures as follows.

a: the number of point pairs that belong to the same target cluster and are assigned to the same closure.

b: the number of point pairs that belong to the same target cluster but are assigned to different closures (implies division of clusters).

c: the number of point pairs that belong to different target clusters but are assigned to the same closure (implies mixing of clusters).

d: the number of point pairs that belong to different target clusters and are assigned to different closures.

$$\mathrm{JI} = \frac{a}{a+b+c} \tag{3.1}$$

$$\mathrm{RI} = \frac{a+d}{a+b+c+d} \tag{3.2}$$

$$\mathrm{QJI} = \frac{a+b}{a+b+c} \tag{3.3}$$

JI is one of the well-known external clustering validity indices. It takes values between zero and one, one indicating the target clustering is achieved. RI is also known as the simple matching coefficient. While JI focuses on point pairs correctly assigned to the same cluster, RI also takes into account point pairs correctly assigned to different clusters. Both indices penalize division of clusters as well as mixing them.

The purpose of the NC algorithm is not clustering but neighborhood construction. Therefore, division of target clusters does not need to be penalized, but mixing clusters is a more serious problem. A “pure” neighborhood should have a minimum number of points mixed from other clusters. QJI is a measure to quantify such mixes. It is a relaxed version of JI and it only penalizes mixing of points from different clusters.

Our final measure PPN also penalizes only mixing of clusters. It indicates the fraction of points having a “pure” neighborhood (no neighbors from other clusters). PPN is calculated as follows. Suppose the target cluster labels of six points in a data set are 1, 1, 2, 2, 2, 3; and the neighborhood construction algorithm assigns the closure labels 1, 1, 2, 2, 3, 1 to these points. Then, in closure 1 two points (i.e. points 1 and 2) are in the same target cluster 1 out of three points, and assignment of point 6 to closure 1 causes mixing of target clusters 1 and 3. As all the points in closure 2 (i.e. points 3 and 4) are in the same target cluster 2, there does not exist any mixing of clusters. Closure 3 is a singleton so no cluster mix is observed as well. Hence, closures 1, 2 and 3 contribute to PPN with 2, 2 and 1, respectively. To sum up, PPN is found as (2 + 2 + 1) / 6 = 5 / 6. PPN does not penalize separation of point 5 from points 3 and 4, but it penalizes mixing of point 6 with points 1 and 2 in the first closure.

Among these four measures we think that RI and PPN reflect the performance of the neighborhood construction better, as they emphasize placing points from different clusters in different closures.

A neighborhood should be homogeneous as similar points are placed together. Thus, the distances between the neighboring points are expected to have a small variance, whereas the variance of the distances to the points in the complementary set of the neighborhood is expected to be large. This implies that local properties around the points are extracted properly. We check the homogeneity of the neighborhoods by calculating the average of variances over the closures and the average of variances over the complementary sets of the closures in the data set. These two measures together indicate the homogeneity of the neighborhoods.

We used KNN and ε-neighborhood for comparison. To the best of our knowledge, there is not a unique method to determine the number of neighbors, *k*, for the KNN. We used two *k* values, 5% and 10% of the number of points in the data set, and these two settings are labeled KNN1 and KNN2, respectively. These values ensure that *k* is smaller than the size of the smallest cluster in the data set. In some of the previous work (Koontz et al. 1976; Wong and Lane 1983; Karypis et al. 1999), *k* is set to a fixed value within the range from 2 to 30. Our settings for *k* fall into this range.

We applied the procedure used by Ester et al. (1996) for setting the value of ε. A k-distance graph is constructed for the whole data set and a threshold point is determined to find the maximum k-distance value. This maximum k-distance value gives the value of ε. As suggested by Ester et al. (1996) we set k = 4 in the graph and calculate a different ε value for each data set.

3.4.3. Summary of Results

The algorithm was coded in Matlab 7.9 and run on a PC with Intel Core2 Duo 2.33 GHz processor and 2 GB RAM. For an example data set with three clusters, the neighborhoods of two points constructed by the competing approaches are shown in Figure 3.9. Both intercluster and intracluster density variations occur in all the three clusters. Although points 140 and 166 are in the same cluster, the density around point 140 is much higher than the density around point 166 and their neighborhood properties are quite different. In KNN1, KNN2, ε-neighborhood and NC there are 9, 18, 39 and 10 neighbors to point 140, respectively. The same figures for point 166 are 9, 18, 3 and 17. None of the NC neighbors is from a different cluster, whereas both KNN2 and ε-neighborhood produce mixtures from other clusters. NC results in three closures, whereas KNN1, KNN2 and ε-neighborhood merge the two spiral clusters in Figure 3.9 and end up with two closures. Due to the mixing of clusters, homogeneity in the neighborhood is not ensured by either of KNN1, KNN2 and ε-neighborhoods.

The detailed results for the selected data sets are presented in Tables C.1, C.2, C.4, C.5, C.7 and C.8 in Appendix C. The average, standard deviation, minimum and maximum of JI, RI, QJI, PPN, the variance of distances within the neighborhood and in the complementary set of the neighborhood are summarized in Tables 3.1 and 3.2 for the two groups of data sets. We test the statistical significance of the differences among competing approaches in terms of the mean values of performance measures. For both data set groups the NC algorithm performs significantly better than the remaining three neighborhood approaches at 5% significance level in terms of PPN and QJI. ε-neighborhood is the second best, followed by KNN1 and KNN2. Relatively small standard deviations of NC performance measures show that it also yields robust results in cluster mixes.

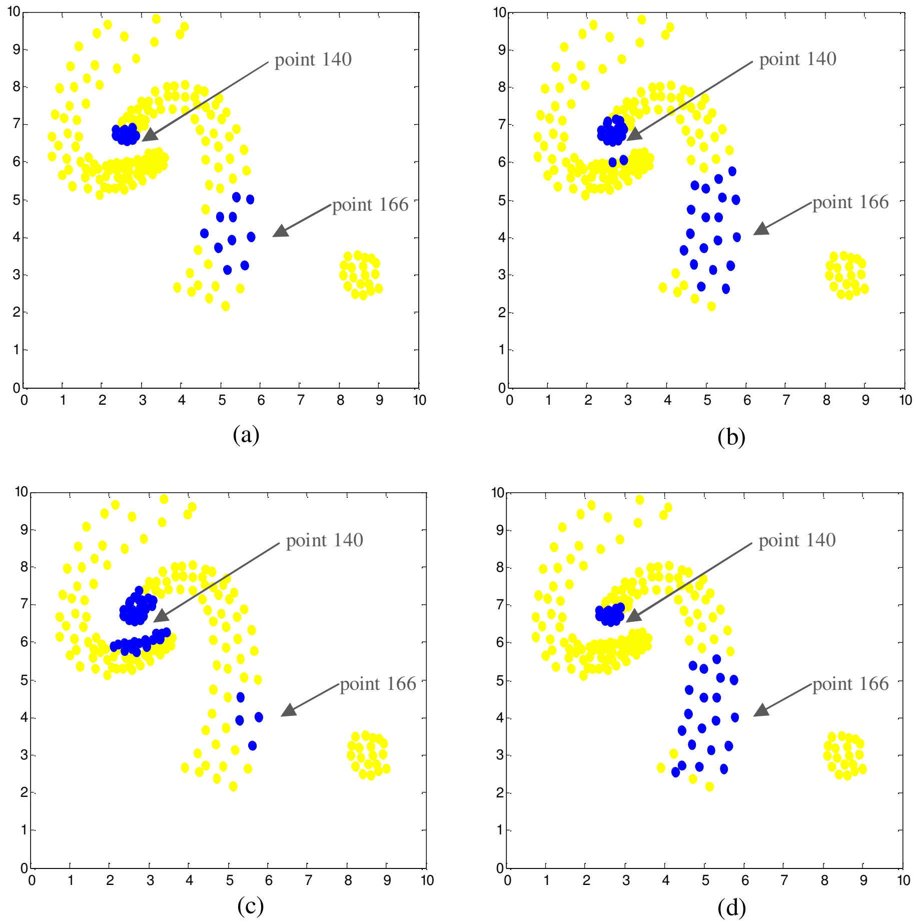


Figure 3.9 Example neighborhood sets for points 140 and 166 in data-c-cc-nu-n_v2. (a) KNN1, (b) KNN2, (c) ε-neighborhood, (d) NC algorithm

Although there is not a significant difference between KNN1 and KNN2 in both groups of data sets, KNN1 has slightly higher average performance, indicating that smaller *k* values in KNN result in fewer mixes from other clusters. In group 1 data sets JI and RI of NC are better than those of ε-neighborhood whereas in group 2 ε-neighborhood has slightly higher JI than NC. In both groups of data sets NC outperforms ε-neighborhood in terms of PPN and QJI, whereas JI and RI results for the two algorithms do not differ significantly. This implies that there are fewer cluster mixes in NC than in ε-neighborhood whereas the number of divided clusters in NC is higher than in ε-neighborhood. The JI and RI values of both KNN1 and KNN2 are significantly lower than ε-neighborhood and NC.

The worst-case performance of NC (minimum of JI, RI, QJI and PPN) is better than the remaining three approaches in the group 1 data sets. The best performance of all four approaches (maximum of JI, RI, QJI and PPN) indicates that there exists at least approach for which target clusters are achieved in the group 1 data sets.

For group 2 the worst-case performance of NC is no more the best among the four competing approaches, instead ε-neighborhood wins in this case. The worst performance of NC in group 2 is observed for the data set D_1211. In this data set separation between two of the clusters is no larger than compactness of one of these clusters. That is, as the distance between clusters decreases, NC results in cluster mixes. In fact, this constitutes the main limitation of NC. In addition, KNN1 and KNN2 cannot find the target clusters in any of the data sets in group 2 whereas target clusters are achieved in the best performances of both ε-neighborhood and NC.

In both groups of data sets the NC algorithm yields by far the smallest variance within the neighborhood, indicating that it results in more homogeneous neighborhoods. The highest variance in the complementary sets of the neighborhoods is also observed in NC. Note that the variance results of ε-neighborhood in group 2 data sets are close to those of NC. Besides, NC and ε-neighborhood yield a more homogeneous neighborhood than KNN1 and KNN2.

KNN1, KNN2, ε-neighborhood and NC find the target cluster labels exactly in 13, 8, 13 and 21 data sets, respectively, among the 69 data sets from the aggregation of both groups. When we compare the neighborhood construction algorithms in terms of cluster mixes, KNN1 and KNN2 mix clusters in 57 and 62 data sets, respectively. ε-neighborhood has cluster mixes in 24 data sets, and this figure is 18 for NC. The cluster mixes in NC are mostly outlier mixes. This explains the reason for PPN and QJI levels still being higher than 0.90.

Results of Full Factorial Experiments for NC

We conduct a factorial design and a full factorial experiment to understand the capabilities of the NC algorithm. According to our full factorial experimental design, the main and interaction effect plots are presented in Figures 10 through 13 for PPN and RI. As seen in Figures 10 and 12 all factors except the existence of the outliers affect both performance measures significantly. As the intercluster density varies, performance of NC drops in both PPN and RI. PPN also decreases in the case of the intracluster density variation. Smooth density change has a more pronounced negative effect on PPN than random density variation does. However, random density variation in clusters slightly improves the RI performance. This may be attributed to the neighborhood construction done in four interacting steps. Smooth change in density gives rise to unavoidable breaks. As the distance between clusters decreases, the accuracy of the NC algorithm worsens.

There is not a significant interaction among factors as Figures 11 and 13 show. As the clusters become closer and there is smooth density variation in the clusters, it becomes difficult to come up with a purified neighborhood and PPN decreases sharply as shown in Figure 3.11. In this case, the largest distances between point pairs in the same cluster is the same as the separation between two clusters, and points from other clusters are taken as core neighbors since they are relatively closer.

The run times of the three approaches for each group are presented in Tables C.3, C.6 and C.9 in Appendix C. As expected, the more sophisticated NC has significantly longer run times than KNN and ε-neighborhood. The run time increases as the number of points in the data sets and the number of dimensionality of the data set increase.

Table 3.1 Performance comparison of KNN1, KNN2, ε-neighborhood and NC for group 1 data sets

		KNN1	**KNN2**	**ε-neighborhood**	**NC algorithm**
JI	**average**	0.79	0.74	0.86	0.88
	std.dev.	0.21	0.22	0.21	0.13
	min.	0.31	0.31	0.31	0.56
	max.	1.00	1.00	1.00	1.00
RI	**average**	0.80	0.75	0.88	0.91
	std.dev.	0.21	0.22	0.19	0.10
	min.	0.31	0.31	0.33	0.66
	max.	1.00	1.00	1.00	1.00
QJI	**average**	0.79	0.74	0.87	0.99
	std.dev.	0.21	0.22	0.21	0.02
	min.	0.31	0.31	0.31	0.91
	max.	1.00	1.00	1.00	1.00
PPN	**average**	0.85	0.81	0.90	0.99
	std.dev.	0.17	0.18	0.17	0.01
	min.	0.50	0.45	0.50	0.94
	max.	1.00	1.00	1.00	1.00
VWN*	**average**	0.57	0.71	0.49	0.22
	std.dev.	0.44	0.41	0.44	0.38
VCN**	**average**	0.15	0.06	0.74	0.94
	std.dev.	0.31	0.22	0.35	0.22

*VWN : Variance within the neighborhood
**VCN : Variance in the complementary set of the neighborhood

Table 3.2 Performance comparison of KNN1, KNN2, ε-neighborhood and NC for group 2 data sets

		KNN1	**KNN2**	**ε-neighborhood**	**NC algorithm**
JI	**average**	0.26	0.25	0.76	0.74
	std.dev.	0.04	0.00	0.25	0.26
	min.	0.24	0.24	0.33	0.25
	max.	0.38	0.25	1.00	1.00
RI	**average**	0.28	0.25	0.89	0.89
	std.dev.	0.10	0.00	0.15	0.17
	min.	0.24	0.24	0.51	0.41
	max.	0.60	0.25	1.00	1.00
QJI	**average**	0.26	0.25	0.82	0.88
	std.dev.	0.04	0.00	0.24	0.21
	min.	0.24	0.24	0.57	0.39
	max.	0.38	0.25	1.00	1.00
PPN	**average**	0.30	0.29	0.86	0.91
	std.dev.	0.06	0.00	0.19	0.17
	min.	0.28	0.28	0.38	0.31
	max.	0.50	0.29	1.00	1.00
VWN*	**average**	0.95	1.00	0.12	0.01
	std.dev.	0.16	0.00	0.12	0.01
VCN**	**average**	0.05	0.00	0.98	0.99
	std.dev.	0.15	0.00	0.03	0.02

*VWN : Variance within the neighborhood
**VCN : Variance in the complementary set of the neighborhood

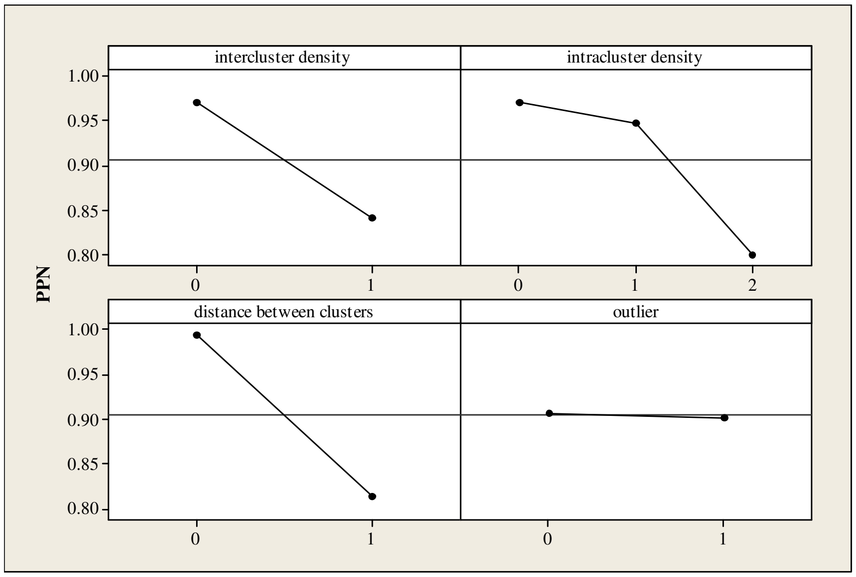


Figure 3.10 Main effect plots of group 2 data sets for PPN

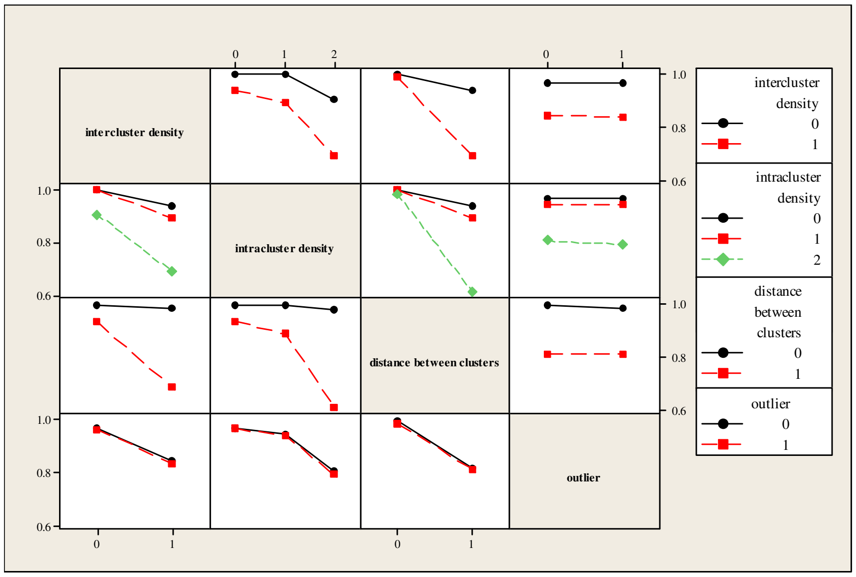


Figure 3.11 Interaction effect plots of group 2 data sets for PPN

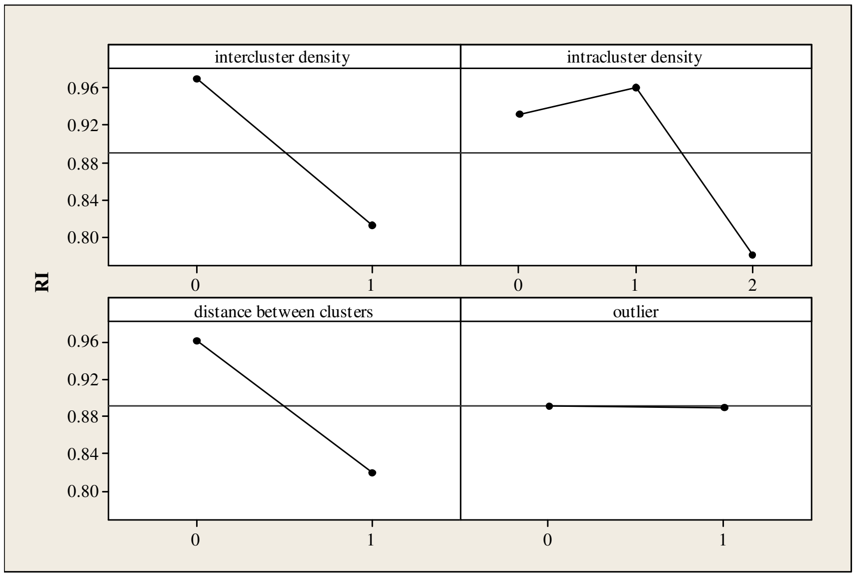


Figure 3.12 Main effect plots of group 2 data sets for RI

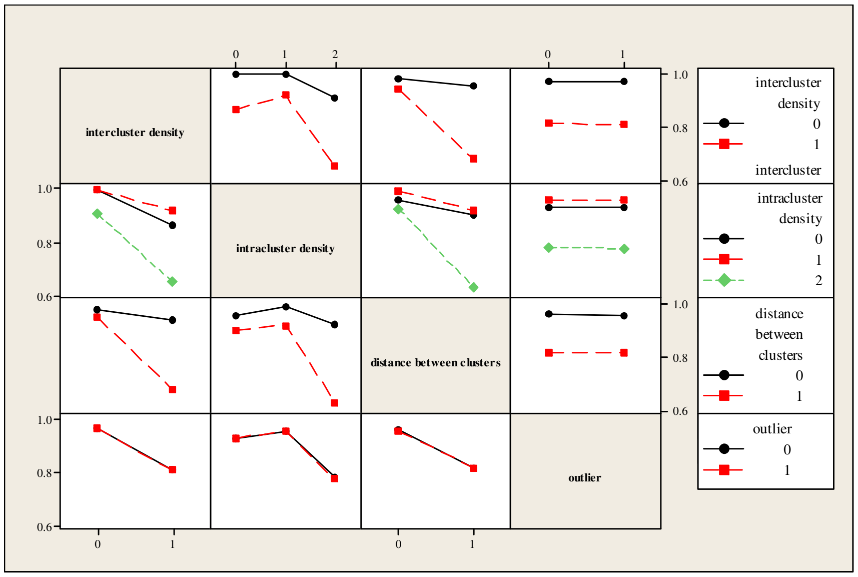


Figure 3.13 Interaction effect plots of group 2 data sets for RI

We also examine the relationship between the characteristics of the data sets (i.e. the minimum separation-to-compactness ratio (MSCR), the coefficient of variation of the edge lengths in MST of the whole data set (CV1), and the average of the coefficient of variations of the edge lengths in individual cluster MSTs (CV2)) and the performance of the NC algorithm. The scatter plots in Figure 3.14 show that there is positive correlation between RI and MSCR (0.54), and between PPN and MSCR (0.53). CV1 and CV2 are negatively correlated with the two performance measures. CV1 has correlations of -0.53 and -0.48 with RI and PPN, respectively. The same figures are -0.56 and -0.52 for CV2. That is, as the clusters become more separated from each other, it gets easier to find the target clusters. Besides, both intracluster and intercluster density variations disrupt the performance of the algorithm.

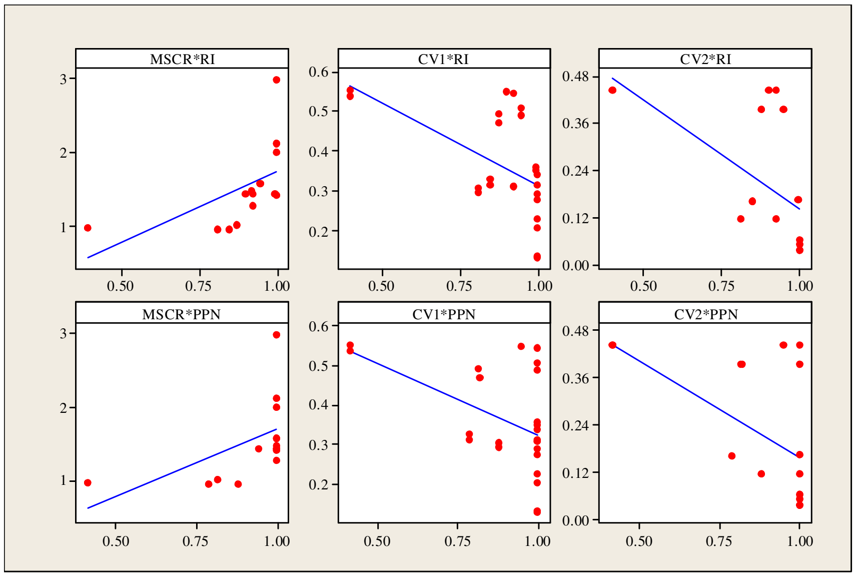


Figure 3.14 Scatter plots for the data set characteristics (MSCR, CV1 and CV2) versus the performance measures (RI and PPN)

The results indicate that the NC algorithm acts as a preprocessing step before any clustering attempt and yields quite homogeneous neighborhoods. It handles density differences in the local regions up to a certain extent and it prevents cluster mixes. In order to ensure the homogeneity of the neighborhoods NC is characterized by placing more weight to the local view rather than the global. Particularly, ignoring the global view results in a larger number of closures than the number of target clusters in the data sets whose clusters become visible only at a relatively high resolution.

Main limitation of NC occurs when the distance between the clusters in a data set is equal to or smaller than the maximum distance within the cluster. In this case cluster mixes might occur as the points from other clusters become the core neighbors.

Unlike NC, ε-neighborhood needs an input parameter and it is crucial to set this parameter correctly. Since ε-neighborhood is based only on the distance information, it fails when there exist density differences in the neighborhood. Besides, the use of a single distance threshold constrains the performance of the approach for the cases having different density regions in the data set and causes cluster mixes in the neighborhoods. As the distance parameter is set considering the whole data set, ε-neighborhood has a more global view than NC and results in fewer closures. Performances of KNN1 and KNN2 fall behind ε-neighborhood and NC because KNN does not consider distance, density and connectivity during neighborhood construction. Although the value of *k* is set smaller than the minimum cluster size, cluster mixes occur in the neighborhoods. Lower *k* values perform slightly better as revealed in some results.

3.5. Discussion of NC Results

In the previous work only proximity and direction information is taken into account to explore the local properties in constructing neighborhoods. However, these are not sufficient when there are density variations in a local region. As a remedy density-based connectivity through GG is used in the proposed NC algorithm. The experimental results indicate that the idea is effective and relatively more purified neighborhoods are constructed. As the number of cluster mixes in the purified neighborhood is small, it enables effectivity in clustering. Thus, NC can be used as a preprocessing step for a neighborhood-based clustering method. If the succeeding clustering algorithm is specialized in the merging operations, target clusters can be found with more ease and accuracy.

The focus on local properties limits the global view of NC which causes tight and homogeneous neighborhoods. In order to handle the data sets where the distances between clusters are small compared to within cluster distances, a collective evaluation of the proximity, density and connectivity information is needed. NC can also be improved for isolation of outliers from the clusters. Finally, ways of reducing the run time of NC can be developed.

Although the main motivation of the NC algorithm is extraction of local properties for clustering, NC can also be used as a nearest-neighbor classifier for the classification problem. Moreover neighborhoods constructed by NC can be used in calculating connectivity for cluster validation purposes.

CHAPTER 4

A DENSITY-BASED CLUSTERING APPROACH IN GRAPH THEORETIC CONTEXT

In this chapter, we propose a new density-based clustering algorithm. A graph theory context is adopted to address arbitrary shapes and heterogeneous densities in two or higher dimensional space. Unknown number of clusters is assumed. First, the density-based connectivity relations in the neighborhood are considered in closure (subcluster) formation. Next, a hierarchical agglomeration scheme is provided in order to come up with the final clusters.

In Section 4.1, the density-based and hierarchical algorithms are reviewed briefly. The three-phase heuristic (NOM) is presented in Section 4.2. Experimental results are presented in Section 4.3, and results are discussed in Section 4.4.

4.1. Literature Review

The main ideas behind the density-based clustering algorithms are connectivity and density, and these two characteristics facilitate handling clusters with arbitrary shapes. Salient works in density-based clustering include DBSCAN (Ester et al. 1996), OPTICS (Ankerst et al. 1999) and GDBSCAN (Sander et al. 1998). The reader can refer to Chapter 3.1 for the main properties of these algorithms.

Hierarchical agglomerative clustering methods construct clusters in stages. Among these CURE (Guha et al., 1998) uses a fixed number of representative points to define the clusters. Agglomeration of a cluster pair is conducted considering the minimum distance between representatives and this is repeated until the given number of clusters is achieved. Although CURE can handle arbitrary shapes, the parameters including the number of representative points, the number of clusters and the shrink factor should be set a priori. One of the complications of CURE is handling intracluster and intercluster density variations. CHAMELEON (Karypis et al., 1999) uses *k*-NN to partition the data set. Merging of these partitions depends on the graph connectivity. That is, relative inter-connectivity and relative closeness are calculated between each cluster pair and compared with a given threshold. Like CURE, CHAMELEON can extract arbitrary shaped clusters with different sizes and densities, but cannot avoid problems due to density variations within clusters as distances are not measured relative to neighborhoods. Since we are interested in arbitrary shaped clusters with varying densities, we propose the following procedure for merging subclusters.

4.2. Neighborhood Construction – Outlier Detection – Merging (NOM) Algorithm

Our algorithm is composed of three phases: neighborhood construction, outlier detection and merging (İnkaya et al. 2010a and 2010b). The first phase uses ideas from (a) density-based algorithms to handle arbitrary shapes and (b) graph theoretic representations to address varying densities. A neighborhood is constructed for each data point using the proximity and connectivity information. The second phase focuses on outlier detection. Local Outlier Factor (LOF) proposed by Breunig et al. (2000) is revised for the neighborhoods obtained. In the third phase, a hierarchical agglomeration is performed where closures are merged considering the improvement in separation-to-compactness ratio subject to consistency with their neighborhood.

4.2.1. Notation in NOM

In describing NOM, we use the following additional notation to the NC algorithm in Chapter 3.

m, n	indices for clusters
d_{ij}^{GG}	Gabriel Graph (GG) distance between points i and j
lrd_i	local reachability distance for point i
LOF_i	local outlier factor for point i
$i(m)$	point i in cluster m
MST_m	set of edges in the Minimum Spanning Tree (MST) of the points in cluster m
$\mathrm{MST}_{i(m)}$	set of edges in the MST of the points in the neighborhood of point i in cluster m
GG_{ij}	set of edges in the GG of the points that are in the ball centered at the midpoint of points i and j with diameter d_{ij}
NGG_m	set of clusters in the GG neighborhood of cluster m
C_m	set of points in cluster m
sep_{mn}	single link separation between clusters m and n
$comp_{(i)m}$	compactness for the neighborhood of point i in cluster m

4.2.2. Phases of NOM Algorithm

Three phases of the NOM algorithm are described below.

Phase 1. Neighborhood construction

This is done with the NC algorithm described in Chapter 3. The resultant CS_i's are the neighborhoods of points i.

Phase 2. Outlier detection

An outlier is a point that shows abnormal behavior relative to all clustering in a data set. In the literature, there are algorithms that extract both clusters and outliers, such as CURE (Guha et al., 1998) and DBSCAN (Ester et al., 1996). However, they specialize in the detection of global outliers, and neither intercluster nor the intracluster density variations are considered. In Breunig et al. (2000), points that are outlying relative to their local neighbors are defined as local outliers. They use a parameter to define the number of points in a neighborhood and compute a Local Outlier Factor (LOF) for each point using this neighborhood. LOF represents the degree of being an outlier based on relative comparison of the average reachability distances of a point and its neighbors.

We are interested in both global and local outliers. Thus, we identify the outliers using a revised version of LOF. Instead of using a fixed parameter to define the size of the neighborhood, we use the neighborhoods constructed in step 1. As the NC algorithm makes use of GG connectivity, resulting neighborhoods can have different sizes and arbitrary shapes. Euclidean distance calculation may mislead the density calculation as it considers the direct links only. In GG distance calculation, reachability is ensured via indirect edges with shorter edge lengths, so we claim that it compensates for the intracluster density variations. Thus, we consider the GG distance between two points in local reachability calculation.

The GG distance takes into account the connectivity between two points. It is the edge with the maximum length in the GG of the points circumscribed by the ball passing through points i and j, i.e. $d_{ij}^{GG} = \max_{(k,l)\in \mathrm{GG}_{ij}} \{d_{kl}\}$. Then, the revised local reachability density and LOF becomes $lrd_i = \left(\frac{\sum_{j\in \mathrm{CS}_i} d_{ij}^{GG}}{|\mathrm{CS}_i|} \right)^{-1}$ and $LOF_i = \frac{\sum_{j\in \mathrm{CS}_i} \frac{lrd_j}{lrd_i}}{|\mathrm{CS}_i|}$. Given a threshold level, a, if $LOF_i > a \max_{j\in \mathrm{CS}_i} \{LOF_j\}$, then point i is called a local outlier. In other words, the reachability density around the neighborhood of a local outlier is a times greater than the reachability density around its neighboring points. This implies that a local outlier shows inconsistency in terms of reachability density in its locality.

Phase 3. Merging

At the end of the first phase, we have closures Cl_m obtained from the NC algorithm. After outliers are separated in the second phase, NC closures may consist of divided clusters. As the first two phases take into account density variations in the neighborhood, they depend on the local view. The whole data set is not considered, so there is a lack of global view in the clustering solution. As a remedy, a hierarchical agglomerative procedure is used for merging the neighboring clusters. In order to consider both global and local patterns in the data, improvement in the separation-to-compactness ratio and dispersion of the neighbors are taken into account as the two merging criteria. Clusters subject to merging are determined by using the GG. Two clusters are in the same GG neighborhood if the ball drawn across the nearest two points of a cluster pair does not include any points from other clusters. The following two criteria are then checked for merging.

Criterion 1. Improvement in the separation-to-compactness ratio

We define the potential compactness of a cluster as the most inconsistent edge in the neighborhoods it contains. MSTs are constructed to identify the connections with the minimum total length in a cluster and in all its relevant neighborhoods. Then, each edge in the cluster's MST is compared with the edges in the MSTs of the neighborhoods within cluster *m*. Potential compactness of cluster *m* is defined as $pcomp_m = \max_{(i,j)\in \mathrm{MST}_m} \left\{ \frac{d_{ij}}{comp_{i(m)}}, \frac{d_{ij}}{comp_{j(m)}} \right\}$ where compactness value for the neighborhood of point *i* in cluster *m* is $comp_{i(m)} = \max_{(p,q)\in \mathrm{MST}_{i(m)}} \left\{ d_{pq} \right\}$.

If the current cluster had to be divided, the edge that would define the separation would most probably be the most inconsistent edge with its neighborhood identified by $pcomp_m$.

Let the candidate clusters for merging be 1 and 2 where $(i^*, j^*) = \underset{i\in \mathrm{C}_1,\, j\in \mathrm{C}_2}{\arg\min} \left\{ d_{ij} \right\}$. Then $d_{i^*j^*}$ is the separation between clusters 1 and 2. Merging them will eliminate the separation $d_{i^*j^*}$ and it will replace the potential compactness for the merged cluster.

A new separation value will emerge between the merged cluster and the cluster nearest to either 1 or 2.

We find out whether the current separation-to-compactness ratio will improve after the merging. We normalize the separation to account for heterogenity and calculate the current separation-to-compactness ratio as $csep_{12} = \max\left\{\frac{d_{i^*j^*}}{comp_{i^*(1)}}, \frac{d_{i^*j^*}}{comp_{j^*(2)}}\right\}$ and $current\_sc = \frac{csep_{12}}{\max\{pcomp_1, pcomp_2\}}$.

We consider the lower bound $lb = \min_{\substack{m\in \mathrm{NGG}_1 \\ n\in \mathrm{NGG}_2 \\ m\neq 2, n\neq 1}} \{sep_{1m}, sep_{2n}\}$ as the possible separation value after merging. If we merge clusters 1 and 2, the bound on the new separation-to-compactness ratio becomes $new\_sc \geq \frac{lb/d_{i^*j^*}}{csep_{12}}$ where $d_{i^*j^*}$ is used to normalize the lower bound on separation, and $csep_{12}$ becomes the new normalized compactness of the merged cluster.

If *new_sc* is greater than *current_sc*, we conclude that the separation-to-compactness ratio improves after merging. However, this might still be an incorrect signal for merging, especially for the heterogeneous data sets with large distance variations between clusters. Although the ratio seems improving, the new compactness value after merging might be inconsistent with its neighborhood. For this reason, a second check is conducted for the consistency of the neighborhood.

Criterion 2. Heterogeneity of edge lengths in the neighborhood

If the candidate clusters for merging satisfy the first criterion, we consider the separation $csep_{12}$ between these two clusters as the potential compactness. To merge, this new edge should be consistent with the neighborhoods of its end points. Hence, merging is performed if this edge does not worsen the existing dispersion of edge lengths in the neighborhoods, that is

$$csep_{12} \leq \max\left\{\frac{\max_{(i,j)\in \mathrm{MST}_{i^*(1)}}\{d_{ij}\}}{\min_{(i,j)\in \mathrm{MST}_{i^*(1)}}\{d_{ij}\}}, \frac{\max_{(i,j)\in \mathrm{MST}_{j^*(2)}}\{d_{ij}\}}{\min_{(i,j)\in \mathrm{MST}_{j^*(2)}}\{d_{ij}\}}\right\}. \tag{4.1}$$

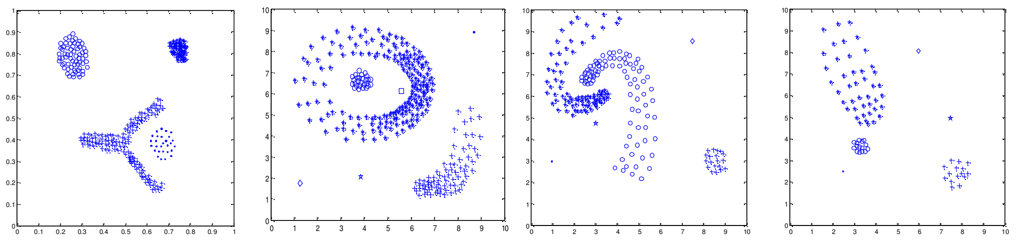
The pseudocode of the NOM algorithm is provided in Section D.1 in Appendix D.

4.2.3. Computational Complexity of NOM

The time complexity of the NC algorithm is MO(n^3) explained in Section 3.3.4 whereas computational complexity of the outlier detection phase is O(n) as outlier check is performed for every point in the data set. Merging continues until none of the cluster pairs satisfy the two merging criteria simultaneously. To sum up, the overall time complexity of NOM is MO(n^3).

4.3. Experimental Results for NOM

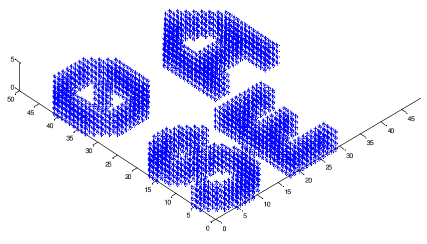
Performance of NOM is tested on three groups of data sets. Group 1 data sets are taken from the literature (Frank and Asuncion 2010, Sourina 2008, İyigün 2008) whereas groups 2 and 3 are 3-dimensional control groups to explore the capabilities of NOM. The details of data generation mechanisms for groups 2 and 3 are explained in Section B.1 in Appendix B. The data set properties and plots for three groups are presented in Sections B.2 through B.6 in Appendix B, respectively. Note that there are 45 and 24 data sets in groups 1 and 2, respectively. Target clusters are either given by the data source or found by visual inspection. The plots of some example data sets are provided in Figures 4.1 and 4.2.

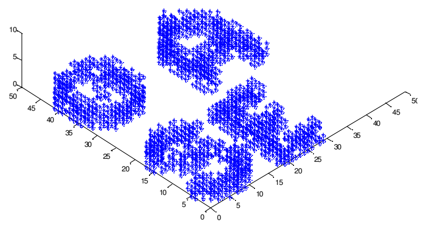


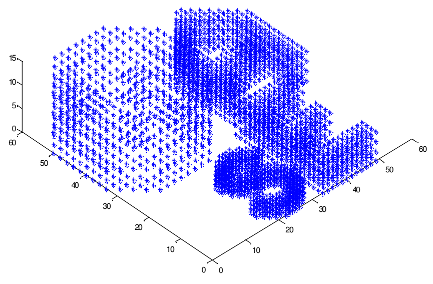
(a) (b) (c) (d)

Figure 4.1 Group 1 data sets (a) train2, (b) data-c-cc-nu-n, (c) data-uc-cc-nu-n, (d) data-c-cv-nu-n

Four performance criteria are used in evaluating the results: the number of clusters, Jaccard index (JI), Rand index (RI) and quasi-Jaccard index (QJI). JI and RI are well-known external cluster validity indices. JI focuses only on the number of point pairs that belong to the same target cluster and assigned to the same cluster whereas RI also considers the number of point pairs that belong to different target clusters and assigned to different clusters. Both of them penalize the divisions and mixes of target clusters. In NC we work on neighborhood construction and we aim to have no mixes from other clusters in the neighborhoods. In order to measure this, we use the relaxed version of JI, namely QJI, which penalizes only the number of point pairs that belong to the same target cluster and assigned to different clusters. Each measure is calculated for the target clustering solution versus found solutions. The algorithm is coded in Matlab 7.9 and run on a PC with Intel Core2 Duo 2.33 GHz processor and 2 GB RAM.


(a)


(b)


(c)

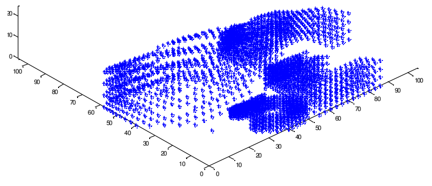

(d)

Figure 4.2 Group 2 data sets (a) D_0000: no intercluster density difference, no intracluster density variation, distant clusters, no outlier. (b) D_0100: no intercluster density difference, random intracluster density variation, distant clusters, no outlier. (c) D_1010: clusters with intercluster density difference, no intracluster density variation, close clusters, without outlier. (d) D_1211: clusters with intercluster density difference, smooth intracluster density variation, close clusters, with outlier.

The performances of NC, outlier detection (OD) and NOM after merging are compared with the results of *k*-means, single-linkage (SL) and DBSCAN approaches. In our comparison *k*-means represents the partitional clustering approach and SL the hierarchical clustering approach. SL also has a graph theoretic view as it has an analogy with MST construction. DBSCAN is selected as a representative of the density-based clustering algorithms. In order to have a fair comparison among these algorithms, *k*-means is run for several values of *k* in the range between 2 and 10% of the points in the data set with increments of 1, and the one with the best JI is used. In the same manner, for DBSCAN, among several *MinPts* settings the one with the best JI is selected for comparison.

The only parameter in NOM, the threshold level a, is set to 2 after pilot runs. The details of the results for some sample data sets are given in Tables D.1 through D.9 in Appendix D. In Tables 4.2 and 4.3 the summary of the results for the entire group 1 and group 2 data sets are provided. Clustering results for an example data set is provided in Figure 4.3. For this data set, JI values with *k*-means, single linkage and DBSCAN algorithms are 0.59, 0.49 and 0.50, respectively. Results of both NC and NOM are superior with respective JI values of 0.98 and 1.

According to Table 4.1, NOM gives the best average and minimum values of JI and RI over 45 data sets in group 1, as well as the smallest standard deviation. For QJI the best average performance and the smallest standard deviation are achieved by NOM, but *k*-means is better in terms of the minimum. That is, NOM results in clustering solutions close to target clusters. Moreover, the number of cluster mixes is fewer in NOM on the average.

For group 1 data sets, which include arbitrary shapes, intercluster and intracluster density variations, NOM gives the best performance among all the clustering algorithms. Using density-based connectivity through GG, NC is the initial phase for detecting both arbitrary shapes and density changes in the clusters. Outlier detection based on the neighborhoods ensures separation of such points in less dense regions. Merging is performed wherever the separation-to-compactness ratio indicates an increase. The relative evaluation of compactness and separation values according to the neighborhoods in clusters helps handling arbitrary shapes and density differences.

In data sets in which clusters are well-separated and there are ruptures in the intracluster density variations (e.g. data_circle_20_1_5_10 and data_mix_uniform_ normal), NOM solution has more clusters than the target solution whereas the clustering solutions obtained by single-linkage and DBSCAN are better. The main reason is the lack of a global view in NOM. In particular, both the NC and the outlier detection phases of NOM have a local view as the decisions are made depending on the information gathered from the neighborhoods. Merging in the third phase tries to bring about a global perspective by checking the improvement at a larger scale, that is, neighborhoods of the clusters instead of points. However, the scale we consider seems to be insufficient to fully realize this global perspective. Particularly, current separation value ($csep_{12}$) and new separation-to-compactness ratio (*new_sc*) need a more global assessment scheme.

In group 1 experiments target clusters are achieved in 9, 32, 17, 13, 16 and 23 data sets for *k*-means, single linkage, DBSCAN, NC, outlier detection phase of NOM, and NOM, respectively. Note that merging operations in the third phase worsen the performance of NOM in three data sets. In fact these three data sets are the only ones that have JI smaller than 0.80 in NOM. One of them (train3) having the worst performance in JI (0.59), includes noise rather than a few outliers. JI is calculated greater than 0.90 after NC and outlier detection phases and most of the noise is detected as outlier. However, in the merging phase of NOM noise is perceived as a cluster showing similar density properties, so most of these points are merged and clusters made up of noise are formed.

We also tested the noise removed version of this data set and NOM was successful in finding the target clusters in this version. As a result, we can infer that NOM is not capable of handling noise. The remaining two data sets that have JI smaller than 0.80 (data_circle_5_10_8_12 and data_circle_3_10_8_12) include intermingled clusters. JI is greater than 0.75 after NC and outlier detection phases. However, the close proximity between the clusters prevents the algorithm from detecting different density regions by the separation-to-compactness ratio, and the clusters are merged in the third phase. Consequently the limitations of NOM are handling data sets with intermingled clusters and noise present in the space between the clusters.

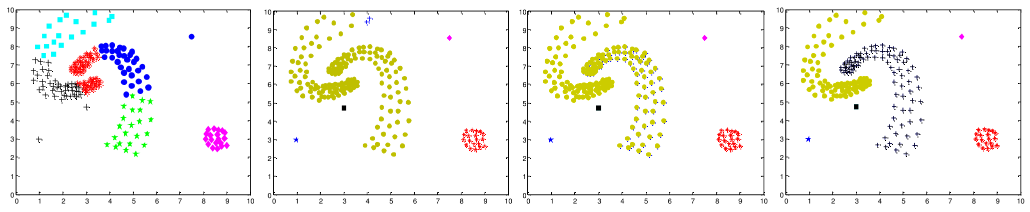
Group 2 is used to explore the main limitations and strengths of NOM further. In this controlled experiment target clusters are achieved in 12, 6, 8, 7 and 7 data sets with single linkage, DBSCAN, NC, outlier detection, and NOM, respectively. *k*-means could not find the target clusters in any of the data sets in group 2, although it seems the best in terms of RI. The letters in group 2 are non-convex, but the shapes are not intertwined. Thus, the center calculation in *k*-means is still useful, and *k*-means shows an average performance in all data sets. As seen from Table 4.1 NOM is no more the best performer, and DBSCAN and *k*-means have higher JI averages. However, both algorithms find the target clusters in fewer data sets than NOM. Single linkage, having the highest number of successes, does not show good performance in the entire group. DBSCAN having the highest JI achieves the target clusters in only 6 data sets. NOM finds the target clusters in 7 data sets but its JI average is only 0.758. In fact, NOM works well in certain data sets as seen in Tables D.1 and D.2 in Appendix D, and performance becomes poor for a certain group. Factorial analysis is conducted to determine the data set properties for which NOM has poor and superior performance.

The effects of the four factors in Table B.1 in Appendix B on NOM's performance (RI) are presented in Figure 4.4 (a). When the density differs among the clusters and the distance between clusters is close (intercluster distance is equal to the distance between the points in the same cluster), RI decreases. The negative effect of smooth density variation is higher than the random intracluster density variation. Note that the existence of outliers does not have a significant effect on the performance of NOM. According to Figure 4.4 (b) the negative effect of the smooth density change increases when the intercluster distance is close. When we exclude the data sets having these properties, the remaining have JI values higher than 0.80. Thus, NOM is capable of handling data sets with intracluster density variations and intercluster density differences when the distance between the clusters is greater than the distance between the closest points in the same cluster. Otherwise, the mixing of clusters seems unavoidable.

To summarize, despite its high performance in JI, RI and QJI, *k*-means cannot find the target clusters. Single linkage performs well when there is no intercluster density difference. DBSCAN mixes outliers and its performance decreases dramatically when there is intracluster density variation (either random change or smooth change) and clusters are close. NOM can handle data sets having arbitrary shapes, intercluster density differences and intracluster density variations, but it fails when clusters are extremely close or when there is noise. To sum up, each clustering approach has its own weaknesses and strengths depending on the characteristics of the data set taken.

Execution times of competing approaches and each phase of NOM are given in Tables D.3, D.6 and D.9 in Appendix D for selected data sets. Execution times of NOM are significantly higher compared to *k*-means, single-linkage and DBSCAN. It spends much time for GG construction, especially for the data sets having a large number of points. Outlier detection takes less time as it requires only one pass of the entire data set. Merging time increases when the number of closures generated by NC (divided clusters) is higher than the number of target clusters (e.g. data_circle). As the dimensionality of the data set increases, the execution times of NOM increase significantly.

The computational complexity of *k*-means, single-linkage and DBSCAN algorithms are O(kn), O(n^2) and O(n log$n$), respectively, whereas NOM has a higher computational complexity with MO(n^3).



(a) (b) (c) (d)

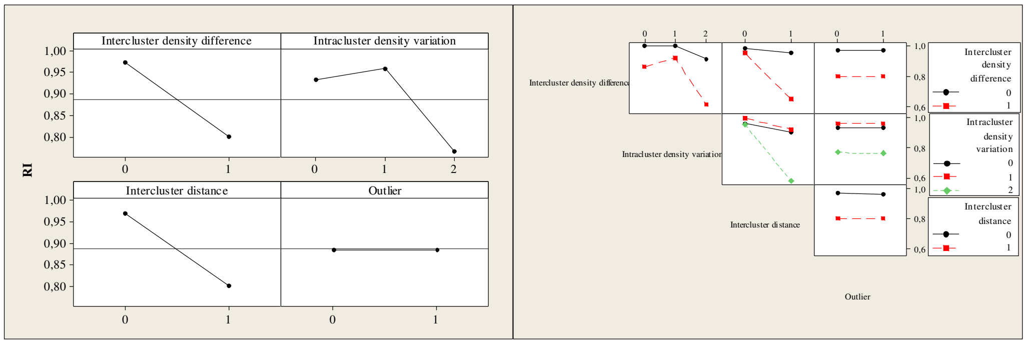
Figure 4.3 Clustering results for data-uc-cc-nu-n: (a) *k*-means, (b) Single linkage, (c) DBSCAN, (d) NOM

Table 4.1 Summary results for group 1 data sets

		***k*-means**	**Single linkage**	**DBSCAN**	**NC**	**Outlier detection**	**NOM**
JI	**average**	0.756	0.937	0.940	0.875	0.875	**0.955**
	std.dev.	0.231	0.163	0.139	0.128	0.137	**0.088**
	min	0.278	0.453	0.504	0.558	0.456	**0.591**
RI	**average**	0.856	0.955	0.963	0.908	0.908	**0.967**
	std.dev.	0.138	0.119	0.095	0.101	0.107	**0.065**
	min	0.580	0.532	0.531	0.659	0.639	**0.648**
QJI	**average**	0.954	0.947	0.972	0.996	0.998	**0.981**
	std.dev.	0.087	0.145	0.097	0.016	0.012	**0.080**
	min	**0.659**	0.460	0.504	0.905	0.916	0.593

Table 4.2 Summary results for group 2 data sets

		***k*-means**	**Single-linkage**	**DBSCAN**	**NC**	**Outlier detection**	**NOM**
JI	**average**	0.858	0.774	**0.877**	0.740	0.739	0.758
	std.dev.	**0.127**	0.255	0.183	0.257	0.257	0.256
	min	**0.623**	0.328	0.559	0.248	0.248	0.247
RI	**average**	**0.962**	0.887	0.960	0.905	0.891	0.886
	std.dev.	**0.036**	0.153	0.060	0.170	0.166	0.196
	min	**0.895**	0.567	0.843	0.412	0.395	0.285
QJI	**average**	0.938	0.789	**0.939**	0.878	0.878	0.867
	std.dev.	**0.041**	0.238	0.119	0.206	0.206	0.218
	min	**0.876**	0.381	0.674	0.305	0.306	0.259



(a) (b)

Figure 4.4 (a) Main effects of factors on RI, (b) Interaction effects

4.4. Discussion of NOM Results

NOM is a new density-based clustering algorithm, which uses graph theoretic concepts such as proximity and connectivity as well as density of points in a data set. It has three phases, namely neighborhood construction, outlier detection, and merging of subclusters. It assumes that the number of clusters is unknown. Compared to some other clustering approaches, one of the advantages of NOM is that no parameters need to be set in the neighborhood construction, and only a single parameter (threshold level a) is needed in the rest of NOM.

NOM is tested on a number of data sets having various properties and compared with some well-known competing approaches. When the intercluster distances are larger than the intracluster distances, NOM is capable of finding clustering solutions close to the target clusters with arbitrary shapes and different densities. Density, distance and connectivity based mechanisms in NOM successfully reveal the local characteristics inherent in the data set. Moreover, NOM can detect the outliers in these data sets although it is not successful with noise. Even in the first phase of NOM, the closures obtained after the neighborhood construction are the same as the target clusters for some data sets. Evaluation of compactness and separation measures relative to the neighborhood densities strengthens the capabilities of NOM in handling arbitrary shapes and density variations.

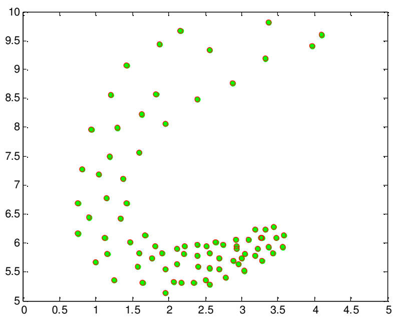
Main limitation of NOM is the lack of collective information from a global perspective. The interrelations among the points are evaluated taking a local view by focusing on the pairings of clusters, and this results in excessive division of target clusters. More information is needed to handle close clusters having intracluster density variations. Besides more globally sensitive mechanisms than the proposed phase 3 of NOM can be developed to test merging of divided clusters. Another complication of NOM is high execution times, but these durations can be reduced using efficient coding schemes.

CHAPTER 5

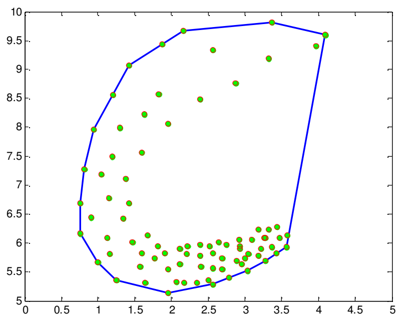
CONSTRUCTION OF EXTERNAL SHAPES

An external shape is the polygonal representation of the boundary of a connected finite set of points. Although this concept is introduced in computational geometry, it is also used in clustering for the representation of clusters and the formation of clusters from closures/subclusters. Firstly, a finite set of points in abstract form are used for clustering purposes, yet generating the external shapes of these points delineates the clustering results. Secondly, discriminative properties of clusters are defined by the points on the boundary or the external shape. As the boundary points of different closures/subclusters are adjacent, the dissimilarity among these boundary points can be explored further to yield the final clusters. The internal points are already connected, and there is little need to consider them during this process. Thus, dealing with the boundary points of subclusters reduces the number of data points to be processed and hence reduces the demanding memory requirements. This data set reduction also strengthens the scalability of a clustering algorithm.

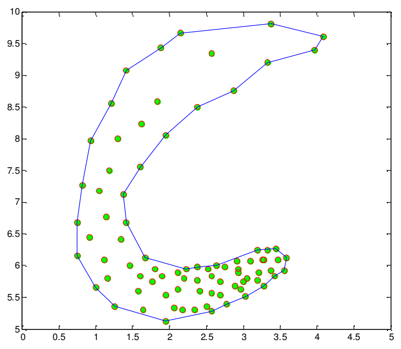
A convex hull represents the external shape of a finite set of points in a particular form. However, the external shape of a cluster may inherently be non-convex in which case we need to find a non-convex hull. Generation of a non-convex hull is relatively more complicated. In fact, there is not a unique non-convex hull of a set of points. Depending on the degree of detail on the boundary, several non-convex hulls can be generated. An example is given in Figure 5.1. Figure 5.1 (a) shows the original data set. The lines connecting the boundary data points in Figure 5.1 (b) define the convex hull. Figures 5.1 (c) and (d) provide examples for a smooth and a ragged (more detailed) non-convex hull. The ragged non-convex hull in Figure 5.1 (d) distinguishes the external shape of the data set better than the ones in Figures 5.1 (b) and (c).



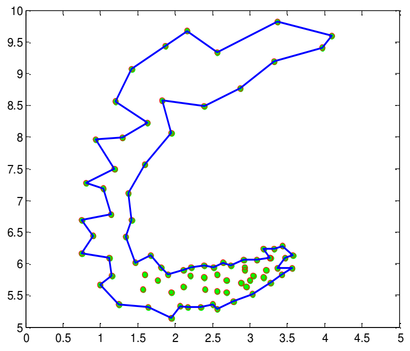
(a)



(b)



(c)



(d)

Figure 5.1 Example external shapes for a set of points. (a) Original data set. (b) Convex hull. (c) A smooth non-convex hull. (d) A ragged non-convex hull.

In this chapter, we focus on generation of external shapes for the purpose of clustering formation. We propose two algorithms, DTC and IS, designed to find the external shape of a finite set of points and extract the external shape (boundary or outline) of given clusters/closures in a data set. The former is a major issue in classical pattern recognition. The latter is particularly useful in finding the external shapes of the closures or (sub)clusters generated by our neighborhood construction algorithm NC. Both of the proposed algorithms are not a substitute for a clustering algorithm as they do not perform any division or merging operations on the existing closures/(sub)clusters.

Section 5.1 introduces the preliminary concepts used in boundary construction. Section 5.2 gives the literature on the external shape construction. The contribution of the work is also clarified in this section. In Section 5.3, the DTC algorithm is explained in detail and some examples are presented. Section 5.4 presents the IS algorithm and its examples. Experimental results for both DTC and IS are presented in Section 5.5. Finally we conclude with the capabilities and limitations of both algorithms in Section 5.6.

5.1. Preliminary Concepts

Triangulation decomposes a polygon, which is composed of a finite set of points, into a set of triangles. In triangulation of a polygon if an edge belongs to a single triangle, it is referred to an *outer edge*, and the edges that are shared by other triangles are *inner edges*. The outer edges form the boundary (*outer frame*) of the polygon. An example is shown in Figure 5.2.

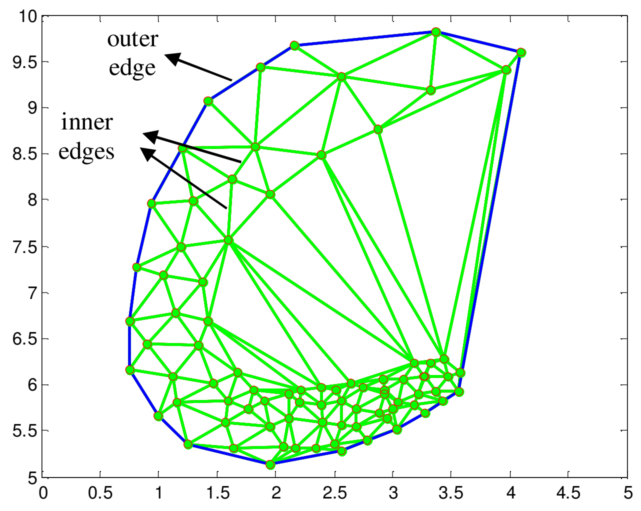


Figure 5.2 Triangulation of a data set.

Delaunay triangulation (DT) is a special triangulation which forms (d+1)-simplices in d-dimensional space. Note that a d-dimensional simplex (d-simplex) is a polytope which is the convex hull of its d+1 vertices. Examples for 2- and 3-simplices are presented in Figure 5.3. DT helps to find the convex hull of a point set and to identify the topological structure of a graph. Main properties of DT are: (1) An edge is inserted between the two points p and q, if the ball having diameter d_{pq} and passing through points p and q is empty. (2) The circumsphere defined for every simplex is empty and does not contain any other points. (3) The nearest neighbor of a given point is always connected to that point by an edge of a simplex (Goodman and O'Rourke, 2004).

Triangle (2-simplex)
Tetrahedron (3-simplex)

Figure 5.3 Examples of 2- and 3-simplices

The non-convex hull of a set of points is not unique as we have shown in Figure 5.1. Thus, the desired degree of detail (smoothness or raggedness) in the external shape needs to be controlled by some parameter(s). In the literature a set of parameters are discussed in Section 5.2 to control the degree of detail.

A boundary is a set of connected points, i.e. each point on a boundary has a degree of at least two. Disconnectivity arises when a point on the boundary does not satisfy this degree condition, and it causes undesired discontinuities on the boundary.

In this chapter, it is assumed that points in the same closure/cluster are connected. Thus, breaking some points off the closure/cluster and forming a new closure/cluster cause disconnectivity.

5.2. Literature Review

In computational geometry there exists a unique convex hull for a finite set of points and there are efficient and fast methods to find the convex hull, such as Quickhull by Barber et al. (1996). In this section we review the non-convex hull generation approaches found in literature.

Non-convex hull generation methods can be classified into two: influence-region based and edge based approaches. In the influence-region based approaches each point has an influence region, such as a ball or a square, and the connectivity of these influence-regions helps to identify the non-convex parts of the shape. In an early work (Richards 1977) a space filling hull is generated using the union of the balls with radius *r*. If the intersection of a pair of balls includes points on the border, an edge is inserted between these points and they constitute the space filling graph which denotes the boundary of the shape. Examples for the space filling hull and the space filling graph are provided in Figures 5.4 (c) and (d) for a θ shaped letter.

A fundamental work (Edelsbrunner et al. 1983) generalizes the convex hull of a point set by using balls with radius $1/\alpha$. The real valued parameter α gives the degree of detail in the non-convex shape. For positive α, the external shape is formed by taking the intersection of the balls with radius $1/\alpha$, and it includes only some obvious extreme points. The disk becomes a halfplane with $\alpha = 0$, and the external shape is equivalent to the intersection of these halfplanes, i.e. the convex hull. The complement of the ball with radius $-1/\alpha$ is considered for the negative values of α. As α goes to $-\infty$, the shape converges to the point set itself. α-shape has deficiencies for the point sets having density variations because a single α setting is used. As a remedy, weighted α-shape is introduced (Edelsbrunner 1992), and the flexibility is ensured by defining a specific weight for each point. In the same work the relationship between the space filling hull and the weighted α-shape is explored in detail and it is shown that these two approaches are equivalent for a certain weight setting.

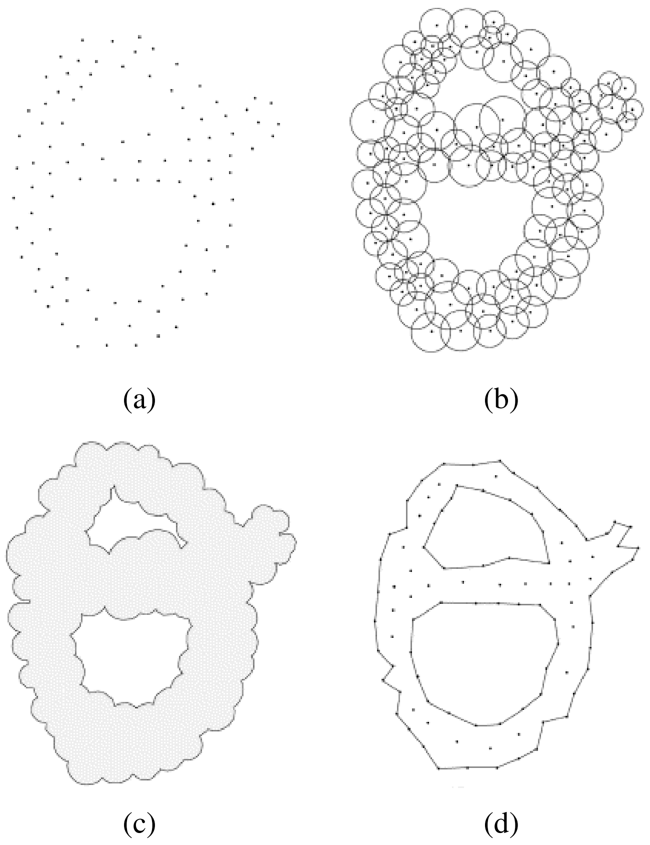


Figure 5.4 An example for space filling hull and space filling graph. (a) Data set. (b) Weighted balls for the data set. (c) Space filling hull. (d) Space filling graph. (Melkemi and Djebali 2001)

In the *s*-shape proposed by Chaudhuri et al. (1997) a point set is partitioned into square grids with side length *s*. A procedure is proposed to optimize the parameter *s*. The external shape is obtained by taking the union of these square grids, hence the shape includes zigzags. In order to overcome this limitation the *r*-shape is introduced. The points are classified as *r*-interior or *r*-extreme points using the balls with radius *r*. Edges are inserted between *r*-extreme points if the boundaries of the balls with radius *r* have a nonempty intersection on the boundary of the non-convex hull. In this work, it is also shown that the *r*-shape is a subgraph of the α-shape.

In the edge based approaches usually an initial convex hull of the shape is generated using either the Voronoi diagram or its dual method DT. Then, the edges on the convex hull are deleted or combined, or new edges are inserted according to certain rules.

Split and merge procedure proposed by Garai and Chaudhuri (1999) is one of the edge based procedures. Starting with the initial convex hull, the splitting algorithm deletes the edges longer than a certain multiple of the average edge length in a neighborhood. The shape obtained after splitting may include zigzags, so merging operations are performed on these zigzags to obtain a smooth boundary. In a certain locality of a shape, points subject to merging form a polygon and the obtuse angles in the polygon are considered for merging. The number of outer edges on a polygon and the polygon area are associated with the degree of detail in the external shape, so two thresholds are determined considering the desired degree of detail. Hence, merging is carried out if the newly formed polygon satisfies these thresholds.

In order to obtain the A-shape (Melkemi and Djebali 2000) first Voronoi diagram and DT of the original point set are constructed, followed by construction of an artificial point set A. In fact, point set A comes with a parameter that determines the degree of detail in the non-convex hull. If the original points and points in A are contained in the same triangle in DT (i.e. if they are neighbors), then the edge between the original points in DT becomes a boundary edge. Weighted A-shape (Melkemi and Djebali 2001) gives a weight to each point in the data set to handle the density variations.

Alani et al. (2001) use the Voronoi diagram to approximate the spatial regions in geographic information systems. In addition to the original data set, an extra data set lying outside the original points need to be supplied. As in the neighborhood idea used by Melkemi and Djebali (2000) the Voronoi cells having neighbors from the additional data set constitute the boundary of the region.

Taking the DT as the initial shape, the χ (chi) algorithm (Duckham et al. 2008) removes the longest edges if the edge is longer than a prespecified threshold and the degree of each point is at least two. In this method setting of the length threshold is crucial in order to have the desired details in the non-convex shape.

Most of the literature we have discussed up to now assume 2-dimensional space. Edelsbrunner and Mücke (1992) extend the α-shape to 3-dimensional space, and Melkemi (2003) proposes the 3-dimensional A shape. Some authors such as Duckham et al. (2008), Chaudhuri et al. (1997), and Garai and Chaudhuri (1999) claim that the proposed approach in their work can be generalized to higher dimensions. However, to the best of our knowledge, there is not an actual implementation of these approaches for higher dimensions.

In order to control the degree of detail in a non-convex shape, a parameter is used almost in all methods in the literature. This makes the correct parameter setting very crucial. Most of the methods propose a procedure to set this parameter properly, but the procedure is still affected by the data set properties such as density variations and characteristics of the non-convex parts.

In this work, a parameter-free approach, namely DT Cropping (DTC) is proposed for generation of non-convex hulls. DTC is an edge based approach and a modified version of χ algorithm by Duckham et al. (2008). DTC can be used to construct the external shape of a finite set of points and to find the external shape of each cluster/closure of points. Use of DTC is limited to 2-dimensional space.

In higher dimensional space there is very limited work in external shape construction. We propose a new edge based approach, namely the Ideal Simplex (IS). IS also starts with the DT construction. The idea behind IS is that in the ideal case the simplices in a DT are expected to have equilateral faces, and an elongation is a potential indicator for non-convexity in the point set. The elongation of a simplex on the boundary is compared with an elongation threshold which controls the degree of detail of an external shape.

5.3. External Shape Construction in 2-dimensional Space

The external shape construction algorithm DTC is based on two main ideas. Points that belong to different closures/clusters but are in the same simplex (a triangle in 2-dimensional space) in DT are on the boundary. This idea is also used by Melkemi and Djebali (2000) and Alani et al. (2001). The second idea is the deletion of longer edges on the boundary if there exists a path between the two points that has all shorter edges. In fact, Duckham et al. (2008) sets out from a similar idea but the edge removal is based on a prespecified parameter in their work. That is, in their proposal an edge is removed if it is longer than a given threshold. The main difference of our work is that DTC is a parameter-free approach. Before proceeding further, basic definition of DT is presented below.

5.3.1. The Delaunay Triangulation Cropping (DTC) Algorithm

Let D be a finite set of points in 2-dimensional space. Suppose that D is composed of *c* disjoint clusters/closures. Given the cluster labels, DTC tries to find the external shape of each of the clusters. Note that D may include only one cluster (*c* = 1) and this case is equivalent to finding the external shape of the entire set of points.

The steps of DTC are as follows.

Step 1. Construct DT for D.

Step 2. Determine the edges on the initial boundary of each cluster by applying the two conditions below.

> Condition 1. If an edge in DT belongs to a single triangle and is not shared by other triangles, then this edge is on the boundary.
>
> Condition 2. If a triangle in DT includes points from different clusters, each of these points is on the boundary of the corresponding cluster. The edge of this triangle between the two adjacent points that are in the same cluster is a boundary edge.

Step 3. For every cluster in D find the non-convex parts of the boundary.

Step 3.1. If all the edges on the current boundary have been evaluated and no edge is deleted, terminate the algorithm. Otherwise, select the (next) longest edge (*e*') on the boundary that has not been evaluated yet.

Step 3.2. Check if both end points of *e*' have a degree higher than two. If yes mark *e*', go to Step 3.3. Otherwise, go to Step 3.1.

Step 3.3. In the triangle that has *e*' as an edge, if both of the other (inner) edges are shorter than *e*', then delete *e*'. Instead, let the two inner edges be on the boundary. Return to Step 3.1.

In Step 1, the DT and the convex hull of the point set is constructed using the Quickhull algorithm (Barber et al., 1996). Step 2 finds a rough initial boundary, which is not necessarily convex. Condition 1 specifies the points on the outer frame of the data set whereas Condition 2 determines the points on the boundaries of two clusters facing each other with simplices running across. Step 3 fine-tunes the external shape by detecting further non-convex parts of the boundary. The longest edge on the initial boundary (*e*') is selected as a candidate for deletion. The degrees of the end points of *e*' are then checked. Both end points should have a degree higher than two so that they are still connected in case *e*' is deleted. Otherwise, the edge deletion disrupts the connectivity and subclusters may occur. In this case, *e*' is not considered further for deletion. If *e*' satisfies the degree constraint, a last check for edge deletion is conducted. That is, if both inner edges of the triangle to which *e*' belongs are shorter than *e*', then it is possible to connect the end points of *e*' in an indirect manner using these two edges both shorter than the direct distance between them. Thus, *e*' is deleted and the two inner edges are taken as the new boundary edges. This results in a finer non-convex boundary.

An external shape construction example for a data set with a single cluster is shown in Figure 5.5. Figure 5.5 (a) is the result of Step 1, the DT of the point set. Next, Condition 1 is applied in Step 2. If data set includes only one cluster, Condition 2 is not used. At the end of Step 2 the dark points in Figure 5.5 (b) become the initial boundary points. An edge deletion example in Step 3 is provided in Figure 5.5 (c). Edge *e'* is eliminated and the inner edges become the new edges on the boundary. Edge deletion operations are conducted until no edges remain to be removed. In Figure 5.5 (c), in the following iterations, edges *a*, *b*, *c* and *d* are deleted in addition to edge *e*'. Dark points in Figure 5.5 (d) show the final boundary points found by DTC.

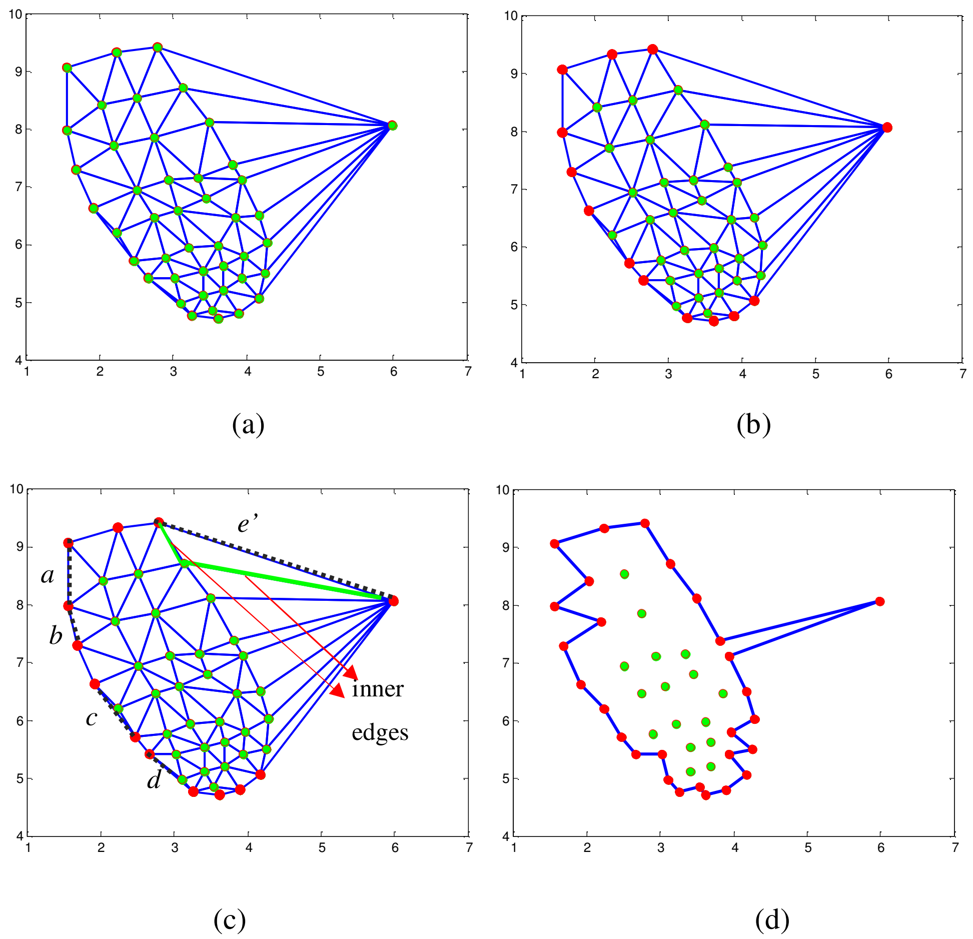


Figure 5.5 External shape construction example for a data set with a single cluster in 2-dimensional space. (a) DT constructed in Step 1 of DTC. (b) Initial boundary (dark points) found in Step 2 of DTC. (c) Finding non-convex part of the boundary in Step 3 of DTC. (d) Final boundary found by DTC.

The boundary points found by DTC in an example data set with three clusters are shown in Figure 5.6. The data set has two spiral shaped clusters and intracluster density variation. Due to the density variation, the number of boundary points found by DTC is large and there exist zigzags on the external shape. Note that this non-smooth structure is observed especially for the regions with density variation.

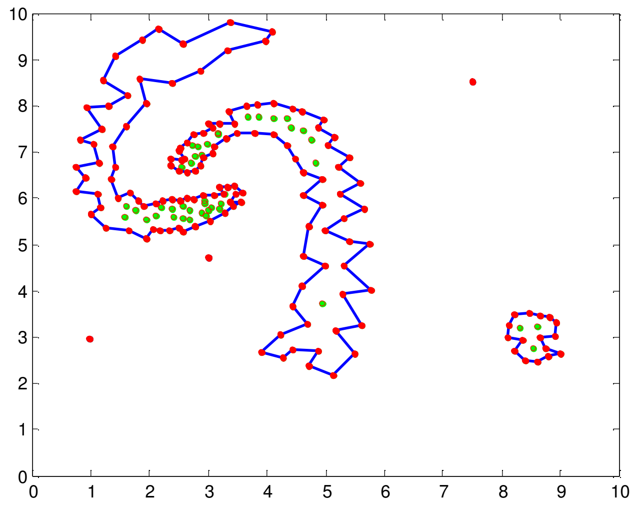


Figure 5.6 External shape (dark points) for a data set with three clusters in 2-dimensional space.

5.3.2. Time Complexity of DTC

2-dimensional DT construction in Step 1 has a time complexity of O(*n* log*n*). Step 2 determines the edges on the boundary by checking all the triangles in the DT. As the maximum number of edges in a DT is bounded by 3*n*, the worst-case time complexity of Step 2 is O(*n*). Deletion of every boundary edge is considered in Step 3 and deletion operations are conducted until there is no improvement. Thus, it is not possible to determine a bound for the time complexity of Step 3. Although Steps 1 and 2 have an overall time complexity of O(*n* log*n*), it is not possible to infer the overall time complexity of DTC.

5.4. External Shape Construction in *d*-dimensional Space

The IS algorithm generates the external shape of a finite set of points in *d*-dimensional space where $d \geq 2$. It is based on the adjacency information gathered from DT and it uses the idea introduced in DTC such that if a simplex is composed of points from different clusters, these points are on the boundary of the closures/clusters they belong to. In addition to this, DT avoids “elongated” simplices with unbalanced edge lengths as it maximizes the minimum angle in the simplices. For this reason, elongated simplices are used in IS to detect the non-convex regions. Elongation thresholds are used to control the degree of detail in non-convex parts of the boundary.

5.4.1. Elongation Measures

In this work, two versions of elongation are considered.

Case 1. Elongation due to the inner angles of a simplex.

Inner angles of a simplex may be distorted compared to the ideal case and this distortion causes obtuse angles in the simplex. In this case elongation is measured as the ratio between the distance of the farthest vertex and the distance of the closest vertex to the center of gravity of the simplex. Let simplex$_s$ be the set of vertices of simplex s, g be the center of gravity of the simplex, and d_{ij} be the Euclidean distance between points i and j. Then, the first elongation measure is calculated as $em_1 = \dfrac{\max\limits_{i \in \text{simplex}_s} \{d_{ig}\}}{\min\limits_{i \in \text{simplex}_s} \{d_{ig}\}}$.

Case 2. Elongation due to the edge lengths of a simplex.

Elongation may cause large differences in edge lengths of a simplex and, instead of obtuse angles, acute inner angles may emerge. Elongation is then measured as the ratio of the maximum edge length to the minimum edge length in the simplex. That is, the second elongation measure is $em_2 = \dfrac{\max\limits_{i \neq j \in \text{simplex}_s} \{d_{ij}\}}{\min\limits_{i \neq j \in \text{simplex}_s} \{d_{ij}\}}$.

For 2-dimensional space the ideal triangle (simplex) and two elongation examples are presented in Figure 5.7.

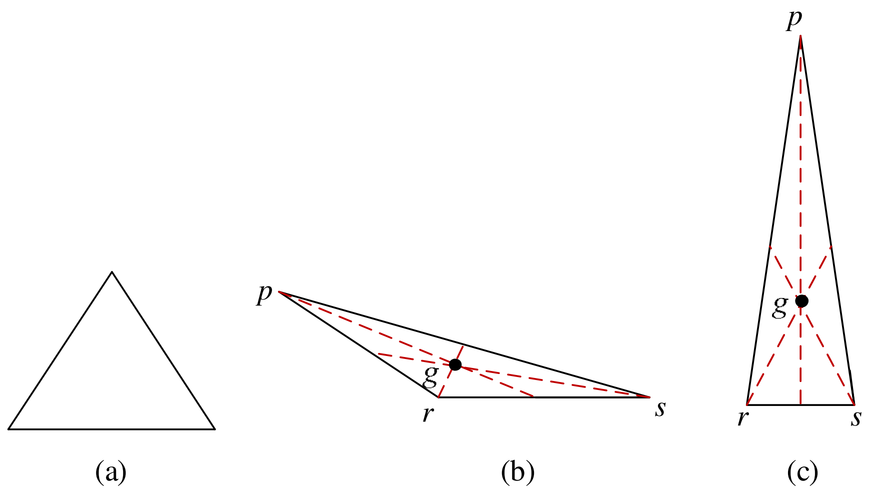


Figure 5.7 Elongation example. (a) The ideal case of an equilateral triangle. (b) Elongation due to an obtuse inner angle of a simplex, $em_1 = \frac{d_{pg}}{d_{rg}}$. (c) Elongation due to the edge lengths, $em_2 = \frac{d_{pr}}{d_{rs}}$.

5.4.2. Ideal Simplex (IS) Algorithm

Let D be a finite set of points in *d*-dimensional space where $d \geq 2$. Suppose that D is composed of *c* disjoint closures/clusters. Given the cluster labels, IS tries to find the external shape of each cluster. Main steps of the IS algorithm are given as follows where the elongation thresholds for em_1 and em_2 are $threshold_1$ and $threshold_2$, respectively.

Step 1. Construct the set of artificial points A (to be explained later) for D.

Step 2. Construct DT for the union of D and A.

Step 3. Determine the edges on the current boundary of each cluster (to be explained later).

Step 4. Find the non-convex parts of the boundary.

Step 4.1. Find the elongated simplices on the current boundary by applying the following two conditions.

Condition 1. A simplex is elongated if $em_1 > threshold_1$.

Condition 2. A simplex is elongated if $em_2 > threshold_2$.

Step 4.2. Place an artificial point at the center of gravity of each simplex found elongated, provided the simplex is not blocked (to be explained later). Add this point to new (temporary) set of artificial points TA.

Step 5. Construct DT for the union of D, A and TA.

Step 6. Check the feasibility of the newly added artificial points in set TA. Delete the artificial points that are isolated or that cause isolation of original points (to be explained later). Block the simplices that contain deleted artificial points.

Step 7. If set TA is not empty, update set A as the union of A and TA, and return to Step 2. Otherwise, terminate the algorithm.

Edge Identification

In Step 1, a least volume hypercube is constructed to enclose the original data set. The hypercube is represented with its vertices and midpoints of its faces. Thus, 3^d-1 artificial points are placed in set A to define the hypercube where *d* denotes the number of dimensions.

DT of the original points and artificial points is constructed in Steps 2 and 5 using the Quickhull algorithm (Barber et al., 1996).

Step 3 finds the boundary edges considering the following four cases for the simplices in DT.

Case 3.1. If the vertices of a simplex are original data points from the same cluster, then all these points are interior points.

Case 3.2. If the vertices of a simplex are original data points from different clusters, then these points are on the boundary of the clusters they belong to.

Case 3.3. If the vertices of a simplex are an artificial point and original data points from different clusters, then the original points are on the boundary of the clusters they belong to.

Case 3.4. If the vertices of a simplex are artificial point(s) and original point(s) from the same cluster, the original points are on the boundary of the cluster they belong to.

Blocking Simplices

Step 4 finds the non-convex parts of the shape by adding artificial points inside the elongated simplices. The two elongation measures discussed previously are calculated for each simplex on the current boundary. For a simplex, if either of these measures is greater than the given thresholds, a new artificial point at the center of gravity of the simplex is inserted.

IS finds the boundary of given clusters and is not allowed to divide the clusters. In doing this, IS should not allow disconnectivity or isolation of boundary points. If insertion of a new artificial point inside an elongated simplex was found to cause disconnectivity in a previous iteration, that simplex has been blocked in step 6. In this case, even though it is classified as elongated, a new artificial point is not added inside this simplex, and consequently its designated boundary edges are maintained. The blocking of such simplices prevents the algorithm from repeating the same disconnectivity checks during the iterations. Thus, in Step 4, the artificial point is inserted only if the elongated simplex under consideration is not blocked.

In Step 5 DT is reconstructed using both the original data set and all the artificial points added up to then.

Step 6 checks the feasibility of new artificial points in the DT. We extract the boundary of each cluster assuming that the original data points in the same cluster form a connected graph. That is, clusters as they stood in the original data set are not allowed to be divided. There are two cases that might cause disconnectivity, hence lead to undesired cluster divisions.

Case 6.1. Isolation of an original point: An original point should be connected to at least two other original points, otherwise it is isolated. If this condition is violated, then the artificial points adjacent to the original point are reexamined and the one nearest to the original point is deleted from set TA to ensure connectivity. An example in 2-dimensional space is shown in Figure 5.8. The original point γ becomes isolated. In this case, the artificial point α is deleted from TA so that the original point γ preserves its connectedness to the original point β.

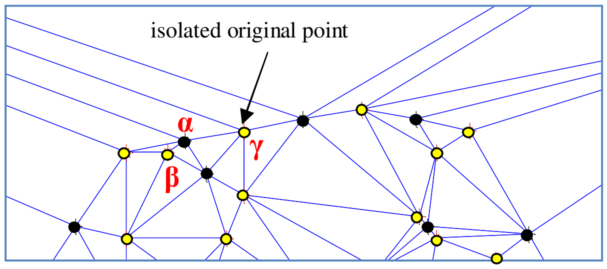


Figure 5.8 An example for the isolation of an original point (light and dark circles denote the original and the artificial points, respectively)

Case 6.2. Isolation of an artificial point: Artificial points carve out the exterior of the data set to discover the non-convex segments. Hence, an artificial point in set TA should be adjacent to at least another artificial point added up to then so that non-convex parts in the exterior of the data set are chipped gradually. If an artificial point inserted in Step 4 is surrounded only by the original points, it means that artificial point forms a hole in the data set. To avoid such disconnectivities, adjacent neighbors of each artificial point added in the current iteration are examined. If all these neighbors are original points, then the artificial point is deleted from set TA. Figure 5.9 shows an example in 2-dimensional space. Point ε will be removed from set TA because if not removed an undesired hole occurs in the data set.

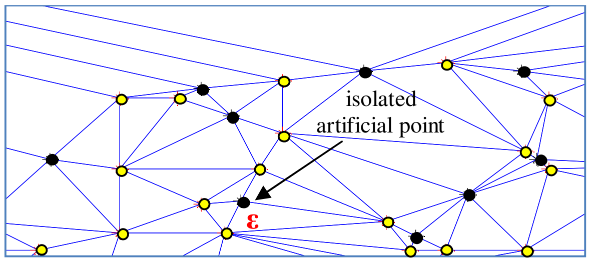


Figure 5.9 An example for the isolation of an artificial point (light and dark circles denote the original data points and the artificial points, respectively)

In Step 6, if a new artificial point does not satisfy the above conditions and is deleted, its simplex is added to the set of blocked simplices.

After deletion of the artificial points that disrupt connectivity, Step 7 checks the termination condition. The IS algorithm terminates with the boundary points found in Step 3. Otherwise, with the newly added artificial points the algorithm continues with Step 2.

5.4.3. Parameter Settings in IS

Elongation parameters (*threshold*$_1$ and *threshold*$_2$) in IS control the degree of detail in the external shape. In an ideal simplex with equilateral 2-faces, the two elongation measures, em_1 and em_2, take the value of 1. Thus, elongation thresholds greater than 1 are considered.

The effect of the elongation thresholds on the external shape generation are tested with two parameter settings: 2 (low) and 2.5 (high). In Figures 5.10 and 5.11 examples of external shapes generated by IS with these two settings are shown. As Figures 5.10 (b) and 5.11 (b) show the face of shape S in 2-dimensional view, all the points except the dark circles are on the boundary. In both figures the dark circles represent the points that are expected to be on the boundary but not found by the IS algorithm. These missed boundary points give rise to unnecessary carving of the shapes at their boundaries. IS misses more boundary points with the high setting of the elongation thresholds as seen in Figure 5.11, whereas the “corners” of shape S become more visible in the low setting as in Figure 5.10. When we calculate the Jaccard Index (JI) and the Rand Index (RI) for the expected versus boundary points, the JI value is 0.86 for both settings whereas the RI value is slightly higher, 0.90 for the low setting and 0.91 for the high setting, respectively. These mean that targeted boundary points occur in 90% of the total points identified as boundary in either settings, and with a high setting boundary-non-boundary discretion is successful in 91% of chosen points. Although the JI and RI values do not differ significantly, the percentage of the points correctly labeled as the boundary points (0.99) is slightly better with the low setting than the same figure (0.97) with the high setting. This is an indicator of high setting potentially missing true boundary points. IS generates these external shapes in 26.12 seconds and 22.49 seconds with the low and the high settings, respectively. As expected, more simplices are categorized as elongated with the low setting and the run time increases.

For higher accuracy in external shape generation, we set both elongation thresholds to 2 from now on.

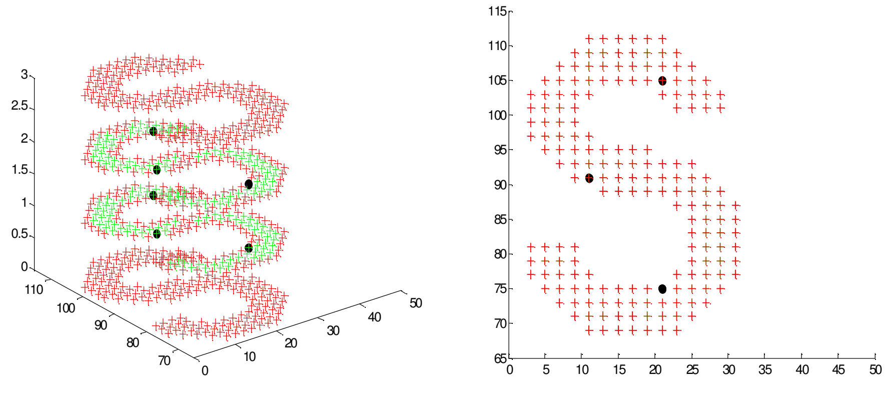

(a) (b)

Figure 5.10 An example external shape generated by IS with *threshold*$_1$ = *threshold*$_2$ = 2. (a) shape S in 3-dimensional view, (b) face of shape S in 2-dimensional view (bold + signs denote the boundary points found, dark circles denote the missing boundary points)

5.4.4. Time Complexity of IS

For a data set having n points, the worst case time complexity of Step 1 is O(n log$n$). Steps 2 and 5 are governed by the DT construction having time complexity O(n log$n$) for $d \leq 3$ and O($n^{\lfloor d/2 \rfloor} / \lfloor d/2 \rfloor !$) for $d \geq 3$. In the first iteration, DT is constructed from scratch. In the subsequent iterations, the use of an incremental DT construction method is possible, as new artificial points are added. Steps 3 and 4 are performed for every simplex in DT, hence their worst-case time complexity can be calculated as the maximum number of facets of the polytope forming the data set which is O($n^{\lceil d/2 \rceil}$). Since Steps 2-7 are repeated until no new artificial point is added, we cannot determine the overall time complexity of the algorithm, but we can conclude that it is M$O(n^{\lfloor d/2 \rfloor} / \lfloor d/2 \rfloor !)$ where M denotes the number of repetitions of the repeat-until loop. Note that the Quickhull algorithm can handle data sets up to 9 dimensions so our implementation of IS is limited with 9-dimensional data sets.

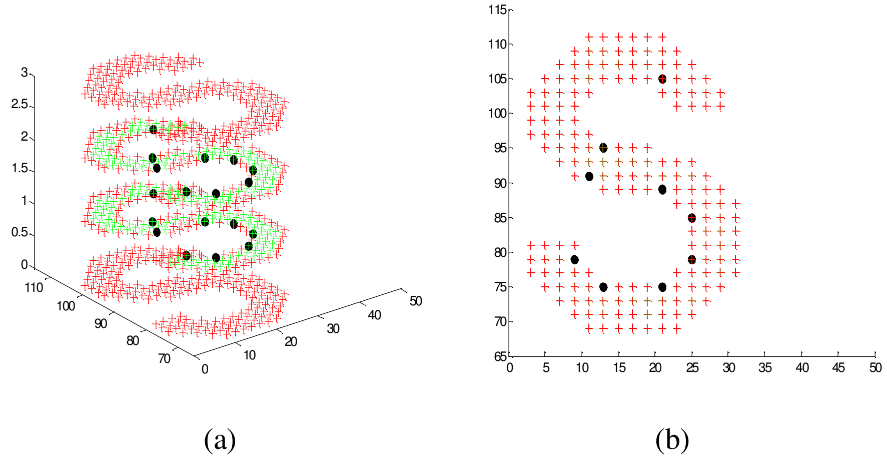


Figure 5.11 An example external shape generated by IS with *threshold*$_1$ = *threshold*$_2$ = 2.5. (a) shape S in 3-dimensional view, (b) face of shape S in 2-dimensional view (bold + signs denote the boundary points found, dark circles denote the missing boundary points)

5.5. Experimental Results for the DTC and IS Algorithms

The DTC and IS algorithms are coded in Matlab. We test the performance of the algorithms using the data sets given in Appendix B. In addition to these, we also use data sets composed of single letters borrowed from our group 2 data set D_0000. We first illustrate the performance of the two algorithms in finding the target boundary on examples of single letters. Then, we report the data set reduction performance of the algorithms for all data sets given in Appendix B.

5.5.1. Finding the Target Boundary

Our first aim is to compare the results of the algorithms with the target boundary. For each data set the points that constitute the external shape are determined by visual inspection, and the target points are labeled as such. Thus, the performance of the algorithms is tested by comparing the target boundary points with the boundary points found by DTC and IS. Three performance measures are used for this purpose, namely JI, RI and the ratio of the number of true points found on the boundary by the proposed algorithm to the number of true points on the target boundary (RPT).

Figure 5.12 shows the external shapes generated by DTC and IS in 2-dimensional space for an S-shaped data set. Figures 5.12 (a) and (b) display that DTC cannot discover the upper and lower circular cavities of letter S denoted by dark circles. DT fills the circular space in the cavities by crossing triangles. The inner edges of each such induced triangle are longer than the current edge on the boundary and the current edge is not deleted by DTC. Thus, DTC stops early on at the beginning of both cavities. Consequently, the circular cavities remain undetected. On the other hand, using the elongation idea instead of edge lengths and adding new artificial points (those digging into the cavity), IS can detect and carve out the circular cavities in the external boundary as seen in Figures 5.12 (c) and (d). In this example, the number of elongated triangles that have been found feasible (unblocked) is 77.

Table 5.1 indicates that IS outperforms DTC in all performance measures but time. The run time of IS is significantly longer than that of DTC, but IS can find the target boundary with the exception of a single point.

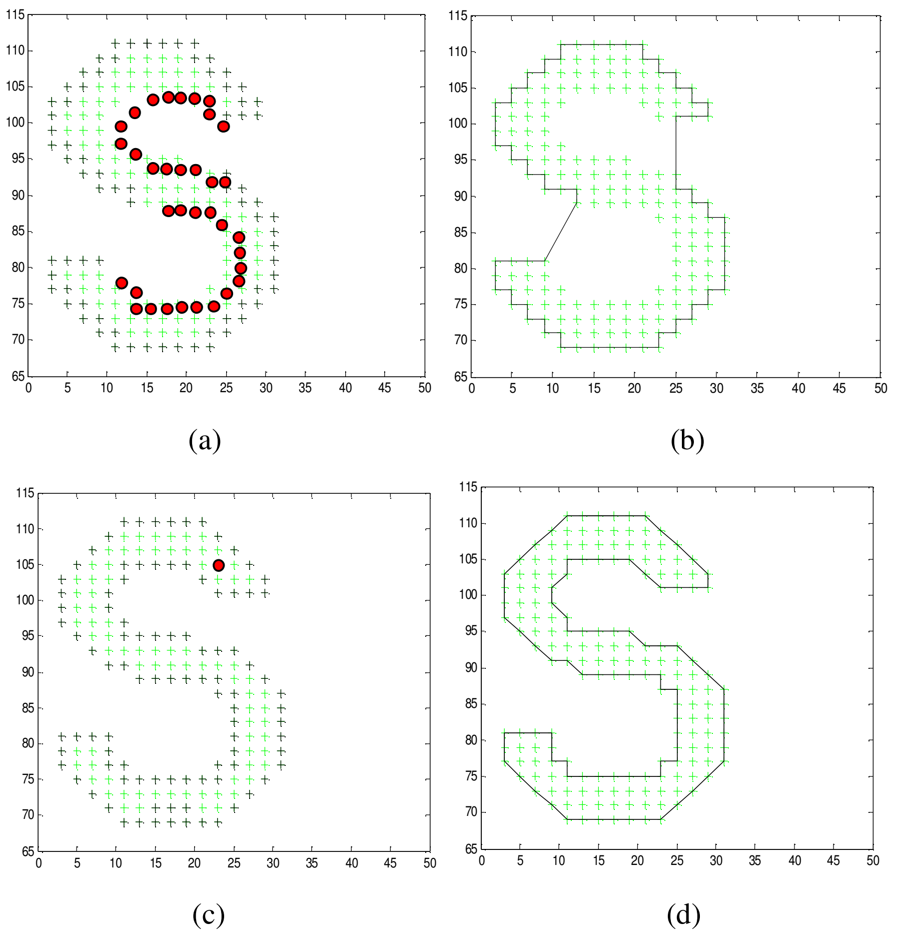


Figure 5.12 External shape examples for an S-shaped data set in 2-dimensional space. (a)-(b) external shape generated by DTC. (c)-(d) external shape generated by IS (bold + signs denote boundary points, dark circles denote the missing boundary points).

Table 5.1 Performance of DTC and IS for 2-dimensional S-shaped data set

Algorithm	**# of points in the data set**	**# of points on the target boundary**	**# of points found on the boundary**	**JI**	**RI**	**RPT**	**Time (sec.)**
DTC	179	87	76	0.39	0.56	0.81	0.24
IS	179	87	94	0.82	0.91	0.99	27.27

DTC is designed for only 2-dimensional space, so in 3-dimensional space the external shapes are generated by IS only. External shapes for 3-dimensional A-shaped and E-shaped data sets are shown in Figure 5.13.

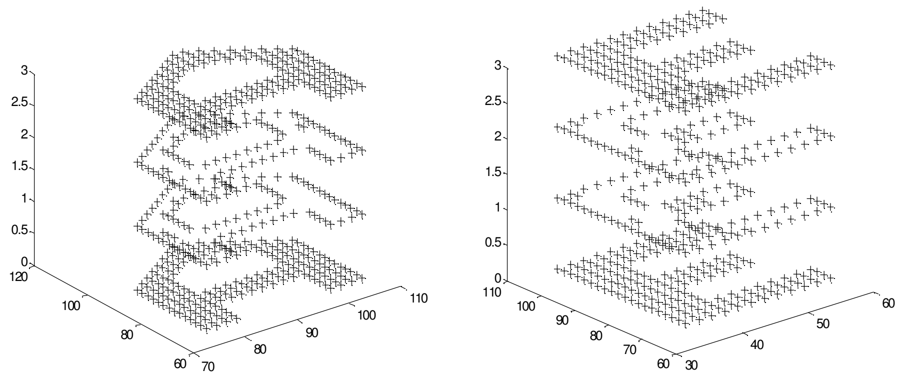


Figure 5.13 External shape examples generated by IS in 3-dimensional space. (a) external shape of the A-shaped data set. (b) external shape of the E-shaped data set (bold + signs denote the boundary points).

Table 5.2 shows the performance of IS on 3-dimensional data sets. Although JI and RI values are not 1, RPT values are equal to 1 and the points on the target boundary are discovered correctly. In fact, the external shape found by IS includes more points than the target boundary. For example, in Figure 5.14 (b) the six points marked with dark circles are identified on the boundary by IS, when in fact they do not lie on the target boundary. The low setting on the threshold for detecting elongation (2.0 in this case) is the reason for these extra zigzags on the boundary.

Table 5.2 Performance of IS for 3-dimensional data sets

Data set	# of points in the data set	# of points on the target boundary	# of points found on the boundary	JI	RI	RNPNT	Time (sec.)
data_S	716	530	555	0.86	0.90	0.99	197.07
data_A	760	558	578	0.92	0.95	1.00	218.86
data_E	672	504	524	0.91	0.94	1.00	155.07
data_O	704	520	564	0.83	0.88	1.00	285.10

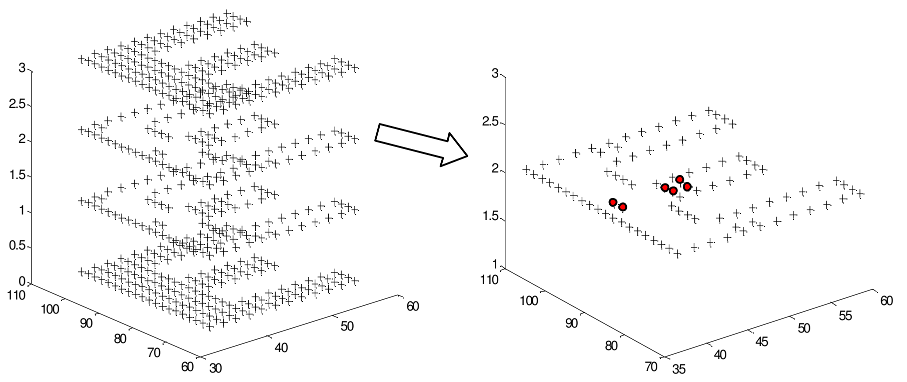


Figure 5.14 External shape generated by IS for the 3-dimensional E-shaped data set. (a) 3-dimensional view (b) 2-dimensional view (bold + signs denote the boundary points found, dark circles denote the points that should not be on the boundary).

5.5.2. Data Set Reduction Performance

We also examine the performance of DTC and IS for all the data sets in terms of the total number of points in the data set, the total number of points found on the boundary, the time, and the percentage of reduction in the data set. DTC is tested only on 2-dimensional group 1 data sets introduced in Appendix B. Experiments with IS are conducted on group 1, group 2 and group 3 data sets.

The results for individual data sets are presented in Appendix E. For groups 1, 2 and 3 data sets these results are summarized in Table 5.3. Compared to DTC, IS provides more reduction in the number of points, but it spends more time on the average.

Table 5.3 Summary of the results for data sets

		Group 1		Group 2	Group 3
		DTC	**IS**	**IS**	**IS**
% Reduction	**Average**	42.96	52.13	21.12	15.97
	Min	1.52	3.03	7.70	9.96
	Max	74.29	82.86	27.77	19.07
Time (sec.)	**Average**	17.68	2004.57	9720.60	190.37
	Min	0.12	0.38	1550.63	41.72
	Max	87.07	14057.06	55914.38	502.43

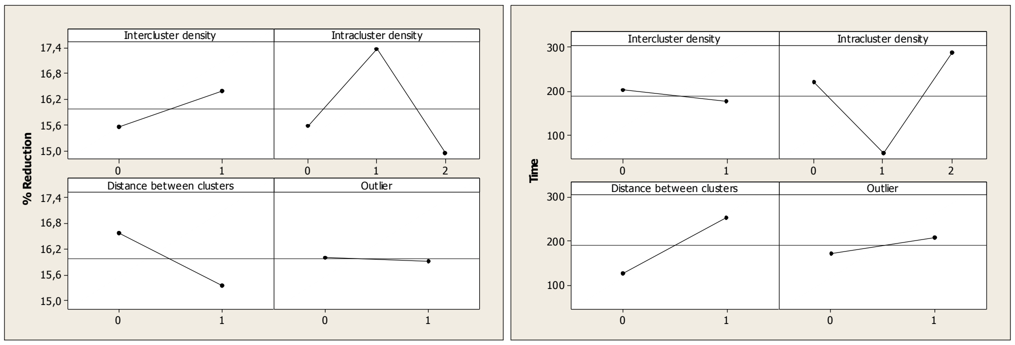
When we compare the performance of DTC and IS with group 1 data sets, DTC finds more points on the boundary than IS (seen in Tables E.1 and E.2). Thus, the percentage of reduction in the total number of points is less for DTC than IS. The differences in sizes of boundary point sets are larger especially for the data sets with density variations. As DTC is parameter-free, it eliminates all the outer edges on the boundary that are longer than the inner edges. However, in IS in order to eliminate an edge, the maximum edge length of a triangle should be at least twice as much as the minimum edge length, or the maximum distance between a point and the center of gravity of a triangle should be at least twice as much as the minimum distance between a point and the center of gravity. These make the edge elimination relatively easier in DTC and generate more boundary points especially for the data sets having density variations. In summary, the elimination criterion in DTC is more relaxed than the criteria in IS.

As Tables E.1 and E.2 in Appendix E show, run times of IS are in general significantly longer than those of DTC for group 1 data sets. For example, the run time for data-uc-cc-nu-n is 1.67 seconds with DTC and 5.53 seconds with IS. Construction of DT several times and feasibility checks take more time in IS. Nevertheless for the data sets in which IS yields significantly more reduction relative to DTC, IS spends less time. For example, DTC results in a 44.55% reduction in dataX in 2 seconds whereas the percentage of reduction in IS is 70.30% in 0.38 seconds. This result is reasonable as DTC eventually generates significantly more points on the boundary.

Factor Effects

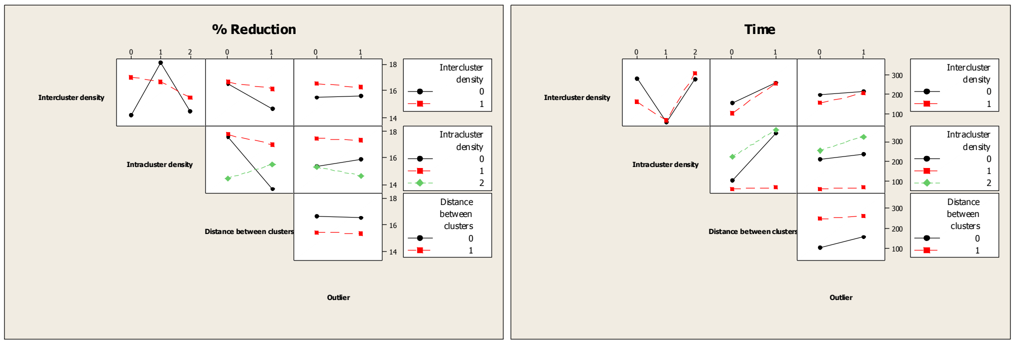
We use a full factorial experimental design for group 2 and group 3 data sets to explore the effects of four factors: intercluster density difference, intracluster density variation, distance between clusters and existence of outliers. Detailed results for these data sets are provided in Tables E.4 and E.5 in Appendix E. For group 3 data sets, the main and interaction effect plots in terms of percentage of point reduction and time are presented in Figures 5.15 and 5.16, respectively. As seen in Figure 5.15, all the factors except existence of outliers affect the reduction percentage significantly. No intercluster density difference and no or smooth intracluster density variation worsen the reduction percentage, whereas higher distance between clusters, presence of intercluster density difference and random intracluster density variation improves the reduction percentage. Actually, random deletion decreases the total number of points, so combining the effects of the proposed algorithm and random intracluster density variation, the percentage of reduction significantly improves. Intercluster density difference and existence of outliers do not significantly affect the time spent for external shape formation. For no or smooth intracluster density variation and lower distance between clusters the time spent increases.

According to Figure 5.16 there exists significant interaction between the intercluster density difference and the intracluster density variation, as well as the intracluster density variation and the distance between clusters in terms of percentage of reduction. That is, when there exist both intercluster density difference and intracluster density variation (i.e. random and smooth), the percentage of reduction decreases. However, when there exist random intracluster density variation and no intercluster density difference, the percentage of reduction improves. Besides, close clusters worsen the percentage of reduction for no and random density variation, whereas the percentage of reduction increases for the combination of smooth density variation and close clusters.



(a) (b)

Figure 5.15 Main effect plots for group 3 data sets. (a) % reduction. (b) time.



(a) (b)

Figure 5.16 Interaction effect plots for group 3 data sets. (a) % reduction. (b) time.

5.6. Discussion of the Results for DTC and IS

The purpose of both DTC and IS algorithms is to discover non-convex external shapes. In fact, they have mainly two functions: finding the external shape of a finite set of points and finding the external shapes of the clusters/closures in a data set.

DTC works only on 2-dimensional data sets. In 2-dimensional space, elimination of an edge is equivalent to elimination of a facet. However, it is difficult to adapt the same edge elimination idea to higher dimensions since a facet of a simplex in higher dimensional space is composed of several edges, and all the edges forming the facet need to be checked to test the possibility of elimination. This brings about an exponential increase in time complexity to form the desired non-convexity, and may not prove to be functioning right all the time. Hence we limit the use of DTC to two dimensions.

DTC is successful in the generation of the external shapes when the edge length in the clearing of a non-convex cavity is larger than the width or diameter of the cavity (that is the cavity “widely open”). In this case, DT forms triangles such that the inner edges in the cavity are shorter than the outer edge of the triangle (crossing the clearing of the cavity). Thus, elimination of the outer edge works properly. As DTC is a parameter-free algorithm, the external shape found is not necessarily smooth and may include excessive zigzags due to excessive carving. Such non-smooth boundaries are observed especially for the data sets with density variations. The advantage of this non-smooth pattern is that it helps increase the accuracy in representing a shape. However, number of points on the boundary becomes excessive as well.

IS is designed for generation of external shapes in higher dimensional spaces. In IS the degree of the detail of the external shape is controlled by the elongation thresholds. The experiments with IS on 2- and 3-dimensional data sets show that the proposed algorithms are satisfactory in the generation of external shapes. The number of points found on the boundary is relatively small, and a significant reduction in the data set is achieved especially when the data set includes convex shapes.

The main limitation of IS is its run time, as DT is constructed several times, and the boundary condition and the feasibility checks are performed repeatedly until the boundary becomes stable.

CHAPTER 6

A SWARM INTELLIGENCE BASED APROACH FOR CLUSTERING: ANT COLONY OPTIMIZATION

In this chapter, we focus on the application of Swarm Intelligence (SI) to the clustering problem. First, we review the major issues in the design of a SI based algorithm, namely the identification of agents and the type of swarm.

Agents define the variables of a problem and they directly affect the search procedure. Thus, agent representation plays a key role in the SI design. Taking into account the importance of the agent representation, we propose a classification for the SI based clustering algorithms in terms of agent representation. We discuss the use of each agent representation scheme and its complications.

SI includes particle swarm optimization, ant colony optimization and other swarm (e.g. bees, wasps, termites) based algorithms. In this dissertation we particularly concentrate on ACO among these due to the natural tendency of clustering in an ant colony. We introduce the ACO-based clustering (ACO-C) algorithm, which deals with data sets having the following properties:

(1) The number of clusters is unknown.

(2) Clusters are separated spatially.

(3) Clusters may have arbitrary shapes.

(4) There may be density variations within the clusters.

(5) Clusters may have density differences across each other.

We define ACO-C with two preprocessing steps: neighborhood construction and data set reduction. Considering the connectivity, the density and the spatial relations among the points, the neighborhood of each point is constructed using the

NC algorithm in Chapter 3. Besides, subclusters (closures) are formed using the connectivity relations between the neighborhoods of points. Taking these subclusters as input, two main decisions of ACO-C become merging the divided subclusters and breaking-off the outliers. The interior points of the closures are already connected, hence it is sufficient to consider in ACO-C only the data points on the boundaries of closures. This provides data set reduction and supports the scalability of the clustering algorithm. The boundary points of the closures are determined by either Delaunay Triangulation Cropping (DTC) or Ideal Simplex (IS) algorithms introduced in Chapter 5. DTC extracts the boundary in 2-dimensional data sets whereas IS steps in for higher dimensions.

To sum up, ACO-C takes the neighborhoods, the closures, and the boundary points of closures as input, and it particularly focuses on the merging of subclusters and breaking-off the outliers.

We provide a review of SI based algorithms in the first section of this chapter. The details of ACO-C are explained in the second section. Next, a factorial design is performed to set the parameters of ACO-C. Then, the performance of ACO-C is tested on various data sets in order to explore the strengths and weaknesses of the algorithm.

6.1. Swarm Intelligence Based Algorithms for Clustering

A swarm is composed of a group of simple but autonomous agents that interact with each other and/or with their environment. Although there are only decentralized control mechanisms among the agents, the swarm can achieve complex and difficult tasks. The main reason behind this is the collective behavior emerging from the swarm.

Working principles of a swarm are affected from the behavior of an autonomous agent, the communication methods among the agents and with the environment, and moves of the agents. In SI metaheuristic these concepts correspond to agent representation, neighborhood, decision rule, exploration and exploitation mechanisms, respectively.

The starting point in the design of a SI based algorithm is agent representation, and the other characteristics of SI build upon this ground. Agent representation is directly related with the potential solutions to be obtained by the algorithm. That is, an agent either directly represents a set of variable(s) in the solution, or works as a means to construct the solution. Agent representation must be capable of covering all the feasible solutions and the optimal solution, and it determines the size of the solution search space.

Neighborhood is directly related with the locality of the agents, i.e. their moves. Thus, it determines the possible solution components in the next move of an ant, and a decision rule helps to select a solution component in the neighborhood. In fact, decision rules for solution construction are tools for the exploitation and exploration properties of ACO.

6.1.1. Classification of SI based Clustering Algorithms

Taking into account the importance of the agent representation, we classify the SI based clustering literature in terms of the agent representation. The four classes are: (1) cluster-data point assignment, (2) representatives of clusters, (3) direct point-agent matching, (4) search agent representation. In this section we discuss the main characteristics of these representation schemes, and provide the strengths and the weaknesses of each category. In addition to these, we analyze the relation between the SI based algorithms (PSO, ACO, OSIB explained in Chapter 2) and the representation categories.

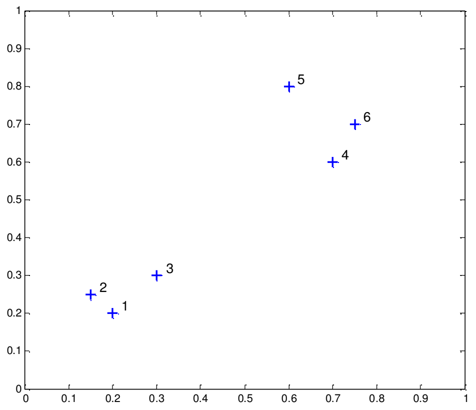
(1) Cluster-data point assignment

Cluster assignments of all points are represented by an agent. In Figure 6.1, an example for this type of representation is depicted for a data set with six points. For a clustering solution with two clusters, the representation is a 6-dimensional vector in which each dimension shows the clustering assignment of the associated point. Furthermore, the number of attributes in the data set does not affect the representation scheme.

Clustering solution

Cluster 1 → $C_1 = \{1, 2, 3\}$

Cluster 2 → $C_2 = \{4, 5, 6\}$



Point #	1	2	3	4	5	6
Agent →	1	1	1	2	2	2

Figure 6.1 An example for the cluster-point assignment representation

There is only one study that uses cluster-data point assignment in PSO, namely Jarboui et al. (2007). Combinatorial PSO is implemented for the clustering problem in this work. The position of a particle (agent) represents the clustering assignments of the data points. Generally, movements of particles in PSO are in continuous space. For this reason, a dummy vector maps the clustering solution represented by the particle taking values from the set {-1, 1, 0} according to the solution status, i.e. the local best cluster, the global best cluster or none of these, respectively. Velocity ensures to move the particle by combining three components: the best position of the particle, the swarm's best position and the random component. As the velocity is a continuous variable, the new position of the particle is discretized using a threshold and the dummy vector.

In ACO, there are several articles reported that use this representation scheme (Wang and Wei 2009, Chen and Chen 2006, Ho and Ewe 2005, Runkler 2005, Prabhaharan et al. 2005). They all have the same pheromone update mechanism, that is, pheromone substance is associated with each cluster-data point assignment. In every iteration some of the pheromone regarding the assignment evaporates, whereas the promising cluster-data point assignments are rewarded by increasing pheromone. Amount of increase is determined by the inverse of the objective function value, which is the within cluster distance variance. Timing of the pheromone update, i.e. local or global update, and the decision rule for solution construction affect the convergence speed and they are different in each article.

In fuzzy clustering, Runkler (2008) uses cluster-data point assignment scheme in a wasp swarm optimization algorithm. Cluster-data point assignment is made using stochastic tournament selection (wasp swarm tournament). Assignment probabilities are determined according to the distance between cluster centers and points. Since neither direct nor indirect information exchange is observed among the wasps, this article does not directly benefit from the collective behavior of the swarm. Instead, it ensures the assignment of the resources to the wasps according to their contribution to the swarm.

The articles in this category take the minimization of the within cluster distance in Euclidean space as the clustering objective. Therefore, these algorithms are suitable for spherical and compact clusters.

(2) Representatives of clusters

An agent shows all the representative points of the clusters such as centers of clusters, median points of clusters, medoid points of clusters, etc. In this scheme, the number of clusters or the number of representatives needs to be provided a priori. In SI a data point is assigned to the cluster having the nearest representative to the data point. In Figure 6.2, an example agent is given for a data set with six points with two attributes for each. If the number of clusters is given as two, then the agent corresponds to a 4-dimensional vector.

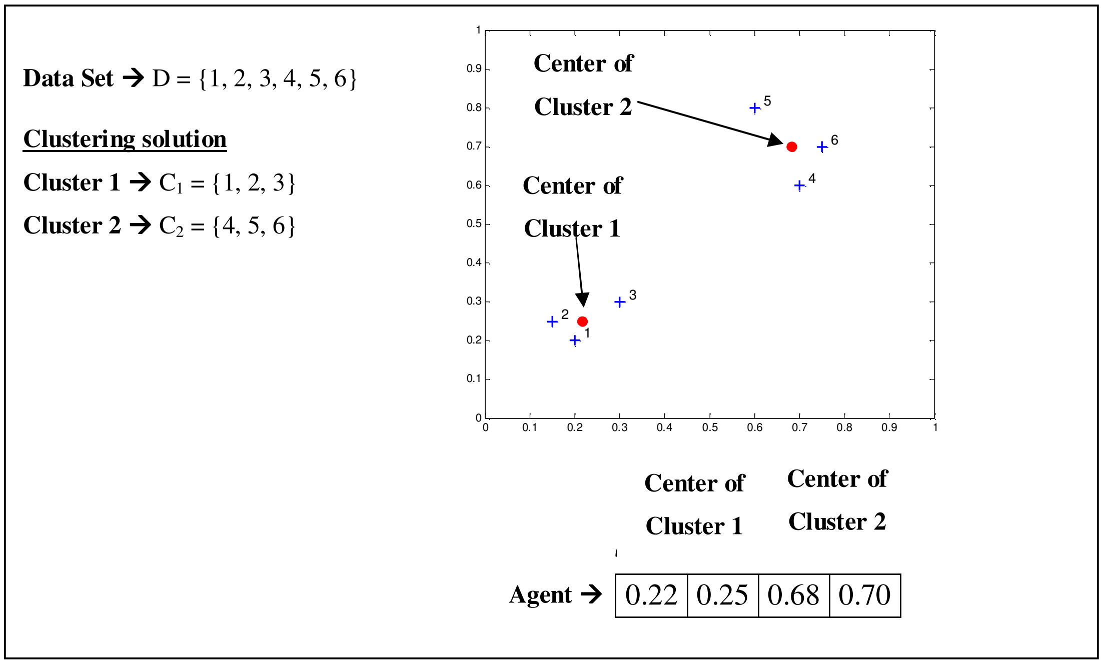


Figure 6.2 An example for representatives of clusters approach

In the PSO literature Ahmadi et al. (2010), Kao et al. (2008), Omran et al. (2006), Paterlini and Krink (2006), Omran et al. (2005) and Omran et al. (2002) use this representation scheme. Except Omran et al. (2006), others assume that the number of clusters is given, and the velocity of a particle (agent) is adjusted according to particle's local best and the swarm's global best. Thus, the moves of the particles change the coordinates of the representative points. Kao et al. (2008) propose a hybrid algorithm combining Nelder-Mead simplex algorithm and PSO. Nelder-Mead is used for local search, whereas PSO is used as an exploration tool. Omran et al. (2006) assumes that the number of clusters is unknown, and they start with a set of potential representatives of clusters. A binary particle is a c-dimensional vector where c is the cardinality of the set of potential representatives of clusters. Thus, each cluster representative is associated with a dimension in the particle, and the associated value is equal to 1, if the cluster representative is used in the current solution, otherwise it takes value of 0. Hence, a binary particle determines the number of clusters and the cluster representatives used in the solution. The coordinates of potential representatives are optimized using the k-means algorithm.

Finally, clustering solutions are evaluated using validity indices, Dunn's index and Davies Bouldin index.

Santosa and Ningrum (2009) apply cat swarm optimization to the clustering problem using the representatives of clusters. Each cat represents a cluster center, and the moves of a cat changes the coordinates of the associated cluster. In Fathian et al. (2007) an agent represents the cluster centroids. Although this algorithm is inspired from honey-bee mating, a crossover operator (queen mates with drones) and local search (improvement of worker bees) are used. Thus, it rather fits the principles of evolutionary algorithms, and emergent properties of swarm are not utilized. To the best of our knowledge, there is not any work on ACO that uses the representatives of clusters scheme.

Most of the algorithms in this category minimize within cluster distances as the objective function. A few studies consider clustering validity indices, such as Dunn's index and Davies Bouldin index, or use multiple objectives, such as minimization of within cluster distances and maximization of intercluster distances. However, the method of handling multiple objectives is quite naïve (by taking the weighted sum of objectives). The main drawback of this representation is being limited to forming of spherical and compact clusters.

(3) Direct point-agent matching

There is one-to-one correspondence with agents and points in this representation scheme. That is, each agent is "loaded" with a data point. Agents are scattered in 2-dimensional search space, and they move in the search space such that agents carrying similar points are positioned close. This representation is an attempt to determine the relative position of the points.

This type of representation is used in PSO by Picarougne et al. (2007), and Veenhuis and Köppen (2006). Both algorithms have two phases. In the first phase, similar points are gathered, whereas dissimilar ones are set apart. Taking the output of the first phase, second phase retrieves the clusters. Cluster retrieval process does not benefit from swarm optimization properties; instead a hierarchical algorithm is applied to construct the final cluster boundaries. The number of clusters is assumed to be unknown in both algorithms. The difference between these algorithms is their velocity update mechanisms which imply different search mechanisms.

Points are assigned to ants in Azzag et al. (2007) which is classified in the OSIB group. They introduce a hierarchical ant based algorithm to build a decision tree. Ants move the data points close to the similar points and away from the dissimilar ones. In this work, cluster retrieval process is not used. As far as we know, direct agent-point matching is not applied in ACO.

This type of representation helps to determine the relative positions of the points. However, as the number of points increases, the number of agents increases as well, and this leads to scalability problems. In addition, a separate cluster retrieval operation suffers from lack of integration with SI.

(4) Search agent representation

In this representation, an agent does not have a direct matching with the points. Instead, it is a means to carry the similar points to the same neighborhood and the dissimilar points away from each other. Agents (carriers) moving in the search space "visit" the data points to achieve this.

Xiao et al. (2004) introduce two hybrid SOM-PSO algorithms that benefit from this representation. In one of these algorithms, SOM is responsible of cluster assignments, and PSO is used to explore the weights found by SOM. Hence, particles correspond to the weight set of SOM. In the second algorithm, a number of SOMs are used, and the weight set of each of them is considered as a particle. Like the former one, PSO focuses on the improvement of the weight set of SOM. The velocity of a particle is affected from its previous velocity, its best local value in the past, and the global best.

Sinha et al. (2007) and Tsai et al. (2004) use search agent representation in ACO. In these articles, ants generate subtours by inserting edges between connected data points. Pheromone value is updated for each edge connecting two points. The closer the distance between the two points, the higher the released pheromone concentration. In the first phase of the algorithm, the edges between similar points become denser in terms of pheromone concentration. The next phase is the cluster retrieval process where clusters are formed using a hierarchical clustering algorithm.

Boryczka (2009), Kao and Fu (2006), Yang and Kamel (2006), Handl et al. (2006) and Martin et al. (2002) use search agent representation in their OSIB algorithms. Points are scattered on a 2-dimensional search space. A search agent (ant) picks up a point in the search space and drops it off near the points similar to it. These picking up and dropping off operations are conducted using probabilities that are calculated based upon the similarity of the points in the neighborhood. Here, search agents work as if they were forming a topographic map. After forming this pseudo-topographic map, a cluster retrieval process is applied to find the boundaries of clusters.

Search agent representation helps to find the number of clusters, however, generally, it requires a cluster retrieval process to run in isolation. This representation allows for arbitrary shaped clusters.

6.1.2. A Discussion about the Classification of Agent Representations

Table 6.1 summarizes the number of articles we could locate in the literature published until 2010 classified with respect to the representation type. All representation schemes are used in PSO, and the one most preferred among them is the representatives of clusters. The main reasoning behind this observation is that PSO generally deals with continuous optimization problems. As representatives of clusters operate in the continuous domain, (particle-)representatives of clusters matching fits perfectly.

As far as we know, representatives of clusters and direct point-agent representation schemes are not used in ACO. An ant typically constructs a solution from scratch as a proposal, whereas representatives of clusters and direct point-agent based approaches are generally based on improvement. In other words, it seems as if the points of view of ACO and these representation schemes are not compatible. The applicability of these representation schemes in ACO can still be a further research topic.

In OSIB, although all four types of representations are used, the most preferred one is the search agent. Search agent is particularly applied in ant based system in which data points are considered as the larvae/brood/dead bodies, and ants (search agents) are responsible of the classification of these. This prevalence can be attributed to the analogy between the ants' behaviors (larvae feeding, brood sorting, corpse clustering) and the clustering problem.

Table 6.1 Summary of the number of papers with respect to representation

		Representation			
	Total # of articles	**Cluster-point assignment**	**Representatives of clusters**	**Direct point-agent**	**Search agent**
PSO	10	1	6	2	1
ACO	7	5	-	-	2
OSIB	9	2	1	2	4

As we claim that the capabilities of a SI algorithm are affected from the agent representation scheme, we summarize the relation between the agent representation schemes and the general characteristics of the SI applied in clustering problem in Table 6.2. Below we provide a brief comparison of representation schemes with respect to problem characteristics.

- Number of clusters
    - In the literature the number of clusters is given a priori for the cluster-point assignment representation.
    - In the representatives of clusters approach, an upper bound is defined on the (unknown) number of clusters and these representative points are activated or deactivated during the iterations.
    - In the direct point-agent and the search agent representations it is not necessary to supply the number of clusters a priori. Both of these determine the relative position of the points with respect to each other.

However, an extra cluster retrieval algorithm is needed after the SI based algorithm which eventually yields the number of clusters. This clustering retrieval process does not utilize the collective power of SI, on the contrary, hierarchical algorithms with thresholds of minimum similarity or maximum dissimilarity are usually used for this purpose. Hence, this limits the role of SI in finding the final clusters.

- Arbitrary shapes
    - In the cluster-point assignment, within cluster variance/distance is minimized in Euclidean space, and this results in spherical and compact clusters. In addition, this may result in failure in extracting the clusters with different sizes and different densities.
    - Representatives of clusters approach assumes that a data point is assigned to a cluster such that the distance between the corresponding cluster representative and the point is minimum. This assumption does not work successfully with arbitrary shaped clusters.
    - Direct point-agent and search agent representations can handle arbitrary shapes. SI is used to extract the relations among the data points. In order to determine the clusters, an independent cluster retrieval process is executed after SI.

- Data types
    - Up to now no special similarity or dissimilarity measures are used in agent representation. Euclidean distances are used with all data types. Limited number of suggested approaches uses Hamming distance when there are binary or categorical data. In addition, the effect of the similarity/dissimilarity measure on the clustering algorithm has not been studied so far.

- Multiple objectives
    - Most of the literature focus only on the compactness objective (i.e. minimization of within cluster variance/distance) and the resulting clusters are consequently limited to spherical shapes.
    - Approaches with multiple objectives take the weighted summation of the conflicting objectives, and make it the fitness function. However, weighted summation is a naïve approach to address multiple objectives. Need for setting the weights and possibility of missing some nondominated solutions are the main drawbacks of this approach. Generally, cluster-point assignment or representatives of clusters schemes are used in these applications.
    - We could not locate any studies using point and search agent representations in a multiobjective clustering context.

- Constraint handling
    - As far as our search has shown SI based algorithms are not applied to constrained clustering problem. However, some of the current work can be adapted to the constraints using repair or penalty approaches.

- Scalability
    - There are scalability problems for the cluster-point assignment and direct point-agent representations. As there are as many agents as the data points, the memory required for agents increase significantly in large data sets.
    - Representatives of clusters approach depends only on the number of attributes and the number of clusters given a priori. Hence, it is capable of handling large data sets.
    - The number of points and the number of attributes do not affect the SI work that is based on search agent representation.

Table 6.2 Comparison of the agent representations with respect to problem characteristics

Agent Representation / Characteristics of clustering problem	Cluster-point assignment			Representatives of clusters			Direct point-agent matching			Search agent		
	PSO	ACO	OSIB	PSO	ACO	OSIB	PSO	ACO	OSIB	PSO	ACO	OSIB
Number of clusters	*			✓	*		PCR	*	PCR	PCR		
Arbitrary shapes	*			✓	*		✓	*	✓	✓		
Mixed data type	✓			✓	*		✓	*	✓	✓		
Multiobjective	*	✓	*	✓	*		*	*	*	*		
Constraint handling	*			*			*			*		
Scalability	✗			✓	*	✓	✗	*	✗	✓		

✓	Applied in the literature
*	Not applied in the literature but can be a further research topic
✗	Not possible
PCR	Post Cluster Retrieval is needed

6.2. The ACO-based Clustering (ACO-C) Algorithm

ACO inspires from the behavior of real ants. As ants search for food on the ground, they deposit a substance called pheromone on their paths. The concentration of the pheromone substance on the paths helps to direct the colony to the food sources. That is, ants release more pheromone on the promising (shortest) paths, and more ants are directed towards these. As a result, ant colony finds the food sources with effectivity by relating the ant activity with the environment.

In this section, the intelligence of an ant colony is used to extract the clusters, so an ACO based clustering (ACO-C) algorithm is proposed. The outline of ACO-C is shown in Figure 6.3. It takes the NC outputs, i.e. neighborhoods of points and subclusters (closures), as input. After the boundary formation algorithm finds the closure boundaries, ACO-C focuses on two operations: detection of outliers and merging the divided clusters. The details of ACO-C are given in the following sections.

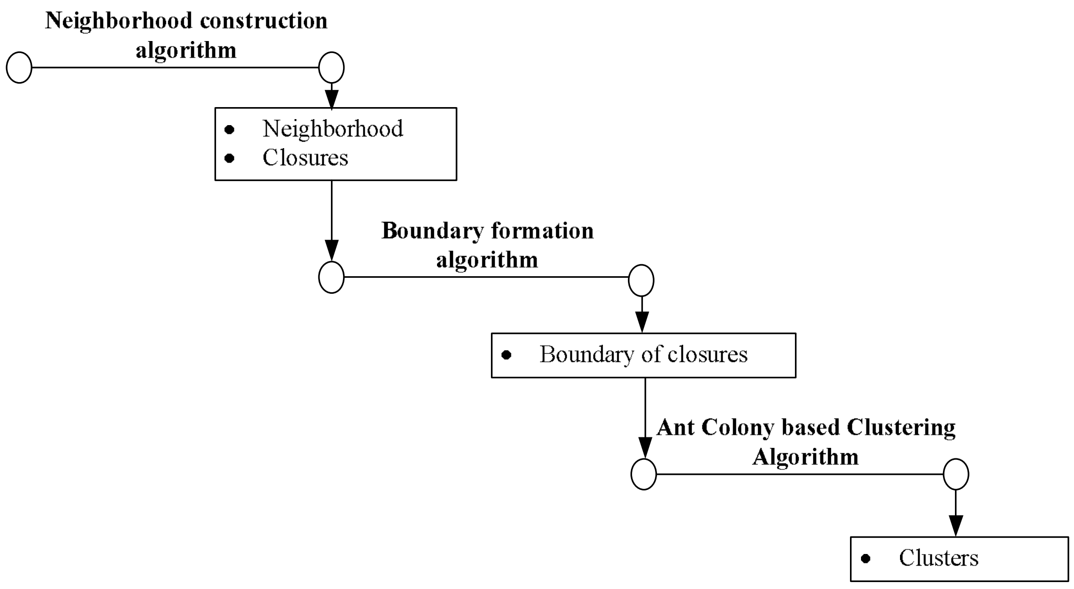

Neighborhood construction algorithm
• Neighborhood
• Closures
Boundary formation algorithm
• Boundary of closures
Ant Colony based Clustering Algorithm
• Clusters

Figure 6.3 Outline of the ACO-based algorithm

6.2.1. Ant (Agent) Representation

An ant represents a search agent in ACO-C, and it is a means to identify the data points that are in the same cluster. That is, an ant constructs a total clustering solution by forming a network. In this network an edge between a pair of data points means that these two data points are located in the same cluster.

There are two preprocessing steps for ACO-C: neighborhood construction and data set reduction (boundary formation). At the beginning of the algorithm ants have the neighborhood and closure information generated by the NC algorithm. Recall that closures formed at the end of the NC algorithm have two complications:

outlier mixes and divided clusters. As a remedy, ACO-C focuses on the detection of outliers and merging of the clusters judged to be divided in NC.

Figures 6.4 and 6.5 show the example data sets for outlier mixing and divided closures, respectively. In Figure 6.4, NC results in 4 closures, and closures 1 and 4 have outlier mixes. The boundary of each closure frames the data points in the closure by a closed piecewise continuous curve. Data points on the curve are on the boundary and the remaining ones are interior points. In Figure 6.5 (a) NC yields 8 closures. However, closures 4, 5, 6 and 7 depicted in higher resolution in Figure 6.5 (b) belong to the same cluster in the target (true) clustering.

To sum up, both tasks are related with the boundaries of the closures. It is assumed that interior points in a closure are already connected and there is no need to spend any effort for cluster assignments of these interior points. Therefore, an ant tracks a path only on the boundaries of the closures, and it connects either the data points on the boundary of the same closure or on the boundaries of different closures. As the interior points are excluded from further consideration for clustering, the data set is reduced. In other words, boundary formation in Chapter 5 contributes to the scalability of the ACO-C algorithm.

In every iteration of ACO-C, an ant constructs complete tours in the data set. Connected points present a cluster. During tour construction each point is connected to exactly two other points for convenience forming a complete tour. Such a restriction makes it easier to extract arbitrary shaped clusters and reduces computational requirements.

In outlier detection, it is expected that ants will explore the potential to skip the outliers, which are located on the boundary, in their tours. In order to merge the divided closures, edge insertion between the boundary points of different closures should be allowed as well as edge insertion between the boundary points of the same closure. An edge connecting the boundary points from two different closures is called a *bridge*. Such bridges are added before ACO-C starts, and they become part of the path an ant can take. In Figure 6.6, the dashed lines that connect two closures form bridges. The details of the bridge construction are explained in the next section.

Figure 6.7 shows an example solution constructed by an ant. The boundary points and associated closure boundaries are displayed in Figure 6.7 (a). The path visited by the ant corresponds to a five-cluster solution in Figure 6.7 (b). The ant skips the outlier point in closure 1 while taking her tour for cluster 1, and this outlier forms cluster 3. The outlier in closure 4 is detected in a similar way. Crossing the bridges between closures 1 and 2, the ant generates cluster 1.

This representation serves three main purposes in clustering: data set reduction, determination of the number of clusters, and extraction of arbitrary shaped clusters. Data set reduction is due to restricting the data points used in the clustering only onto the boundaries. The number of tours generated by an ant is not given a priori. Instead, an ant implies the number of tours (they emerge), i.e. clusters, using the connectivity and distance relations. To sum up, the representation automatically suggests the number of clusters (whether it is the ideal, is to be seen), and it is possible to obtain arbitrary shaped clusters. Besides, it provides a potential to test the edges away from the core neighboring points.

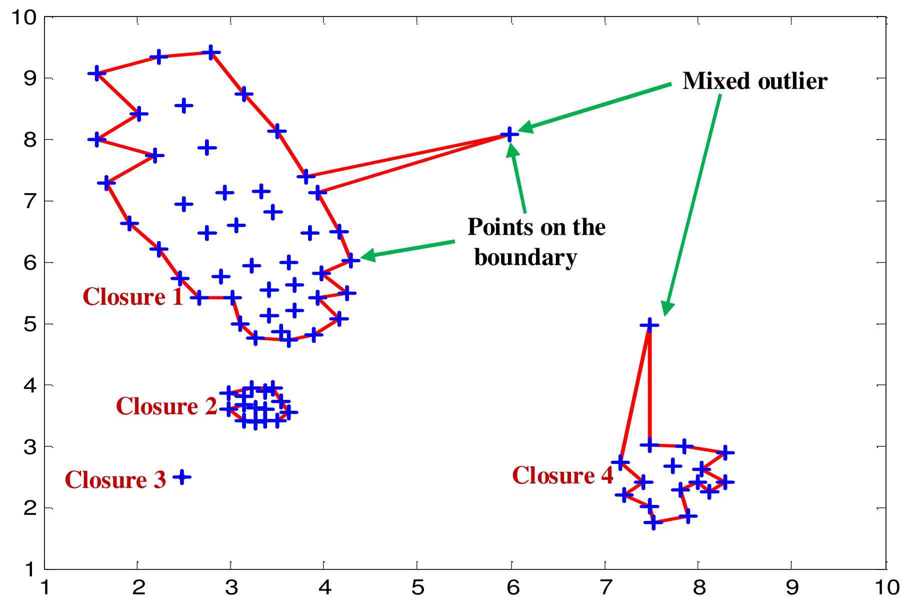


Figure 6.4 An example data set for outlier mixing

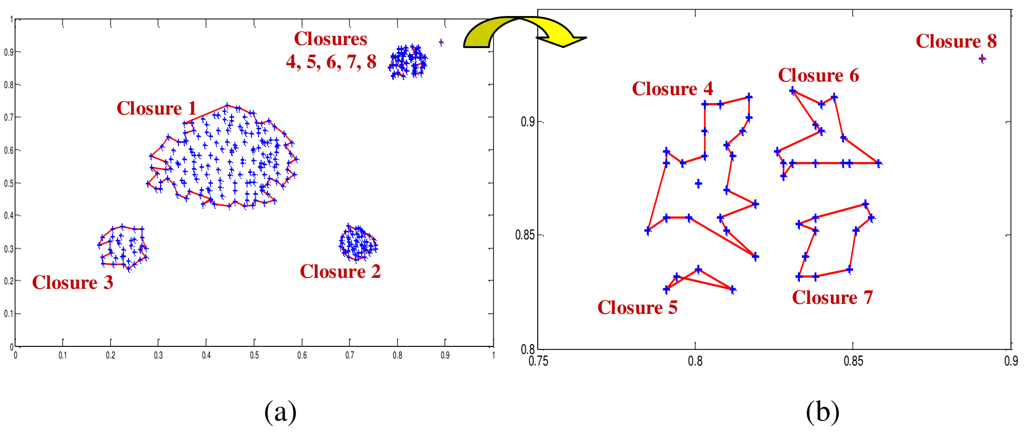


Figure 6.5 An example data set for divided closures (a) The entire data set, (b) In a higher resolution, only the five closures that are part of the cluster in the upper right corner

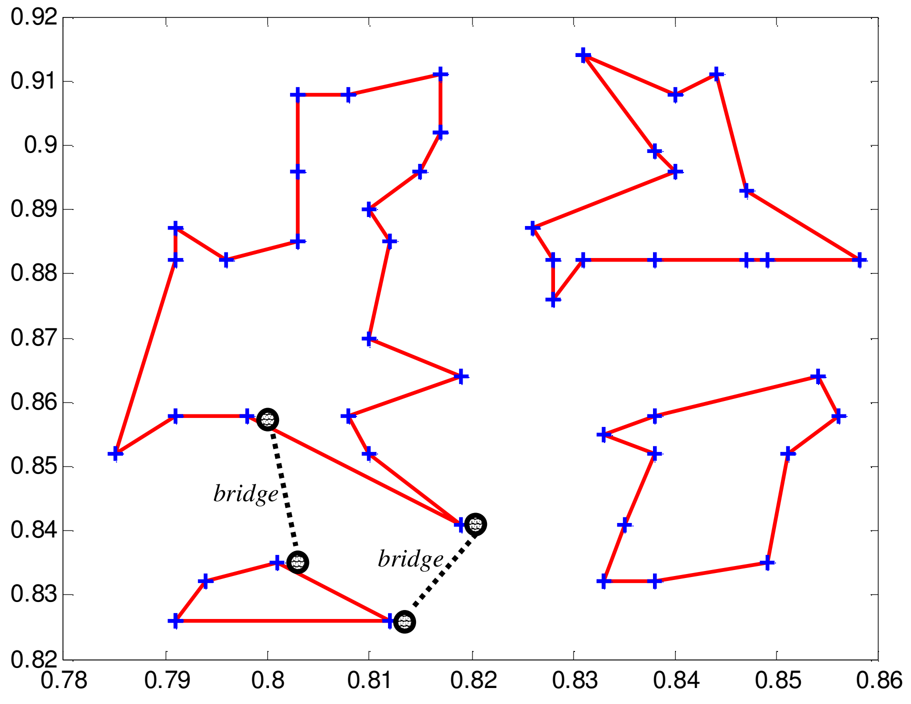
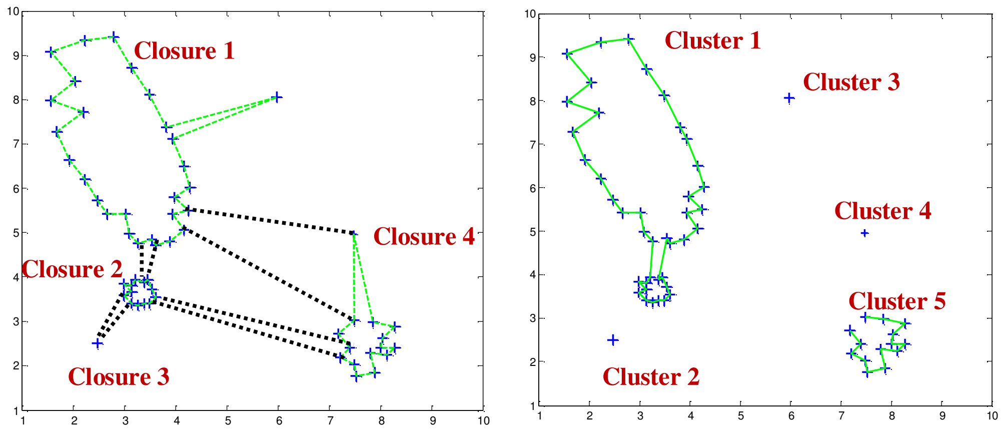


Figure 6.6 An example for bridge construction



(a) (b)

Figure 6.7 An example for solution construction (a) Closures and potential bridges between closures, (b) Ant's path which suggests to a five-cluster solution

6.2.2. Neighborhood of a Data Point in ACO-C

The ultimate aim of clustering is to extract clusters such that they are compact and connected, moreover they are well-separated from each other. The neighborhood definition is crucial as it shows the potential points to be visited by the ant during its tour or cluster construction. Taking into account the 2-degree connections, there are four important issues regarding the neighborhood definition by an ant: (1) It can insert edges between a point and two other points adjacent to it. (2) It can skip some adjacent points to break off one or more points including the outliers. (3) It can connect adjacent points from boundaries of different closures to merge the divided closures by including them in its subtour (i.e. jump or cross the void). (4) It can visit only a single point and leave for another subtour to distinguish the outliers or singleton clusters.

In order to find the ACO-C neighborhood of a point, Delaunay triangulation is used as it gives the proximity and connectivity relations among the points. Next, a tour is constructed passing through the boundary points using the adjacency information gathered from the DT. Then, taking into account the 2-degree restriction for each data point, we classify the neighbors of a point as follows.

Direct neighbors: If boundary points of a closure are in the same simplex in the DT, then these points are direct neighbors.

Indirect neighbors: An indirect neighbor of a given point satisfies the following two conditions.

(1) The given point and its indirect neighbor are not in the same simplex, i.e. they are not direct neighbors.

(2) Indirect neighbor of a given point is the direct neighbors of that point's direct neighbors.

Distant neighbor: A distant neighbor of a given point satisfies the following.

(1) The given point and its distant neighbor are in adjacent closures.

(2) The given point and its distant neighbor are in the same simplex.

(3) The shortest distance between two closures is an edge of the same simplex.

Nowhere: It is a dummy point. Nowhere does not have a degree limitation, and many points in a data set can be connected to nowhere. Connection to nowhere implies that the point under consideration does not have another adjacent point that should be placed in the same cluster with it. If 2-degree requirement of a point is all consumed by nowhere, then that point is an outlier/singleton. If two neighboring points are connected to each other, and both of them are also connected to nowhere, then the ant's tour forms a path with two end points connected to nowhere. Hence, the two points on a straight line form a 2-point cluster.

Direct and indirect neighbors are already connected as a result of the NC algorithm. Examples for direct and indirect neighbors in a 2-dimensional data set are presented in Figure 6.8. Indirect neighbors of point *i* in the figure ensure connectivity when one of its direct neighbors is detached as an outlier. Beyond the finite space, "nowhere" also stands as a neighbor of the point.

Merging the closures becomes possible via distant neighbors as shown in Figure 6.9. An ant can visit more than one closure in its tour, so entry to and departure from a closure must be allowed. Considering the 2-degree limitation, we place at least two bridges between a pair of closures to ensure incoming and outgoing edges to and from a closure. In Figure 6.9, two bridges between closures A and B are shown by dashed lines.

A point classified as an outlier satisfies the 2-degree restriction via the nowhere node. If a point first connects to nowhere, then it forms a singleton cluster, which means that it is an outlier. Hence, connection to nowhere is favored particularly for outliers.

In the 2-dimensional space, the ACO-C neighborhood of point i, NES_i, is composed of direct and indirect neighbors, distant neighbors, and nowhere. In higher dimensions, NES_i includes direct neighbors, distant neighbors, and nowhere.

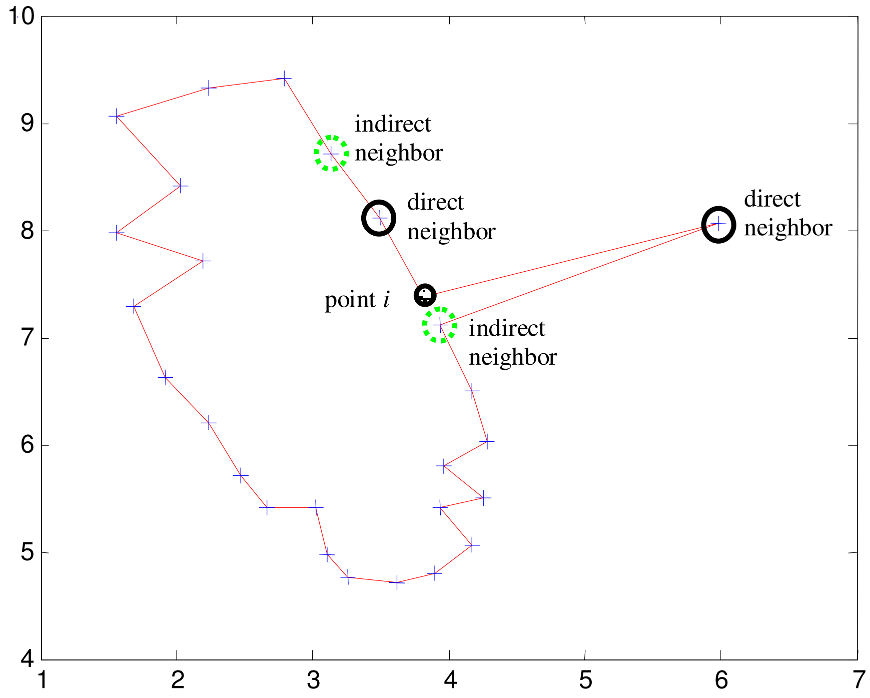


Figure 6.8 An example for direct and indirect neighbors of point i in a 2-dimensional data set

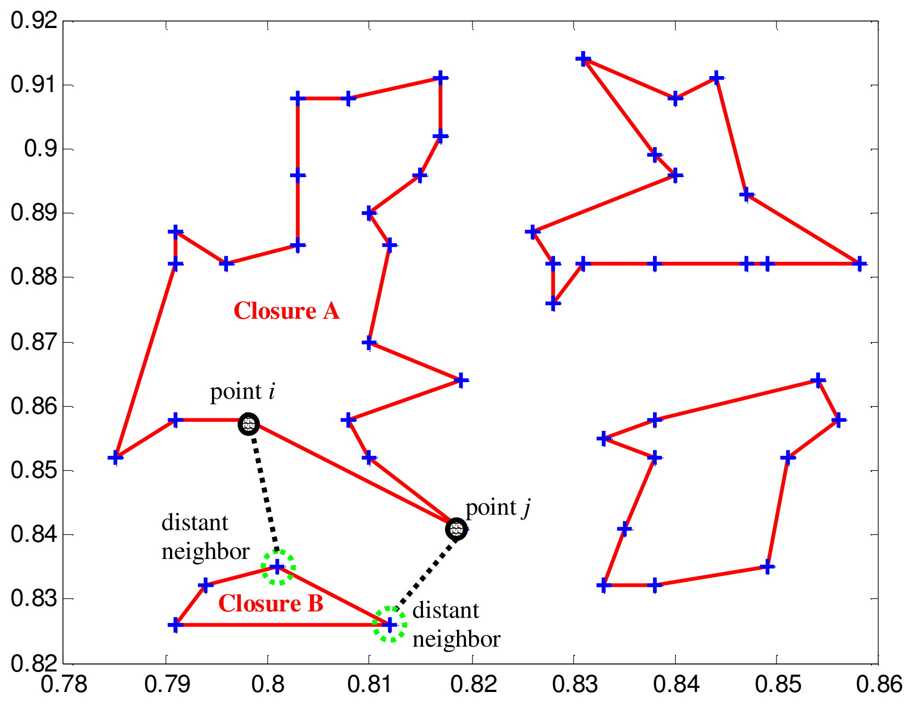


Figure 6.9 An example for distant neighbors of points *i* and *j* in a 2-dimensional data set

In the neighborhood construction the main difference between 2- and higher dimensions is the indirect neighbors. A simplex in 2-dimensional space is equivalent to a triangle, so facets of a closure (boundary) are composed of edges. Thus boundary points on these edges satisfy 2-degree requirement by default. The indirect neighbors are added to the neighborhood of a point, lest the outlier skips (break-offs) disrupt the connectivity of the closure due to the 2-degree restriction.

In 3-dimensions, triangles form the facets of a closure (boundary) and each point has a higher degree than two by construction. Thus, outlier detection does not cause disconnectivity during an ant's tour, and there is no need to consider the indirect neighbors. In the 3-dimensional space, direct neighbors of point *i* is shown in Figure 6.10. For higher dimensions the same conditions hold.

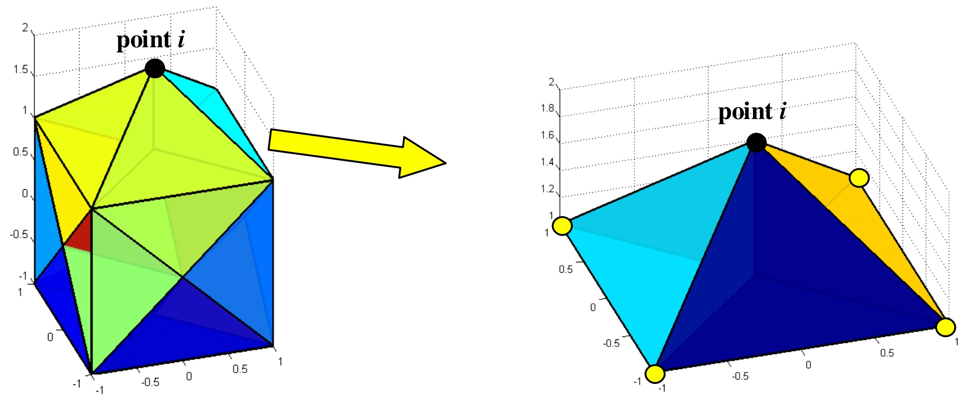


Figure 6.10 An example for direct neighbors (gray circles) of point *i* in a 3-dimensional data set

6.2.3. The ACO-C Algorithm

We give the notation below and present the outline of the ACO-C algorithm in Figure 6.11.

Notation

Index

i, *j*	: indices for data points
m, *n*	: indices for clusters
s	: index for ants
t	: index for iteration number

Problem Parameters

D	: set of data points in the data set
no_points	: number of points in the data set
dimension	: number of attributes of a point in the data set

Algorithm Parameters

no_ants	: number of ants
max_iter	: maximum number of iterations
ρ	: evaporation rate for the pheromone

Variables

$degree_i$	: number of points currently connected to point *i* (current degree of point *i*)
d_{ij}	: Euclidean distance between points *i* and *j*
τ_{ij}	: pheromone value of the edge (*i*, *j*)
p_{ij}	: probability of connecting (placing an edge between) points *i* and *j*
inc_comp_i	: incumbent compactness value for point *i*
inc_sep_i	: incumbent separation value for point *i*
$cluster_s$	: number of clusters in the solution generated by ant *s*
sep_{sm}	: separation value of cluster *m* in the solution generated by ant *s*
$comp_{sm}$	: compactness value of cluster *m* in the solution generated by ant *s*
comp_s	: compactness vector (a one by $cluster_s$ vector) carrying compactness values of all clusters in the solution generated by ant *s*
sep_s	: separation vector (a one by $cluster_s$ vector) carrying separation values of all clusters in the solution generated by ant *s*

Sets

$\mathrm{D_o}$	: set of unvisited points, $\mathrm{D_o} = \{ i : i \in \mathrm{D} \text{ and } degree_i < 2\}$
NS_i	: set of neighbors of point *i* (includes only direct neighbors and indirect neighbors)
NES_i	: set of extended neighbors of point *i* (nowhere node, direct neighbors, indirect and distant neighbors, if any)
NCES_i	: set of currently available extended neighbors of point *i*, $\mathrm{NCES}_i = \{ j : j \in \mathrm{NES}_i \text{ and } degree_j < 2\}$
C_{ms}	: set of points in cluster *m* following the tour by ant *s*

MST_{ms}	: set of edges in the MST of the points in cluster *m* following the tour by ant *s*
$\text{MST}_{m(i),\text{s}}$	: set of edges in the MST of the points in the neighborhood of point *i*, NS_i, in cluster *m* following the tour by ant *s*
SC	: set of solutions constructed by the ants in the current iteration
SN	: set of nondominated solutions
SE_s	: set of edges in the current clustering solution following the tour by ant *s*

```
The ACO-C Algorithm

Step 0. Preprocessing (Neighborhood construction, data set reduction)
Step 1. Initialization ($max\_iter$, $no\_ants$, $\rho$)
For $t = 1,.., max\_iter$
    For $s = 1,.., no\_ants$
        Step 2. Connecting the points
        Set $D_o = D$ and $NCES_k = NES_k, \forall k \in D$.
        Set $m = 1$.
        While $D_o \neq \varnothing$
            2.1. Point selection
            Select point $i$ from $D_o$ at random.
            Initialize the set of points in the current cluster $m$, $C_{ms} = \{i\}$.
            While $NCES_i \neq \varnothing$ and $i \neq$ "nowhere"
                2.2. Edge construction
                Construct edge $(i,j)$ where $j \in NCES_i$ using
                probabilities based on $\tau_{ij}$.
                2.3. Update sets
                Add $j$ to $C_{ms}$, remove $j$ from $D_o$ and from all $NCES_k$'s
                that contain it, set $i = j$.
            End while
            Set $m = m + 1$ and start a new cluster.
        End while
        Step 3. Clustering evaluation
        Step 4. Local search
    End for
    Step 5. Pheromone update
    Step 6. Non-dominated set update
End for
```

Figure 6.11 The ACO-C Algorithm

The details of each step are explained as follows.

Step 0. Preprocessing

Preprocessing is composed of three phases.

Phase 1. Neighborhood construction

Neighborhood of each point i, CS_i, and closures are constructed by the NC algorithm described in Chapter 3.

Phase 2. Data set reduction (boundary formation)

Boundaries of the closures are determined by DTC and IS algorithms for 2- and higher dimensional data sets, respectively. The details of the DTC and IS algorithms are given in Chapter 5.

Phase 3. Neighborhood in ACO-C

Using the adjacency information gathered from Delaunay triangulation, neighborhood of point i, NES_i, is constructed from the direct, indirect, distant neighbors and nowhere. This step is repeated for every point in D.

Step 1. Initialization

The parameters in ACO, i.e. the number of ants (*no_ants*), the number of maximum iterations (*max_iter*), and the evaporation rate (ρ), are initialized. We conduct a factorial design explained in Section 6.3 to guide this initialization.

There is a relationship between the evaporation rate and the pheromone deposits on the solution components. In each iteration, pheromone amounts on the solution components evaporate at a given rate, and some of this is recovered through the release of pheromone by the ants visiting if any. Thus, convergence rate of the algorithm depends on the initial pheromone levels and the evaporation rate. In order to avoid premature convergence, we initialize the pheromones as the inverse of the evaporation rate, $\tau_{ij} = 1/\rho$ for $\forall i, j \in \mathrm{D}$, to let pheromone removal take adequately long.

Closures obtained at the end of the NC algorithm are taken as the initial clustering solution. Thus, for each point, incumbent compactness and incumbent separation values (i.e. inc_comp_i and inc_sep_i) are initialized using the separation and the compactness values of the NC closure to which point i belongs.

Steps 2-6 that are explained below are repeated until the entire ant colony performs the maximum number of iterations, *max_iter*.

Step 2. Connecting the points

Each time an ant starts clustering, the set of unvisited points, D_o, is initialized as the entire data set, D. For each point in the data set, the set of currently available extended neighbors, $NCES_i$, is initialized as its complete set of extended neighbors, NES_i. The current number of clusters is set to 1. Every time an ant starts a new subtour, the number of clusters is incremented by 1.

Steps 2.1-2.3 are repeated until the set of unvisited points, D_o, becomes empty.

2.1. *Point selection*

A point, say point i, is selected at random from the set of unvisited points, D_o. Suppose that ant s constructs cluster m. Then, the first element of this cluster, C_{ms}, becomes point i.

Suppose that the current point visited by the ant is point i. While the set of currently available extended neighbors for point i, $NCES_i$, is not empty, and point i does not correspond to the nowhere node, a subtour (cluster) is constructed by repeating Steps 2.2 and 2.3.

2.2. *Edge selection*

The ant inserts an edge between the current point i and a point selected from the set of currently available neighbors of point i, $NCES_i$. As the pheromone amount on an edge, τ_{ij}, represents the tendency for the edge (i, j) to occur in a cluster based upon past solutions, probability of selecting the edge (i, j) is calculated as follows.

$$p_{ij} = \frac{\tau_{ij}}{\sum_{l \in \mathrm{NCES}_i} \tau_{il}} \quad \text{for } \forall j \in \mathrm{NCES}_i \tag{6.1}$$

In the next move, the ant continues its visits starting from point *j*. During the construction of clusters both locality and connectivity issues are taken into account, locality by the use of points from NCES_i and connectivity by recurrence of the linkage relation.

2.3. *Update sets*

Point *j* selected in Step 2.2 is added to the current cluster, $\mathrm{C}_{ms} = \mathrm{C}_{ms} \cup \{j\}$. As point *j* becomes a visited point, it is eliminated from D_o and from all NCES_k's that include point *j*.

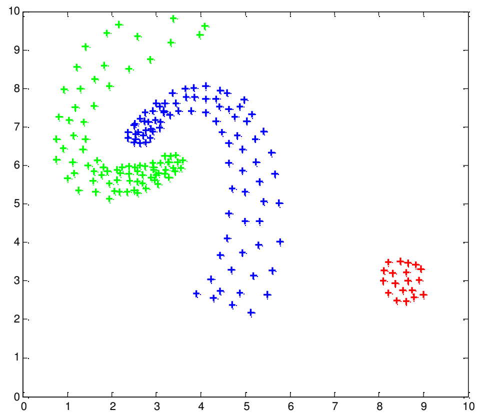
Point *j* becomes the next point for connection, i.e. $i = j$. If the set of currently available extended neighbors, NCES_i, is not empty for point *i*, and point *i* does not correspond to the nowhere node, Steps 2.2 and 2.3 are repeated. Otherwise, chain-like connectivity relations in the subtour (cluster) are broken, and the construction of the associated subtour ends up. Hence, a new subtour is initialized by incrementing the cluster index, *m*, by 1. The construction of the new subtour starts in Step 2.1.

As the result of this iterative process, a clustering solution is generated by ant *s*.

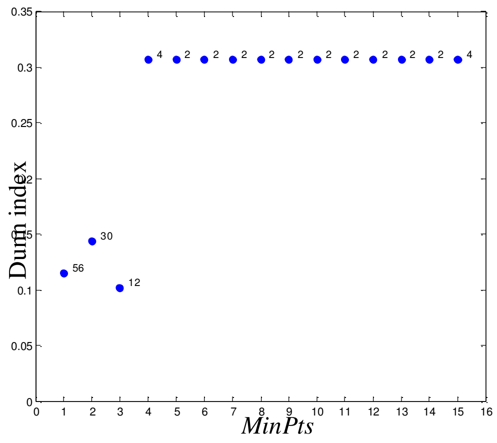
Step 3. Clustering evaluation

SI based algorithms guide the search by evaluating and reflecting the performance of the solutions in general. Thus, it is assumed that the target clustering solution will be sought by testing the optimality of an evaluation function. However, quantifying and combining the clustering objectives (compactness, connectivity and separation) for any data set is not a trivial task. Particularly, for the data sets with arbitrary shaped clusters and density variations, traditional compactness objectives like maximum edge in a cluster or distance between the points in a cluster and the cluster's representative point, or traditional connectivity measures like *k*-NN do not reflect the quality of the clustering properly. An example data set is displayed in Figure 6.12 (a). The maximum edge length in the spiral clusters is the larger than the intercluster distance between these two spiral clusters. Furthermore, the cluster center of a spiral cluster is not located within the cluster, so the total distance between the cluster points and the cluster center of a spiral cluster becomes large. Thus, both compactness calculations may mislead.

We have also evaluated the performance of a traditional validity index, i.e. Dunn index (Halkidi et al. 2002b) in this data set. For generation of arbitrary shaped clusters, we use DBSCAN algorithm (Ester et al. 1996). The clustering solutions generated by DBSCAN are evaluated using Dunn index with different *MinPts* settings (within the range of 1 to 15), and the results are presented in Figure 6.12 (b). In the figure, the number of clusters found by each setting is shown as well, e.g. 56 clusters are found when *MinPts* is set to 1. Dunn index measures the worst separation to the worst compactness ratio, so the highest Dunn index implies good clustering. Although the highest Dunn index (0.31) is achieved for the solutions with 2 and 4 clusters, the target clustering includes 3 clusters with a Dunn index of 0.09. To sum up, Dunn index is not a proper measure to evaluate the quality of the clustering.



(a)



(b)

Figure 6.12 (a) An example data set with arbitrary shaped clusters and density variations. (b) Dunn index values for the clustering solutions generated by DBSCAN with different *MinPts* settings.

Optimization based methods and metaheuristics generally use a compactness based measure. These have objectives like minimization of total variance/distance in the cluster or minimization of total variance/distance between the points in a cluster and the cluster's representative point. However, the number of clusters needs to be given a priori, and by their nature resulting clusters have spherical or ellipsoid shapes in general.

In an effort to find the target clusters, some researchers suggest clustering validity indices as their objective, since these indices help to quantify the quality of the clustering solution and determine the number of clusters in a data set (Halkidi et al. 2002a and 2002b). In a clustering validity index the intercluster property (i.e. separation) and the intracluster properties (i.e. compactness and connectivity) of a clustering solution need to be measured properly. Next, measurements of these properties are combined in a scalar value.

Optimization of a validity index as such for clustering has some complications. Firstly, some of the validity indices (like the modified Hubert Γ statistic) are monotonically non-increasing functions so the optimal value might not correspond to the target cluster. Secondly, the combination operation in the final step of the validity index calculation is like "cutting toothpicks by an ax", as some important information about compactness, separation and connectivity might be overlooked.

As far as we know, there is not a generally accepted objective function in evaluating the clustering solutions with arbitrary shapes, different densities and unknown number of clusters. Therefore, we introduce two alternative evaluation functions for clustering. Two main contributions of these evaluation functions are: (1) Compactness and separation values for the clusters are not combined, instead, both of them are considered in the comparisons. (2) Compactness and separation are measured relative to the neighborhoods.

Alternative 1. Clustering Evaluation Relative to Neighborhood (CERN)

Compactness of a cluster is measured in two parts: connectivity ratio and relative compactness. The base point in the connectivity ratio calculation is the NC closures. In the clustering solution each cluster is evaluated in terms of each closure. In doing this, $connect_{msp}$ is calculated as the number of points left connected identically in cluster *m* (generated in ACO-C by ant *s*) and in closure *p* (generated by

NC) divided by the number of points in closure p, i.e. $connect_{msp} = \begin{cases} |\mathrm{C}_{ms} \cap \mathrm{Cl}_p| / |\mathrm{C}_p|, & \mathrm{C}_{ms} \cap \mathrm{Cl}_p \neq \varnothing \\ 1, & \text{otherwise} \end{cases}$.

In the ideal case, $connect_{msp}$ takes a value of 1 which implies that either cluster m and closure p fully overlap, or cluster m and closure p do not have any common elements. To calculate the connectivity ratio of cluster m (generated in ACO-C by ant s), $connect_{msp}$ is multiplied over all closures. Thus, merging multiple closures that are entirely contained in the cluster does not harm a connectivity value of 1, whereas connectivity gets less than 1 when there is an outlier in the closure. Connectivity ratio helps to cover arbitrary shapes and density variations in the clusters.

We define the relative compactness of cluster m, r_comp_{ms}, with respect to the most inconsistent edge within its neighborhood. For this purpose, given a solution generated by ants, the minimum spanning tree (MST) is constructed for each cluster, and each edge in the MST is evaluated relative to the edges in its neighborhood. That is, each edge in the MST is compared with the edge contained in the cluster with the maximum length in the neighborhoods relative to the end points. More formally, $r\_comp_{ms} = \max\limits_{(i,j)\in \mathrm{MST}_{ms}} \left\{ \dfrac{d_{ij}}{\max\limits_{(k,l)\in \mathrm{MST}_{m(i),s}} \{d_{kl}\}}, \dfrac{d_{ij}}{\max\limits_{(k,l)\in \mathrm{MST}_{m(j),s}} \{d_{kl}\}} \right\}$.

MST is a graph such that the sum of the edge lengths in the graph is the minimum and the graph is connected with no cycles. Hence, MST accommodates both arbitrary shapes and density variations.

Our objective is to form clusters that span the connected points while ensuring consistency in the neighborhoods. Hence, we consider the inverse of the connectivity ratio as the weight of the relative compactness. Compactness of cluster m generated in by ant s is found as the product of the inverse of the connectivity ratio and the relative compactness, i.e. $comp_{ms} = \dfrac{r\_comp_{ms}}{\prod\limits_{p=1}^{no\_closures} connect_{msp}}$.

In the division of closures case, a small relative compactness value is obtained, but connectivity is disrupted. On the other hand, in the merging of closures case, the relative compactness value increases, and the connectivity is still ensured. To sum up, the proposed compactness measure is built upon the trade-off between the connectivity and the relative compactness.

Separation of cluster m generated in ACO-C by ant s, sep_{ms}, is based on the nearest cluster to cluster m. Suppose that nearest two points are i^* and j^* between cluster m and its nearest neighbor, i.e. $(i^*, j^*) = \arg\min_{i,j,n} \{ d_{ij} : i \in \mathrm{C}_{ms}, j \in \mathrm{C}_{ns} \}$. The distance between points i^* and j^* is evaluated relative to the neighborhoods of points i^* and j^*, i.e. $sep_{ms} = \min \left\{ \frac{d_{i^* j^*}}{\max\limits_{(k,l) \in \mathrm{MST}_{m(i^*),s}} \{d_{kl}\}}, \frac{d_{i^* j^*}}{\max\limits_{(k,l) \in \mathrm{MST}_{m(j^*),s}} \{d_{kl}\}} \right\}$.

Figure 6.13 provides an example for the relative compactness and separation calculation. In order to calculate the relative compactness of cluster 1, we consider the MST of cluster 1 which is displayed in the figure. Each edge (i, j) in the MST is evaluated according to the sets of neighbors for points i and j, NS_i and NS_j. That is, the edge (i, j) is normalized with respect to the maximum edge in the MSTs which are constructed using the intersection points of NS_i and NS_j with the points in cluster 1. This normalization is conducted for each edge in the cluster MST. Among them the maximum is selected as the most inconsistent edge or the relative compactness of cluster 1, r_comp_{1s}. The minimum edge length between cluster 1 and the nearest cluster 2 is edge (i^*, j^*), and separation of cluster 1 becomes sep_{1s} which is normalized with respect to the maximum edge in the MSTs of the intersection of the points in NS_{i^*} and NS_{j^*} with the points in cluster 1.

Finally, quality of a clustering solution generated by ant s is represented by the worst separation, $\min_m \{sep_{ms}\}$, and the worst compactness, $\max_m \{comp_{ms}\}$, and these two performance measures are used for comparing the clustering solutions in a bicriteria setting.

This evaluation mechanism is advantageous for solving arbitrary shaped clusters or cases with large intracluster density variations. However, the evaluation scheme may mislead ACO in heterogeneous data sets. If the variance of the intercluster distances is high, CERN smoothes out the magnitude information gathered from the edge lengths and gets indifferent to widely varying clearances. To overcome this sensitivity, an alternative evaluation function is proposed as follows.

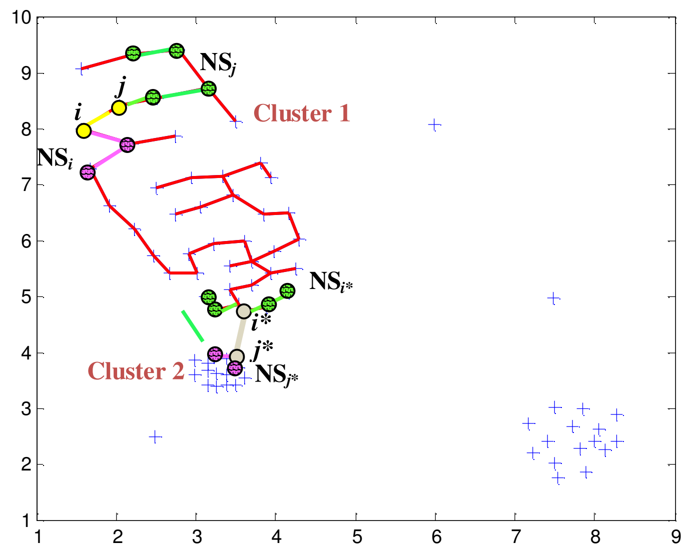


Figure 6.13 An example for compactness and separation calculation

Alternative 2. Weighted Clustering Evaluation Relative to Neighborhood (WCERN)

The idea behind this evaluation scheme is similar to CERN. For each cluster, both compactness and separation values are calculated relative to the neighborhoods. The main difference in WCERN is that the length of an edge is used as the weight factor. Thus, compactness and separation of cluster m are calculated as follows.

$$comp_{ms} = \max_{(i,j)\in \mathrm{MST}_{ms}} \left\{ \frac{d_{ij}^2}{\max\limits_{(k,l)\in \mathrm{MST}_{m(i),s}} \{d_{kl}\}}, \frac{d_{ij}^2}{\max\limits_{(k,l)\in \mathrm{MST}_{m(j),s}} \{d_{kl}\}} \right\} \frac{1}{\prod\limits_{p=1}^{no\_closures} connect_{msp}} \tag{6.2}$$

$$sep_{ms} = \min \left\{ \frac{d_{i^*j^*}^2}{\max\limits_{(k,l)\in \mathrm{MST}_{m(i^*),s}} \{d_{kl}\}}, \frac{d_{i^*j^*}^2}{\max\limits_{(k,l)\in \mathrm{MST}_{m(j^*),s}} \{d_{kl}\}} \right\} \tag{6.3}$$

This evaluation scheme takes both the edge lengths and its relative size in its neighborhood into account. Typically, in the relative compactness calculation of clusters with intracluster density variations, the long edges will shrink due to the relativity implied by the ratio. In the separation calculation, the minimum edge length between the clusters is inflated using the neighborhoods with modest variation.

Just as in CERN, quality of a clustering solution generated by ant *s* is determined as the worst separation, $\min_m\{sep_{ms}\}$, and the worst compactness, $\max_m\{comp_{ms}\}$.

To sum up, in step 3, the separation and compactness values of each cluster, i.e. sep_{ms} and $comp_{ms}$, are calculated using evaluation functions from either alternative 1 or alternative 2. sep_s and comp_s vectors are formed to be used in step 4. The quality of the clustering solution becomes $\min_m\{sep_{ms}\}$ and $\max_m\{comp_{ms}\}$ in each alternative.

Step 4. Local search

Randomness in ACO ensures exploration in the search space thoroughly, and local search is a crucial mechanism to strengthen ACO's exploitation property. In ACO-C we apply local search to each clustering solution constructed and evaluated. Conditional merging operations are performed in the local search. That is, if the two clustering evaluation criteria do not worsen after merging two clusters, clusters under consideration are merged.

Two conditions used in the local search are as follows.

(1) If candidate clusters to be merged are non-singleton clusters, check whether the worst separation, $\min_m\{sep_{ms}\}$, and the worst compactness, $\max_m\{comp_{ms}\}$ after merging, are at least as good as those of the two clusters before merging. If this condition holds, candidate clusters are merged.

(2) If there exists a singleton cluster (potential outlier) in the candidate clusters to be merged, check whether the worst compactness, $\max_m\{comp_{ms}\}$, after merging induces an improvement jointly in both the worst separation, $\min_m\{sep_{ms}\}$, and the worst

compactness, $\max_{m}\{comp_{ms}\}$, before merging. If this condition holds, candidate clusters are merged.

The clustering solutions at the end of the local search are accumulated in the set of solutions SC constructed by the ants in the current iteration.

Step 5. Pheromone update

Ants direct the solution to the promising solution regions by releasing pheromone. The amount of pheromone released is correlated with the solution quality. Pheromone update is performed for each solution component (i.e. edges in the connectivity) so that the effect of the solution component is well reflected in the pheromone accumulation for correct guidance.

In the ACO literature pheromone level for each solution component is proportional to the evaluation function value of the entire solution. However, the basis of our clustering approach is local density, connectivity and proximity relations. That is, local properties are rather dominant in clustering. So, definition of the solution quality by a single global measure ignores local characteristics (compactness of a cluster and separation between a pair of clusters). Pheromone update by a single measure for the entire solution may easily mislead the ant colony. For example, suppose a data set with three target clusters. In a solution we identify one of the clusters correctly whereas the remaining two are mixed. If we combine the compactness and separation values of individual clusters in one global measure, we may classify the entire solution as poor, missing the correctly identified cluster.

We use the local information of each data point in the pheromone update. That is, in the pheromone update of an edge in the solution, instead of using an overall compactness or separation value, we use the compactness and separation values of the specific cluster which the edge belongs to. For each point in the data set, the incumbent compactness value, i.e. the minimum compactness value obtained so far, inc_comp_i, and incumbent separation value, i.e. the maximum separation value obtained so far, inc_sep_i, are kept in the memory. Incumbent values denote the compactness and separation of an ideal cluster which includes point *i*. These compactness and separation values are not necessarily obtained from the same clustering solution. We compare the compactness (separation) of edge (*i*, *j*) in the current solution with the best of the incumbent compactness (separation) of its end points *i* and *j*. More pheromone is released if any of the best incumbent is improved. The choice of the best incumbent value for the comparison is a rather conservative update scheme, but it helps to avoid rapid convergence.

If edge (*i*, *j*) is visited in the subtours generated by any of the ants, the pheromone update mechanism includes both evaporation and pheromone release as below. Otherwise, the pheromone on edge (*i*, *j*) only evaporates.

$$\begin{aligned}\tau_{ij} &= (1-\rho)\tau_{ij} + \frac{1}{2\times no\_ants}\rho\left(\frac{\min\{inc\_comp_i, inc\_comp_j\}}{comp_{ms}}\right)\tau_{ij} \\ &+ \frac{1}{2\times no\_ants}\rho\left(\frac{sep_{ms}}{\max\{inc\_sep_i, inc\_sep_j\}}\right)\tau_{ij} \quad \text{for } (i,j)\in \mathrm{SE}_s, \forall s\end{aligned} \tag{6.4}$$

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad \text{for } (i,j)\notin \mathrm{SE}_s, \forall s \tag{6.5}$$

In ACO-C, we consider a bicriteria setting, in which each ant in the colony generates a clustering solution in every iteration. Improvement in both incumbents for all the solutions generated in an iteration may increase the accumulated pheromone amount on the edges dramatically, and this may bring about the rapid convergence risk. Thus, we need to stabilize the amount of pheromone deposited by the ants. In order to avoid the inflation of pheromone deposit, we use two weight factors in the pheromone release expression, namely ½ and 1/*no_ants*. The former one compensates for the bicriteria setting, and the latter accommodates for the ant colony effect. In addition, the amount of pheromone released by the ant is scaled using the evaporated pheromone, $\rho\tau_{ij}$.

Step 6. Nondominated set update

Performance of a clustering solution is evaluated using two criteria: the worst compactness and the worst separation values. The ultimate aim is to minimize the worst compactness and maximize the worst separation.

We evaluate the clustering solutions in a bicriteria setting, and propose a set of nondominated solutions. This way not only alternative clustering solutions, but also their trade-offs can be seen.

Suppose two clustering solutions generated by ants are $s1$ and $s2$. Solution $s1$ dominates solution $s2$ if solution $s1$ is better than solution $s2$ in either compactness or separation, and it is at least as good as solution $s2$ in the remaining criterion. More formally, $s1$ dominates $s2$ if one of the following holds:

i. $\max_m \{comp_{m,s1}\} < \max_m \{comp_{m,s2}\}$ and $\min_m \{sep_{m,s1}\} \geq \min_m \{sep_{m,s2}\}$.

ii. $\min_m \{sep_{m,s1}\} > \min_m \{sep_{m,s2}\}$ and $\max_m \{comp_{m,s1}\} \leq \max_m \{comp_{m,s2}\}$.

If there does not exist any other clustering solutions dominating solution $s1$, then solution $s1$ is a nondominated solution.

We update the set of nondominated solutions, SN, at the end of each iteration. That is, the clustering solutions constructed by the ants in the current iteration, SC, and the set of nondominated solutions present so far, SN, are checked for domination. If a clustering solution in SN is dominated by a solution in SC, then it is eliminated from SN and the new nondominated solution is added to SN. Using the updated SN, incumbent compactness and separation values (inc_comp_i and inc_sep_i) are also updated for each point in the data set. Every addition to SN does not necessarily improve the incumbents.

If the maximum number of iterations is not exceeded, Steps 2-6 are repeated. Otherwise, ACO-C terminates with the nondominated solutions in set SN. The ultimate aim of ACO-C is that the target clustering solution is included in the set of nondominated clustering solutions. Besides, not only a reasonable size of set SN which facilitates the evaluation of the decision makers, but also set SN which includes nondominated solutions within a wide range of the two criteria is preferred.

6.2.4. Computational Complexity of the ACO-C Algorithm

The worst-case time complexity of step 1 in ACO-C is O(n^2) due to the initialization of the pheromone amounts of the edges defined according to the neighborhoods. Step 2 constructs solutions in O(n^2) time. In step 3, the compactness calculation requires MST construction for each cluster in the data set, and each edge in the MST is evaluated according to the neighborhood of its end points. Thus, MST is constructed once more for the neighborhood of these end points. Hence, compactness calculation takes O(n^3) in the worst-case. Separation calculation includes finding the single-link edges between the clusters and evaluating them according to the edges in the MST of their neighborhoods. Therefore, worst-case time complexity of separation is O(n^3). The total time complexity of step 3 becomes O(n^3). The local search in step 4 is repeated for all cluster pairs until no improvement, so we cannot determine the total time complexity of step 4. Still, using the time complexity of the clustering evaluation function, we can infer that step 4 has MO(n^3) time complexity where M denotes the number of cluster pairs evaluated until no improvement. Step 5 updates the pheromone values of all the edges, so it takes O(n^2) time. Finally, step 6 updates the set of the nondominated solutions. Nondomination check is repeated for every solution generated in the current iteration and for every nondominated solution so far. Since we cannot determine an upper bound on the number of nondominated solutions, it is not possible to make a time complexity analysis for step 6.

We repeat steps 2 to 6 for every ant in the colony for maximum number of iterations. Consequently, the overall time complexity of ACO-C is MO(n^4) where M stands for the number of repetitions of the repeat-until loop.

6.3. Experimental Results for the ACO-C Algorithm

We test the performance of the ACO-C algorithm empirically. The algorithm was coded in Matlab 7.9 and run on a PC with Intel Core2 Duo 2.33 GHz processor and 2 GB RAM. The parameters in ACO strongly affect the performance, so, in a pilot group of data sets, we conduct a full factorial experiment in order to set the ACO-C parameters. After setting the parameters we test the performance of ACO-C in two groups of data sets.

6.3.1. Data Sets and Performance Evaluation Criteria

In our experiments we used two groups of data sets, namely groups 1 and 3. Group 1 data sets include various shapes of clusters (circular, elongated, spiral, etc.), they are with intracluster and intercluster density variations, and contain outliers. There are 45 data sets in this group, and 15 of these are used in the pilot experiment for parameter setting. Group 3 data sets are 3-dimensional data sets that are generated to test the strengths and weaknesses of the proposed approach in a controlled experiment. Properties and plots for groups 1 and 3 data sets are given in Appendix B.

Two performance evaluation alternatives are proposed in ACO-C: CERN and WCERN. We test the performance of both evaluation schemes, which are described in Section 6.2.

ACO-C yields a set of nondominated clustering solutions. Thus, we keep track of the number nondominated solutions in the set. The performance of a nondominated clustering solution is evaluated using three criteria: Jaccard index (JI), quasi-Jaccard index (QJI) and Rand index (RI) in every nondominated solution. JI and RI are well-known external cluster validity indices. JI focuses only on the number of point pairs that belong to the same target cluster and assigned to the same cluster, whereas RI also considers the number of point pairs that belong to different target clusters and assigned to different clusters. QJI is a relaxed version of JI, and it penalizes only the number of point pairs that belong to the same target cluster and assigned to different clusters.

The ultimate aim of ACO-C is that the target clustering solution is included in the set of nondominated solutions. In order to check this, we report the maximum JI, QJI and RI values in the set of nondominated solutions. Finally, the iteration number in which the target clustering is achieved is stored.

6.3.2. Parameter Settings for the ACO-C Algorithm

The main parameters of ACO-C are the number of ants (*no_ants*), the maximum number of iterations (*max_iter*) and evaporation rate (ρ). These three parameters have an important role on the exploration power and the convergence rate of ACO-C. Effect of parameters in ACO design has been addressed in the literature (Dorigo and Stützle 2004). However, there is not a unique method for parameter setting in ACO. Thus, we used a full factorial experimental design in order to determine the best parameter setting in ACO-C.

Three factors used in the experimental design and their levels are presented in Table 6.3. In addition to the number of ants (*no_ants*) and evaporation rate (ρ), the effect of evaluation function (FE) on the performance of ACO-C is explored further. In the experiments we set the maximum number of iterations, *max_iter*, to three times the number of points in the data set, *no_points*, and record the iteration number in which the target clustering is found (if it has). Thus, we do not consider the maximum number of iterations (*max_iter*) as a factor in our experimental design.

Table 6.3 Factors in ACO-C

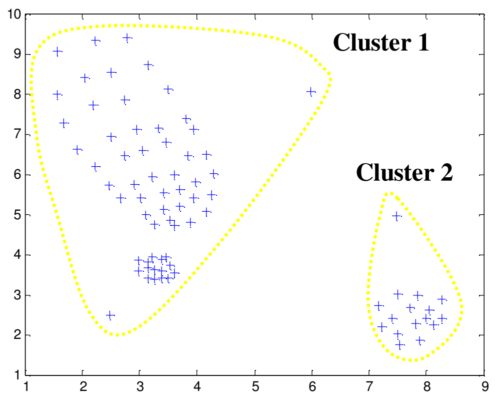
Factors	**Level 0**	**Level 1**
evaluation function, EF	CERN	WCERN
evaporation rate, ρ	0.01	0.05
number of ants, *no_ants*	5	10

In the 2^3 factorial design each factor combination is coded using the factor levels. That is, S_000 denotes the parameter setting in which all three factors are set to their level 0. The results are presented in Tables F.1 through F.8 in Appendix F. Table 6.4 reports the count of data sets out of 15 for which the target clustering is included among the set of nondominated solutions, SN. The number of such data sets varies between 11 and 13 for different parameter settings. However, only the CERN setting yields the target clusters for set data-uc-cv-nu-n. Besides, only the settings S_000 and S_111 find the target clusters for data-c-cv-nu-n and data-uc-cc-nu-n, respectively. Finding the target clusters of data_circle is possible only with the parameter settings, S_100 and S_110. When there exist intracluster density differences in the data set (e.g. data-uc-cv-nu-n and data-c-cv-nu-n), experiments with CERN can find the target clusters. WCERN is more effective when the intercluster and the intracluster edge lengths are significantly different.

An example for the set of nondominated solutions generated for data set data-c-cv-nu-n is as follows. ACO-C yields three nondominated clustering solutions when CERN is used as the evaluation function, and evaporation rate and number of ants are set to 0.01 and 5, respectively. The resulting nondominated clustering solutions and CERN values of each cluster are shown in Figure 6.14. Target clustering solution is included as seen in Figure 6.14 (c). When WCERN is applied instead of CERN, the number of nondominated solutions is less by one, and the target clustering solution is not one of these (Figure 6.15).

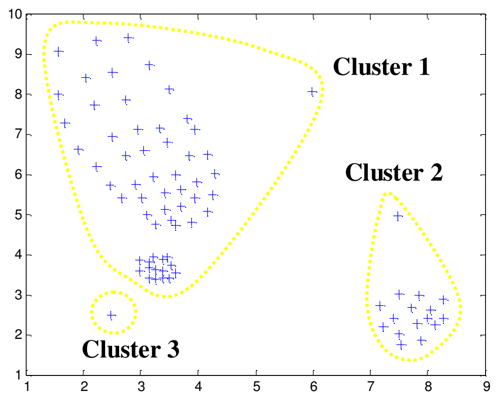
Table 6.4 Performance of ACO-C

Parameter Setting	**# of data sets for which target is covered in SN**	**Uncovered data sets**
S_000	13	data-uc-cc-nu-n, data_circle
S_001	11	data-c-cv-nu-n, data-uc-cc-nu-n, data-c-cc-nu-n_v2, data_circle
S_010	12	data-c-cv-nu-n, data-uc-cc-nu-n, data_circle
S_011	12	data-c-cv-nu-n, data-uc-cc-nu-n, data_circle
S_100	12	data-c-cv-nu-n, data-uc-cv-nu-n, data-uc-cc-nu-n
S_101	11	data-c-cv-nu-n, data-uc-cv-nu-n, data-uc-cc-nu-n, data_circle
S_110	12	data-c-cv-nu-n, data-uc-cv-nu-n, data-uc-cc-nu-n
S_111	12	data-c-cv-nu-n, data-uc-cv-nu-n, data_circle



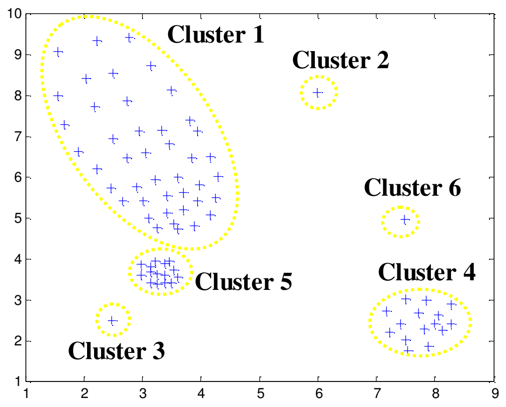
(a)

	Clusters	
	1	2
comp.	4.36	4.81
sep.	7.78	7.78



(b)

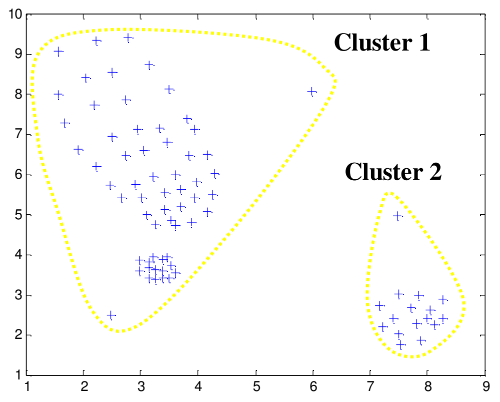
	Clusters		
	1	2	3
comp.	3.45	4.36	0
sep.	4.81	7.78	4.81



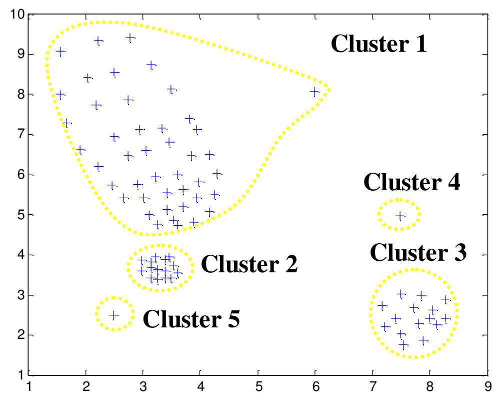
(c)

	Clusters					
	1	2	3	4	5	6
comp.	1	0	0	0.91	1.22	0
sep.	3.16	4.36	4.81	3.16	4.36	3.45

Figure 6.14 Nondominated clustering solutions for data-c-cv-nu-n for the factorial setting, S_000, (a) JI = 0.54, (b) JI = 0.57, (c) JI = 1.



(a)



(b)

	Clusters	
	1	2
comp.	5.43	1.14
sep.	25.45	5.33

	Clusters				
	1	2	3	4	5
comp.	1.14	0.24	0.81	0.00	0.00
sep.	5.33	1.32	1.68	1.13	2.25

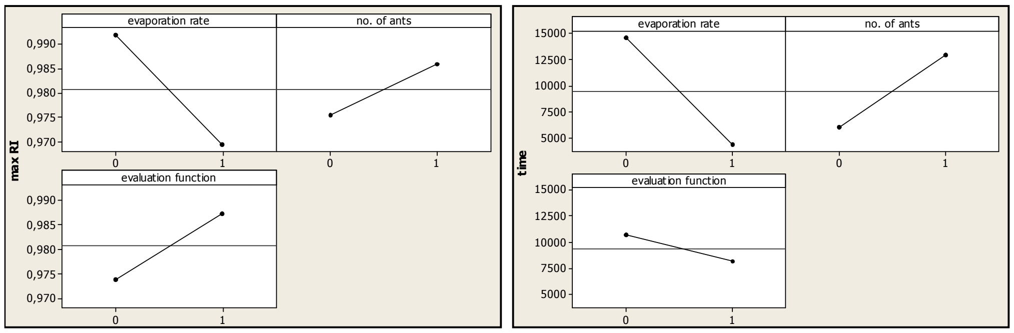
Figure 6.15 Nondominated clustering solutions for data-c-cv-nu-n for the factorial setting, S_100. (a) JI = 0.54. (b) JI = 0.97.

The effects of the three factors are analyzed for the two performance measures, namely the mean of the maximum RI values of nondominated solutions, and the average of computation time. The main effects plots for maximum RI and time are presented in Figure 6.16. The low setting of the evaporation rate slows the convergence rate and prevents ACO-C from finding suboptimal (missing the target) solutions. However, the time spent in ACO-C with this setting of the evaporation rate is three times larger. Increasing the number of ants used in ACO-C provides a slight improvement in maximum RI in return for negative effects on time. Although WCERN performs better than CERN in terms of both the maximum RI and time, CERN reaches the target clustering in some data sets (e.g. data-uc-cv-nu-n, data-c-cv-nu-n). Figure 6.17 underlines no significant interaction among the factors for either of the performance measures.

To sum up, the production runs are performed with the following parameter settings. Despite the increase in the time spent, evaporation rate is set to level 0 (0.01) due to its superior performance in maximum RI. The increase in the number of ants increases the time spent more significantly than the maximum RI, so we set the number of ants to level 0 (5). The nondominated solutions generated by WCERN and

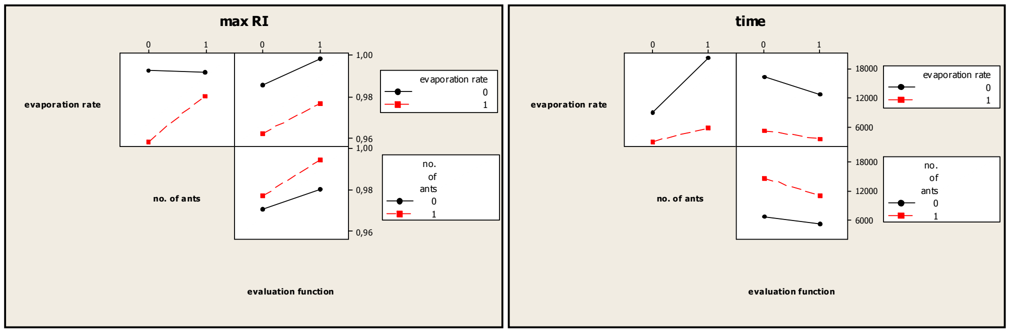
CERN can be different due to the different rewarding mechanisms of the clustering solutions. Thus, both WCERN and CERN are used for the evaluation purposes in independent runs of ACO-C, and the union of the resulting sets of nondominated solutions is taken as the final solution set. Consequently, we conduct our production runs with parameter settings S_000 and S_100.

In order to set the maximum number of iterations, we check the iteration number in which the target clustering is found (#ITA) and the ratio of this iteration number to the number of points in the data set (#ITA / #P) from Tables F.1 and F.8 in Appendix F. In both parameter settings #ITA is less than twice the number of data points (the maximum # ITA / #P ratio is 1.30 for train2 in S_000), so we set the maximum number of iterations as twice the number of data points.



(a) (b)

Figure 6.16 Main effect plots for (a) max RI. (b) time.



(a) (b)

Figure 6.17 Interaction effect plots for (a) max RI. (b) time.

6.3.3. Convergence of the ACO-C Algorithm

Convergence of an ACO algorithm can be analyzed from two perspectives: convergence in value and convergence in solution (Dorigo and Blum 2005). Convergence in value implies that ACO generates the optimal solution throughout the iterations at least once, i.e. selection probability of the optimal solution components is positive. Convergence in solution ensures that ACO repeats the optimal solution after reaching a steady state.

In ACO-C we initialize the pheromone values at $1/\rho$ as in T'kindt et al. (2002). Thus, the minimum amount of pheromone on an edge decreases to $(1-\rho)^{max\_iter}/\rho$ at the end of ACO-C due to evaporation. To ensure convergence in value, we set the maximum number of iterations, *max_iter*, so that the minimum amount of pheromone is a positive value.

An example is provided for data set data-c-cv-nu-n in Figure 6.18. Four closures are generated for this data set at the end of the NC algorithm, and two of them (closures 1 and 4) have outlier mixes. The point in closure 3 is already classified as an outlier at the end of NC. ACO-C results in three nondominated solutions as shown in Figure 6.14 when this data set is run with S_000 setting. The trajectories of the pheromone values for the extended neighbors of points 10 and 55 are presented in Figures 6.19 to 6.20, respectively.

ACO-C results in a set of nondominated solutions, so, instead of convergence to a single optimal solution, we consider convergence of the set of nondominated solutions ACO-C. We keep track of the set of nondominated solutions throughout the iterations. An example is displayed for data set data-c-cv-nu-n in Figure 6.21. The set of nondominated solutions stabilizes after iteration number 70 as the figure indicates.

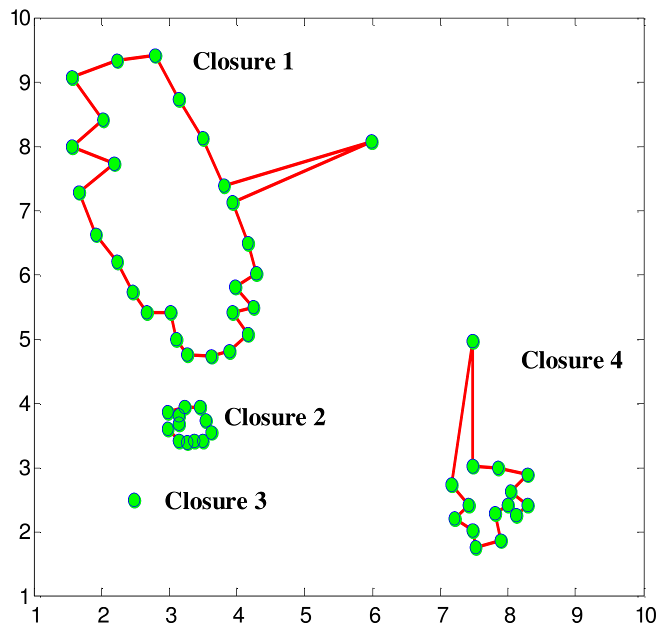
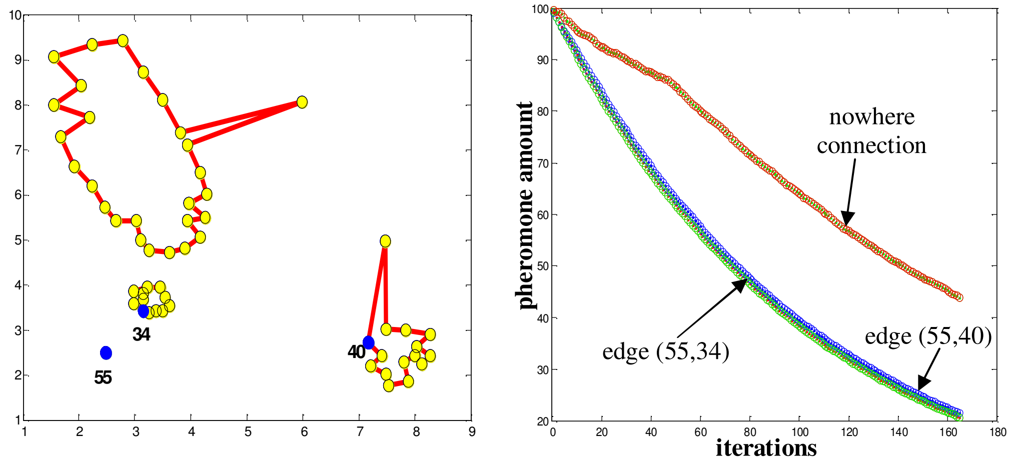
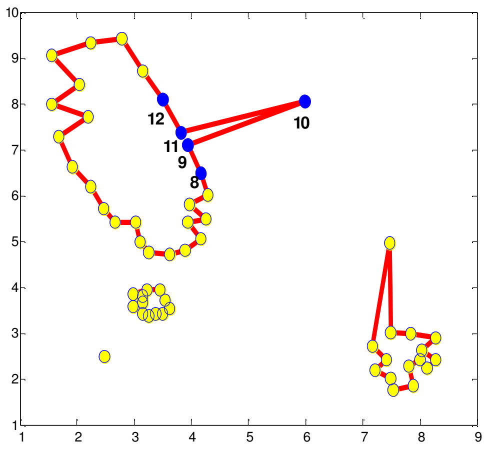


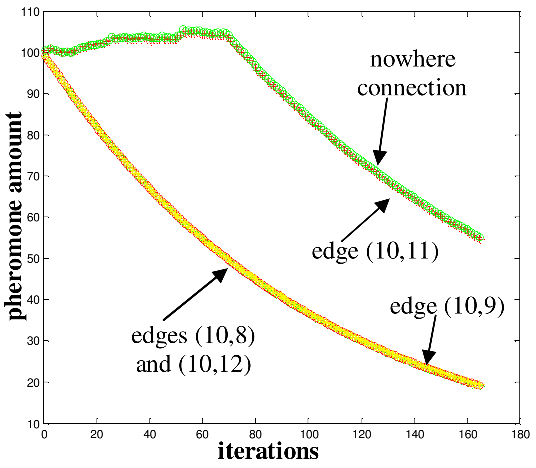
Figure 6.18 NC closures for data set data-c-cv-nu-n



(a) (b)

Figure 6.19 (a) Extended neighbors of point 55. (b) Trajectories of the pheromone values for the extended neighbors of point 55.





(a) (b)

Figure 6.20 (a) Extended neighbors of point 10. (b) Trajectories of the pheromone values for the extended neighbors of point 10.

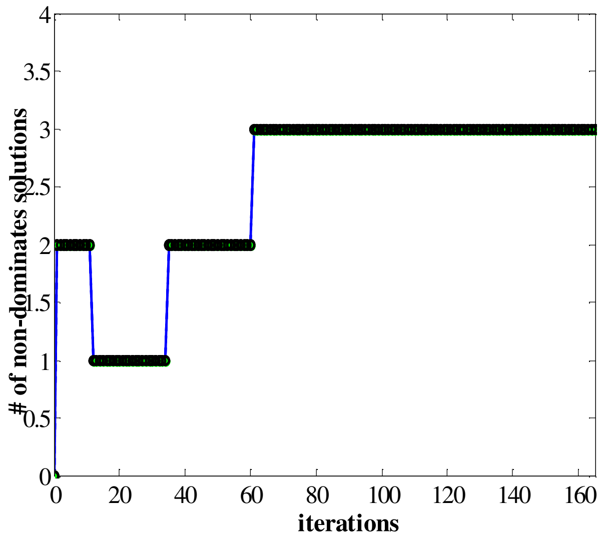


Figure 6.21 Convergence analysis for the example data set, data-c-cv-nu-n

6.3.4. Experiments with the ACO-C Algorithm

We test the performance of ACO-C in group 1 and group 3 data sets using the parameter settings in Section 6.3.2. Both evaluation methods, CERN and WCERN, are examined in these experiments. We present the details of the results for data sets in Tables F.9 through F.12 in Appendix F.

Results of ACO-C in Group 1 Data Sets

In group 1 data sets, ACO-C finds the target clusters in 21 data sets out of 45 using CERN as the evaluation method, and the same figure is 27 for WCERN. Taking the union of the resulting nondominated solutions generated by both evaluation mechanisms, we observe that target clusters are achieved in 29 data sets. Both CERN and WCERN find the target clusters in 19 data sets out of these 29 data sets. Only CERN setting achieves the target clusters in 2 data sets, and the contribution of WCERN setting is active in 8 data sets. When we examine the performances of the two evaluation functions in terms of mixing of clusters, CERN and WCERN mix clusters in 5 and 6 data sets, respectively.

The results of ACO-C in group 1 data sets are summarized in Table 6.5. In the table, the maximum JI, QJI and RI values are reported over the nondominated sets. The number of nondominated solutions generated by CERN and WCERN is within a range of 1 to 6. Hence, we can infer that the size of nondominated sets is conceivable for practical use.

As both CERN and WCERN can extract the target clusters, the maximum JI, QJI and RI values are all 1. The average performance of CERN and WCERN in terms of the maximum JI, QJI and RI do not differ significantly, and we can conclude that ACO-C is able to discover clusters close to the target clusters on the average. Nevertheless, the minimum and average values of these performance measures demonstrate the deficiencies of both methods, and they are worthwhile to elaborate.

The minimum values for the maximum JI in set SN are obtained for data_mix_uniform_normal and data_circle_1_20_1_19 with CERN and WCERN, respectively. In fact, neither CERN nor WCERN can find the target clusters of data_mix_uniform_normal and data_circle_1_20_1_19. In data set data_mix_uniform_normal, ACO-C perceives the data points around one of the target clusters as “local outliers”. Data set data_circle_1_20_1_19 is composed of two intermingling clusters in which the compactness of a target cluster is larger than the separation between the two clusters, in spite of scaling of these measures relative to neighborhoods with WCERN, the minimum value of the maximum RI in set SN is observed for the same data set as in the case of JI, namely data_circle_1_20_1_19, whereas CERN yields the least of the maximum RI’s in data set data_circle_20_1_5_10. Different from JI, RI rewards the data points correctly assigned to different clusters. Hence, we can infer that the small JI value of data_mix_uniform_normal arises from the division of clusters, so its RI value does not yield the minimum.

The minimum of the maximum QJI across the data sets is significantly higher for CERN compared to WCERN. The data set having the highest number of cluster mixes with CERN setting is data set train3 which includes noise. However, the highest number of cluster mixes is observed for data set data_circle_1_20_1_19 which includes intermixed clusters. As the intracluster and intercluster distances are very close, the weights used in WCERN cause mixing of clusters with a larger proportion.

Table 6.5 Summary of ACO-C results in group 1 data sets

	CERN				**WCERN**			
	#SN*	**max. JI in SN**	**max. QJI in SN**	**max. RI in SN**	**#SN***	**max. JI in SN**	**max. QJI in SN**	**max. RI in SN**
Min.	1	0.62	0.90	0.87	1	0.69	0.69	0.75
Max.	5	1	1	1	6	1	1	1
Average	2.86	0.94	0.99	0.97	2.98	0.96	0.98	0.98
Std. Dev.	1.21	0.08	0.02	0.04	1.21	0.08	0.07	0.05

#SN*: number of nondominated solutions
Min., Max., Average and Std.Dev. are for 45 data sets.

Consequently, CERN particularly contributes to the identification of target clusters distinguished in low resolution like data-c-cv-nu-n and data-uc-cv-nu-n. Taking into account the edge lengths, WCERN is more powerful in the extraction of target clusters visible in higher resolution such as data_circle and data_circle_1_20_1_11. Nevertheless, there exist 16 data sets for which neither CERN nor WCERN could find the target clusters. Typically, ACO-C has difficulty in detecting the target clusters when there exist data points that look like "local outliers" around the clusters. The relative evaluation mechanisms acting in both CERN and WCERN are very sensitive to density and distance changes, so these points are labeled as separate clusters. Moreover, we may apply ACO-C to a data set with noise. Although ACO-C yields the general structure of the target clusters, it forms clusters by enclosing noise as well. Another limitation of ACO-C is the intermingling clusters having very close intracluster and intercluster distances. In this case, the relative evaluation mechanism does not result in target clusters.

Results of ACO-C in Group 3 Data Sets

Group 3 is designed as a controlled experiment. We conduct a full factorial design, however, the increase in the number of dimensions and the number of data points inflates the execution times dramatically. Hence, we are able to complete the experiments for 12 data sets out of 24, for which intercluster density difference is set to no difference.

Both CERN and WCERN find the target clusters in the same 8 data sets out of 12. The summary of the results are shown in Table 6.6. The number of nondominated solutions generated by CERN and WCERN do not differ significantly. The average performances of both settings in terms of maximum JI, maximum QJI and maximum RI are also similar. In both settings, the minimum values of the maximum JI in set SN are obtained for the same data set, namely DS_0211. Furthermore, the data set DS_0211 has the lowest values of the QJI and RI values.

The performance of CERN and WCERN depends on the closures found by the neighborhood construction algorithm, a preprocessing step of ACO-C, due to the connectivity term in compactness evaluation (explained in Section 6.2.3). When the closures obtained from neighborhoods of NC result in divided clusters, CERN and WCERN stand as appropriate measures for merging them. However, when closures include cluster mixtures, CERN and WCERN no longer ensure distinguishing the mixed clusters. That is, forced breaking of the cluster mixes that are supplied by the NC algorithm inevitably worsens the connectivity. This results in a significant increase in the compactness objective, whereas separation does not change. In short, using such an evaluation scheme is not capable of neatly identifying the case of cluster mixes and tightly neighboring clusters. We can infer that the evaluation mechanisms relative to the neighborhoods do not work successfully in such data sets. On the other hand, for the cases of neighborhoods without any cluster mixes, performance measuring by CERN and WCERN ensure identification of the target clusters even if the clusters are located very close.

For both evaluation settings we analyze the effects of three factors, i.e. intracluster density, intercluster distance and existence of outliers, on two performance measures, i.e. maximum JI and time. The main and the interaction effects with CERN setting are displayed in Figures 6.22 and 6.23, respectively. The same effects for WCERN are provided in Figures 6.24 and 6.25.

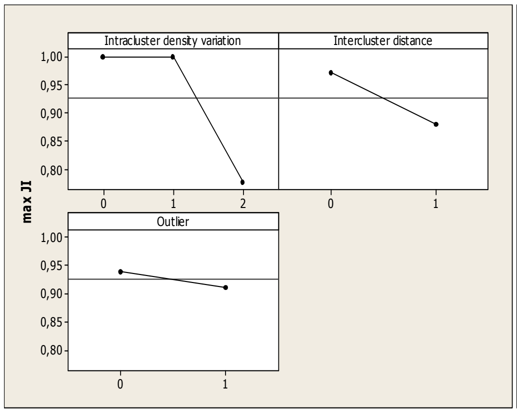
In both settings, the figures indicate that smooth density change disrupts the performance of ACO-C in terms of maximum JI. Even gradual changes in distances cannot be noted as local neighborhood properties. The speed in smooth changes in density may be the cause. Closer intercluster distance also decreases the maximum JI, whereas the effect of outliers' presence is negligible. The interaction plots demonstrate that there exists interaction between close intercluster distance and smooth intracluster density variation in terms of maximum JI. Since cluster mixes are observed in the neighborhood when there exist both smooth intracluster density change and close clusters, this interaction complies with the deficiency of the evaluation schemes. Interactions between the other factors are not significant.

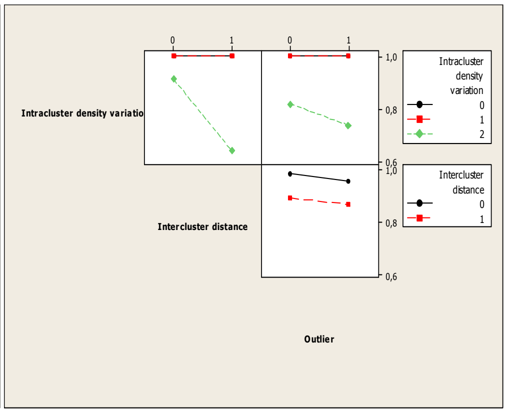
The time spent in ACO-C decreases for the level of random intracluster density variation due to the decrease in the number of points. Execution time of ACO-C is negatively affected from close intercluster distances, and outliers do not have significant effect on time.

Table 6.6 Summary of ACO-C results in group 3 data sets

	CERN				WCERN			
	#SN*	**max. JI in SN**	**max. QJI in SN**	**max. RI in SN**	**#SN***	**max. JI in SN**	**max. QJI in SN**	**max. RI in SN**
Min.	1	0.61	0.75	0.70	2	0.71	0.82	0.77
Max.	4	1	1	1	5	1	1	1.00
Average	2.42	0.93	0.96	0.95	3.25	0.93	0.97	0.96
Std. Dev.	0.90	0.14	0.09	0.10	0.97	0.12	0.06	0.07

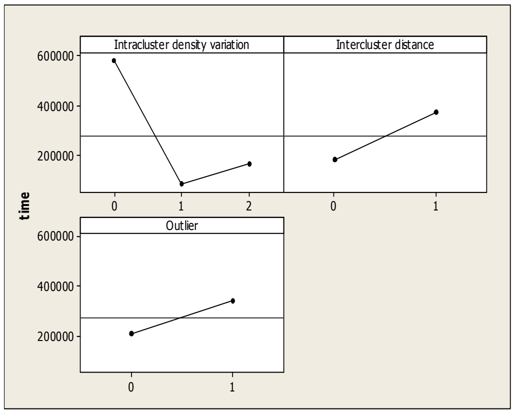
#SN*: number of nondominated solutions
Min., Max., Average and Std. Dev. are for 12 data sets.

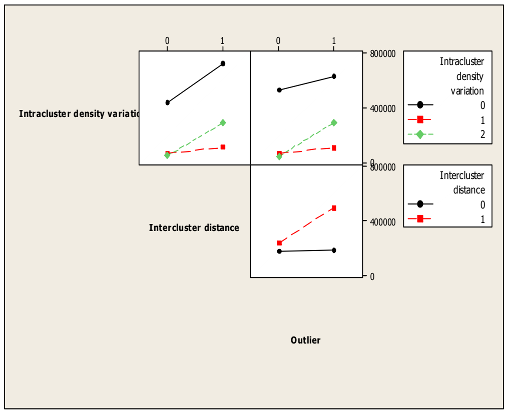




(a) (b)

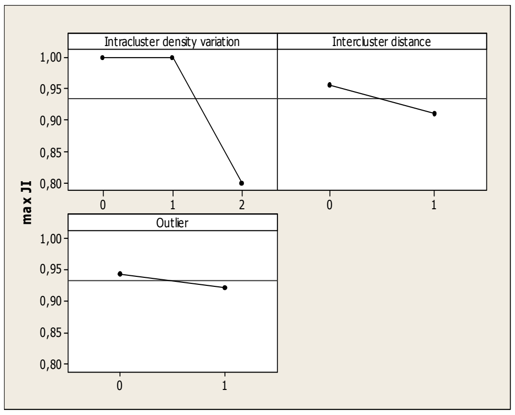
Figure 6.22 (a) Main effects of factors on max JI, (b) Interaction effects with CERN setting

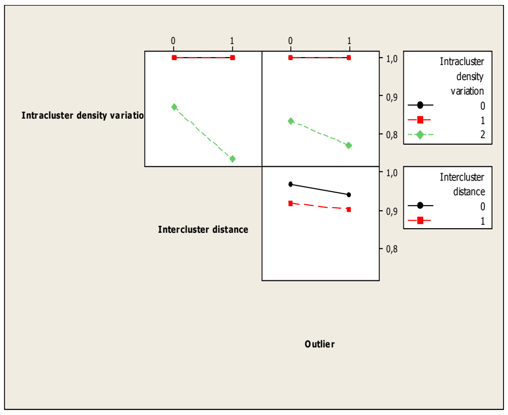




(a) (b)

Figure 6.23 (a) Main effects of factors on time, (b) Interaction effects with CERN setting





(a) (b)

Figure 6.24 (a) Main effects of factors on max JI, (b) Interaction effects with WCERN setting

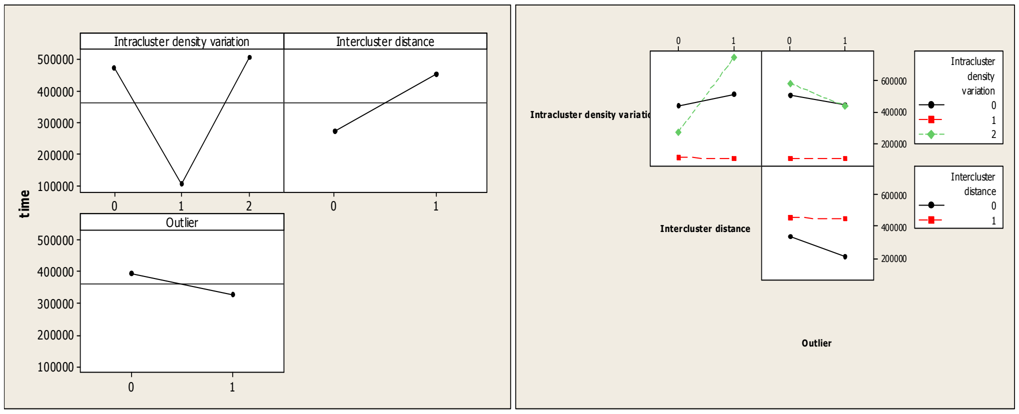


Figure 6.25 (a) Main effects of factors on time, (b) Interaction effects with WCERN setting

Data Set Reduction and Execution Time of ACO-C

The data set reduction (boundary formation) algorithms are different for 2- and higher dimensional space. Hence, we analyze the percentage of data set reduction in these two algorithms separately. We present a summary of the data set reduction percentages in Table 6.7.

On the average, the algorithm that is designed for 2-dimensional space, namely DTC, ensures a higher reduction compared to the algorithm that is able to work in higher dimensions. Nevertheless, the data set reduction typically depends on the properties of the data set. Thus, we analyze the correlation between the data set properties (explained in Appendix B) and the data set reduction.

In Figures 6.26 and 6.27, we demonstrate the scatter plots of the data set properties having significant correlations with the data set reduction for 2- and higher dimensional data sets, respectively. In 2- and higher dimensional spaces, the data set reduction is more for the data sets with well-separated clusters, i.e. when the minimum separation-to-compactness ratio (MSCR) is high. In fact, this is relevant to the DT construction in the proposed algorithms. We display an example in Figure 6.28. The short intercluster distance between the two clusters in Figure 6.28 (a) provide to discover the non-convex parts of the boundary (shown as dashed line), whereas in Figure 6.28 (b) the larger intercluster distances bring about skinny triangles between clusters 1 and 2, and the non-convex region in cluster 1 could not be detected. When the distance between the two clusters is smaller, the cavity is identified by the algorithm. To sum up, higher MSCR yields higher data set reduction. Maximum compactness also has a negative correlation with data set reduction. Large distances in a cluster bring about zigzags on the boundary, and this causes a decrease in the data set reduction.

Table 6.7 The percentages of data set reduction

	2-dimensional data sets	**Higher dimensional data sets**
Min.	1.52%	4.90%
Max.	74.29%	53.84%
Average	42.79%	19.11%
Std. Dev.	20.42%	12.41%

#SN*: number of nondominated solutions
Min., Max., Average and Std. Dev. are for 42 2-dimensional data sets in group 1 and 27 higher dimensional sets in groups 1 and 3.

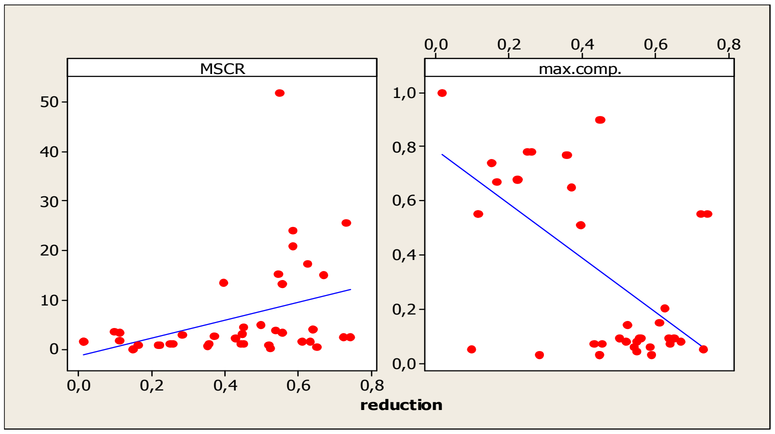


Figure 6.26 Scatter plots of MSCR and maximum compactness versus data set reduction for 2-dimensional data sets (correlation coefficients are 0.371 and 0.418, respectively)

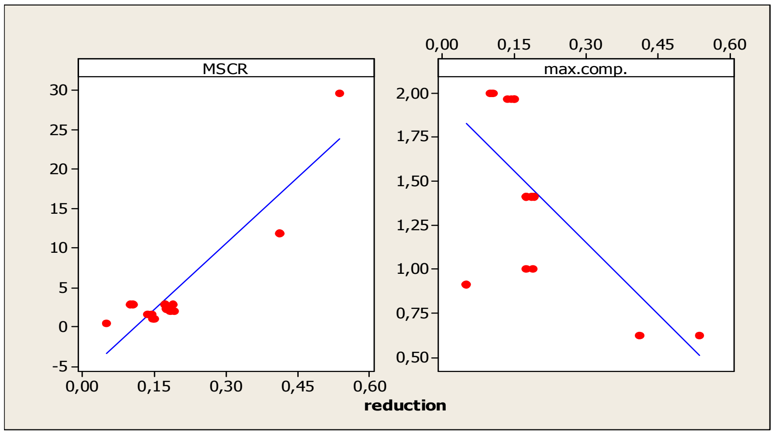
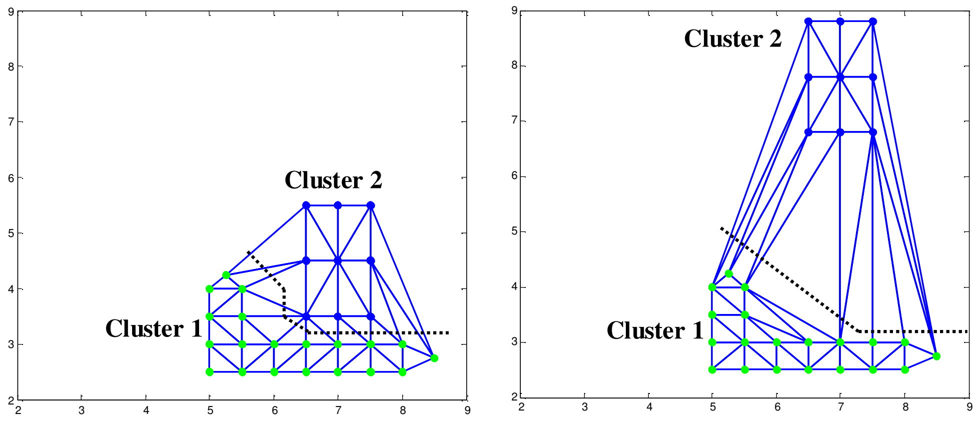


Figure 6.27 Scatter plots of MSCR and maximum compactness versus data set reduction for higher dimensional data sets (correlation coefficients are 0.925 and 0.581, respectively)



(a) (b)

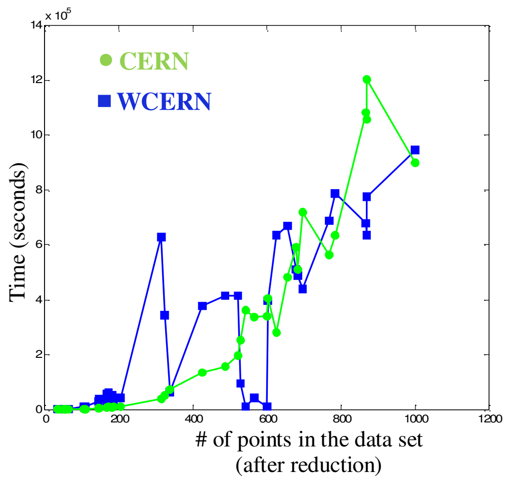
Figure 6.28 An example for Delaunay triangulation construction

Considering the data set reduction, the execution times of ACO-C are provided in Tables F.9 to F.12 in Appendix F. Figures 6.29 (a) and (b) display the number of points in the data set (after reduction) versus the execution time with 2- and higher dimensional data sets, respectively. For each data set CERN and WCERN settings are shown separately. The results in the figures indicate that despite the data set reduction the execution times increase dramatically with the increase in the number of points and the dimensions. The time spent with the two evaluation settings does not show a significant difference.

We have already determined a lower bound for the time complexity of ACO-C in Section 6.2.4. Nevertheless, we analyze the time spent in each step of ACO-C to gain an understanding of actual times. In Figure 6.30 we demonstrate the percentage of time spent in each step of ACO-C for a sample of data sets. The figure points out that the bottleneck operation is step 4, namely the local search, on the average.

In order to understand the complexity of the local search algorithm, we analyzed the number of cluster pairs evaluated for merging and the time spent in the local search steps in an example data set with 55 data points, i.e. data-c-cv-nu-n. The average number of cluster pairs evaluated per ant per iteration is 21.56. In an iteration, an ant spends 0.39 seconds in the local search on the average. On the average, 3.05% of this time is spent for the selection of the clusters to be merged, and the calculation of the compactness and separation measures (explained in Section 6.2.3) after merging takes 96.55% of this time. Remaining 0.40% is spent for checking the improvement in compactness and separation measures. These figures show that the evaluation of the new solution is the most time consuming operation in the local search. Thus, lengthy comparisons of the many underlying cluster combinations are mainly to blame for the increase in the local search time.

Inspite of its computational burden, the local search compensates the undesirable divisions of the clusters. It strengthens the exploitation property of ACO-C. For this reason, we need to develop a bounding mechanism on the compactness and separation calculations or new local search mechanisms.



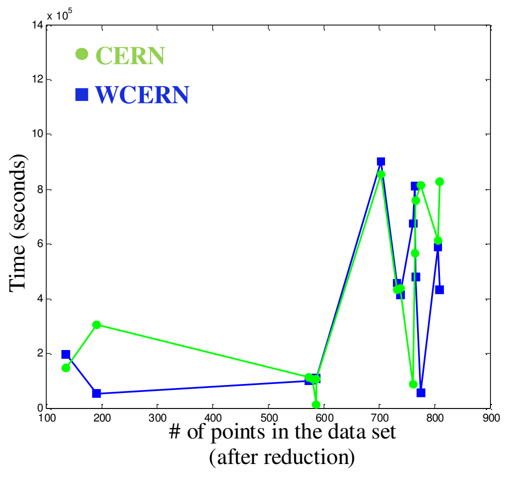


Figure 6.29 The number of points in the data set (after reduction) versus the execution time (in seconds) with (a) 2-dimensional data sets. (b) higher dimensional data sets.

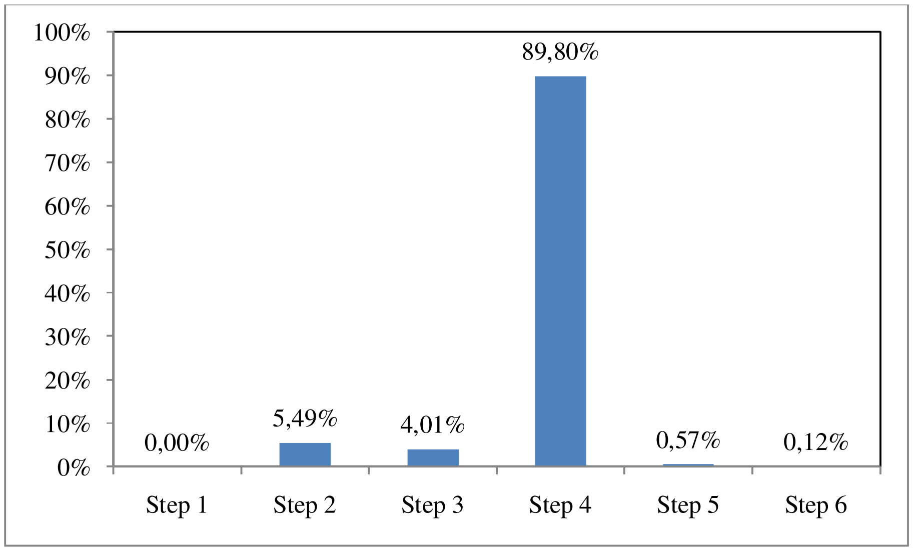


Figure 6.30 The average of the total time spent in each step of ACO-C

Comparison of the ACO-C Algorithm with the Competing Clustering Approaches

The performance of ACO-C is compared with the results of *k*-means, single-linkage, DBSCAN, NC closures, outlier detection of NOM, and NOM. In our comparison *k*-means represents the partitional clustering approach and single-linkage the hierarchical clustering approach. DBSCAN is selected as a representative of the density-based clustering algorithms. In order to have a fair comparison among these algorithms, *k*-means and single-linkage are run for several values of *k* (i.e. the number of clusters in the single-linkage) in the range between 2 and 10% of the points in the data set with increments of 1, and the one with the best JI is used for each algorithm. In the same manner, for DBSCAN, among several *MinPts* settings the one with the best JI is selected for comparison. For ACO-C, we consider the union of the nondominated solutions with CERN and WCERN settings. In order to have a fair comparison in terms of time, we use the summation of the execution times of both CERN and WCERN for ACO-C.

The results are summarized in Tables 6.8 and 6.9 for group 1 and group 3 data sets, respectively. In both groups, the single-linkage algorithm gives the highest number of data sets for which the target clusters are found, and ACO-C and NOM follow single-linkage. In group 1, ACO-C has the best JI and RI values, and NOM and single-linkage come after ACO-C. Furthermore, QJI values of NC, outlier detection and ACO-C outperform the remaining approaches.

In group 3, single-linkage correctly finds the target clusters in all data sets. Albeit DBSCAN and *k*-means are superior to ACO-C in terms of JI and RI, in ACO-C the number of data sets for which target clusters are achieved is higher compared to both algorithms. In group 3, ACO-C outperforms NOM.

The computational complexity of *k*-means, single-linkage, DBSCAN, NOM and ACO-C algorithms are O(kn), O(n^2), O(n log$n$), MO(n^3), and MO(n^4), respectively. Hence, ACO-C has the highest computational complexity among these competing approaches.

Main limitations of NC, NOM and ACO-C are high execution times compared to *k*-means, single-linkage and DBSCAN. In this context, improvements are required.

Table 6.8 Comparison of ACO-C with *k*-means, single-linkage, NC closures, outlier detection of NOM, and NOM for group 1 data sets (45 data sets)

		***k*-means**	**Single-linkage**	**DBSCAN**	**NC**	**Outlier detection**	**NOM**	**ACO-C**
# of data sets TC* is found		9	32	17	13	16	23	29
	average	0.756	0.937	0.94	0.875	0.875	0.955	0.971
	std.dev.	0.231	0.163	0.139	0.128	0.137	0.088	0.054
JI	**min**	0.278	0.453	0.504	0.558	0.456	0.591	0.806
	average	0.856	0.955	0.963	0.908	0.908	0.967	0.989
	std.dev.	0.138	0.119	0.095	0.101	0.107	0.065	0.022
RI	**min**	0.58	0.532	0.531	0.659	0.639	0.648	0.895
	average	0.954	0.947	0.972	0.996	0.998	0.981	0.994
	std.dev.	0.087	0.145	0.097	0.016	0.012	0.08	0.021
QJI	**min**	0.659	0.46	0.504	0.905	0.916	0.593	0.901
	average	0.723	6.484	2.580	64272	6387	70006	531233
	std.dev.	0.769	8.262	3.187	408825	14273	421898	368465
	min	0.045	0.381	0.034	1.240	1.076	2.382	1047
time	**max**	2.714	32.597	12.857	2714383	82117	2820103	1458101

TC*: Target clusters

Table 6.9 Comparison of ACO-C with *k*-means, single-linkage, NC closures, outlier detection of NOM, and NOM for group 3 data sets (12 data sets)

		***k*-means**	**Single-linkage**	**DBSCAN**	**NC**	**Outlier detection**	**NOM**	**ACO-C**
# of data sets TC* is found		0	12	6	3	3	3	8
	average	0.968	1	0.990	0.927	0.928	0.884	0.941
	std.dev.	0.030	0	0.024	0.124	0.124	0.199	0.106
JI	**min**	0.903	1	0.922	0.669	0.669	0.404	0.708
	average	0.992	1	0.998	0.977	0.977	0.948	0.958
	std.dev.	0.008	0	0.006	0.042	0.042	0.104	0.077
RI	**min**	0.975	1	0.981	0.888	0.888	0.653	0.771
	average	0.975	1	0.999	0.949	0.951	0.893	0.974
	std.dev.	0.016	0	0.002	0.105	0.104	0.192	0.056
QJI	**min**	0.941	1	0.994	0.725	0.728	0.422	0.822
	average	0.423	3.076	1.051	6676	863	7551	768912
	std.dev.	0.083	1.694	0.208	7264	652	7219	466625
	min	0.301	1.448	0.760	1903	257	2494	122351
time	**max**	0.575	6.471	1.251	24099	2250	24388	1378745

TC*: Target clusters

6.4. Discussion of the ACO-C Algorithm

ACO-C is a novel clustering algorithm which is based on ACO, one of the renowned swarm intelligence approaches. It takes the connectivity and proximity relations extracted by the neighborhood construction algorithm as input. In ACO-C, we combine the connectivity, proximity, density and distance information with the exploration and exploitation capabilities of ACO in a bicriteria setting. In order to handle unknown number of clusters and arbitrary shaped clusters with density variations, we suggest a new ant representation scheme and two clustering evaluation mechanisms relative to neighborhoods. ACO-C yields a set of nondominated solutions which hopefully includes the target clustering. Furthermore, we take into account scalability using a data set reduction algorithm as a preprocessing tool.

The performance of ACO-C is tested on various data sets. The results indicate that when our neighborhoods reflect the density, connectivity and proximity relations among the points with the least number of cluster mixes, and within the neighborhoods of the points subject to separation, the intracluster distances are not greater than the separation value (i.e. intercluster distance), ACO-C is able to find clustering solutions very close to the target clusters. In such data sets, ACO-C outperforms the other competing approaches in terms of JI, QJI and RI. Particularly, the bicriteria evaluation mechanisms with measures relative to the neighborhoods enhance the extraction of the arbitrary shaped clusters and density variations.

However, ACO-C has complications in the following cases exclusively or in combinations: (1) The data set has intermingling clusters. (2) The data set includes noise. (3) The neighborhoods constructed include several cluster mixes, and at the same time, the intercluster distance between two clusters is not greater than the intracluster distances within the neighborhoods of the closest points between these two clusters. (4) There exist clusters with local outliers.

Particularly, single-linkage and DBSCAN have superior performance for the data sets having clusters visible in the high resolution. The intense use of local perspectives and the neighborhoods in ACO-C enhances a sensitive evaluation of density, connectivity and distance issues, and this provides not only homogeneous clusters but also divided clusters. On the other hand, the top-down hierarchy in single-linkage captures the global view inherent in the clusters, and fewer divided clusters are likely seen.

Although single-linkage delivers nice results using a simple minded approach in short times, actually, it cannot find the number of clusters in a data set automatically. Instead it forms a hierarchy of data points, and the number of clusters is determined via a cutoff point. As we select the clustering solution with the best JI value in our single-linkage experiments, this procedure resolves the unknown number of clusters issue and single-linkage gains advantage of this. In compactness and separation calculations, the single-linkage method does not evaluate the edges relative to the neighborhoods. Thus, the performance of the single-linkage method worsens, when there exist longer edges within a cluster compared to the edge lengths subject to separation. ACO-C is able to find the target clusters in such data sets.

ACO-C achieves a reasonable number of non-dominated solutions to be used for practical purposes. Although the data set reduction mechanism provides a significant improvement in certain data sets, the time complexity of ACO-C is still high. Particularly, we need to develop mechanisms to reduce execution times.

CHAPTER 7

CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH

In this dissertation, we have considered the clustering problem in data sets with unknown number of clusters having arbitrary shapes and which display density variations. This problem can be observed in spatial data sets such as those in geographic information systems, city planning, image segmentation for computer aided systems and biomedical applications. Our main motivation has been automation of this type of clustering tasks in order to facilitate the identification of distinct properties of the data sets.

We introduced a new clustering methodology for this purpose. The proposed methodology included mechanisms for the extraction of neighborhood characteristics in a data set and the reduction of a data set. Two clustering approaches were developed based on neighborhood construction, namely a hierarchical density-based clustering algorithm and a swarm intelligence based clustering algorithm. We demonstrated the strengths and the weaknesses of the proposed methods empirically.

We started by extracting the neighborhood characteristics of the data points to enable effectivity in clustering. In addition to the widely used proximity and direction concepts in the literature, our parameter-free neighborhood construction approach takes into account the density-based connectivity facilitated by Gabriel graph and determines a unique neighborhood for each data point. We tested the performance of the proposed approach on various data sets particularly in arbitrary shaped and well-separated clusters with density variations, the proposed approach yields effective and relatively more purified neighborhoods compared to other proximity and distance based neighborhood approaches. The resulting tight and homogeneous neighborhoods facilitate the succeeding clustering process. Typically, the algorithm has limited performance in two cases, namely data sets with noise and data sets in which the intercluster distances are small compared to intracluster distances. Accordingly, the algorithm needs to be extended using the analysis of the proximity, density and connectivity information in a more global and collective perspective.

The proposed neighborhood construction algorithm is a generic approach for identification of local characteristics. Hence, it can be used as a preprocessing step for any kind of clustering method in spatial data sets. Since the uniquely constructed neighborhoods are composed of points with similar characteristics, the algorithm may also be used as a nearest-neighbor classifier for the classification problem. Another extension of the neighborhood construction algorithm can be clustering validation. The neighborhoods constructed for data points can form a basis for connectivity comparisons.

Secondly, we examined the scalability problem in clustering through the use of external shape generation in computational geometry. We suggested the use of the boundary points only for outlier detection and merging purposes in clustering. Thus, given a partitioning (subclusters or clusters) of a data set, we introduced two approaches based on Delaunay triangulation for formation of non-convex cluster boundaries. The former approach is a parameter-free algorithm, and it works in 2-dimensional space. The latter is based on user specified elongation thresholds in order to control the degree of non-convexity to be considered, and it is designed to work in higher dimensions. We analyzed the accuracy and the percentage of data set reduction for both algorithms using numerical experiments. The former approach was found to be effective in the generation of the external shapes when the edge length in the clearing of a non-convex cavity is larger than the width or diameter of the cavity. The latter produced satisfactory results when the elongation thresholds are set properly. We achieved significant reduction percentages for the data sets especially in convex clusters. The proposed approaches can also be used for visualization of cluster boundaries and external shape generation in computational geometry.

Thirdly, we proposed a three-phase clustering algorithm, namely NOM, for data sets with spatial characteristics. The algorithm includes neighborhood construction, outlier detection, and merging of subclusters phases, which combine density and distance information with graph theoretic concepts like proximity and connectivity. The criteria we define for outlier detection and merging of clusters are based on the assessment of the edge lengths relative to their neighborhoods in order to handle arbitrary shapes and density variations in distance calculations. We evaluated the performance of the algorithm using data sets having various properties, and we conducted a benchmark analysis with some well-known competing approaches. The numerical results indicated that the proposed approach is capable of finding clustering solutions close to the target clusters with arbitrary shapes and different densities when the intercluster distances are larger than the intracluster distances. Due to the lack of global view in the algorithm, excessive number of cluster divisions is observed in the data sets having clusters in a relatively higher resolution. Moreover, the proposed approach had limited success in data sets with noise.

Finally, we inquired the applicability of swarm intelligence to clustering using ACO. We used the neighborhoods and the subclusters, and the external shapes of the subclusters in order to address locality and scalability issues. We proposed a new ant representation scheme that is capable of finding the number of clusters, arbitrary shapes and density variations. For ACO, we suggested a bicriteria evaluation setting in which compactness and separation measures are considered relative to the neighborhoods. The performance of ACO applied to clustering, namely ACO-C, was tested using two groups of data sets with arbitrary shaped clusters, intracluster and intercluster density variations. One group was taken from open sources, and the other was for controlled experiments. The numerical results were compared with the well-known clustering approaches. Finally, the capabilities and deficiencies of the proposed approach were elaborated.

ACO-C is capable of finding the target number of clusters and extracting the well-separated arbitrary shaped clusters with density variations when proper neighborhoods are provided. It can handle data sets with outliers, whereas its performance is limited with noise. Furthermore, smooth density changes within the clusters affect the performance of ACO-C negatively. With these results, ACO-C outperforms the well-known clustering approaches and our three-phase clustering algorithm.

The main limitation of our proposed approaches is their long execution times. Thus, there is need for the development of efficient algorithms. Particularly, in ACO-C fast and efficient local search algorithms and bounding mechanisms in the clustering evaluation are worthwhile to examine further. Moreover, parallelization and distributed computing can be considered to speed up our work.

There is also room for improvement in the neighborhood construction, NOM and ACO-C algorithms. In order to prevent mixing of clusters for the data sets in which intercluster distances are smaller compared to within cluster distances, the proximity, density and connectivity information can be considered in a collective manner. Furthermore, evaluation mechanisms with a more global view can be developed in order to resolve the division of clusters in these three approaches. Isolation of outliers is another improvement direction for the neighborhood construction.

As for computational experiments, the performance of the proposed methodology can be tested on real-life clustering problems. Hence, the capabilities and the weaknesses of the proposed approach can be identified thoroughly.

In this work, we aim to perform clustering, boundary formation and outlier detection in the data sets with unknown number of clusters having arbitrary shapes, intracluster and intercluster density variations. Consideration of these challenging issues all together makes our clustering task quite complicated, and this results in excessive computation time. As a remedy, decomposition of the clustering problem into manageable subproblems can be considered in the future.

The inherent clustering pattern in a data set can be observed in different scales of local and global perspectives. Moving from local to global scale, positioning on the appropriate scale is a challenge that requires collective information of distance, density, direction, and proximity. In this dissertation, we deepened our understanding on the local perspective, and we also barely touched upon the global scale. Nevertheless, the identification of the appropriate level of global and local scale at the right moment along the computations is still a field to explore further.

An interesting future search direction can be constrained clustering. Typically, constraints represent the problem-specific information and the preferences of a domain expert in a clustering problem, and their guidance enables the effectivity and efficiency in clustering. Thus, it is possible to incorporate constraint handling mechanisms to the proposed approaches.

In nature, several types of swarm struggle with obstacles during their daily-life activities such as food search and sheltering. Nevertheless, they are able to survive using the collective intelligence. Thus, swarm intelligence is a promising field for constrained clustering due to the analogy between obstacles faced by a swarm and constraints in clustering.

Another extension for future research can be handling data types. In this dissertation, we considered numerical data sets and measured dissimilarity using Euclidean distances. It is possible to explore the effect of other similarity/dissimilarity measures on the proposed approaches.

REFERENCES

Aarts, E. and Korst, J., 1989. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons.

Abraham, A., Das, S., and Roy, S., 2008. Swarm intelligence algorithms for data clustering. In Maimon, O. and Rokach, L., editors, *Soft Computing for Knowledge Discovery and Data Mining*, 279-313. Springer, Berlin.

Agrawal R., Gehrke J., Gunopulos D., and Raghavan P., 1998. Automatic subspace clustering of high dimensional data for data mining applications. *In: Proceedings of the ACM SIGMOD International Conference on Management of Data*, 94-105.

Ahmadi, A., Karray, F., and Kamel, M.S., 2010. Flocking based approach for data clustering. *Natural Computing*, 9 (3), 767-791.

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

Alani, H., Jones, C.B., and Tudhope D., 2001. Voronoi-based region approximation for geographical information retrieval with gazetteers. *International Journal of Geographical Information Science*, 15 (4), 287-306.

Alexander, F. E., Storm, H., Olsen, J., Teppo, L., Carli, P.M., Michaelis, J., Petridou, E., Terracini, B., Boyle, P., Van Der-Does-Van Den Berg, A., Peris-Bonet, R., Adami, H., Ekbom, A., Draper, G., Muir, C. S., Mckinney, P., Vatten, L., Levi, F., Mcwhirther, W, Plesko, I., Pompe-Kirn, V., Rahu, M., 1998. EUROCLUS: Clustering of childhood leukemia in Europe. In Baig, S.S, editor, *Cancer Research Supported under BIOMED 1*, IOS Press, Amsterdam.

Anderberg, M.R., 1973. *Cluster Analysis for Applications*. Academic Press, New York.

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

Azzag, H., Venturini, G., Oliver, A., and Guinot, C., 2007. A hierarchical ant based clustering algorithm and its use in three real-world applications. *European Journal of Operational Research*, 179 (3), 906-922.

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

Barber, C.B., Dobkin, D.B., and Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software, 22 (4), 469-483.

Basu, S. and Davidson, I., 2011. KDD 2006 Tutorial Clustering with Constraints: Theory and Practice. (http://www.ai.sri.com/~basu/kdd-tutorial-2006, last accessed on March 2, 2011).

Ben-Dor A., Shamir R., and Yakhini Z., 1999. Clustering gene expression patterns. *Journal of Computational Biology*, 6, 281–297.

Berkhin, P., 2006. A survey of clustering data mining techniques. In Kogan, J., Nicholas, C., and Teboulle M., editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25-71. Springer, Berlin.

Bezdeck, J.C., Ehrlich, R., and Full, W., 1984. FCM: Fuzzy *c*-means algorithm. *Computers and Geoscience*, 10, 191-203.

Bilgin, C., Demir, C., Nagi, C., and Yener, B., 2007. Cell-graph mining for breast tissue modeling and classification. *In: Proceedings of the Twenty-ninth Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Lyon, France, 5311-5314.

Birch, J.M., Alexander, F.E., Blair1, V., Eden, O.B., Taylor, G.M., and McNally, R.J.Q., 2000. Space-time clustering patterns in childhood leukemia support a role for infection. *British Journal of Cancer*, 82 (9), 1571-1576.

Blum, C., 2005. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2 (4), 353-373.

Boryczka, U., 2009. Finding groups in data: cluster analysis with ants. *Applied Soft Computing*, 9 (1), 61-70.

Breunig, M.M., Kriegel, H.P., Raymond, T.N., and Sander, J., 2000. LOF: identifying density-based local outliers. *ACM SIGMOD Record*, 29 (2), 93-104.

Burl, M.C., Asker, L., Smyth, P., Fayyad, U., Perona, P., Crumpler, L., and Aubele, J., 1998. Learning to Recognize Volcanoes on Venus. *Machine Learning*, 30 (2-3), 165-194.

Chaudhuri, A.R., Chaudhuri, B.B., and Parui, S.K., 1997. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68 (3), 257-275.

Chaudhuri, B.B., 1996. A new definition of neighborhood of a point in multi-dimensional space. *Pattern Recognition Letters*, 17 (1), 11-17.

Chen, A.P. and Chen, C.C., 2006. A new efficient approach for data clustering in electronic library using ant colony clustering algorithm. *The Electronic Library*, 24 (4), 548-559.

Chu, S.C., Roddick, J.F, Su, C.J., and Pan, J.S., 2004. Constrained ant colony optimization for data clustering. In Zhang, C., Guesgen, H. W., and Yeap, W. K., editors, *Lecture Notes in Computer Science, Volume 3157. PRICAI 2004: Trends in Artificial Intelligence*, pages 534-543. Springer, Berlin.

Dahlhaus, E., 2000. Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition, *Journal of Algorithms*, 36 (2), 205-240.

Dave, R.N., 1992. Adaptive fuzzy *c*-shells clustering and detection of ellipses. *IEEE Transactions on Neural Networks*, 3 (5), 643-662.

Deneubourg J.L., Goss S., Frank N., Sendova-Franks A., Detrain C., and Chretien L., 1991. The dynamics of collective sorting: robot-like ant and ant-like robot. In Meyer, J.A. and Wilson, S.W., editors, *In: Proceedings of the First European Conference on Simulation of Adaptative Behavior: From Animal to Animats*, MIT Press, Cambridge, pages 356–365.

Deng, M., Liu, Q., Cheng, T., and Shi, Y., 2011. An adaptive spatial clustering algorithm based on Delaunay triangulation. *Computers, Environment and Urban Systems*, 35 (4), 320-332.

Dockerty, J.D., Skegg, D.C., Elwood, J.M., Herbison, G.P., Becroft, D.M., Lewis, M.E., 1999. Infections, vaccinations, and the risk of childhood leukemia. *British Journal of Cancer*, 80 (9), 1483-1489.

Dorigo, M. and Blum, C., 2005. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344 (2-3), 243-278.

Duckham, M., Kulik, L., Worboys, M., and Galton, A., 2008. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41 (10), 3224-3236.

Eberhart, R.C. and Shi, Y., 2001. Particle swarm optimization: Developments, applications and resources. *In: Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, Korea, 81-86.

Edelsbrunner, H., 1992. Weighted alpha shapes. *Technical Report UIUCDCS-R-92-1760*, Department of Computer Science, University of Illinois, Urbana, IL.

Edelsbrunner, H. and Mücke, E.P., 1992. Three-dimensional alpha shapes. *In: Proceedings of the 1992 Workshop on Volume Visualization*, Boston, MA, 75-82.

Edelsbrunner, H., Kirkpatrick, D.G., and Seidel, R., 1983. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29 (4), 551-559.

Ertoz, L., Steinbach, M., and Kumar, V., 2003. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. *In: Proceedings of the International Conference on Data Mining*, 47-58

Eschrich, S., Ke, J., Hall, L., and Goldgof, D., 2003. Fast accurate fuzzy clustering through data reduction. *IEEE Transaction on Fuzzy Systems*, 11 (2), 262-270.

Ester, M., Kriegel, K.P., Sander J., and Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, 226-231.

Estivill-Castro, V. and Lee, I., 2000. AMOEBA: Hierarchical clustering based on spatial proximity using Delaunay diagram. *In: Proceedings of the Ninth International Symposium on Spatial Data Handling,* Beijing, China.

Estivill-Castro, V. and Lee, I., 2001. AUTOCLUST+: Automatic clustering of point-data sets in the presence of obstacles. *Temporal, Spatial, and Spatio-Temporal Data Mining, Lecture Notes in Computer Science*, 2007/2001, 133-146.

Fathian, M., Amiri, B., and Maroosi, A., 2007. Application of honey-bee mating optimization algorithm on clustering. *Applied Mathematics and Computation*, 190 (2), 1502-1513.

Frank, A. and Asuncion, A., 2010. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. (http://archive.ics.uci.edu/ml, last accessed on February 21, 2011).

Gabriel, K.R. and Sokal, R.R., 1969. New statistical approach to geographic variation analysis. *Systematic Zoology*, 18 (3), 259-278.

Garai, G. and Chaudhuri, B.B., 1999. A split and merge procedure for polygonal border detection of dot pattern. *Image and Vision Computing*, 17 (1), 75-82.

Garey, M.R. and Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman.

Goil S., Nagesh H. and Choudhary A., 1999. MAFIA: Efficient and scalable subspace clustering for very large data sets. *Technical Report CPDC-TR-9906-010*, Northwestern University, USA.

Goodman, J.E. and O'Rourke, J., 2004. *Handbook of Discrete and Computational Geometry*. Chapman & Hall.

Gower, J., 1971. Coefficients of association and similarity, based on binary (presence-absence) data: An evaluation. *Biometrics*, 27, 857–871.

Grosan, C., Abraham, A., and Chis, M., 2006. Swarm intelligence in data mining. In Abraham, A., Grosan, C., and Ramos, V., editors, *Swarm Intelligence in Data*

Mining, *Studies in Computational Intelligence, Volume 34*, pages 1-20. Springer, Berlin.

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

Guha, S., Rastogi, R., and Shim, K., 2000. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25 (5), 345-366.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M., 2002a. Cluster validity methods: Part I. *ACM SIGMOD Record*, 31 (2), 40-45.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M., 2002b. Cluster validity methods: Part II. *ACM SIGMOD Record*, 31 (3), 19-27.

Han, J. and Kamber, M., 2001. *Data Mining: Concepts and Techniques*. Morgan Kaufman, Massachusetts.

Handl, J. and Knowles, J., 2004a. Evolutionary multiobjective clustering, *In: Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, 1081-1091.

Handl, J. and Knowles, J., 2004b. Multiobjective clustering with automatic determination of the number of clusters. *Technical Report TR-COMPSYSBIO-2004-02*, University of Manchester Institute of Science and Technology, Manchester, UK.

Handl, J. and Knowles, J., 2005. Exploiting the trade-off—the benefits of multiple objectives in data clustering. *In: Proceedings of Third International Conference on Evolutionary Multicriterion Optimization*, 547-560.

Handl, J. and Knowles, J., 2007. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11 (1), 56-76.

Handl, J. and Meyer, B., 2007. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1, 95-113.

Handl, J., Knowles, J. and Dorigo M., 2006. Ant-based clustering and topographic mapping. *Artificial Life*, 12 (1), 35-61.

Hastie, T., Tibshirani, R., and Friedman, J. H., 2009. *The Elements of Statistical Learning*. 2nd ed., Springer, Berlin.

He, Y. and Chen, L., 2003. A novel nonparametric clustering algorithm for discovering arbitrary shaped clusters. *In: Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia*, 1826-1830.

Hinneburg, A. and Kiem, D., 1998. An efficient approach to clustering large multimedia databases with noise. *In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, New York, NY, 58-65.

Ho, C.K. and Ewe, H.T., 2005. Performance of ant colony optimization algorithm on the dynamic load-balanced clustering problem in ad hoc networks. In Carbonell, J. G. and Siekmann J., editors, *Computational Intelligence and Security, Lecture Notes in Computer Science, Volume 3801*, pages 622-629. Springer-Verlag, Berlin.

Hoffmann W., Terschueren C., and Richardson D. B., 2007. Childhood leukemia in the vicinity of the Geesthacht nuclear establishments near Hamburg, Germany. *Environmental Health Perspective,* 115, 947–952.

Höppner, F., Klawonn, F., and Kruse, R., 1999. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition.* John Wiley & Sons, New York.

Hruschka, E.R., Campello R.J.G.B., Freitas A.A., and Carvalho, A.C.P.L.F., 2009. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, 39 (2), 133-155.

Ichino, M. and Yaguchi, H., 1994. Generalized Minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*. 24 (4), 698-708.

Iglesia, B. and Reynolds, A., 2005. The use of meta-heuristic algorithms for data mining. *In: Proceedings of the First International Conference on Information and Communication Technologies*, Karachi, Pakistan, 34- 44.

İnkaya, T., Kayalıgil, S., and Özdemirel, N.E., 2010a. A new density-based clustering approach in graph theoretic context. *In: Proceedings of Fourth European Conference on Data Mining*, 1-8.

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

Islier, A.A., 2005. Group technology by an ant system algorithm. *International Journal of Production Research*, 43 (5), 913-932.

İyigün, C., 2008. Probabilistic Distance Clustering. Ph.D. Dissertation. Rutgers University, New Brunswick, New Jersey.

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

Jarboui, B., Cheikh, M., Siarry, P., and Rebai, A., 2007. Combinatorial particle swarm optimization for partitional clustering problem. *Applied Mathematics and Computation*, 192 (2), 337-345.

Jaromczyk, J.W. and Toussaint, G.T., 1992. Relative neighborhood graphs and their relatives. *In: Proceedings of the IEEE*, 80 (9), 1502-1517.

Kao, Y. and Fu, S.C., 2006. An ant based clustering algorithm for manufacturing cell design. *International Journal of Advanced Manufacturing Technology*, 28 (11/12), 1182-1189.

Kao, Y.T., Zahara, E., and Kao, I.W., 2008. A hybridized approach to data clustering. *Expert Systems with Applications*, 34 (3), 1754-1762.

Karypis, G., Han, E., and Kumar, V., 1999. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32 (8), 68-75.

Klein, D., Kamvar, S.D., and Manning, C.D., 2002. From instance-level constraints to space-level constraints: Making the most prior knowledge in data clustering. *In: Proceedings of the Nineteenth International Conference on Machine Learning*. 307-313.

Klein, R.W. and Dubes, R., 1989. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22 (2), 213-220.

Kohonen, T., 1990. The self-organizing map. *Proceedings of the IEEE*, 78 (9), 1464-1479.

Koontz, W.L.G., Narendra, P.M., and Fukunaga, K., 1976. A graph-theoretic approach to nonparametric cluster analysis. *IEEE Transactions on Computers*, C-25 (9), 936-944.

Lee, I. and Estivill-Castro, V., 2006. Fast cluster polygonization and its applications in data-rich environments. *GeoInformatica*, 10 (4), 399-422.

Liu, D., Nosovskiy, G. V., and Sourina, O., 2008. Effective clustering and boundary detection algorithm based on Delaunay triangulation. *Pattern Recognition Letters*, 29 (9), 1261-1273.

Lozano J.A. and Larranaga P., 1999. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20 (9), 911-918.

Lu, S.Y. and Fu, K.S., 1978. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8 (5), 381-389.

Lumer, E.D. and Faieta, B., 1994. Diversity and adaptation in populations of clustering ants. In Cliff, D., Husbands, P., Meyer, J.A., and Wilson, S.W., editors, *In: Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, Volume 3, pages 501-508, MIT Press/Bradford Books, Cambridge, MA.

MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. *In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley, 281-297.

Martin, M., Chopard, B., and Albuquerque, P., 2002. Formation of an ant cemetery: Swarm intelligence or statistical accident?. *Future Generation Computer Systems*, 18 (7), 951-959.

Melkemi, M., 2003. Three-dimensional shapes of a finite set of points. *International Journal of Pattern Recognition and Artificial Intelligence*, 17 (2), 301-318.

Melkemi, M. and Djebali, M., 2000. Computing the shape of a planar points set. *Pattern Recognition*, 33 (9), 1423-1426.

Melkemi, M. and Djebali M., 2001. Weighted A-shape: A descriptor of the shape of a point set. *Pattern Recognition*, 34 (6), 1159-1170.

Merkle, D., Middendorf, M., and Scheider, A., 2005. Decentralized packet clustering in router-based networks. *International Journal of Foundations of Computer Science*, 16 (2), 321-341.

Mitchell, T., 1997. *Machine Learning*. McGraw-Hill, New York, NY.

Nagesh H., Goil S., and Choudhary A., 2001. Adaptive grids for clustering massive data sets. *In: Proceedings of the First SIAM International Conference on Data Mining*, Chicago, IL.

Ng, R.T. and Han, J., 1994. Efficient and effective clustering methods for spatial data mining. *In: Proceedings of the Twentieth Conference on VLDB*, Santiago, Chile, 144-155.

Ng, R.T. and Han, J., 2002. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, 14 (5), 1003-1016.

Nosovskiy, G.V., Liu, D., and Sourina, O., 2008. Automatic clustering and boundary detection algorithm based on adaptive influence function. *Pattern Recognition*, 41 (9), 2757-2776.

O'Callaghan, J.F., 1975. An alternative definition for neighborhood of a point. *IEEE Transactions on Computers*, 100 (11), 1121-1125.

Omran, M., Engelbrecht, A.P., and Salman, A., 2005. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19 (3), 297-321.

Omran, M., Salman, A., and Engelbrecht, A.P., 2002. Image classification using particle swarm optimization. *In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, Singapore.

Omran, M., Salman, A., and Engelbrecht, A.P., 2006. Dynamic clustering using particle swarm optimization with application in image segmentation. *Pattern Analysis and Applications*, 8 (4), 332-344.

Paterlini, S. and Krink. T., 2006. Differential evolution and particle swarm optimisation in partitional clustering. *Computational Statistics and Data Analysis,* 50 (5), 1220-1247.

Piatetsky-Shapiro, G., Brachman, R., Khabaza, T., Kloesgen, W., and Simoudis, E., 1996. An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. *In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

Picarougne, F., Azzag, H., Venturini, G., and Guinot, C., 2007. A new approach of data clustering using a flock of agents. *Evolutionary Computation*, 15 (3), 345-367.

Poli, R., Kennedy, J. and Blackwell, T., 2007. Particle swarm optimization an overview. *Swarm intelligence*, 1 (1), 33-57.

Prabhaharan, G., Muruganandam, A., Asokan, P., and Girish, B.S., 2005. Machine cell formation for cellular manufacturing systems using ant colony system approach. *International Journal of Advanced Manufacturing Technology*, 25 (9/10), 1013-1019.

Raghavan, V.V. and Birchard, K., 1979. A clustering strategy based on a formalism of the reproductive process in natural systems, *In: Proceedings of the Second Annual International ACM SIGIR Conference on Information Storage and Retrieval*, 10-22.

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

Richards, F.M., 1977. Area, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering*, 6, 151-176.

Rose, K., Gurewitz, E., and Fox, G. C., 1993. Constrained clustering as an optimization problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (8), 785-794.

Runkler, T.A., 2005. Ant colony optimization of clustering models. *International Journal of Intelligent Systems*, 20, 1233-1251.

Runkler, T.A., 2008. Wasp swarm optimization of the c-means clustering model, International Journal of Intelligent Systems, 23 (3), 269-285.

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

Santosa, B. and Ningrum, M.K., 2009. Cat swarm optimization for clustering. *In: Proceedings of International Conference of Soft Computing and Pattern Recognition*, 54-59.

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

Senator, T., Goldberg, H. G., Wooton, J., Cottini, M.A., Umarkhan, A. F., Klinger, C. D., Llamas, W. M., Marrone, M. P., and Wong, R. W. H., 1995. The financial crimes enforcement network AI system (FAIS): Identifying potential money laundering from reports of large cash transactions. *Artificial Intelligence Magazine*, 16 (4), 21-39.

Sharan R. and Shamir R., 2000. CLICK: A clustering algorithm with applications to gene expression analysis. *In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 307-316.

Sheikholeslami, G., Chatterjee, S., and Zhang, A., 1998. WaveCluster: A multiresolution clustering approach for very large spatial databases. *In: Proceedings of the Twenty-fourth Conference on VLDB*, 428-439.

Shelokar, P.S., Jayaraman, V.K., and Kulkarni, B.D., 2004. An ant colony approach for clustering. *Analytica Chimica Acta*, 509 (2), 187-195.

Sinha, A.N., Das, N., and Sahoo, G., 2007. Ant colony based hybrid optimization for data clustering. *Kybernetes*, 36 (1/2), 175-191.

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).

Spinello, R.A., 1997. *Case Studies in Information and Computer Ethics*. Prentice Hall, Upper Saddle River, N.J.

Sung C. and Jin H., 2000. A tabu-search-based heuristic for clustering. *Pattern Recognition*, 33, 849-858.

T'kindt, V., Monmarché, N., Tercinet, F., and Laügt D., 2002. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research,* 142 (2), 250-257.

Tan, P.N., Steinbach, M., and Kumar, V., 2005. *Introduction to Data Mining*. Addison-Wesley.

Toussaint, G.T., 1980. The relative neighborhood graph of a planar set. *Pattern Recognition*, 12 (4), 261-268.

Toussaint, G.T., 2002. Proximity graphs for nearest neighbor decision rules: Recent progress. *In: Proceedings of the Thirdy-fourth Symposium on the on Computing and Statistics, INTERFACE*.

Trejos, J., Piza, E., Murillo, A., and Pacheco, A., 2006. Comparison of metaheuristics for partitioning in cluster analysis. *XIII Congreso Latino-Iberoamericano de Investigaci´on Operativa*, Montevideo.

Tsai, C.F., Tsai, C.W., Wu, H.C., and Yang, T., 2004. ACODF: A novel data clustering approach for data mining in large databases. *Journal of Systems and Software*, 73 (1), 133-145.

Urquhart, R., 1982. Graph theoretical clustering based on limited neighborhood sets. *Pattern Recognition*, 15 (3), 173-187.

Veenhuis, C. and Köppen, M., 2006. Data swarm clustering. *Studies in Computational Intelligence*, 34, 221-241.

Wagstaff, K.L., 2002. Intelligent Clustering with Instance Level Constraints. Ph.D. Dissertation. Cornell University, Ithaca.

Wang, W., Yang, J., and Muntz, R.R., 1997. STING: A statistical information grid approach to spatial data mining. *In: Proceedings of the Twenty-third Conference VLDB*, Athens, Greece, 186-195.

Wang, Y. and Wei P.C., 2009. Data clustering method based on ant swarm intelligence. *In: Proceedings of Second IEEE International Conference on Computer Science and Information Technology*, 358-361.

Wong, M.A. and Lane, T., 1983. A kth nearest neighbor clustering procedure. *Journal of the Royal Statistical Society*, 45 (3), 362-368.

Xiao, X., Dow, E.R., Eberhart, R., Miled, Z.B., and Oppelt, R.J., 2004. A hybrid self-organizing maps and particle swarm optimization approach, *Concurrency and Computation-Practice & Experience*, 16 (9), 895-915.

Xu R. and Wunsch D., 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16 (3), 645-678.

Yang, Y., and Kamel, M.S., 2006. An aggregated clustering approach using multi-ant colonies algorithms, *Pattern Recognition*, 39 (7), 1278-1289.

Yousri, N.A., Kamel, M.S., and Ismail, M.A., 2008. A Novel Validity Measure for Clusters of Arbitrary Shapes and Densities. *In: Proceedings of International Conference of Pattern Recognition,* Tampa, USA.

Zahn C.T., 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20 (1), 68-86.

Zhang, T., Ramakrishnan, R., and Linvy, M., 1997. BIRCH: An efficient data clustering method for very large data sets. *Data Mining and Knowledge Discovery*, 1(2), 141-182.

Zhao, F., Hong, Y., Yu, D., Yang, Y., Zhang, Q., and Yi, H., 2007. A hybrid algorithm based on particle swarm optimization and simulated annealing to holon task allocation for holonic manufacturing system. *International Journal of Advanced Manufacturing Technology*, 32 (9/10), 1021-1032.

APPENDIX A

SWARM INTELLIGENCE APPLICATIONS IN CLUSTERING

In this part, we present the clustering studies about Swarm Intelligence.

Table A.1 Particle Swarm Optimization applications for clustering

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Particle representation**	**Homogeneity**	**Velocity matching**	**Locality**	**Collision avoidance**	**Flock centering**
Omran et al. 2002	Given	Yes	No	Image	Yes	Weighted sum of minimization of within cluster distance and maximization of intercluster distance	Partitional algorithm	Cluster centroids are represented by a particle.	All agents use the same behavioral model.	- Particle's previous velocity - Best of particle's past - Global best	Particle is affected from its past.	-	Towards global best

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Particle representation	Homogeneity	Velocity matching	Locality	Collision avoidance	Flock centering
Xiao et al. 2004	Not given	Yes	No	Mixed	No	Maximization of within-cluster similarity	2 hybrid SOM-PSO algorithms are proposed: - Block SOM/PSO: SOM first clusters data set and generates a group of weights. PSO improves clustering. - Alternating SOM/PSO: Weights are trained by SOM and PSO in alternating fashion. Several SOMs are trained. Each SOM is treated as a particle, swarm is run for a number of iterations.	Particle denotes the complete weight set of SOM.	All agents use the same behavioral model.	- Particle's previous velocity - Best of particle's past - Global best	Particle is affected from its past.	-	Towards global best

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Particle representation	Homogeneity	Velocity matching	Locality	Collision avoidance	Flock centering
Omran et al. 2005	Given	Yes	No	Image	Yes	2 objectives are studied: - Weighted sum of minimization of within cluster distance and maximization of intercluster distance. - Weighted sum of minimization of within cluster distance, maximization of intercluster distance and minimization of quantization error.	- Partitional clustering algorithm - 2 versions of PSO are considered: gbest PSO and GCPSO.	Cluster centroids are represented by a particle.	- All agents use the same behavioral model in gbest PSO. - In GCPSO, velocity of global best particle is updated using a different function.	In gbest PSO: - Particle's previous velocity - Best of particle's past - Global best In GCPSO, the global best particle is updated by resetting the particle's position to the global best position, and using the previous velocity and a random component.	Particle is affected from its past.	-	Towards global best for both algorithms

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Particle representation	Homogeneity	Velocity matching	Locality	Collision avoidance	Flock centering
Omran et al. 2006	Not given	No	No	Image	No	- Dunn's index, validity index by Turi (2001) and S_Dbw validity index	- Partitional algorithm - First, large clusters are formed. Then, by using binary PSO, number of clusters is determined. K-means is used to refine the clusters. - Position of the particle (0 or 1) can be updated using sigmoid function.	Particle denotes that a cluster centroid is used or not. (Binary particle)	All agents use the same behavioral model.	- Particle's previous velocity- Best of particle's past- Global best	Particle is affected from its past.	-	Towards global best

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Particle representation**	**Homogeneity**	**Velocity matching**	**Locality**	**Collision avoidance**	**Flock centering**
Paterlini and Krink 2006	Given	No	No	Mixed	No	- Minimization of trace within criterion - Minimization of variance ratio criterion - Minimization of Marriott's criterion	Partitional clustering algorithm	Cluster centroids are represented by a particle.	All agents use the same behavioral model.	- Particle's previous velocity - Best of particle's past - Global best	Particle is affected from its past.	-	Towards global best
Veenhuis and Köppen 2006	Not given	Yes	No	Mixed	No	- Maximization of within-cluster similarity	- Partitional algorithm - Clustering is performed in the datoid space. - Similarity and dissimilarity distance functions are defined. These take into account both real distance and similarity distance between the points.	- Each point is represented by a particle (datoid). - Position of the particle in 2-dimensional plane denotes the position of the datoid.	- All dataoids use the same behavioral model.	- Particle's previous velocity - Nearest similar particle's velocity - Nearest dissimilar particle's position - Nearest similar particles' center	-Datoid is affected from its k-nearest similar neighbors.	Avoidance of dissimilar dataoids	Towards nearest similar particles' center

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Particle representation**	**Homogeneity**	**Velocity matching**	**Locality**	**Collision avoidance**	**Flock centering**
Jarboui et al. 2007	Given	No	No	Mixed	No	- Minimization of within-cluster variation (Sum of squared Euclidean distances between each object and its cluster center) - Variance ratio	- Partitional algorithm - Combinatorial PSO is applied. - Particles define a dummy variable which translates the continuous state of the problem to combinatorial state.	- Each point is represented by a particle. - Particle denotes the cluster assignment of the corresponding point.	- All particles use the same behavioral model.	- Particle's previous velocity - Best of particle's past - Global best	- Particle is affected from its past.	-	-

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Particle representation	Homogeneity	Velocity matching	Locality	Collision avoidance	Flock centering
Picarougne et al. 2007	Not given	Yes	No	Mixed		- Maximization of within-cluster similarity	- Partitional algorithm - Ideal distance is defined. It depends on the similarity of the two points. Distance between agents should converge to this ideal distance so that relative distances between agents can be provided. - Data are dynamically visualized. - Disorder of the flock is defined by spatial entropy and it is used for stopping criterion.	- Each point is represented by a particle. - Position of the particle in 2-dimensional plane denotes the position of the point.	All agents use the same behavioral model.	Velocity calculation: - Particle's previous velocity - Attraction, rejection or constant alignment of the agents according to their similarity and current positions. Amplitude calculation: - Minimum speed - Agents in a group are slower than agents traveling alone.	Neighbors of an agent are agents located with a distance smaller than or equal to the threshold distance.	Avoidance of dissimilar agents	Points with no neighbors will be fast to catch up a group, whereas points in a group will move slower.

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Particle representation**	**Homogeneity**	**Velocity matching**	**Locality**	**Collision avoidance**	**Flock centering**
Kao et al. 2008	Given	No	No	Mixed	No	- Minimization of within cluster distance (sum of Euclidean distances between each point and its cluster center)	- Partitional algorithm - Nelder-Mead (NM) simplex search and PSO are combined. - NM provides local search. PSO ensures exploration. K-means is used in seeding the initial population.	Cluster centroids are represented by a particle.	- (N+1) particles use NM operator. - 2N particles use PSO operator.	- Particle's previous velocity - Best of particle's past - Global best	2N particles are evenly divided into N neighborhoods.	-	Towards global best

Table A.1 Particle Swarm Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Particle representation**	**Homogeneity**	**Velocity matching**	**Locality**	**Collision avoidance**	**Flock centering**
Ahmadi et al. 2010	Given	No	No	Mixed	No	- Optimization of a validity index (compactness, separation, combined version, Dunn's index, Turi's validity index, S_Dbw)	- Problem is decomposed into small subproblems using multiple cooperative swarms. Global optimum is ensured by cooperation among these. - Multiple swarms represent the cluster centers. Each swarm searches for the associated cluster center. - For the given the cluster centers, K-means is used for cluster assignments.	Each particle in a swarm denotes the cluster centers.	Each swarm optimizes the corresponding cluster centroid.	- Particle's previous velocity - Best of particle's past - Global best	Each swarm has its own search space and it searches for the correspon ding cluster centroid.	There is cooperatio n among multiple swarms.	- Towards global best

Table A.2 Ant Colony Optimization applications for clustering

Author(s) and Year	Problem Characteristics					Clustering objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Ant representation	Neighborhood	Pheromone update	Decision rule for solution construction	Exploration	Exploitation
Tsai et al. 2004	Not given	Yes	No	Mixed	No	- Minimization of within-cluster distance	- Density-based approach - ACO is hybridized with simulated annealing, tournament selection (GA), tabu search and density distribution.	Ant is an agent that searches through the solution components (points).	An ant chooses 10 to 15 points randomly. Next point to be visited by ant is determined from these selected points by tournament selection.	- Pheromone is updated for each visited edge which connects two points. - Pheromone density is computed as the summation of trail evaporation and the inverse function of the tour length of ants.	Tournament selection is done on the basis of pheromone intensity. Ant selects next point to visit using tournament selection.	To determine the next point to visit, first 10-15 points are selected randomly (then tournament selection is used).	-Ant selects next point to visit using tournament selection. - Simulated annealing ensures the reduction of number of points visited by each ant in every iteration. - Edges having greater pheromone density are merged in order to form the clusters.

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Ant representation	Neighborhood	Pheromone update	Decision rule for solution construction	Exploration	Exploitation
Ho and Ewe 2005	Given	No	Cluster capacity constraint: cluster cardinality cannot exceed maximum load.	N*	No	- Minimization of total variance of the load of the clusters	- Problem is dynamic as points are mobile during the problem solution process. - 2 points can connect if their Euclidean distance is below the transmission range. - Visibility measure is defined as a weight associated with each point. It indicates the number of uncovered neighboring nodes that a node can cover.	- Ant represents a clustering assignment of all points.	- Each ant can select a point from the allowed set. - Allowed set is composed of points that are not in the partial solution yet.	- Pheromone value of each point is updated according to visibility factor, previous pheromone value of the point and objective function value of the ant.	- Selection probability is calculated using visibility and pheromone values.	- Selection of points (solution components) is performed using visibility and pheromone values.	

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Ant representation	Neighborhood	Pheromone update	Decision rule for solution construction	Exploration	Exploitation
Runkler 2005	Given	No	No	N*	No	- Minimization of within-cluster variation (Sum of squared Euclidean distances between each point and its cluster center)	- Hard c-means is adapted. Cluster assignments of points are done by ACO.	Ant represents a clustering assignment of a point.	A point can be assigned to any cluster.	- Pheromone update is performed for cluster assignment of each point. - Pheromone values are updated using the incumbent solution and objective function value of the solution.	A point can be assigned to any cluster according to a probability proportional to the pheromone value.	Selection of solution components is made by using pheromone values.	

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering objective	Method							
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Ant representation	Neighborhood	Pheromone update	Decision rule for solution construction	Exploration	Exploitation
Chen and Chen 2006	Given	No	No	N*	No	- Minimization of within-cluster variation (sum of squared Euclidean distances between each point and its cluster center)	Hierarchical algorithm	Ant represents clustering assignment of all points.	In each iteration, ant moves along the points which do not belong to its working memory.	- Local pheromone update: Ant simultaneously updates the amount of pheromone on its visited paths using the inverse of within cluster variance. - Global pheromone update: After all ants built their solutions, the amount of pheromone on the global best path is updated.	- A random number and a threshold value control whether exploration or exploitation will be emphasized in the solution construction. - If the random number is smaller than threshold, exploitation is emphasized. Otherwise, exploration is used.	Cluster assignment of the point is performed on a probabilistic basis proportional to the pheromone values weighted by the inverse of the distance of the edge.	Point is assigned to the cluster with the highest pheromone value weighted by the inverse of the distance. - A percentage of the furthest points are selected. These points are assigned to the clusters with the closest centroids.

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering objective**	**Method**						
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Ant representation**	**Neighborhood**	**Pheromone update**	**Decision rule for solution construction**	**Exploration**	**Exploitation**
Prabhaharan et al. 2005	Given	No	No	Mixed	Yes	- Minimization of total intercellular moves (y) - Minimization of total cell load variation (within cluster variance) (x).	Assignments are done according to a index, x/(1+y).	Particle denotes the cluster (cell) assignments of the points (machines).	Possible cluster assignments of a point (machine) constitute the neighborhood.	- Local pheromone update: Ant updates the amount of pheromone on its visited edges by the pheromone value times evaporation rate. - Global pheromone update: After all ants built solutions, the amount of pheromone on the global best path is updated as the objective value and the previous pheromone value of the edge times evaporation rate.	- A random number and a threshold value control whether exploration or exploitation is emphasized in the solution construction. - If the random number is smaller than threshold, exploitation is emphasized, otherwise, exploration is used.	Cluster (cell) assignment of the point (machine) is performed on a probabilistic basis proportional to the pheromone values of the edges.	Point (machine) is assigned to the cluster with the highest pheromone value.

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering objective**	**Method**							
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Ant representation**	**Neighborhood**	**Pheromone update**	**Decision rule for solution construction**	**Exploration**	**Exploitation**
Sinha et al. 2007	Not given	Yes	No	Mixed	No	- Minimization of within-cluster distance	- Density-based approach - ACO is hybridized with simulated annealing, tournament selection (GA), tabu search and density distribution.	Ant is an agent that searches through the points.	An ant chooses 10 to 15 points randomly. Next point to be visited by ant is determined from these selected points by tournament selection.	- Pheromone value of an edge is updated when an ant travels an edge. - Pheromone density is the inverse function of the length of the edges.	- Tournament selection is done on the basis of pheromone intensity. Ant selects next point to visit using tournament selection.	- Ant selects next point to visit using tournament selection. - Tabu search restricts the path of an ant, i.e. an ant cannot visit the same point more than once on the same path.	- Simulated annealing ensures the reduction of number of points visited by each ant in every iteration. - Edges having greater pheromone density are merged in order to form the clusters.

Table A.2 Ant Colony Optimization applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering objective	Method						
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Ant representation	Neighborhood	Pheromone update	Decision rule for solution construction	Exploration	Exploitation
Wang and Wei 2009	Given	No	No	N*	No	Minimization of within-cluster variance	Points are assigned to the clusters.	Ant denotes the cluster assignments of the points.	Possible cluster assignments of a point constitute the neighborhood.	- Pheromone is updated for each edge which shows the assignment of a point to a cluster. - Ant updates the amount of pheromone on its visited edges by the previous pheromone value times evaporation rate.	A point can be assigned to any cluster according to a probability proportional to the pheromone value.	Selection of cluster assignments is made by using pheromone values.	

N*: Numerical

Table A.3 Other Swarm Intelligence based applications for clustering

Author(s) and Year	Problem Characteristics					Clustering Objective	Method					
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Agent representation	Neighborhood	Decision rule for solution construction	Exploration	Exploitation
Martin et al. 2002	Not given	Yes	No		No	Cemetery formation (corpse clustering)	Minimal model, a modified version of Deneubourg's model	Ant is an agent that carries corpses.	- Ants move one cell at a time, but in the same direction for a pre-specified random number of steps. - When a corpse lies on the trajectory of the ant, a new random direction is selected. - When ant faces an obstacle, it chooses a new direction of motion and a free path length.	- If an unloaded ant has a body in its neighboring cells, it is loaded with probability 1. If there are several bodies, it chooses one of them randomly. - After an ant moves at least one step, corpse is dropped if the ant is has corpses in its neighboring cells. Ant drops the corpse in a randomly chosen empty neighboring cell.	Pick-up and drop rules ensure both exploration and exploitation.	

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method					
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Agent representation	Neighborhood	Decision rule for solution construction	Exploration	Exploitation
Handl et al. 2006	Not given	Yes	No	Mixed	No	Maximization of within-cluster similarity	- Partitional algorithm - 2 algorithms are proposed: ATTA-C for cluster retrieval and ATTA-TM for topographic mapping - Agglomerative hierarchical clustering (single or average link) is used for cluster retrieval in ATTA-C.	Ant is an agent that searches through the solution components (points).	- Ant can move to the grids within a predefined radius (radius of perception) surrounding itself. - Ant has a memory, and next step of the ant is towards the position of the best match in the memory. - Neighborhood size increases over time (like VNS).	Probability of picking up or dropping a point is proportional to the similarity between the associated point and the other neighboring points.	- Probability of picking up or dropping a point is proportional to the similarity between the associated point and the other neighboring points. - Neighborhood size is increased over time.	- Neighborhood function gives higher penalty to higher dissimilarity. - Each ant has a memory for the previously carried data. When ant picks a point, it moves the point towards the best matching point in the memory. - Picking up and dropping a point is deterministic in a range.

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method					
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Agent representation	Neighborhood	Decision rule for solution construction	Exploration	Exploitation
Yang and Kamel 2006	Given	Yes	No	Mixed	No	Maximization of within-cluster similarity	- Partitional algorithm - Algorithm has 2 main parts: clustering by using different ant colonies and aggregation of clustering results of these different colonies by a queen ant.	Ant is an agent that searches through the solution components (points).	- Ant in site r moves to a site in the square of sxs sites surrounding itself (site in which ant is located currently) with different speeds.	- Probability of picking up or dropping a point is proportional to the similarity between the associated point and the other neighboring points. - A new similarity matrix is computed by using the clustering results of different colonies.	- Probability of picking up or dropping a point is proportional to the similarity of the point with the other points in the neighborhood. - Each ant moves with a different speed (constant, uniformly random, segmented random, randomly decreasing). - A new similarity matrix is computed using the clustering results of the different colonies.	

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**					
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Agent representation**	**Neighborhood**	**Decision rule for solution construction**	**Exploration**	**Exploitation**
Kao and Fu 2006	Given	No	No	Categorical	No	Maximization of within-cluster similarity	- Partitional algorithm - Manufacturing cell formation is studied. - A modified part similarity coefficient is introduced. - Algorithm terminates when the entropy value becomes steady.	Ant is an agent that searches through the solution components (points).	Each ant has a fixed-length memory which tracks of the location recently laid down. In the loaded status, ant will move towards the location of the most similar point (part) in the memory. In the unloaded status, ant moves randomly.	- Ant picks up or drops a point according to a probability proportional to the similarity between the associated point and its neighbors. - Clusters are refined by k-means. - Points are merged in a hierarchical manner until the desired number of clusters is obtained.	Pick-up and drop probabilities are calculated proportional to the similarity between the associated point and the other neighboring points.	After ant-based clustering step, k-means is applied to improve the cluster quality.

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**					
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Agent representation**	**Neighborhood**	**Decision rule for solution construction**	**Exploration**	**Exploitation**
Boryczka 2009	Not given	No	No	N*	No	Like cemetery formation (corpse clustering)	Basic idea is to pick up or drop a data item on the grid.	Ant is an agent that carries data points.	- Data points are assigned to grids and ants move data points to the neighboring grids with certain probability. - An adaptive neighborhood size is used.	- Probability of picking up and dropping off a data point is affected from the density function in a given neighborhood.	- Pick-up and drop rules ensure both exploration and exploitation. - A cooling procedure is used for the dissimilarities scaling parameter in the neighborhood function so that exploitation is ensured.	

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	**Problem Characteristics**					**Clustering Objective**	**Method**					
	# of clusters	**Arbitrary shapes**	**Constraints**	**Data type**	**Multi-objective**		**Properties**	**Agent representation**	**Neighborhood**	**Decision rule for solution construction**	**Exploration**	**Exploitation**
Azzag et al. 2007	Not given	Yes	No	Mixed	No	Build a tree with maximum similar roots and sufficientl y dissimilar branches.	- Hierarchical algorithm(s) - A stochastic algorithm (SA) and 4 deterministic algorithms (DA) (algorithm with dissimilarity threshold, algorithm with both dissimilarity and similarity thresholds, algorithm with self-adaptive thresholds, algorithm with no thresholds and no parameters) are proposed.	Ant represents a point.	- An ant moves down the tree or forms a new branch. - Random movements are inserted into the stochastic version.	- For DA with no thresholds and no parameters: Ant may disconnect from the current connected ant and go to the top of the tree. - For SA: Ant may move down to the daughter ant randomly. - For other DAs: Ant can either form a new branch connected to the current ant or move down to the most similar daughter ant.	- For deterministic algorithm with no thresholds and no parameters: Ant may disconnect from the current connected ant goes to the support (top of the tree) - For stochastic algorithm: Ant may move down to the daughter ant of current connected ant randomly.	- For deterministic algorithms 1-3: An ant can either form a new branch connected to the current ant or move down to the most similar daughter ant of current connected ant.

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method					
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Agent representation	Neighborhood	Decision rule for solution construction	Exploration	Exploitation
Fathian et al. 2007	Given	No	No	N*	No	Minimization of within-cluster variation (sum of squared Euclidean distances between each object and its cluster center)	-	Cluster centorids are represented by queen and drones.	- New broods are created by exchanging the drone's genes with the queen's (crossover). - Worker bees improve the broods by performing local search (royal jelly).	- Algorithm starts with mating flight: queen (best solution) selects drones in a probabilistic manner to form the list of drones. Then, a drone is randomly selected from the list to create broods. - Fitter broods are replaced with queen(s).	- Drones are potential parents and they go into crossover with queen in a probabilistic manner. - Workers have different mutation heuristics.	- Queen is the best solution and always goes into crossover. Broods are created by taking a weighted average of the queen and selected drone. - Broods are improved using worker bees.

Table A.3 Other Swarm Intelligence based applications for clustering (cont'd)

Author(s) and Year	Problem Characteristics					Clustering Objective	Method					
	# of clusters	Arbitrary shapes	Constraints	Data type	Multi-objective		Properties	Agent representation	Neighborhood	Decision rule for solution construction	Exploration	Exploitation
Runkler 2008	Given	No	No	Mixed	No	Minimization of within-cluster variation (sum of squared Euclidean distances between each object and its cluster center)	- Partitional algorithm - Memory-free WSO is applied.	Particle denotes the cluster assignment of all the points.	Possible cluster assignments are sorted according to the distance between cluster center and the point. This sorting forms the neighborhood structure.	Stochastic tournament selection is used to pick up a solution component (cluster assignment).	Stochastic tournament selection is used to pick up a solution component (cluster assignment). Pick-up probabilities are determined according to cost value (distance between the cluster center and the point).	
Santosa and Ningrum 2009	Given	No	No	N*	No	Minimization of sum of squared-error	Seeking mode ensures exploitation and tracing mode ensures exploration.	Each cat represents a cluster center.	In seeking mode each dimension has a certain mode to move.	Cluster centers are selected via roulette wheel in seeking mode.	In tracing mode, cluster centers move with velocity which is affected from the current position and the best solution.	Cluster centers move with a certain percentage in each dimension.

N*: Numerical

APPENDIX B

PROPERTIES AND GENERATION OF DATA SETS USED IN THE EXPERIMENTS

In this dissertation, three groups of data sets are used in the experiments. Group 1 is composed of 2- and higher dimensional data sets compiled from several sources (Frank and Asuncion 2010, Sourina 2008, İyigün 2008). For some data sets with outliers, another version was created by removing the outliers to see their effect. There are 45 data sets in the first group. We generate the synthetic data sets in groups 2 and 3. The details of the generation scheme are explained in Section B.1.

We quantify data set properties using three measures, namely the minimum separation-to-compactness ratio in the target clustering (MSCR), the coefficient of variation of the edge lengths in MST of the whole data set (CV1), and the average of the coefficient of variations of the edge lengths in individual cluster MSTs (CV2). A high value of MSCR shows that even the cluster with the minimum ratio is well-separated from the others hence a rather trivial data set. A high CV1 for the whole data set indicates significant density variation between/within the clusters. Large values of CV2 show significant density variation within the clusters. The properties of all the data sets in groups 1, 2 and 3 are given in Section B.2.

Finally, plot of data sets are provided in Section B.3.

B.1. Generation of Group 2 and Group 3 Data Sets

Generation mechanism of groups 2 and 3 is similar. Each data set is composed of four clusters, namely the letters S, A, O and E. There is a fifth spherical cluster inside the letter O. These letters are selected due to their non-convex and dissimilar (curly, sharp, cornered, oval with a cluster enclosed) shapes. Each letter is formed from cubes with unit edge length (grid size) and a point is placed at the center of each cube.

Basic letter shapes formed from cubes are shown in Figure B.1 for both groups. The main difference between the two groups is the data set size. The letters are circumscribed in boxes in group 2 approximately 22 by 15 grids in size, whereas the same for a letter in group 3 is approximately 15 by 9 grids. Four grids form the depth of a letter in group 2, and in group 3 the depth of a letter is three grids. To sum up, the size of letters in group 3 is smaller than the ones in group 2.

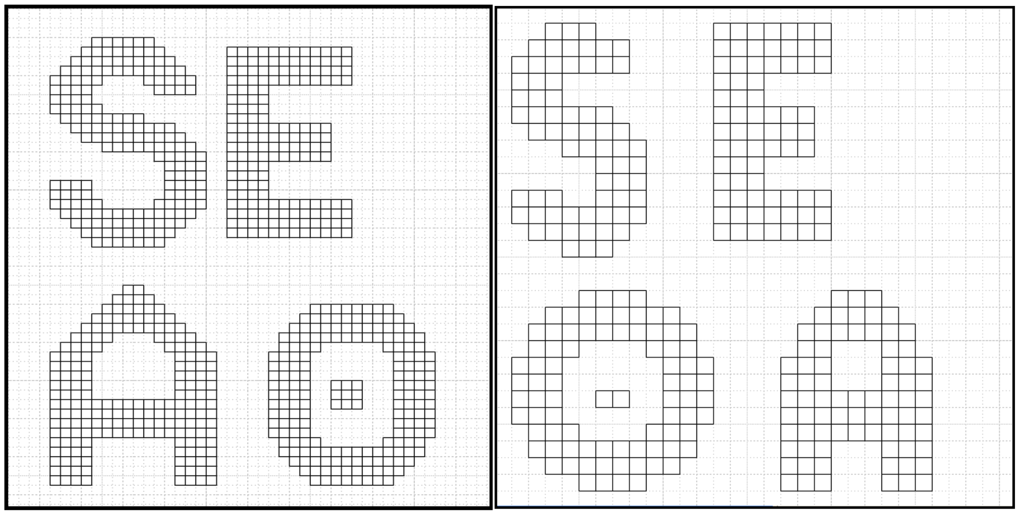

(a) (b)

Figure B.1 Letters formed from cubes in (a) group 2 data sets, (b) group 3 data sets

Following four factors are considered in generating the data sets.

Factor 1. Intercluster density difference (EDD)

Level 1. No difference (EDD1)

All the clusters have the same density. That is, the grid size is 1 and it is identical in each cluster.

Level 2. Density difference (EDD2)

Grid sizes of letters S, A, E and O are 1, $\sqrt[3]{3}$, $\sqrt[3]{3}$ and $\sqrt[3]{9}$, respectively. In 3-dimensional space, these figures produce densities in the proportion of 9: 3: 3: 1 (points in the letter S are nine times denser than points in the letter O).

Factor 2. Intracluster density variation (ADV)

Level 1. No point deletion (ADV1)

Points are uniformly distributed in each cluster.

Level 2. Random deletion (ADV2)

30% of the points are randomly deleted from the uniformly distributed clusters.

Level 3. Smooth change (ADV3)

Density decreases with a trend while moving from North to South and from West to East within a cluster (letter). That is, the grid size gradually increases up to $\sqrt[3]{9}$ times the original size just next to the bottom right corner.

Factor 3. Intercluster distance (DST)

Level 1. Distant (DST1)

Distance between adjacent cluster pairs is determined according to the maximum possible density (grid size) in the nearest adjacent neighborhoods. The distant level is twice the maximum possible grid size in a letter.

Level 2. Close (DST2)

In the close level the distance between adjacent cluster pairs is equal to the maximum possible grid size in a letter.

Factor 4. Outlier (OL)

Level 1. Without outlier (OL1)

There exists no outlier in the data set.

Level 2. With outlier (OL2)

There are 3 outliers inserted in each data set.

Using these factors, a full factorial design is applied. The factors used in data generation are summarized in Table B.1.

Table B.1 Factorial design for the generation of group 2 data sets

		Level 0	**Level 1**	**Level 2**
Factors	Intercluster density difference	EDD1	EDD2	-
	Intracluster density variation	ADV1	ADV2	ADV3
	Intercluster distance	DST1	DST2	-
	Outlier	OL1	OL2	-

Distance between adjacent cluster pairs is determined from the maximum possible grid size in the nearest adjacent neighborhoods. Each letter is positioned as shown in Figure B.2. The properties and plots of these 24 data sets are presented in Sections B.2 and B.3, respectively.

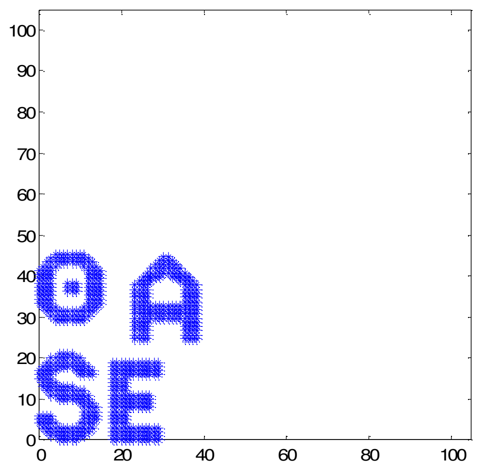


Figure B.2 Example data set in a 2-dimensional view

B.2. Properties of Data Sets

Table B.2 Data set properties for 2-dimensional group 1 data sets

Data set	**# of target clusters**	**# of outliers**	**# of points**	**MSCR**	**CV1**	**CV2**	**min. sep.**	**max. comp.**
data_60	3	0	60	1.50	0.43	0.23	1.46	1.00
data_66	4	0	66	1.50	0.47	0.27	1.27	1.00
data-c-cv-nu-n_v2	3	0	73	1.02	1.04	0.25	0.80	0.78
data-c-cv-nu-n	6	3	76	1.02	1.04	0.25	0.80	0.78
data-c-cv-u-n	5	3	81	2.74	1.13	0.24	1.79	0.65
data-uc-cv-nu-n	6	3	127	0.92	1.04	0.32	0.62	0.67
data-oo_v2	2	0	140	2.52	0.47	0.16	0.46	0.55
data-oo	6	4	144	2.52	1.46	0.16	0.46	0.55
data-uc-cc-nu-n_v2	3	0	188	0.80	0.78	0.42	0.54	0.68
data-uc-cc-nu-n	6	3	191	0.80	1.04	0.42	0.54	0.68
data-c-cc-nu-n2_v2	3	0	192	3.31	0.63	0.24	1.82	0.55
data-c-cc-nu-n2	6	3	195	1.72	0.79	0.24	0.95	0.55
dataX_v2	2	0	200	1.15	0.64	0.63	1.04	0.90
dataX	4	2	202	1.15	0.75	0.63	1.04	0.90
data-c-cc-nu-n_v2	3	0	285	1.07	0.56	0.37	0.82	0.77
train2	4	0	287	2.79	1.23	0.27	0.07	0.03
data-c-cc-nu-n	7	4	289	0.60	0.94	0.37	0.46	0.77
train1_v1	5	1	306	3.02	1.28	0.38	0.05	0.03
train1	6	2	307	3.02	1.39	0.38	0.05	0.03
train3_v1	5	0	361	3.64	1.62	0.26	0.06	0.05
train3	36	30	397	0.03	1.26	0.78	0.02	0.74
data_circle	2	0	700	51.94	2.36	0.59	0.71	0.04
data_mix_uniform_normal	2	0	1000	13.52	1.39	0.71	2.12	0.51
data_circle_1_10_5_10	2	0	1100	3.37	0.79	0.63	0.30	0.09
data_circle_10_1_10_10	2	0	1100	1.60	0.89	0.60	0.12	0.15
data_circle_2_10_2_12	2	0	1200	15.19	0.82	0.61	0.33	0.08
data_circle_2_10_3_12	2	0	1200	5.03	0.73	0.62	0.27	0.09
data_circle_2_10_4_12	2	0	1200	4.58	0.69	0.63	0.22	0.07
data_circle_2_10_5_13	2	0	1200	13.22	1.10	0.61	0.72	0.09
data_circle_2_10_6_12	2	0	1200	2.24	0.63	0.59	0.13	0.07
data_circle_2_10_3_12	2	0	1200	5.03	0.73	0.62	0.27	0.09
data_circle_3_10_8_12	2	0	1300	0.90	0.63	0.62	0.07	0.08
data_circle_5_10_8_12	2	0	1500	0.46	0.61	0.61	0.04	0.09

Table B.2 Data set properties for 2-dimensional group 1 data sets (cont'd)

data_circle1	2	0	1890	3.90	0.67	0.61	0.22	0.06
data_circle2	2	0	1890	4.09	0.70	0.65	0.22	0.07
data_circle_20_1_5_10	2	0	2100	17.22	1.79	0.62	0.39	0.20
data_circle3	2	0	2100	21.00	2.88	0.62	0.71	0.03
data_circle_1_20_1_11	2	0	2100	25.56	0.91	0.63	0.41	0.05
data_circle_1_20_1_13	2	0	2100	24.00	0.76	0.61	0.32	0.06
data_circle_1_20_1_15	2	0	2100	14.99	0.68	0.62	0.23	0.08
data_circle_1_20_1_17	2	0	2100	0.28	0.63	0.61	0.00	0.14
data_circle_1_20_1_19	2	0	2100	1.46	0.64	0.63	0.03	0.09

Table B.3 Data set properties for higher dimensional group 1 data sets

Data set	**# of target clusters**	**# of outliers**	**# of points**	**MSCR**	**CV1**	**CV2**	**min. sep.**	**max. comp.**
iris	3	0	150	0.35	0.60	0.46	0.22	0.91
3d_dataset3	2	0	325	11.87	0.93	0.13	5.94	0.62
3d_dataset4	2	0	1523	29.68	0.71	0.13	5.94	0.62

Table B.4 Data set properties for group 2 data sets

Data set	**# of target clusters**	**# of outliers**	**# of points**	**MSCR**	**CV1**	**CV2**	**min. sep.**	**max. comp.**
D_0000	5	0	2780	3.00	0.21	0.04	3.00	1.00
D_0001	8	3	2783	3.00	0.28	0.04	3.00	1.00
D_0010	5	0	2780	2.00	0.13	0.04	2.00	1.00
D_0011	8	3	2783	2.00	0.29	0.04	2.00	1.00
D_0100	5	0	1975	2.12	0.23	0.06	3.00	1.41
D_0101	8	3	1978	2.12	0.31	0.06	3.00	1.41
D_0110	5	0	1927	1.41	0.13	0.05	2.00	1.41
D_0111	8	3	1930	1.41	0.34	0.05	2.00	1.41
D_0200	5	0	2780	1.58	0.49	0.39	3.10	2.00
D_0201	8	3	2783	1.58	0.51	0.39	3.10	2.00
D_0210	5	0	2780	1.02	0.47	0.39	2.00	2.00
D_0211	8	3	2783	1.02	0.49	0.39	2.00	2.00
D_1000	5	0	2780	1.44	0.31	0.12	4.01	4.33
D_1001	8	3	2783	1.28	0.31	0.12	2.20	4.33
D_1010	5	0	2780	0.96	0.30	0.12	2.01	4.33
D_1011	8	3	2783	0.96	0.31	0.12	2.01	4.33
D_1100	5	0	1925	1.44	0.35	0.17	4.01	4.33
D_1101	8	3	1928	1.44	0.36	0.17	2.20	4.33
D_1110	5	0	1948	0.96	0.32	0.16	2.01	4.33
D_1111	8	3	1951	0.96	0.33	0.16	2.01	4.33
D_1200	5	0	2780	1.49	0.55	0.45	4.06	6.05
D_1201	8	3	2783	1.44	0.55	0.45	4.06	6.05
D_1210	5	0	2780	0.98	0.54	0.45	2.12	6.05
D_1211	8	3	2783	0.98	0.55	0.45	2.12	6.05

Table B.5 Data set properties for group 3 data sets

Data set	# of target cluster s	# of outliers	# of points	MSCR	CV1	CV2	min. sep.	max. comp.
DS_0000	5	0	894	2.83	0.39	0.00	2.83	1.00
DS_0001	8	0	903	2.83	0.44	0.00	2.83	1.00
DS_0010	5	0	894	2.83	0.50	0.36	5.66	2.00
DS_0011	8	0	903	2.83	0.58	0.22	5.66	2.00
DS_0100	5	0	709	2.83	0.41	0.01	2.83	1.41
DS_0101	8	3	712	2.24	0.44	0.01	2.83	1.41
DS_0110	5	0	708	2.00	0.19	0.01	2.83	1.41
DS_0111	8	3	711	2.00	0.30	0.01	2.83	1.41
DS_0200	5	0	894	1.49	0.51	0.36	2.93	1.97
DS_0201	8	3	897	1.49	0.52	0.36	2.93	1.97
DS_0210	5	0	894	1.03	0.45	0.36	2.00	1.97
DS_0211	8	3	897	1.03	0.47	0.36	2.00	1.97
DS_1000	5	0	894	2.83	0.32	0.00	4.00	2.08
DS_1001	8	3	897	2.83	0.34	0.00	4.00	2.08
DS_1010	5	0	894	1.92	0.28	0.00	2.00	2.08
DS_1011	8	3	897	1.92	0.29	0.00	2.00	2.08
DS_1100	5	0	708	2.00	0.35	0.02	5.77	2.94
DS_1101	8	3	711	1.80	0.36	0.02	3.11	2.94
DS_1110	5	0	715	1.92	0.29	0.01	2.89	2.08
DS_1111	8	3	718	1.67	0.30	0.01	2.89	2.08
DS_1200	5	0	894	1.49	0.55	0.36	5.84	4.10
DS_1201	8	3	897	1.49	0.57	0.36	5.39	4.10
DS_1210	5	0	894	0.98	0.54	0.36	3.03	4.10
DS_1211	8	3	897	0.98	0.54	0.36	3.03	4.10

B.3. Plots of Data Sets

Two Dimensional Group 1 Data Sets

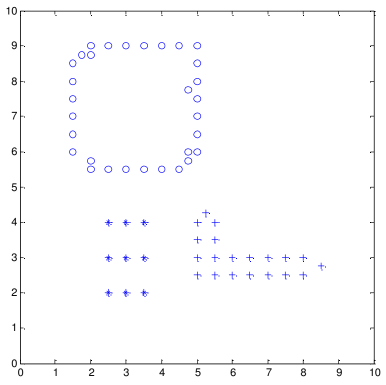


Figure B.3 data_set_60

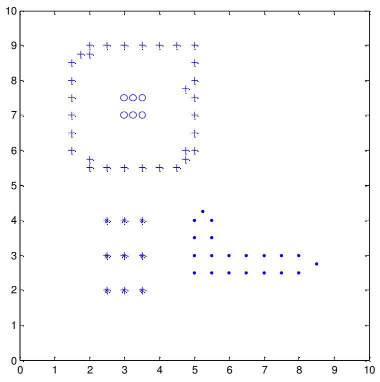


Figure B.4 data_set_66

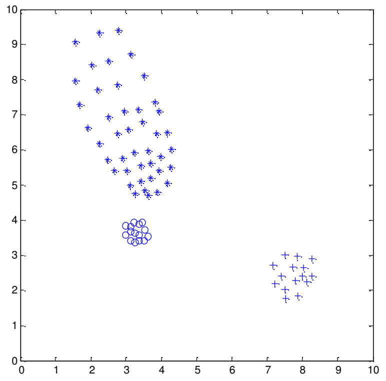


Figure B.5 data-c-cv-nu-n_v

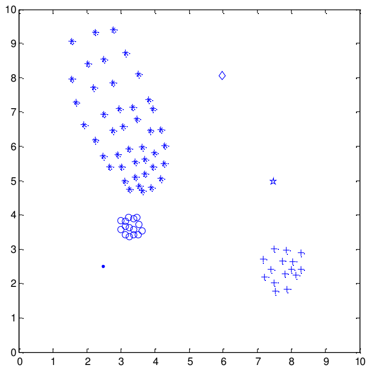


Figure B.6 data-c-cv-nu-n

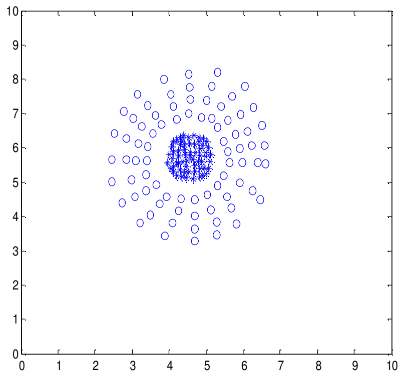


Figure B.7 data-oo_v2

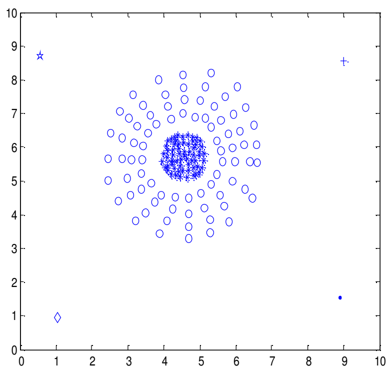


Figure B.8 data-oo

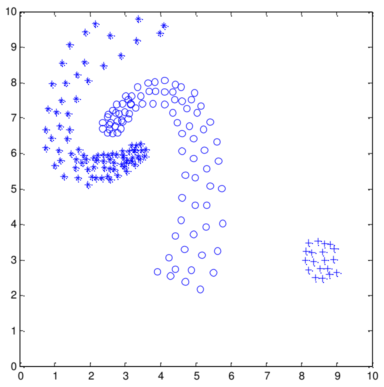


Figure B.9 data-uc-cc-nu-n_v2

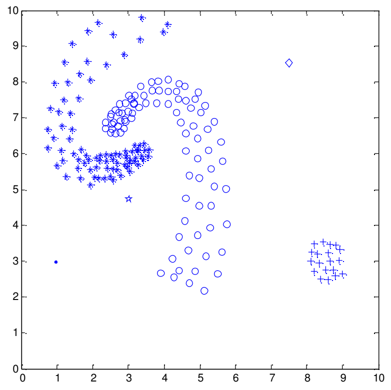


Figure B.10 data- uc-cc-nu-n

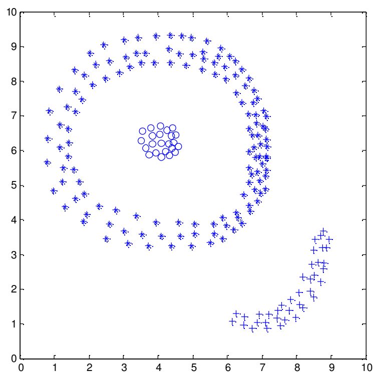


Figure B.11 data-c-cc-nu-n2_v2

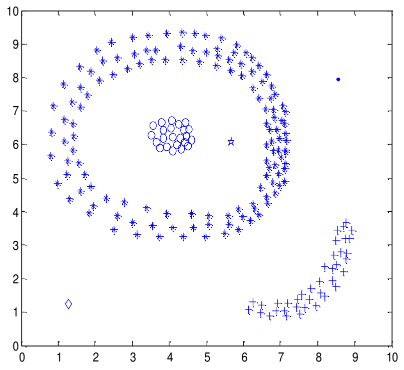


Figure B.12 data-c-cc-nu-n2

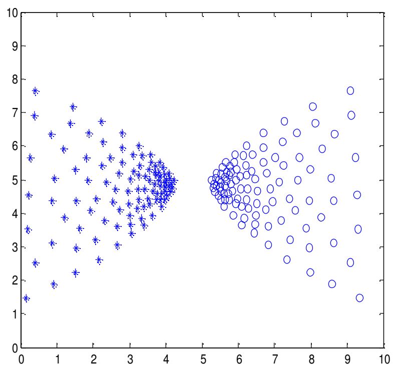


Figure B.13 dataX_v2

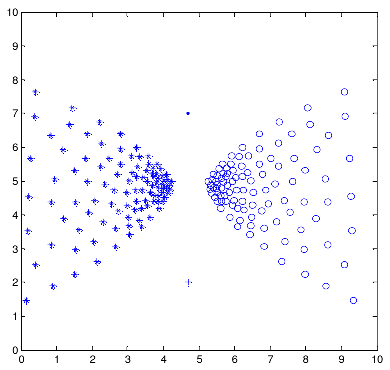


Figure B.14 dataX

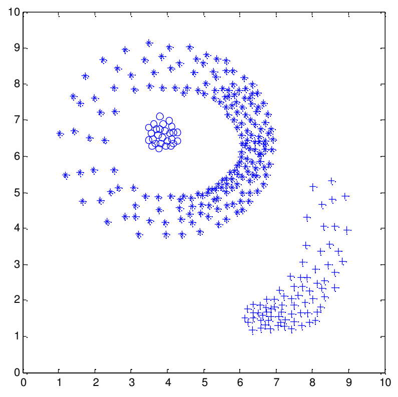


Figure B.15 data-c-cc-nu-n_v2

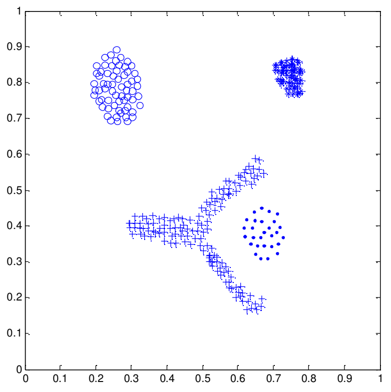


Figure B.16 train2

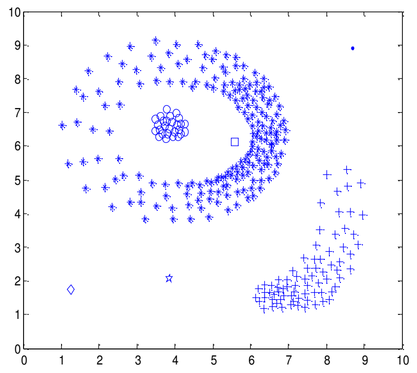


Figure B.17 data-c-cc-nu-n

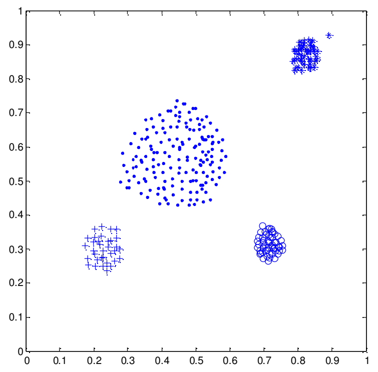


Figure B.18 train1_v1

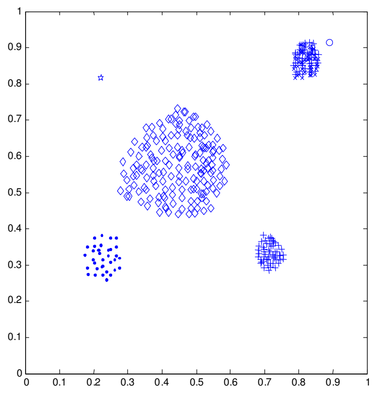


Figure B.19 train1

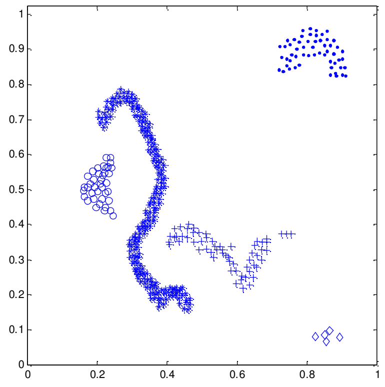


Figure B.20 train3_v1

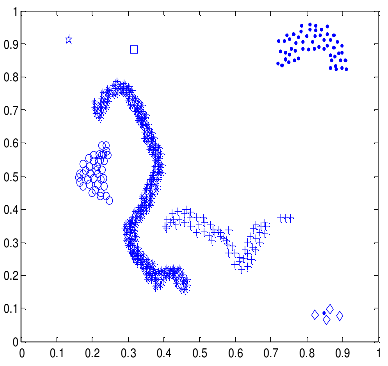


Figure B.21 train3

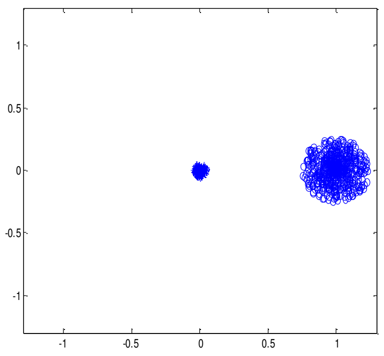


Figure B.22 data_circle

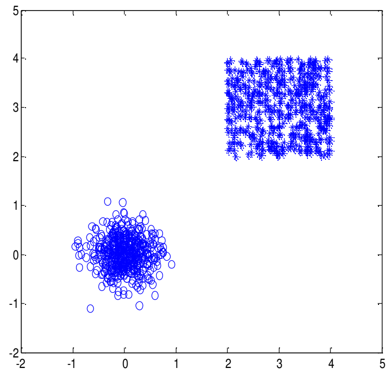


Figure B.23 data_mix_uniform_normal

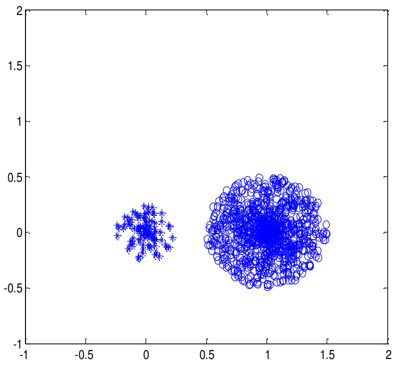


Figure B.24 data_circle_1_10_5_10

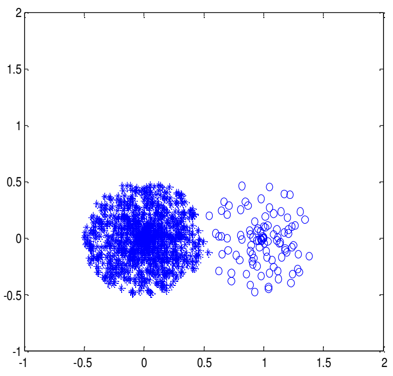


Figure B.25 data_circle_10_1_10_10

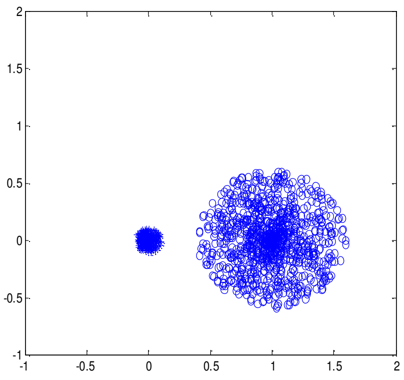


Figure B.26 data_circle_2_10_2_12

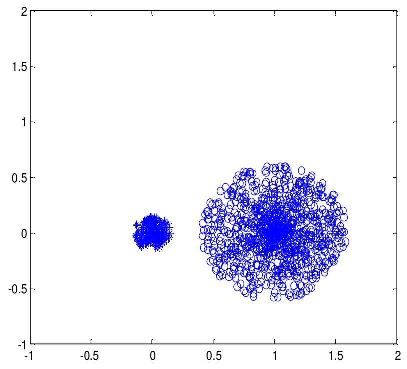


Figure B.27 data_circle_2_10_3_12

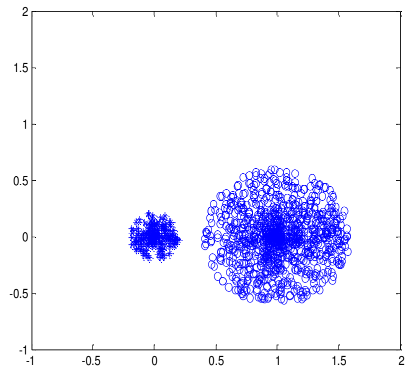


Figure B.28 data_circle_2_10_4_12

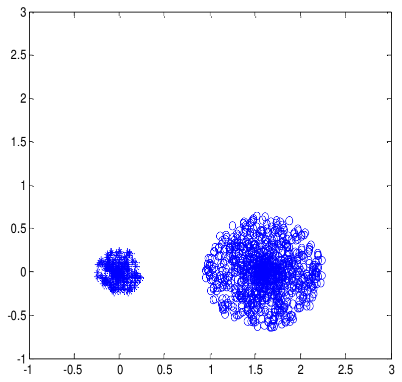


Figure B.29 data_circle_2_10_5_13

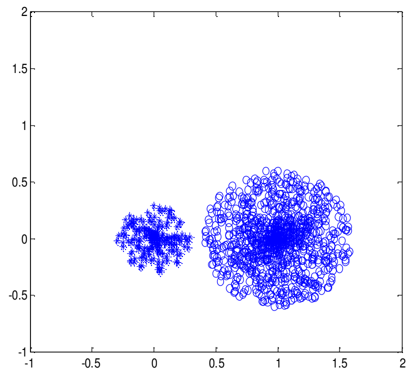


Figure B.30 data_circle_2_10_6_12

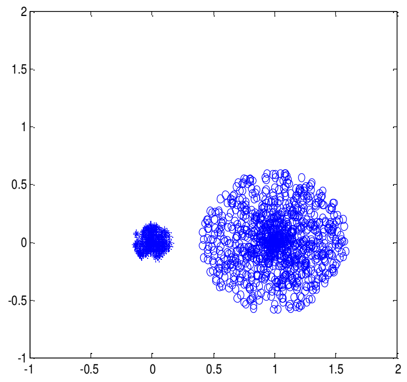


Figure B.31 data_circle_2_10_3_1

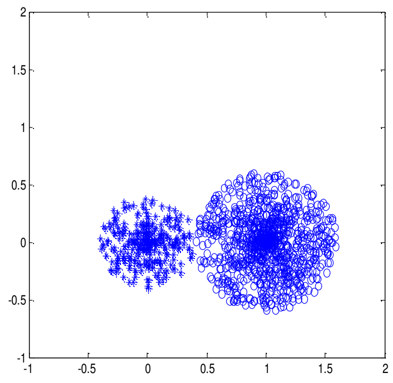


Figure B.32 data_circle_3_10_8_12

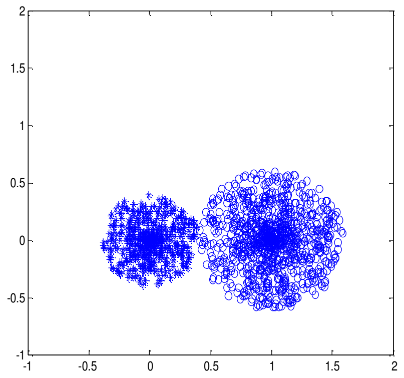


Figure B.33 data_circle_5_10_8_12

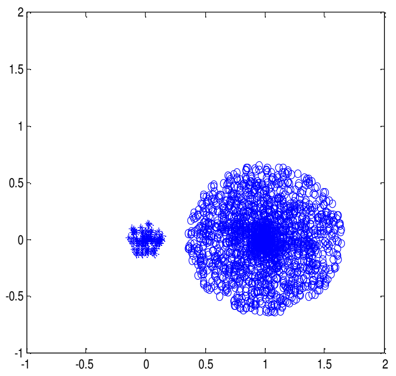


Figure B.34 data_circle1

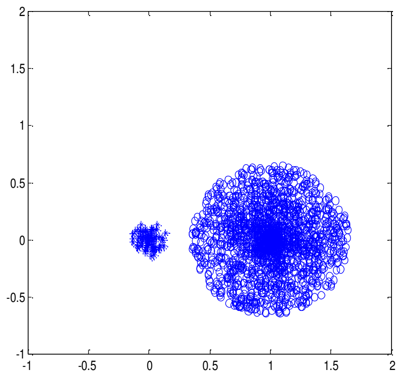


Figure B.35 data_circle2

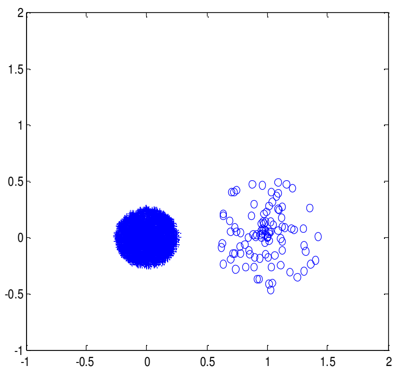


Figure B.36 data_circle_20_1_5_10

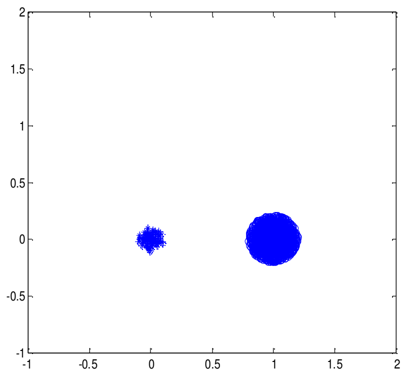


Figure B.37 data_circle3

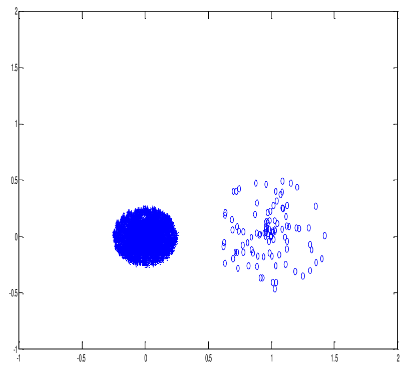


Figure B.38 data_circle_20_1_5_10

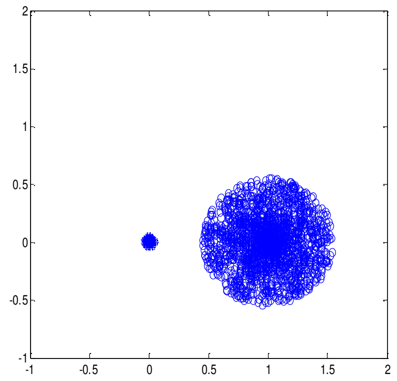


Figure B.39 data_circle_1_20_1_11

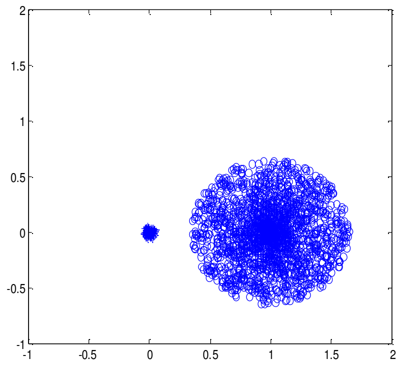


Figure B.40 data_circle_1_20_1_13

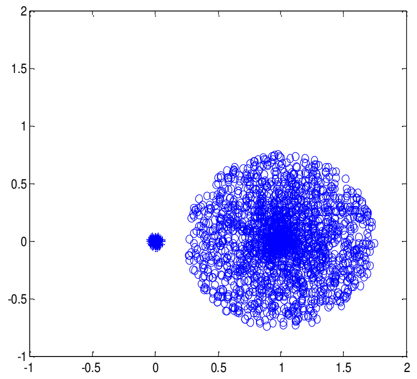


Figure B.41 data_circle_1_20_1_15

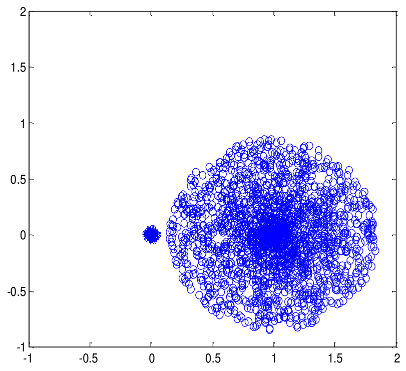


Figure B.42 data_circle_1_20_1_17

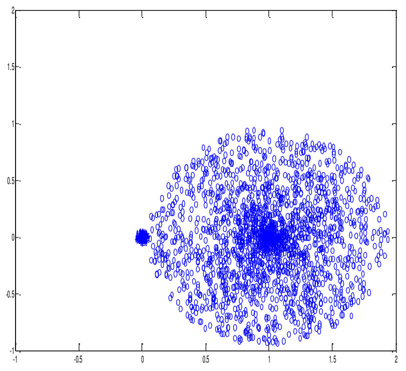


Figure B.43 data_circle_1_20_1_19

Group 2 Data Sets

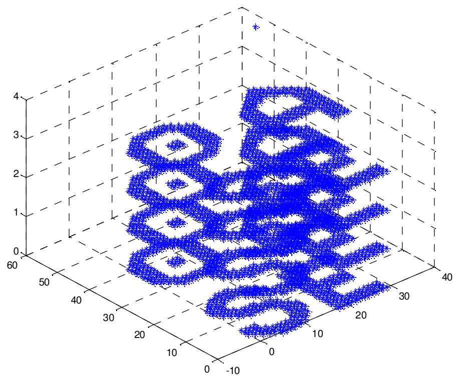


Figure B.44 D_0001

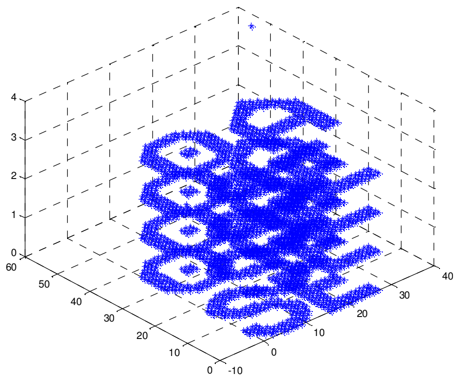


Figure B.45 D_0011

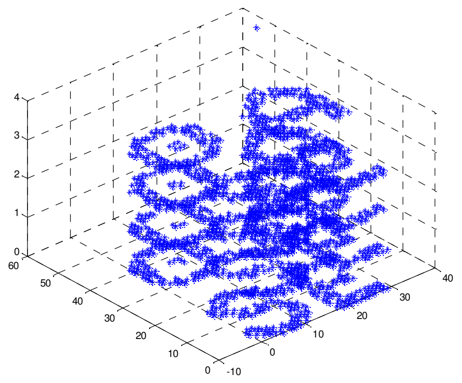


Figure B.46 D_0101

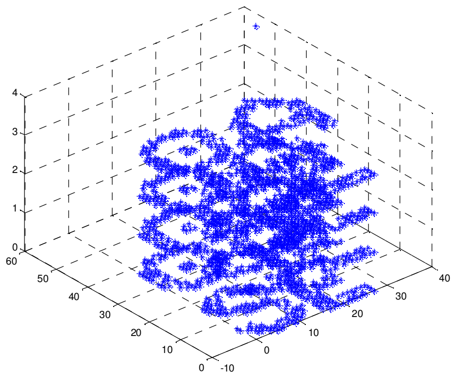


Figure B.47 D_0111

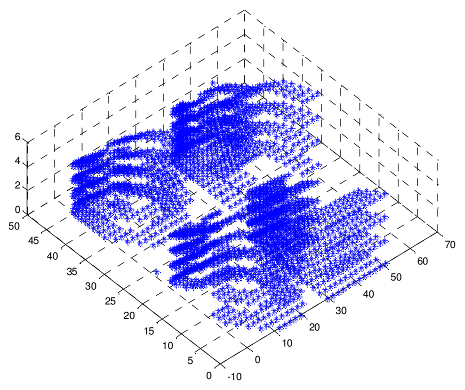


Figure B.48 D_0201

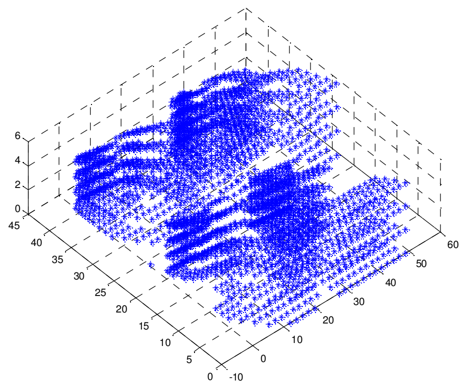


Figure B.49 D_0211

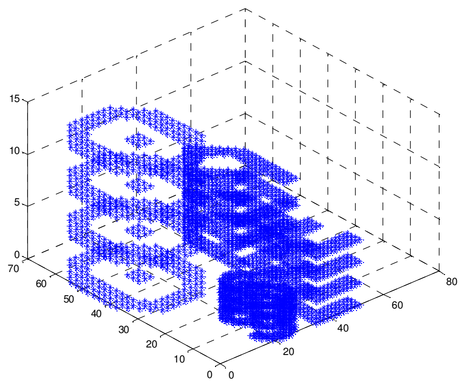


Figure B.50 D_1001

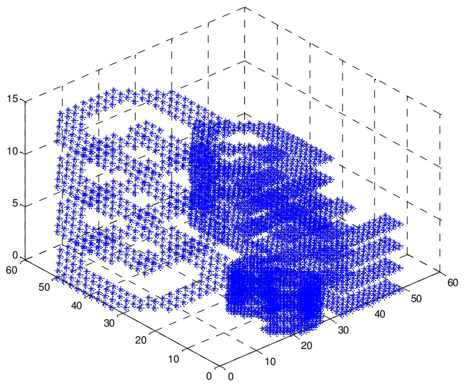


Figure B.51 D_1011

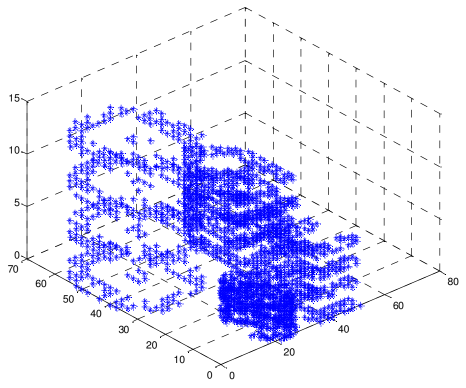


Figure B.52 D_1101

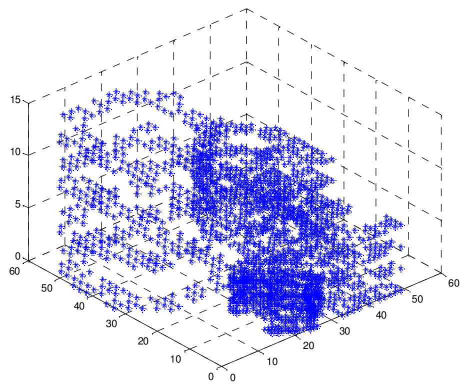


Figure B.53 D_1111

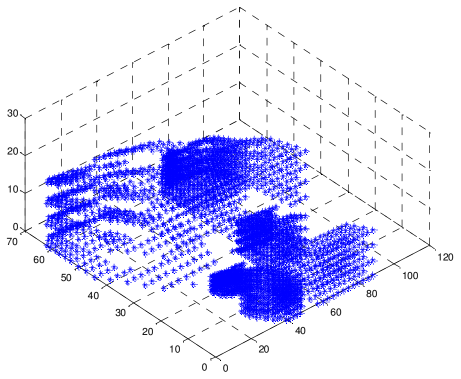


Figure B.56 D_1201

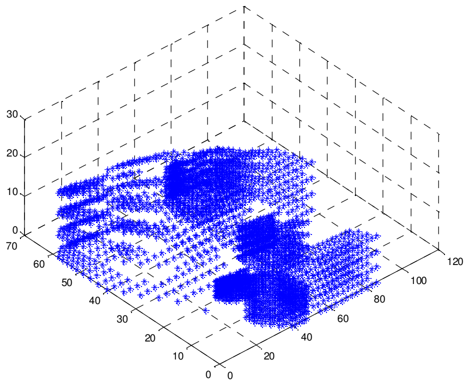


Figure B.57 D_1211

Group 3 Data Sets

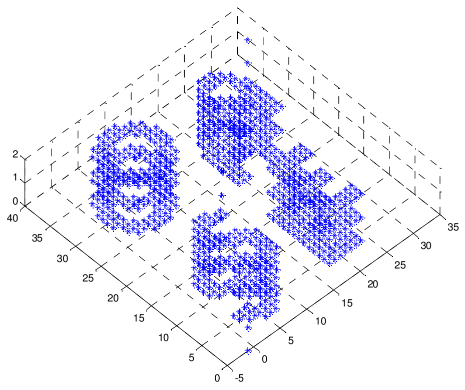


Figure B.54 DS_0001

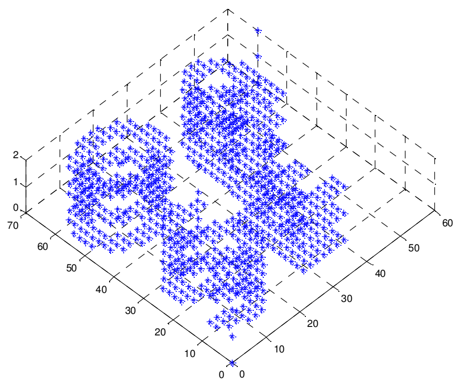


Figure B.55 DS_0011

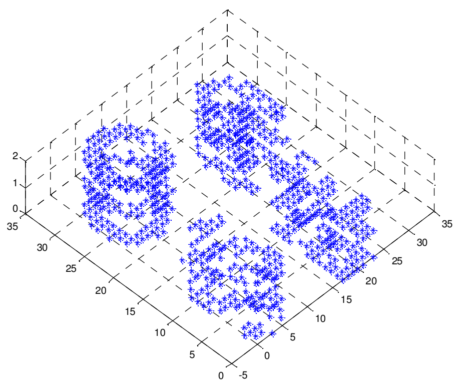


Figure B.56 DS_0101

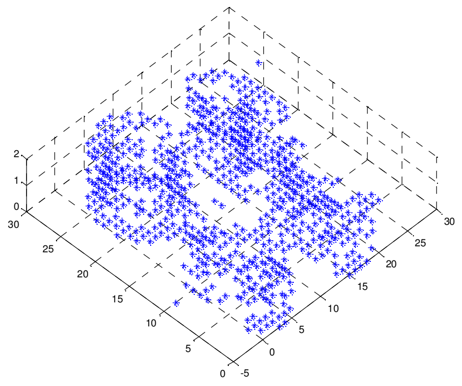


Figure B.57 DS_0111

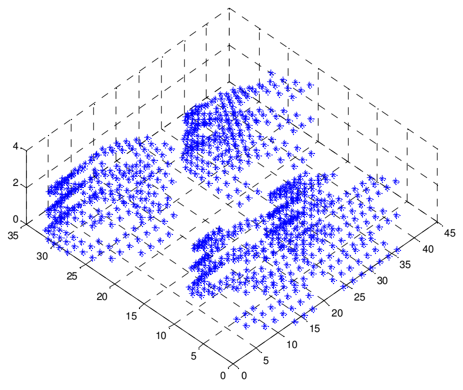


Figure B.58 DS_0201

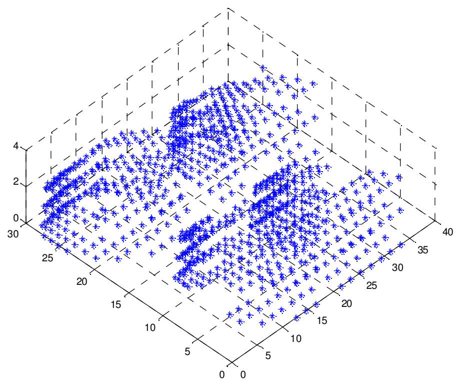


Figure B.59 DS_0211

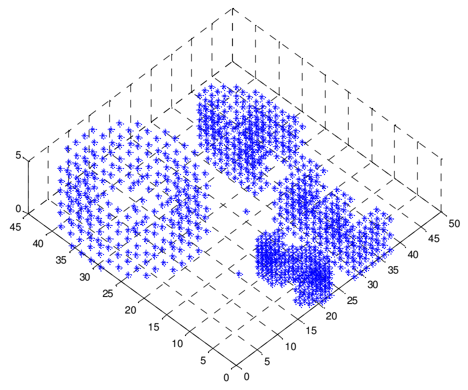


Figure B.60 DS_1001

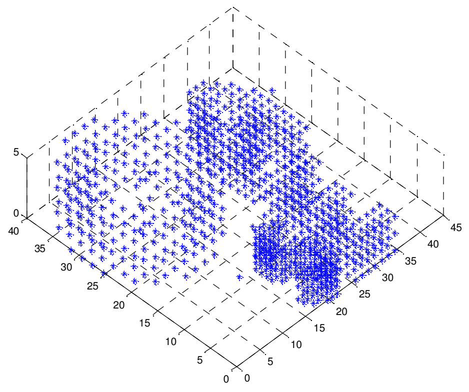


Figure B.61 **DS_1011**

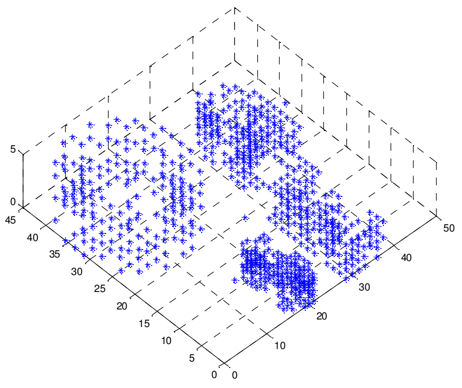


Figure B.62 DS_1101

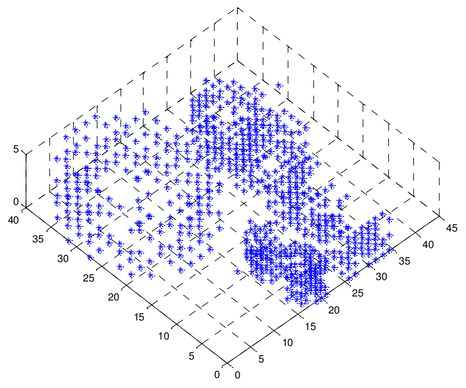


Figure B.63 DS_1111

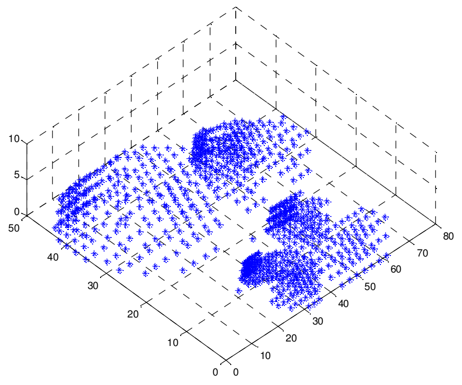


Figure B.64 DS_1201

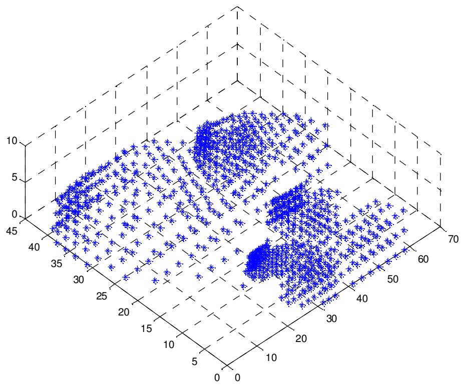


Figure B.65 DS_1211

APPENDIX C

THE NC ALGORITHM AND THE EXPERIMENTAL RESULTS

We present the pseudocode of the NC algorithm in Section C.1, and we provide the experimental results of the NC algorithm in Tables C.1 through C.9 in Section C.2. Column headings in Tables C.1 through C.9 are explained as follows.

#TC	: number of target clusters
#C	: number of clusters found by the corresponding algorithm

C.1. The NC Algorithm

Step 1. Core candidate set (CC_i) construction by direct connectivity.

```
For $i = 1,..,|D|$
    Sort all remaining points in D in non-decreasing order of distance to point $i$,
    and form ordered set, $T_i$.
    Set $j = 0$ and initialize $CC_i = \varnothing$.
    Repeat
        Set $j = j + 1$ and move to the next nearest neighbor $j$ of point $i$ in $T_i$.
        Calculate the density (i.e. number of points), $density_{ij}$, in the hyperball
        passing through points $i$ and $j$ and having diameter $d_{ij}$.
        If $density_{ij} = 0$
            $CC_i = CC_i \cup \{j\}$.
        End if
    Until $density_{ij} \neq 0$
    Set $core\_flag_i = j - 1$.
End for
```

Step 2. Break point candidate set (BC$_i$) construction by density tracking.

```
For $i = 1,..,|D|$
    Set $j = core\_flag_i$ and initialize $BC_i = CC_i$.
    Repeat
        Set $j = j + 1$ and move to the next nearest neighbor $j$ of point $i$ in $T_i$.
        Calculate $density_{ij}$ in the hyperball passing through points $i$ and $j$ and
        having diameter $d_{ij}$.
        If $density_{ij} - density_{i,j-1} \geq 0$
            $BC_i = BC_i \cup \{j\}$.
        End if
    Until $density_{ij} - density_{i,j-1} < 0$
    Set $break\_flag_i = j - 1$.
End for
```

Step 3. Potential candidate set (PC$_i$) construction by indirect connectivity checks.

```
For $i = 1,..,|D|$
    Set $j = break\_flag_i$ and $flag = 1$. Initialize $PC_i = BC_i$.
    Repeat
        Set $j = j + 1$ and move to the next nearest neighbor $j$ of point $i$ in $T_i$.
        Calculate $density_{ij}$ in the hyperball passing through points $i$ and $j$ and
        having diameter $d_{ij}$.
        If $density_{ij} - density_{i,j-1} \geq 0$
            $PC_i = PC_i \cup \{j\}$.
        Else if $BC_i \cap BC_{T_i(j)} \neq \varnothing$
```

```
                PC$_i$ = PC$_i$ $\cup$ {$j$}.
            Else
                Set $flag$ = 0.
            End if
        Until $flag$ = 0
        Set CS$_i$ = PC$_i$.
End for
```

Step 4. Candidate set (CS$_i$) construction by mutuality tests.

```
Repeat
    For $i$ = 1,..,|D|
        Set $j$ = 0 and $candidate\_flag_i$ = 0.
        Repeat
            Set $j = j + 1$ and move to the next nearest neighbor $j$ of point $i$
            in CS$_i$.
            If $j \in \mathrm{CC}_i$
                If $\mathrm{CC}_i \cap \mathrm{CS}_j = \varnothing$
                    Remove all neighbors $k = j$,..,|CS$_i$| from CS$_i$.
                    Set $candidate\_flag_i$ = 1.
                End if
            Else
                If $\mathrm{CS}_i \cap \mathrm{CS}_{\mathrm{CS}_i(j)} = \varnothing$
                    Remove $\mathrm{CS}_i(j)$ and all the neighbors such
                    that $\mathrm{CS}_i(j')$ where $j' > j$ from CS$_i$.
                    Set $candidate\_flag_i$ = 1.
                End if
            End if
        Until $candidate\_flag_i$ = 1 or $j$ = |CS$_i$|
    End for
```

Until $\prod_{i \in \mathrm{D}} candidate\_flag_i = 0$, meaning no more change occurs in any CS_i.

Step 5. Formation of closures (subclusters) by coverage.

```
Set $\mathrm{D}_0 = \mathrm{D}$ and initialize $\mathrm{NCS}_i = \mathrm{CS}_i,\ \forall i \in \mathrm{D}$.
While $\mathrm{D}_0 \neq \varnothing$
    Select $i \in \mathrm{D}_0$ arbitrarily.
    Set $\mathrm{D}_0^{'} = \mathrm{D}_0 \setminus \{i\} \cup \mathrm{NCS}_i$.
    While $\mathrm{D}_0^{'} \neq \varnothing$
        Select $j \in \mathrm{D}_0^{'}$ arbitrarily.
        If $\mathrm{NCS}_i \cap \mathrm{NCS}_j \neq \varnothing$
            $\mathrm{NCS}_i = \mathrm{NCS}_i \cup \mathrm{NCS}_j$ and $\mathrm{NCS}_j = \mathrm{NCS}_i \cup \mathrm{NCS}_j$.
        End if
        Remove $j$ and all points in $\mathrm{NCS}_j$ from $\mathrm{D}_0^{'}$.
    End while
End while
```

```
Set $m = 1$, $i = 1$ and $\mathrm{D}_0 = \mathrm{D}$.
Initialize the set of points in closure $m$, $\mathrm{Cl}_m = \{i\} \cup \mathrm{NCS}_i$. Remove point $i$ and all points in $\mathrm{NCS}_i$ from $\mathrm{D}_0$.
While $\mathrm{D}_0 \neq \varnothing$
    Select $i \in \mathrm{D}_0$ arbitrarily.
    Set $flag = 1$.
    For $k = 1,..,m$
        If $\mathrm{Cl}_m \cap \mathrm{NCS}_i \neq \varnothing$
            $\mathrm{Cl}_m = \mathrm{Cl}_m \cup \{i\} \cup \mathrm{NCS}_i$.
            Set $flag = 0$ and break.
        End if
    End for
    If $flag = 1$
        Set $m = m + 1$ and start a new closure, $\mathrm{Cl}_m = \{i\} \cup \mathrm{NCS}_i$.
    End if
    Remove point $i$ and all points in $\mathrm{NCS}_i$ from $\mathrm{D}_0$.
End while
```

C.2. The Experimental Results of the NC Algorithm

Table C.1 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of PPN and RI for group 1 data sets

		KNN1			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
Data set	**#TC**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**
data_60	3	3	1.00	1.00	1	0.52	0.39	3	1.00	1.00	3	1.00	1.00
data_66	4	4	1.00	1.00	1	0.47	0.33	4	1.00	1.00	4	1.00	1.00
data-c-cv-nu-n_v2	3	3	1.00	1.00	3	1.00	1.00	2	0.80	0.76	3	1.00	1.00
data-c-cv-nu-n	6	3	0.96	0.97	3	0.96	0.97	4	0.79	0.75	4	0.97	0.98
data-c-cv-u-n	5	1	0.59	0.48	1	0.59	0.48	5	1.00	1.00	5	1.00	1.00
data-uc-cv-nu-n	6	1	0.63	0.46	1	0.63	0.46	4	0.78	0.72	5	0.99	0.99
data-oo_v2	2	1	0.56	0.51	1	0.56	0.51	1	0.56	0.51	2	1.00	1.00
data-oo	6	1	0.55	0.48	1	0.55	0.48	5	0.58	0.53	6	1.00	1.00
data-uc-cc-nu-n_v2	3	2	0.56	0.59	2	0.56	0.59	2	0.56	0.59	3	1.00	1.00
data-uc-cc-nu-n	6	2	0.55	0.58	1	0.45	0.40	4	0.56	0.60	4	0.99	0.99
data-c-cc-nu-n2_v2	3	3	1.00	1.00	2	0.80	0.73	3	1.00	1.00	4	1.00	0.99
data-c-cc-nu-n2	6	2	0.87	0.82	1	0.67	0.50	6	1.00	1.00	7	1.00	0.99
dataX_v2	2	2	1.00	1.00	2	1.00	1.00	1	0.50	0.50	2	1.00	1.00
dataX	4	1	0.50	0.49	1	0.50	0.49	3	0.51	0.51	4	1.00	1.00
data-c-cc-nu-n_v2	3	1	0.64	0.49	1	0.64	0.49	3	1.00	1.00	3	1.00	1.00
train2	4	3	0.90	0.91	3	0.90	0.91	4	1.00	1.00	6	1.00	0.99
data-c-cc-nu-n	7	1	0.63	0.48	1	0.63	0.48	5	0.90	0.88	7	1.00	1.00
train1_v1	5	4	1.00	1.00	4	1.00	1.00	5	1.00	1.00	8	1.00	0.98
train1	6	4	0.99	1.00	4	0.99	1.00	6	1.00	1.00	9	1.00	0.98
train3_v1	5	2	0.71	0.63	2	0.71	0.63	7	1.00	1.00	6	1.00	1.00
train3	36	1	0.50	0.31	1	0.50	0.31	2	0.51	0.33	17	0.94	0.97
data_circle	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	14	1.00	0.88

Table C.1 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of PPN and RI for group 1 data sets (cont'd)

		KNN1			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
Data set	**#TC**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**
data_mix_uniform_normal	2	2	1.00	1.00	2	1.00	1.00	10	1.00	0.99	38	1.00	0.83
data_circle_1_10_5_10	2	2	1.00	1.00	1	0.91	0.84	9	1.00	0.99	29	1.00	0.86
data_circle_10_1_10_10	2	1	0.91	0.84	1	0.91	0.84	18	1.00	1.00	22	1.00	0.87
data_circle_2_10_2_12	2	2	1.00	1.00	1	0.83	0.72	7	1.00	0.99	22	1.00	0.85
data_circle_2_10_3_12	2	1	0.83	0.72	1	0.83	0.72	6	1.00	0.99	30	1.00	0.86
data_circle_2_10_4_12	2	1	0.83	0.72	1	0.83	0.72	6	1.00	0.99	37	1.00	0.74
data_circle_2_10_5_13	2	2	1.00	1.00	2	1.00	1.00	6	1.00	0.99	30	1.00	0.81
data_circle_2_10_6_12	2	1	0.83	0.72	1	0.83	0.72	3	1.00	1.00	39	1.00	0.79
data_circle_2_10_3_12	2	1	0.83	0.72	1	0.83	0.72	6	1.00	0.99	30	1.00	0.86
data_circle_3_10_8_12	2	1	0.77	0.65	1	0.77	0.65	6	0.78	0.65	26	1.00	0.85
data_circle_5_10_8_12	2	1	0.67	0.56	1	0.67	0.56	6	0.67	0.55	29	1.00	0.87
data_circle1	2	1	0.95	0.91	1	0.95	0.91	7	1.00	0.99	35	1.00	0.69
data_circle2	2	1	0.95	0.91	1	0.95	0.91	11	1.00	0.98	35	1.00	0.86
data_circle_20_1_5_10	2	1	0.95	0.91	1	0.95	0.91	75	1.00	1.00	52	1.00	0.66
data_circle3	2	1	0.95	0.91	1	0.95	0.91	18	1.00	0.99	41	1.00	0.71
data_circle_1_20_1_11	2	1	0.95	0.91	1	0.95	0.91	13	1.00	0.97	49	1.00	0.74
data_circle_1_20_1_13	2	1	0.95	0.91	1	0.95	0.91	14	1.00	0.97	41	1.00	0.83
data_circle_1_20_1_15	2	1	0.95	0.91	1	0.95	0.91	7	1.00	0.99	47	1.00	0.86
data_circle_1_20_1_17	2	1	0.95	0.91	1	0.95	0.91	6	1.00	0.99	57	1.00	0.80
data_circle_1_20_1_19	2	1	0.95	0.91	1	0.95	0.91	5	0.95	0.90	38	1.00	0.89
iris	2	2	1.00	1.00	2	1.00	1.00	3	1.00	0.98	4	1.00	0.99
3d_dataset3	2	2	1.00	1.00	2	1.00	1.00	127	1.00	0.85	2	1.00	1.00
3d_dataset4	2	2	1.00	1.00	2	1.00	1.00	4	1.00	1.00	2	1.00	1.00

Table C.2 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of JI and QJI for group 1 data sets

Data set	**#TC**	**KNN1**			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
		#C	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
data_60	3	3	1.00	1.00	1	0.39	0.39	3	1.00	1.00	3	1.00	1.00
data_66	4	4	1.00	1.00	1	0.33	0.33	4	1.00	1.00	4	1.00	1.00
data-c-cv-nu-n_v2	3	3	1.00	1.00	3	1.00	1.00	2	0.63	0.63	3	1.00	1.00
data-c-cv-nu-n	6	3	0.94	0.94	3	0.94	0.94	4	0.61	0.61	4	0.95	0.95
data-c-cv-u-n	5	1	0.48	0.48	1	0.48	0.48	5	1.00	1.00	5	1.00	1.00
data-uc-cv-nu-n	6	1	0.46	0.46	1	0.46	0.46	4	0.62	0.62	5	0.98	0.98
data-oo_v2	2	1	0.51	0.51	1	0.51	0.51	1	0.51	0.51	2	1.00	1.00
data-oo	6	1	0.48	0.48	1	0.48	0.48	5	0.51	0.51	6	1.00	1.00
data-uc-cc-nu-n_v2	3	2	0.50	0.50	2	0.50	0.50	2	0.50	0.50	3	1.00	1.00
data-uc-cc-nu-n	6	2	0.49	0.49	1	0.40	0.40	4	0.50	0.50	4	0.98	0.98
data-c-cc-nu-n2_v2	3	3	1.00	1.00	2	0.66	0.66	3	1.00	1.00	4	0.98	1.00
data-c-cc-nu-n2	6	2	0.73	0.73	1	0.50	0.50	6	1.00	1.00	7	0.98	1.00
dataX_v2	2	2	1.00	1.00	2	1.00	1.00	1	0.50	0.50	2	1.00	1.00
dataX	4	1	0.49	0.49	1	0.49	0.49	3	0.50	0.50	4	1.00	1.00
data-c-cc-nu-n_v2	3	1	0.49	0.49	1	0.49	0.49	3	1.00	1.00	3	1.00	1.00
train2	4	3	0.78	0.78	3	0.78	0.78	4	1.00	1.00	6	0.97	1.00
data-c-cc-nu-n	7	1	0.48	0.48	1	0.48	0.48	5	0.79	0.79	7	1.00	1.00
train1_v1	5	4	1.00	1.00	4	1.00	1.00	5	1.00	1.00	8	0.95	1.00
train1	6	4	0.99	0.99	4	0.99	0.99	6	1.00	1.00	9	0.95	1.00
train3_v1	5	2	0.50	0.50	2	0.50	0.50	7	0.99	1.00	6	0.99	1.00
train3	36	1	0.31	0.31	1	0.31	0.31	2	0.31	0.31	17	0.90	0.91
data_circle	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	14	0.84	1.00

Table C.2 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of JI and QJI for group 1 data sets (cont'd)

		KNN1			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
Data set	**#TC**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
data_mix_uniform_normal	2	2	1.00	1.00	2	1.00	1.00	10	0.98	1.00	38	0.56	1.00
data_circle_1_10_5_10	2	2	1.00	1.00	1	0.84	0.84	9	0.99	1.00	29	0.83	1.00
data_circle_10_1_10_10	2	1	0.84	0.84	1	0.84	0.84	18	1.00	1.00	22	0.84	1.00
data_circle_2_10_2_12	2	2	1.00	1.00	1	0.72	0.72	7	0.98	1.00	22	0.79	1.00
data_circle_2_10_3_12	2	1	0.72	0.72	1	0.72	0.72	6	0.99	1.00	30	0.80	1.00
data_circle_2_10_4_12	2	1	0.72	0.72	1	0.72	0.72	6	0.99	1.00	37	0.63	1.00
data_circle_2_10_5_13	2	2	1.00	1.00	2	1.00	1.00	6	0.98	1.00	30	0.74	1.00
data_circle_2_10_6_12	2	1	0.72	0.72	1	0.72	0.72	3	1.00	1.00	39	0.70	1.00
data_circle_2_10_3_12	2	1	0.72	0.72	1	0.72	0.72	6	0.99	1.00	30	0.80	1.00
data_circle_3_10_8_12	2	1	0.65	0.65	1	0.65	0.65	6	0.65	0.65	26	0.76	1.00
data_circle_5_10_8_12	2	1	0.56	0.56	1	0.56	0.56	6	0.55	0.56	29	0.77	1.00
data_circle1	2	1	0.91	0.91	1	0.91	0.91	7	0.99	1.00	35	0.66	1.00
data_circle2	2	1	0.91	0.91	1	0.91	0.91	11	0.98	1.00	35	0.85	1.00
data_circle_20_1_5_10	2	1	0.91	0.91	1	0.91	0.91	75	1.00	1.00	52	0.62	1.00
data_circle3	2	1	0.91	0.91	1	0.91	0.91	18	0.99	1.00	41	0.68	1.00
data_circle_1_20_1_11	2	1	0.91	0.91	1	0.91	0.91	13	0.97	1.00	49	0.72	1.00
data_circle_1_20_1_13	2	1	0.91	0.91	1	0.91	0.91	14	0.97	1.00	41	0.81	1.00
data_circle_1_20_1_15	2	1	0.91	0.91	1	0.91	0.91	7	0.99	1.00	47	0.85	1.00
data_circle_1_20_1_17	2	1	0.91	0.91	1	0.91	0.91	6	0.99	1.00	57	0.78	1.00
data_circle_1_20_1_19	2	1	0.91	0.91	1	0.91	0.91	5	0.90	0.91	38	0.88	1.00
iris	2	2	1.00	1.00	2	1.00	1.00	3	0.97	1.00	4	0.98	1.00
3d_dataset3	2	2	1.00	1.00	2	1.00	1.00	127	0.71	1.00	2	1.00	1.00
3d_dataset4	2	2	1.00	1.00	2	1.00	1.00	4	1.00	1.00	2	1.00	1.00

Table C.3 Comparison of KNN1,KNN2, ε-neighborhood and NC in terms of run time (in seconds) for group 1 data sets

		KNN1		**KNN2**		**ε-neighborhood**		**NC Algorithm**	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
data_60	3	3	0.04	1	0.02	3	0.07	3	1.24
data_66	4	4	0.02	1	0.02	4	0.03	4	1.51
data-c-cv-nu-n_v2	3	3	0.02	3	0.03	2	0.04	3	2.16
data-c-cv-nu-n	6	3	0.03	3	0.03	4	0.07	4	2.36
data-c-cv-u-n	5	1	0.03	1	0.03	5	0.05	5	2.77
data-uc-cv-nu-n	6	1	0.07	1	0.08	4	0.1	5	10.71
data-oo_v2	2	1	0.09	1	0.1	1	0.12	2	14.08
data-oo	6	1	0.09	1	0.11	5	0.13	6	15.15
data-uc-cc-nu-n_v2	3	2	0.16	2	0.18	2	0.19	3	34.27
data-uc-cc-nu-n	6	2	0.19	1	0.18	4	0.27	4	35.71
data-c-cc-nu-n2_v2	3	3	0.16	2	0.19	3	0.18	4	36.14
data-c-cc-nu-n2	6	2	0.17	1	0.19	6	0.18	7	37.72
dataX_v2	2	2	0.18	2	0.21	1	0.24	2	40.48
dataX	4	1	0.19	1	0.22	3	0.24	4	42.29
data-c-cc-nu-n_v2	3	1	0.37	1	0.42	3	0.4	3	119.27
train2	4	3	0.37	3	0.42	4	0.37	6	120.45
data-c-cc-nu-n	7	1	0.37	1	0.42	5	0.45	7	123.37
train1_v1	5	4	0.42	4	0.48	5	0.43	8	146.64
train1	6	4	0.42	4	0.48	6	0.43	9	147.54
train3_v1	5	2	0.59	2	0.67	7	0.57	6	242.31
train3	36	1	0.72	1	0.81	2	0.93	17	318.46
data_circle	2	2	2.27	2	2.66	3	2.15	14	1765.74

Table C.3 Comparison of KNN1,KNN2, ε-neighborhood and NC in terms of run time (in seconds) for group 1 data sets (cont'd)

		KNN1		**KNN2**		**ε-neighborhood**		**NC Algorithm**	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
data_mix_uniform_normal	2	2	4.64	2	5.52	10	4.13	38	3250.87
data_circle_1_10_5_10	2	2	5.68	1	6.85	9	5.09	29	4786.74
data_circle_10_1_10_10	2	1	5.67	1	6.84	18	5.63	22	1735.09
data_circle_2_10_2_12	2	2	6.82	1	8.3	7	6.17	22	3430.11
data_circle_2_10_3_12	2	1	6.79	1	8.27	6	6.02	30	2191.76
data_circle_2_10_4_12	2	1	6.81	1	8.3	6	5.97	37	2681.99
data_circle_2_10_5_13	2	2	6.78	2	8.26	6	5.94	30	3307.2
data_circle_2_10_6_12	2	1	6.81	1	8.25	3	5.96	39	3994.25
data_circle_2_10_3_12	2	1	6.78	1	8.27	6	6.03	30	2672.78
data_circle_3_10_8_12	2	1	8.01	1	9.87	6	7	26	1665.07
data_circle_5_10_8_12	2	1	10.88	1	13.55	6	9.12	29	2504.25
data_circle1	2	1	17.52	1	22.31	7	14.56	35	3414.44
data_circle2	2	1	17.63	1	22.53	11	14.55	35	3433.05
data_circle_20_1_5_10	2	1	21.99	1	28.26	75	19.28	52	9729.73
data_circle3	2	1	22	1	28.62	18	18.12	41	4734.05
data_circle_1_20_1_11	2	1	22	1	28.48	13	17.99	49	10609.76
data_circle_1_20_1_13	2	1	22.04	1	28.37	14	18.09	41	11779.36
data_circle_1_20_1_15	2	1	21.96	1	28.22	7	17.99	47	8461.14
data_circle_1_20_1_17	2	1	21.84	1	28.21	6	18.02	57	9706.72
data_circle_1_20_1_19	2	1	21.93	1	28.02	5	18	38	6495.46
iris	2	2	0.13	2	0.12	3	0.18	4	18.21
3d_dataset3	2	2	0.47	2	0.54	127	0.46	2	11923
3d_dataset4	2	2	10.98	2	13.36	4	9.21	2	2714383

Table C.4 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of PPN and RI for group 2 data sets

Data set	**#TC**	**KNN1**			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
		#C	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**
D_0000	5	2	0.50	0.60	1	0.29	0.25	5	1.00	1.00	5	1.00	1.00
D_0001	8	1	0.29	0.25	1	0.29	0.25	8	1.00	1.00	8	1.00	1.00
D_0010	5	1	0.29	0.25	1	0.29	0.25	5	1.00	1.00	5	1.00	1.00
D_0011	8	1	0.29	0.25	1	0.29	0.25	8	1.00	1.00	7	1.00	1.00
D_0100	5	2	0.50	0.60	1	0.28	0.25	15	1.00	0.97	5	1.00	1.00
D_0101	8	1	0.28	0.25	1	0.28	0.25	18	1.00	0.97	8	1.00	1.00
D_0110	5	1	0.29	0.24	1	0.29	0.24	15	1.00	0.99	5	1.00	1.00
D_0111	8	1	0.29	0.24	1	0.29	0.24	18	1.00	0.99	7	1.00	1.00
D_0200	5	1	0.29	0.25	1	0.29	0.25	6	1.00	1.00	66	1.00	0.95
D_0201	8	1	0.29	0.25	1	0.29	0.25	9	1.00	1.00	67	1.00	0.95
D_0210	5	1	0.29	0.25	1	0.29	0.25	6	1.00	1.00	65	0.82	0.87
D_0211	8	1	0.29	0.25	1	0.29	0.25	9	1.00	1.00	66	0.82	0.87
D_1000	5	1	0.29	0.25	1	0.29	0.25	11	1.00	0.97	14	1.00	0.92
D_1001	8	1	0.29	0.25	1	0.29	0.25	13	1.00	0.97	15	1.00	0.92
D_1010	5	1	0.29	0.25	1	0.29	0.25	10	0.76	0.85	15	0.88	0.81
D_1011	8	1	0.29	0.25	1	0.29	0.25	12	0.76	0.85	15	0.88	0.81
D_1100	5	1	0.29	0.25	1	0.29	0.25	13	0.76	0.85	8	1.00	0.99
D_1101	8	1	0.29	0.25	1	0.29	0.25	15	0.76	0.85	9	1.00	0.99
D_1110	5	1	0.29	0.25	1	0.29	0.25	9	0.52	0.57	13	0.79	0.85
D_1111	8	1	0.29	0.25	1	0.29	0.25	10	0.52	0.57	14	0.79	0.85
D_1200	5	1	0.29	0.25	1	0.29	0.25	8	0.76	0.86	45	1.00	0.92
D_1201	8	1	0.29	0.25	1	0.29	0.25	11	0.76	0.86	44	0.95	0.90
D_1210	5	1	0.29	0.25	1	0.29	0.25	7	0.51	0.58	38	0.41	0.39
D_1211	8	1	0.29	0.25	1	0.29	0.25	10	0.51	0.59	40	0.41	0.39

Table C.5 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of JI and QJI for group 2 data sets

Data set	#TC	KNN1			KNN2			ε-neighborhood			NC Algorithm		
		#C	JI	QJI	#C	JI	QJI	#C	JI	QJI	#C	JI	QJI
D_0000	5	2	0.38	0.38	1	0.25	0.25	5	1.00	1.00	5	1.00	1.00
D_0001	8	1	0.25	0.25	1	0.25	0.25	8	1.00	1.00	8	1.00	1.00
D_0010	5	1	0.25	0.25	1	0.25	0.25	5	1.00	1.00	5	1.00	1.00
D_0011	8	1	0.25	0.25	1	0.25	0.25	8	1.00	1.00	7	1.00	1.00
D_0100	5	2	0.38	0.38	1	0.25	0.25	15	0.88	1.00	5	1.00	1.00
D_0101	8	1	0.25	0.25	1	0.25	0.25	18	0.88	1.00	8	1.00	1.00
D_0110	5	1	0.24	0.24	1	0.24	0.24	15	0.94	1.00	5	1.00	1.00
D_0111	8	1	0.24	0.24	1	0.24	0.24	18	0.94	1.00	7	1.00	1.00
D_0200	5	1	0.25	0.25	1	0.25	0.25	6	0.99	1.00	66	0.78	1.00
D_0201	8	1	0.25	0.25	1	0.25	0.25	9	0.99	1.00	67	0.78	1.00
D_0210	5	1	0.25	0.25	1	0.25	0.25	6	0.99	1.00	65	0.60	0.77
D_0211	8	1	0.25	0.25	1	0.25	0.25	9	0.99	1.00	66	0.60	0.77
D_1000	5	1	0.25	0.25	1	0.25	0.25	11	0.86	1.00	14	0.68	1.00
D_1001	8	1	0.25	0.25	1	0.25	0.25	13	0.86	1.00	15	0.68	1.00
D_1010	5	1	0.25	0.25	1	0.25	0.25	10	0.58	0.67	15	0.37	0.81
D_1011	8	1	0.25	0.25	1	0.25	0.25	12	0.58	0.67	15	0.37	0.80
D_1100	5	1	0.25	0.25	1	0.25	0.25	13	0.59	0.67	8	0.97	1.00
D_1101	8	1	0.25	0.25	1	0.25	0.25	15	0.59	0.67	9	0.97	1.00
D_1110	5	1	0.25	0.25	1	0.25	0.25	9	0.33	0.39	13	0.57	0.70
D_1111	8	1	0.25	0.25	1	0.25	0.25	10	0.33	0.39	14	0.57	0.70
D_1200	5	1	0.25	0.25	1	0.25	0.25	8	0.63	0.67	45	0.68	1.00
D_1201	8	1	0.25	0.25	1	0.25	0.25	11	0.63	0.67	44	0.62	0.92
D_1210	5	1	0.25	0.25	1	0.25	0.25	7	0.36	0.38	38	0.25	0.31
D_1211	8	1	0.25	0.25	1	0.25	0.25	10	0.36	0.38	40	0.25	0.31

Table C.6 Comparison of KNN1,KNN2, ε-neighborhood and NC in terms of run time (in seconds) for group 2 data sets

Data set		KNN1		KNN2		ε-neighborhood		NC Algorithm	
	#TC	#C	Time	#C	Time	#C	Time	#C	Time
D_0000	5	2	39.41	1	51.24	5	29.17	5	1436761894.00
D_0001	8	1	39.86	1	51.69	8	29.31	8	2144832592.00
D_0010	5	1	39.40	1	51.43	5	29.31	5	1800125681.00
D_0011	8	1	39.54	1	51.96	8	29.25	7	3136493251.00
D_0100	5	2	18.86	1	23.39	15	14.76	5	467595257.00
D_0101	8	1	18.95	1	23.38	18	14.82	8	923769738.00
D_0110	5	1	17.88	1	22.10	15	14.25	5	686832794.00
D_0111	8	1	17.91	1	22.26	18	14.15	7	373263598.00
D_0200	5	1	39.80	1	52.55	6	30.08	66	1138221968.00
D_0201	8	1	39.92	1	52.48	9	30.10	67	2248430650.00
D_0210	5	1	39.84	1	52.63	6	30.10	65	3215476728.00
D_0211	8	1	39.85	1	52.48	9	30.16	66	2333344144.00
D_1000	5	1	39.52	1	52.13	11	30.81	14	291329.40
D_1001	8	1	39.69	1	52.75	13	31.01	15	1033781560.00
D_1010	5	1	39.62	1	52.62	10	31.11	15	1013577112.00
D_1011	8	1	40.23	1	52.64	12	31.02	15	1016609053.00
D_1100	5	1	17.91	1	22.38	13	15.45	8	336715385.00
D_1101	8	1	17.97	1	22.41	15	15.44	9	338026006.00
D_1110	5	1	18.38	1	23.09	9	15.73	13	351864525.00
D_1111	8	1	18.51	1	23.13	10	15.81	14	331960810.00
D_1200	5	1	40.08	1	52.74	8	31.70	45	1258736589.00
D_1201	8	1	40.03	1	52.96	11	31.72	44	1266772753.00
D_1210	5	1	40.16	1	53.12	7	31.60	38	1260551276.00
D_1211	8	1	40.17	1	53.70	10	31.79	40	1331819776.00

Table C.7 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of PPN and RI for group 3 data sets

Data set		**KNN1**			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
	#TC	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**	**#C**	**PPN**	**RI**
DS_0000	5	4	0.99	1.00	4	0.99	1.00	5	1.00	1.00	5	1.00	1.00
DS_0001	8	1	0.27	0.24	1	0.27	0.24	8	1.00	1.00	8	1.00	1.00
DS_0010	5	3	0.75	0.87	1	0.27	0.25	298	1.00	0.76	5	1.00	1.00
DS_0011	8	1	0.27	0.24	1	0.27	0.24	301	1.00	0.76	7	1.00	1.00
DS_0100	5	4	1.00	1.00	3	0.75	0.87	5	1.00	1.00	4	1.00	1.00
DS_0101	8	2	0.54	0.63	1	0.29	0.25	10	1.00	1.00	4	0.99	1.00
DS_0110	5	2	0.55	0.75	1	0.29	0.25	9	1.00	1.00	4	1.00	1.00
DS_0111	8	1	0.29	0.25	1	0.29	0.25	12	1.00	1.00	7	1.00	1.00
DS_0200	5	2	0.52	0.65	1	0.27	0.25	5	1.00	1.00	17	1.00	0.98
DS_0201	8	1	0.27	0.25	1	0.27	0.25	8	1.00	1.00	18	1.00	0.98
DS_0210	5	1	0.27	0.25	1	0.27	0.25	4	0.77	0.89	16	0.79	0.89
DS_0211	8	1	0.27	0.25	1	0.27	0.25	7	0.77	0.89	18	0.78	0.89
DS_1000	5	2	0.52	0.65	1	0.27	0.25	5	1.00	1.00	5	1.00	1.00
DS_1001	8	1	0.27	0.25	1	0.27	0.25	8	1.00	1.00	6	1.00	1.00
DS_1010	5	1	0.27	0.25	1	0.27	0.25	4	0.77	0.89	4	0.77	0.89
DS_1011	8	1	0.27	0.25	1	0.27	0.25	6	0.77	0.89	5	0.77	0.89
DS_1100	5	3	0.75	0.87	1	0.26	0.25	5	1.00	1.00	5	1.00	1.00
DS_1101	8	2	0.50	0.61	1	0.26	0.24	8	1.00	1.00	6	1.00	1.00
DS_1110	5	1	0.27	0.25	1	0.27	0.25	4	0.77	0.89	5	1.00	1.00
DS_1111	8	1	0.27	0.24	1	0.27	0.24	7	0.77	0.89	5	1.00	1.00
DS_1200	5	1	0.27	0.25	1	0.27	0.25	19	1.00	0.99	17	1.00	0.98
DS_1201	8	1	0.27	0.25	1	0.27	0.25	20	1.00	0.99	19	1.00	0.98
DS_1210	5	1	0.27	0.25	1	0.27	0.25	17	0.53	0.64	16	0.31	0.31
DS_1211	8	1	0.27	0.25	1	0.27	0.25	18	0.53	0.65	17	0.31	0.31

Table C.8 Comparison of KNN1, KNN2 ε-neighborhood and NC in terms of JI and QJI for group 3 data sets

Data set	**#TC**	**KNN1**			**KNN2**			**ε-neighborhood**			**NC Algorithm**		
		#C	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
DS_0000	5	4	0.99	0.99	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00
DS_0001	8	1	0.24	0.24	1	0.24	0.24	8	1.00	1.00	8	1.00	1.00
DS_0010	5	3	0.66	0.66	1	0.25	0.25	298	0.01	1.00	5	1.00	1.00
DS_0011	8	1	0.24	0.24	1	0.24	0.24	301	0.01	1.00	7	0.99	0.99
DS_0100	5	4	0.99	0.99	3	0.66	0.66	5	1.00	1.00	4	0.99	0.99
DS_0101	8	2	0.40	0.40	1	0.25	0.25	10	1.00	1.00	4	0.98	0.98
DS_0110	5	2	0.50	0.50	1	0.25	0.25	9	0.99	1.00	4	0.99	0.99
DS_0111	8	1	0.25	0.25	1	0.25	0.25	12	0.99	1.00	7	0.99	0.99
DS_0200	5	2	0.41	0.41	1	0.25	0.25	5	1.00	1.00	17	0.92	1.00
DS_0201	8	1	0.25	0.25	1	0.25	0.25	8	1.00	1.00	18	0.92	1.00
DS_0210	5	1	0.25	0.25	1	0.25	0.25	4	0.69	0.69	16	0.67	0.73
DS_0211	8	1	0.25	0.25	1	0.25	0.25	7	0.69	0.69	18	0.67	0.73
DS_1000	5	2	0.41	0.41	1	0.25	0.25	5	1.00	1.00	5	1.00	1.00
DS_1001	8	1	0.25	0.25	1	0.25	0.25	8	1.00	1.00	6	1.00	1.00
DS_1010	5	1	0.25	0.25	1	0.25	0.25	4	0.69	0.69	4	0.69	0.69
DS_1011	8	1	0.25	0.25	1	0.25	0.25	6	0.69	0.69	5	0.69	0.69
DS_1100	5	3	0.66	0.66	1	0.25	0.25	5	1.00	1.00	5	1.00	1.00
DS_1101	8	2	0.38	0.38	1	0.24	0.24	8	1.00	1.00	6	0.99	0.99
DS_1110	5	1	0.25	0.25	1	0.25	0.25	4	0.70	0.70	5	1.00	1.00
DS_1111	8	1	0.24	0.24	1	0.24	0.24	7	0.70	0.70	5	0.99	0.99
DS_1200	5	1	0.25	0.25	1	0.25	0.25	19	0.97	1.00	17	0.92	1.00
DS_1201	8	1	0.25	0.25	1	0.25	0.25	20	0.97	1.00	19	0.92	1.00
DS_1210	5	1	0.25	0.25	1	0.25	0.25	17	0.40	0.41	16	0.25	0.27
DS_1211	8	1	0.25	0.25	1	0.25	0.25	18	0.40	0.41	17	0.24	0.27

Table C.9 Comparison of KNN1,KNN2, ε-neighborhood and NC in terms of run time (in seconds) for group 3 data sets

Data set	**#TC**	**KNN1**		**KNN2**		**ε-neighborhood**		**NC Algorithm**	
		#C	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
DS_0000	5	4	4.18	4	4.88	5	3.72	5	4729.13
DS_0001	8	1	4.29	1	4.94	8	3.75	8	3953.68
DS_0010	5	3	4.12	1	4.81	298	3.47	5	3834.32
DS_0011	8	1	4.25	1	4.92	301	3.56	7	18660.97
DS_0100	5	4	2.59	3	2.96	5	2.41	4	1917.27
DS_0101	8	2	2.61	1	3.00	10	2.38	4	1931.07
DS_0110	5	2	2.58	1	2.98	9	2.32	4	1903.47
DS_0111	8	1	2.61	1	3.00	12	2.35	7	1931.58
DS_0200	5	2	4.20	1	4.88	5	3.78	17	9442.06
DS_0201	8	1	4.19	1	4.94	8	3.80	18	24099.10
DS_0210	5	1	4.17	1	4.89	4	3.78	16	3840.04
DS_0211	8	1	4.19	1	4.93	7	3.82	18	3876.84
DS_1000	5	2	4.13	1	4.81	5	3.87	5	3828.47
DS_1001	8	1	4.15	1	4.87	8	3.90	6	3863.46
DS_1010	5	1	4.12	1	4.83	4	3.86	4	3825.85
DS_1011	8	1	4.15	1	4.89	6	3.93	5	3845.62
DS_1100	5	3	2.57	1	2.95	5	2.44	5	1907.03
DS_1101	8	2	2.60	1	2.98	8	2.45	6	4522.14
DS_1110	5	1	2.63	1	3.02	4	2.48	5	4754.65
DS_1111	8	1	2.65	1	3.08	7	2.53	5	5077.17
DS_1200	5	1	4.19	1	4.92	19	3.97	17	6542.76
DS_1201	8	1	4.21	1	4.94	20	4.02	19	3901.61
DS_1210	5	1	4.18	1	4.97	17	4.01	16	3837.40
DS_1211	8	1	4.23	1	4.94	18	4.02	17	3876.83

APPENDIX D

THE NOM ALGORITHM AND THE EXPERIMENTAL RESULTS

We present the pseudocode of the NOM algorithm in Section D.1, and we provide the experimental results of the NOM algorithm in Tables D.1 through D.9 in Section D.2. Column headings in Tables D.1 through D.9 are explained as follows.

#TC	: number of target clusters
#C	: number of clusters found by the corresponding algorithm

D.1. The NOM Algorithm

```
Phase 1. Neighborhood Construction
Run NC algorithm and obtain $CS_i$, $i = 1,..,|D|$, and $Cl_m$, $m = 1,.., no\_closures$.

Phase 2. Outlier Detection
Set $no\_cluster = no\_closures$.
For $i = 1,..,|D|$
    Calculate local reachability density, $lrd_i$, and local outlier factor, $LOF_i$.
End for
For $i = 1,..,|D|$
    If $LOF_i > k \max_{j \in CS_i} \{LOF_j\}$
        Classify point $i$ as a local outlier and increase the number of clusters,
        $no\_cluster$, by 1.
    End if
End for
```

Phase 3. Merging

```
Set $m = 0$ and $flag1 = 0$.
Repeat
    Set $m = m + 1$, $flag2 = 0$ and $n = 0$.
    Find the set of clusters in the Gabriel graph neighborhood of cluster $m$, $\text{NGG}_m$.
    Calculate the potential compactness of cluster $m$, $pcomp_m$.
    Repeat
        Set $n = n + 1$ and move to the next nearest cluster neighbor $n$ of cluster $m$ in $\text{NGG}_m$.
        Calculate the separation between clusters $m$ and $n$, $csep_{mn}$, and current separation-to-compactness ratio, $current\_sc$.
        Calculate the lower bound for the possible separation after merging clusters $m$ and $n$, $lb$, and the new separation-to-compactness ratio, $new\_sc$.
        If $new\_sc > current\_sc$
            If $csep_{mn} \le \max\left\{ \frac{\max_{(i,j)\in \text{MST}_{i^*(\text{m})}} \{d_{ij}\}}{\min_{(i,j)\in \text{MST}_{i^*(\text{m})}} \{d_{ij}\}}, \frac{\max_{(i,j)\in \text{MST}_{j^*(\text{n})}} \{d_{ij}\}}{\min_{(i,j)\in \text{MST}_{j^*(\text{n})}} \{d_{ij}\}} \right\}$
                Merge clusters $m$ and $n$.
                Decrease the number of clusters, $no\_cluster$, by 1.
                Set $flag1 = 1$ and $flag2 = 1$.
            End if
        End if
    Until $flag2 = 1$ or $n = |\text{NGG}_m|$
    If $flag2 = 1$
        Set $m = 0$ and $flag1 = 0$.
    End if
Until $flag1 = 0$ and $m = no\_cluster$
```

D.2. The Experimental Results of the NOM Algorithm

Table D.1 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of PPN and RI for group 1 data sets

		K-means			Single-linkage			DBSCAN			NC			Outlier Detection			NOM		
Data set	**#TC**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**
data_60	3	2	0.90	0.85	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data_66	4	2	0.83	0.77	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00
data-c-cv-nu-n_v2	3	4	0.84	1.00	3	1.00	1.00	5	0.86	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data-c-cv-nu-n	6	5	0.83	0.97	6	1.00	1.00	3	0.68	0.86	4	0.98	0.97	6	1.00	1.00	6	1.00	1.00
data-c-cv-u-n	5	2	0.97	0.96	5	1.00	1.00	3	1.00	0.98	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
data-uc-cv-nu-n	6	5	0.83	0.98	5	0.99	0.99	4	0.99	0.98	5	0.99	0.99	6	1.00	1.00	6	1.00	1.00
data-oo_v2	2	5	0.76	1.00	7	0.95	1.00	3	0.98	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
data-oo	6	7	0.75	0.98	5	0.53	0.58	2	0.53	0.56	6	1.00	1.00	6	1.00	1.00	6	1.00	1.00
data-uc-cc-nu-n_v2	3	8	0.73	1.00	7	0.60	0.63	8	0.83	0.94	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data-uc-cc-nu-n	6	6	0.73	0.92	6	0.62	0.63	5	0.83	0.93	4	0.99	0.99	6	1.00	1.00	6	1.00	1.00
data-c-cc-nu-n2_v2	3	7	0.63	1.00	3	1.00	1.00	3	1.00	1.00	4	0.99	1.00	4	0.99	1.00	3	1.00	1.00
data-c-cc-nu-n2	6	7	0.64	0.98	6	1.00	1.00	4	1.00	0.99	7	0.99	1.00	7	0.99	1.00	6	1.00	1.00
dataX_v2	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
dataX	4	2	0.99	0.99	4	1.00	1.00	3	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00
data-c-cc-nu-n_v2	3	2	0.88	0.91	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
train2	4	3	0.91	0.90	4	1.00	1.00	4	1.00	1.00	6	0.99	1.00	6	0.99	1.00	4	1.00	1.00
data-c-cc-nu-n	7	2	0.86	0.89	6	1.00	1.00	4	1.00	0.99	7	1.00	1.00	7	1.00	1.00	7	1.00	1.00
train1_v1	5	4	1.00	1.00	5	1.00	1.00	5	1.00	1.00	8	0.98	1.00	8	0.98	1.00	5	1.00	1.00
train1	6	4	1.00	0.99	6	1.00	1.00	5	1.00	1.00	9	0.98	1.00	9	0.98	1.00	6	1.00	1.00
train3_v1	5	6	0.75	0.86	5	1.00	1.00	5	1.00	1.00	6	1.00	1.00	6	1.00	1.00	6	1.00	1.00
train3	36	7	0.79	0.86	10	0.64	0.68	6	0.99	0.92	17	0.97	0.94	20	0.97	0.95	14	0.79	0.78
data_circle	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	14	0.88	1.00	14	0.88	1.00	2	1.00	1.00
data_mix_uniform_normal	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	38	0.83	1.00	39	0.73	1.00	12	0.92	1.00
data_circle_1_10_5_10	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	29	0.86	1.00	29	0.86	1.00	4	0.99	1.00
data_circle_10_1_10_10	2	2	1.00	1.00	7	1.00	1.00	3	0.99	1.00	22	0.87	1.00	23	0.86	1.00	7	0.97	1.00

Table D.1 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of PPN and RI for group 1 data sets (cont'd)

Data set		**K-means**			**Single-linkage**			**DBSCAN**			**NC**			**Outlier Detection**			**NOM**		
	#TC	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**			**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**			**#C**	**RI**
data_circle_2_10_2_12	2	2	0.95	0.98	2	1.00	1.00	3	1.00	1.00	22	0.85	1.00	23	0.85	1.00	8	0.96	1.00
data_circle_2_10_3_12	2	2	0.91	0.95	2	1.00	1.00	3	1.00	1.00	30	0.86	1.00	31	0.86	1.00	11	0.97	1.00
data_circle_2_10_4_12	2	2	0.95	0.97	2	1.00	1.00	2	1.00	1.00	37	0.74	1.00	37	0.74	1.00	11	0.94	1.00
data_circle_2_10_5_13	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	30	0.81	1.00	30	0.81	1.00	8	0.94	1.00
data_circle_2_10_6_12	2	2	0.93	0.96	2	1.00	1.00	2	1.00	1.00	39	0.79	1.00	40	0.79	1.00	10	0.97	1.00
data_circle_2_10_3_12	2	2	0.91	0.95	2	1.00	1.00	3	1.00	1.00	30	0.86	1.00	31	0.86	1.00	11	0.97	1.00
data_circle_3_10_8_12	2	2	0.95	0.98	10	0.99	1.00	3	0.99	1.00	26	0.85	1.00	26	0.85	1.00	2	0.65	0.77
data_circle_5_10_8_12	2	2	0.96	0.98	7	0.99	1.00	3	0.99	1.00	29	0.87	1.00	30	0.87	1.00	30	0.87	1.00
data_circle1	2	2	0.62	0.95	2	1.00	1.00	2	1.00	1.00	35	0.69	1.00	36	0.69	1.00	9	0.96	1.00
data_circle2	2	2	0.65	0.95	2	1.00	1.00	3	0.99	1.00	35	0.86	1.00	36	0.86	1.00	12	0.97	1.00
data_circle_20_1_5_10	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	52	0.66	1.00	53	0.64	1.00	10	0.97	1.00
data_circle3	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	41	0.71	1.00	41	0.71	1.00	9	0.99	1.00
data_circle_1_20_1_11	2	2	0.67	0.95	2	1.00	1.00	3	1.00	1.00	49	0.74	1.00	49	0.74	1.00	9	0.89	1.00
data_circle_1_20_1_13	2	2	0.65	0.95	2	1.00	1.00	3	1.00	1.00	41	0.83	1.00	41	0.83	1.00	5	0.94	1.00
data_circle_1_20_1_15	2	2	0.64	0.95	2	1.00	1.00	3	1.00	1.00	47	0.86	1.00	49	0.86	1.00	8	0.99	1.00
data_circle_1_20_1_17	2	2	0.61	0.95	2	1.00	1.00	2	1.00	1.00	57	0.80	1.00	57	0.80	1.00	14	0.96	1.00
data_circle_1_20_1_19	2	2	0.58	0.95	2	0.91	0.95	2	0.91	0.95	38	0.89	1.00	38	0.89	1.00	7	0.89	0.95
iris	2	3	0.88	0.89	6	0.78	0.69	3	0.78	1.00	4	0.92	1.00	4	1.00	0.92	2	1.00	1.00
3d_dataset3	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
3d_dataset4	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00

Table D.2 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of JI and QJI for group 1 data sets

		K-means			**Single-linkage**			**DBSCAN**			**NC**			**Outlier Detection**			**NOM**		
Data set	**#TC**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
data_60	3	2	0.79	0.79	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data_66	4	2	0.66	0.66	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00
data-c-cv-nu-n_v2	3	4	0.61	1.00	3	1.00	1.00	5	0.66	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data-c-cv-nu-n	6	5	0.59	0.97	6	1.00	1.00	3	0.63	0.81	4	0.95	0.95	6	1.00	1.00	6	1.00	1.00
data-c-cv-u-n	5	2	0.93	0.93	5	1.00	1.00	3	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
data-uc-cv-nu-n	6	5	0.62	0.98	5	0.98	0.98	4	0.98	0.98	5	0.98	0.98	6	1.00	1.00	6	1.00	1.00
data-oo_v2	2	5	0.52	1.00	7	0.89	1.00	3	0.95	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
data-oo	6	7	0.49	0.99	5	0.50	0.50	2	0.50	0.50	6	1.00	1.00	6	1.00	1.00	6	1.00	1.00
data-uc-cc-nu-n_v2	3	8	0.34	1.00	7	0.45	0.57	8	0.59	0.98	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
data-uc-cc-nu-n	6	6	0.59	0.93	6	0.48	0.57	5	0.50	0.97	4	0.98	0.98	6	1.00	1.00	6	1.00	1.00
data-c-cc-nu-n2_v2	3	7	0.29	1.00	3	1.00	1.00	3	1.00	1.00	4	0.98	1.00	4	0.98	1.00	3	1.00	1.00
data-c-cc-nu-n2	6	7	0.28	0.99	6	1.00	1.00	4	1.00	1.00	7	0.98	1.00	7	0.98	1.00	6	1.00	1.00
dataX_v2	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
dataX	4	2	0.98	0.98	4	1.00	1.00	3	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00
data-c-cc-nu-n_v2	3	2	0.80	0.80	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00	3	1.00	1.00
train2	4	3	0.78	0.78	4	1.00	1.00	4	1.00	1.00	6	0.97	1.00	6	0.96	1.00	4	1.00	1.00
data-c-cc-nu-n	7	2	0.78	0.78	6	0.99	0.99	4	0.99	0.99	7	1.00	1.00	7	1.00	1.00	7	1.00	1.00
train1_v1	5	4	1.00	1.00	5	1.00	1.00	5	1.00	1.00	8	0.95	1.00	8	0.95	1.00	5	1.00	1.00
train1	6	4	0.99	0.99	6	1.00	1.00	5	1.00	1.00	9	0.95	1.00	9	0.95	1.00	6	1.00	1.00
train3_v1	5	6	0.39	0.90	5	1.00	1.00	5	1.00	1.00	6	0.99	1.00	6	0.99	1.00	6	0.99	1.00
train3	36	7	0.37	0.90	10	0.46	0.46	6	0.97	0.97	17	0.90	0.91	20	0.91	0.92	14	0.59	0.59
data_circle	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	14	0.84	1.00	14	0.84	1.00	2	1.00	1.00
data_mix_uniform_normal	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	38	0.56	1.00	39	0.46	1.00	12	0.84	1.00
data_circle_1_10_5_10	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	29	0.83	1.00	29	0.83	1.00	4	0.99	1.00
data_circle_10_1_10_10	2	2	1.00	1.00	7	1.00	1.00	3	0.99	1.00	22	0.84	1.00	23	0.84	1.00	7	0.96	1.00

Table D.2 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of JI and QJI for group 1 data sets (cont'd)

		K-means			**Single-linkage**			**DBSCAN**			**NC**			**Outlier Detection**			**NOM**		
Data set	**#TC**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
data_circle_2_10_2_12	2	2	0.93	0.99	2	1.00	1.00	3	1.00	1.00	22	0.79	1.00	23	0.79	1.00	8	0.94	1.00
data_circle_2_10_3_12	2	2	0.88	0.98	2	1.00	1.00	3	1.00	1.00	30	0.80	1.00	31	0.80	1.00	11	0.96	1.00
data_circle_2_10_4_12	2	2	0.93	0.99	2	1.00	1.00	2	1.00	1.00	37	0.63	1.00	37	0.63	1.00	11	0.92	1.00
data_circle_2_10_5_13	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	30	0.74	1.00	30	0.74	1.00	8	0.92	1.00
data_circle_2_10_6_12	2	2	0.90	0.98	2	1.00	1.00	2	1.00	1.00	39	0.70	1.00	40	0.70	1.00	10	0.96	1.00
data_circle_2_10_3_12	2	2	0.88	0.98	2	1.00	1.00	3	1.00	1.00	30	0.80	1.00	31	0.80	1.00	11	0.96	1.00
data_circle_3_10_8_12	2	2	0.93	0.98	10	0.99	1.00	3	0.99	1.00	26	0.76	1.00	26	0.76	1.00	2	0.65	0.65
data_circle_5_10_8_12	2	2	0.93	0.98	7	0.99	1.00	3	0.99	1.00	29	0.77	1.00	30	0.77	1.00	30	0.77	1.00
data_circle1	2	2	0.59	0.97	2	1.00	1.00	2	1.00	1.00	35	0.66	1.00	36	0.66	1.00	9	0.95	1.00
data_circle2	2	2	0.63	0.98	2	1.00	1.00	3	0.99	1.00	35	0.85	1.00	36	0.85	1.00	12	0.97	1.00
data_circle_20_1_5_10	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	52	0.62	1.00	53	0.60	1.00	10	0.96	1.00
data_circle3	2	2	1.00	1.00	2	1.00	1.00	3	1.00	1.00	41	0.68	1.00	41	0.68	1.00	9	0.99	1.00
data_circle_1_20_1_11	2	2	0.65	0.98	2	1.00	1.00	3	1.00	1.00	49	0.72	1.00	49	0.72	1.00	9	0.88	1.00
data_circle_1_20_1_13	2	2	0.63	0.98	2	1.00	1.00	3	1.00	1.00	41	0.81	1.00	41	0.81	1.00	5	0.93	1.00
data_circle_1_20_1_15	2	2	0.61	0.98	2	1.00	1.00	3	1.00	1.00	47	0.85	1.00	49	0.85	1.00	8	0.99	1.00
data_circle_1_20_1_17	2	2	0.58	0.97	2	1.00	1.00	2	1.00	1.00	57	0.78	1.00	57	0.78	1.00	14	0.95	1.00
data_circle_1_20_1_19	2	2	0.55	0.97	2	0.91	0.91	2	0.91	0.91	38	0.88	1.00	38	0.88	1.00	7	0.89	0.91
iris	2	3	0.70	0.83	6	0.57	0.63	3	0.59	0.60	4	0.86	1.00	4	0.86	1.00	2	1.00	1.00
3d_dataset3	2	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00
3d_dataset4	2	2	1	1	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00	2	1.00	1.00

Table D.3 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of time (in seconds) for group 1 data sets

		K-means		**Single-linkage**		**DBSCAN**		**NC**		**Outlier Detection**		**NOM*** **	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
data_60	3	2	0.21	3	0.52	3	0.07	3	1.24	3	2.23	3	3.74
data_66	4	2	0.07	4	0.38	4	0.03	4	1.51	4	1.08	4	2.65
data-c-cv-nu-n_v2	3	4	0.19	3	0.39	5	0.06	3	2.16	3	1.71	3	3.97
data-c-cv-nu-n	6	5	0.05	6	0.38	3	0.04	4	2.36	6	1.91	6	4.37
data-c-cv-u-n	5	2	0.11	5	0.38	3	0.24	5	2.77	5	3.37	5	6.30
data-uc-cv-nu-n	6	5	0.07	5	0.42	4	0.13	5	10.71	6	7.44	6	18.34
data-oo_v2	2	5	0.73	7	0.43	3	1.95	2	14.08	2	13.72	2	28.04
data-oo	6	7	0.11	5	0.43	2	0.23	6	15.15	6	14.28	6	29.67
data-uc-cc-nu-n_v2	3	8	0.09	7	0.48	8	0.13	3	34.27	3	9.30	3	43.81
data-uc-cc-nu-n	6	6	0.10	6	0.48	5	0.12	4	35.71	6	9.37	6	45.33
data-c-cc-nu-n2_v2	3	7	0.32	3	0.49	3	0.38	4	36.14	4	16.69	3	53.49
data-c-cc-nu-n2	6	7	0.11	6	0.49	4	0.19	7	37.72	7	17.06	6	55.42
dataX_v2	2	2	0.83	2	0.50	2	2.34	2	40.48	2	14.94	2	55.74
dataX	4	2	0.10	4	0.51	3	0.13	4	42.29	4	15.00	4	57.67
data-c-cc-nu-n_v2	3	2	0.05	3	0.87	3	0.05	3	119.27	3	33.48	3	153.28
train2	4	3	0.10	4	0.65	4	0.11	6	120.45	6	21.81	4	143.59
data-c-cc-nu-n	7	2	0.24	6	0.66	4	0.26	7	123.37	7	34.31	7	159.49
train1_v1	5	4	0.08	5	0.72	5	0.11	8	146.64	8	22.03	5	170.14
train1	6	4	0.07	6	0.69	5	0.13	9	147.54	9	21.84	6	170.90
train3_v1	5	6	0.07	5	0.80	5	0.12	6	242.31	6	45.25	6	288.13
train3	36	7	0.14	10	0.91	6	0.23	17	318.46	20	51.66	14	377.31
data_circle	2	2	0.10	2	2.17	2	0.12	14	1765.74	14	606.09	2	2390.75
data_mix_uniform_normal	2	2	0.17	2	4.19	3	0.29	38	3250.87	39	295.99	12	3700.85
data_circle_1_10_5_10	2	2	0.96	2	5.39	3	2.07	29	4786.74	29	2451.40	4	7388.68
data_circle_10_1_10_10	2	2	2.14	7	5.55	3	7.11	22	1735.09	23	2472.22	7	4291.73

Table D.3 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of time (in seconds) for group 1 data sets (cont'd)

		K-means		**Single-linkage**		**DBSCAN**		**NC**		**Outlier Detection**		**NOM***	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**			**#C**	**Time**	**#C**	**Time**
data_circle_2_10_2_12	2	2	0.92	2	6.63	3	2.75	22	3430.11	23	2273.90	8	5784.75
data_circle_2_10_3_12	2	2	0.93	2	6.26	3	4.91	30	2191.76	31	2359.58	11	4691.89
data_circle_2_10_4_12	2	2	1.24	2	6.47	2	7.05	37	2681.99	37	1681.65	11	4564.74
data_circle_2_10_5_13	2	2	1.34	2	6.23	2	8.61	30	3307.20	30	2088.10	8	5532.97
data_circle_2_10_6_12	2	2	1.98	2	6.23	2	8.37	39	3994.25	40	1948.46	10	6187.39
data_circle_2_10_3_12	2	2	0.93	2	6.26	3	4.91	30	2191.76	31	2359.58	11	4691.89
data_circle_3_10_8_12	2	2	2.14	10	7.54	3	12.86	26	1665.07	26	2224.98	2	4063.81
data_circle_5_10_8_12	2	2	2.71	7	10.23	3	5.71	29	2504.25	30	2403.26	30	5092.63
data_circle1	2	2	0.29	2	16.69	2	0.47	35	3414.44	36	10983.40	9	15261.26
data_circle2	2	2	0.40	2	16.83	3	0.82	35	3433.05	36	16201.94	12	20495.21
data_circle_20_1_5_10	2	2	0.11	2	19.68	3	0.34	52	9729.73	53	26124.00	10	59456.07
data_circle3	2	2	0.62	2	19.33	3	2.77	41	4734.05	41	30617.86	9	36674.49
data_circle_1_20_1_11	2	2	0.92	2	19.57	3	2.75	49	10609.76	49	15616.89	9	27725.93
data_circle_1_20_1_13	2	2	0.77	2	20.71	3	3.64	41	11779.36	41	18708.61	5	31817.30
data_circle_1_20_1_15	2	2	0.77	2	19.08	3	3.68	47	8461.14	49	19980.85	8	30163.46
data_circle_1_20_1_17	2	2	1.06	2	20.68	2	5.63	57	9706.72	57	17753.30	14	29523.94
data_circle_1_20_1_19	2	2	0.92	2	20.34	2	2.75	38	6495.46	38	21364.07	7	29182.07
iris	2	3	2.19	6	0.57	3	7.45	4	18.21	4	6.24	2	25.02
3d_dataset3	2	2	1.99	2	1.00	2	6.11	2	11923.00	2	399.94	2	12325.35
3d_dataset4	2	2	2.11	2	32.60	2	5.57	2	2714383.00	2	82117.80	2	2796865.30

*** Times for NOM include NC and Outlier Detection times.

Table D.4 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of PPN and RI for group 2 data sets

		K-means			Single-link			DBSCAN			NC			Outlier Detection			NOM		
Data set	**#TC**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**
D_0000	5	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0001	8	4	0.99	0.99	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0010	5	4	0.98	0.98	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0011	8	4	0.98	0.98	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0100	5	4	1.00	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0101	8	4	0.99	0.99	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0110	5	4	0.98	0.97	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0111	8	4	0.98	0.97	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	7	1.00	1.00	7	1.00	1.00
D_0200	5	4	0.97	0.97	5	1.00	1.00	5	1.00	1.00	66	0.95	1.00	68	0.95	1.00	17	0.96	0.99
D_0201	8	4	0.97	0.97	8	1.00	1.00	6	0.99	0.99	67	0.95	1.00	71	0.95	1.00	19	0.96	0.99
D_0210	5	4	0.95	0.94	5	1.00	1.00	5	1.00	1.00	65	0.87	0.82	68	0.87	0.82	17	0.88	0.79
D_0211	8	4	0.95	0.93	8	1.00	1.00	6	0.98	0.98	66	0.87	0.82	71	0.87	0.82	18	0.88	0.79
D_1000	5	4	0.99	0.99	4	0.88	0.76	4	0.99	0.99	14	0.92	1.00	14	0.92	1.00	14	0.92	1.00
D_1001	8	5	0.97	0.98	6	0.88	0.76	4	0.99	0.99	15	0.92	1.00	16	0.92	1.00	16	0.92	1.00
D_1010	5	5	0.90	0.88	10	0.85	0.76	10	0.89	0.99	15	0.81	0.88	15	0.81	0.88	15	0.81	0.88
D_1011	8	5	0.91	0.89	10	0.57	0.51	10	0.89	0.99	15	0.81	0.88	16	0.81	0.88	16	0.81	0.88
D_1100	5	4	1.00	0.99	4	0.88	0.76	4	1.00	0.99	8	0.99	1.00	8	0.99	1.00	8	0.99	1.00
D_1101	8	4	0.99	0.99	6	0.88	0.76	4	0.99	0.99	9	0.99	1.00	9	0.99	1.00	9	0.99	1.00
D_1110	5	5	0.90	0.89	10	0.85	0.77	18	0.84	0.77	13	0.85	0.79	17	0.85	0.79	17	0.85	0.79
D_1111	8	5	0.89	0.87	10	0.57	0.52	18	0.84	0.77	14	0.85	0.79	18	0.85	0.79	18	0.85	0.79
D_1200	5	5	0.96	0.97	4	0.88	0.76	10	0.98	1.00	45	0.92	1.00	46	0.92	1.00	17	0.95	0.99
D_1201	8	4	0.96	0.95	10	0.88	0.76	10	0.98	1.00	44	0.90	0.95	48	0.90	0.95	15	0.94	0.94
D_1210	5	4	0.94	0.93	7	0.58	0.51	10	0.85	0.76	38	0.39	0.41	42	0.40	0.41	15	0.29	0.32
D_1211	8	5	0.90	0.90	10	0.59	0.51	10	0.85	0.76	40	0.39	0.41	45	0.40	0.41	17	0.29	0.32

Table D.5 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of JI and QJI for group 2 data sets

		K-means			Single-link			DBSCAN			NC			Outlier Detection			NOM		
Data set	**#TC**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
D_0000	5	4	0.98	0.98	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0001	8	4	0.98	0.98	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0010	5	4	0.94	0.96	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0011	8	4	0.94	0.96	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0100	5	4	0.98	0.98	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0101	8	4	0.98	0.98	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
D_0110	5	4	0.93	0.95	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
D_0111	8	4	0.92	0.95	8	1.00	1.00	6	0.99	0.99	8	1.00	1.00	7	1.00	1.00	7	1.00	1.00
D_0200	5	4	0.90	0.94	5	1.00	1.00	5	1.00	1.00	66	0.78	1.00	68	0.78	1.00	17	0.84	0.98
D_0201	8	4	0.90	0.94	8	1.00	1.00	6	0.99	0.99	67	0.78	1.00	71	0.78	1.00	19	0.84	0.98
D_0210	5	4	0.81	0.89	5	1.00	1.00	5	1.00	1.00	65	0.60	0.77	68	0.60	0.77	17	0.63	0.73
D_0211	8	4	0.81	0.89	8	1.00	1.00	6	0.94	1.00	66	0.60	0.77	71	0.60	0.77	18	0.63	0.72
D_1000	5	4	0.97	0.98	4	0.67	0.67	4	0.98	0.98	14	0.68	1.00	14	0.68	1.00	14	0.68	1.00
D_1001	8	5	0.89	0.99	6	0.67	0.67	4	0.98	0.98	15	0.68	1.00	16	0.68	1.00	16	0.68	1.00
D_1010	5	5	0.64	0.89	10	0.58	0.67	10	0.56	0.98	15	0.37	0.81	15	0.37	0.81	15	0.37	0.81
D_1011	8	5	0.66	0.90	10	0.33	0.38	10	0.56	0.98	15	0.37	0.80	16	0.37	0.80	16	0.37	0.80
D_1100	5	4	0.98	0.98	4	0.67	0.67	4	0.98	0.98	8	0.97	1.00	8	0.97	1.00	8	0.97	1.00
D_1101	8	4	0.98	0.98	6	0.67	0.67	4	0.98	0.98	9	0.97	1.00	9	0.97	1.00	9	0.97	1.00
D_1110	5	5	0.65	0.90	10	0.58	0.68	18	0.56	0.68	13	0.57	0.70	17	0.57	0.70	17	0.57	0.70
D_1111	8	5	0.62	0.88	10	0.33	0.39	18	0.56	0.68	14	0.57	0.70	18	0.57	0.70	18	0.57	0.70
D_1200	5	5	0.86	0.97	4	0.67	0.67	10	0.92	1.00	45	0.68	1.00	46	0.68	1.00	17	0.80	0.99
D_1201	8	4	0.85	0.91	10	0.67	0.67	10	0.92	1.00	44	0.62	0.92	48	0.62	0.92	15	0.79	0.90
D_1210	5	4	0.80	0.88	7	0.36	0.38	10	0.58	0.67	38	0.25	0.31	42	0.25	0.31	15	0.25	0.26
D_1211	8	5	0.64	0.88	10	0.36	0.38	10	0.58	0.67	40	0.25	0.31	45	0.25	0.31	17	0.25	0.26

Table D.6 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of time (in seconds) for group 2 data sets

		K-means		**Single-link**		**DBSCAN**		**NC**		**Outlier Detection**		**NOM*****	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
D_0000	5	4	4.40	5	292.36	5	13.58	5	1436761894.00	5	94559.45	5	1436856511.00
D_0001	8	4	3.69	8	293.72	6	13.09	8	2144832592.00	8	51118.54	8	2144883768.88
D_0010	5	4	5.34	5	283.95	5	12.41	5	1800125681.00	5	75706.78	5	1800201707.24
D_0011	8	4	3.88	8	267.73	6	13.48	8	3136493251.00	8	83338.22	8	3136576925.47
D_0100	5	4	2.29	5	110.82	5	10.95	5	467595257.00	5	14622.20	5	467609896.69
D_0101	8	4	2.26	8	102.05	6	14.40	8	923769738.00	8	12538.38	8	923782294.18
D_0110	5	4	2.15	5	109.68	5	9.46	5	686832794.00	5	5455.99	5	686838265.96
D_0111	8	4	2.26	8	101.57	6	14.27	8	373263598.00	7	5548.29	7	373269162.48
D_0200	5	4	4.76	5	48.71	5	12.59	66	1138221968.00	68	4456.46	17	1138228858.07
D_0201	8	4	3.74	8	48.79	6	12.66	67	2248430650.00	71	12642.69	19	2248450350.28
D_0210	5	4	6.37	5	48.96	5	12.93	65	3215476728.00	68	18286.87	17	3215501304.02
D_0211	8	4	3.84	8	48.82	6	23.01	66	2333344144.00	71	22100.90	18	2333374030.94
D_1000	5	4	6.08	4	168.81	4	20.09	14	291329.40	14	9770.67	14	301121.38
D_1001	8	5	5.86	6	170.14	4	232.33	15	1033781560.00	16	9587.78	16	1033791169.13
D_1010	5	5	4.02	10	162.47	10	10.89	15	1013577112.00	15	8286.46	15	1013585417.27
D_1011	8	5	3.90	10	164.47	10	14.87	15	1016609053.00	16	8218.61	16	1016617291.82
D_1100	5	4	1.91	4	58.96	4	9.73	8	336715385.00	8	2356.02	8	336717813.29
D_1101	8	4	1.97	6	59.17	4	10.91	9	338026006.00	9	2351.49	9	338028447.80
D_1110	5	5	2.27	10	58.97	18	16.69	13	351864525.00	17	3825.27	17	351868408.21
D_1111	8	5	2.28	10	59.77	18	11.94	14	331960810.00	18	3641.22	18	331964510.91
D_1200	5	5	3.78	4	48.61	10	225.36	45	1258736589.00	46	10584.14	17	1258751359.58
D_1201	8	4	3.81	10	48.66	10	16.31	44	1266772753.00	48	12758.56	15	1266788467.91
D_1210	5	4	3.90	7	48.55	10	15.97	38	1260551276.00	42	3485.56	15	1260554764.68
D_1211	8	5	3.92	10	48.64	10	14.79	40	1331819776.00	45	5800.89	17	1331825587.75

*** Times for NOM include NC and Outlier Detection times.

Table D.7 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of PPN and RI for group 3 data sets

		K-means			**Single-linkage**			**DBSCAN**			**NC**			**Outlier Detection**			**NOM**		
Data set	**#TC**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**	**#C**	**RI**	**PPN**
DS_0000	5	4	1.00	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_0001	8	4	0.99	0.98	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
DS_0010	5	4	1.00	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_0011	8	4	0.99	0.98	8	1.00	1.00	5	1.00	0.99	7	1.00	1.00	8	1.00	1.00	8	1.00	1.00
DS_0100	5	4	1.00	1.00	5	1.00	1.00	5	1.00	1.00	4	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_0101	8	4	1.00	0.99	8	1.00	1.00	5	1.00	1.00	4	1.00	0.99	6	1.00	0.99	6	1.00	0.99
DS_0110	5	4	1.00	1.00	5	1.00	1.00	5	1.00	1.00	4	1.00	1.00	4	1.00	1.00	4	1.00	1.00
DS_0111	8	4	1.00	0.99	8	1.00	1.00	5	1.00	1.00	7	1.00	1.00	7	1.00	1.00	7	1.00	1.00
DS_0200	5	4	1.00	0.99	5	1.00	1.00	5	1.00	1.00	17	0.98	1.00	19	0.98	1.00	13	0.98	0.99
DS_0201	8	4	0.99	0.99	8	1.00	1.00	5	1.00	1.00	18	0.98	1.00	21	0.98	1.00	10	0.86	0.76
DS_0210	5	4	0.98	0.97	5	1.00	1.00	5	0.98	0.97	16	0.89	0.79	18	0.89	0.79	11	0.88	0.77
DS_0211	8	4	0.97	0.97	8	1.00	1.00	5	0.99	0.98	18	0.89	0.78	21	0.89	0.79	11	0.65	0.54
DS_1000	5	5	0.96	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1001	8	5	0.96	0.99	8	1.00	1.00	5	1.00	1.00	6	1.00	1.00	8	1.00	1.00	8	1.00	1.00
DS_1010	5	5	0.94	0.97	4	0.90	0.87	4	1.00	0.99	4	0.89	0.77	4	0.89	0.77	4	0.89	0.77
DS_1011	8	6	0.90	0.92	7	0.53	0.57	4	0.99	0.99	5	0.89	0.77	6	0.89	0.77	6	0.89	0.77
DS_1100	5	4	1.00	0.99	5	0.75	0.80	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1101	8	4	0.99	0.99	10	0.74	0.80	5	1.00	1.00	6	1.00	1.00	6	1.00	1.00	6	1.00	1.00
DS_1110	5	5	0.94	0.96	5	0.73	0.70	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1111	8	6	0.92	0.96	10	0.45	0.62	5	1.00	1.00	5	1.00	1.00	6	1.00	1.00	6	1.00	1.00
DS_1200	5	5	0.96	0.99	6	0.62	0.77	5	1.00	1.00	17	0.98	1.00	18	0.98	1.00	11	0.99	0.99
DS_1201	8	6	0.93	0.99	10	0.60	0.66	5	1.00	1.00	19	0.98	1.00	21	0.98	1.00	10	0.64	0.54
DS_1210	5	5	0.94	0.97	8	0.64	0.53	5	0.96	0.99	16	0.31	1.00	17	0.31	0.31	13	0.30	0.31
DS_1211	8	5	0.94	0.97	8	0.27	0.28	5	0.96	0.99	17	0.31	1.00	18	0.31	0.31	14	0.30	0.31

Table D.8 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of JI and QJI for group 3 data sets

		K-means			Single-linkage			DBSCAN			NC			Outlier Detection			NOM		
Data set	**#TC**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**	**#C**	**JI**	**QJI**
DS_0000	5	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_0001	8	4	0.97	0.97	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00	8	1.00	1.00
DS_0010	5	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_0011	8	4	0.97	0.97	8	1.00	1.00	5	1.00	1.00	7	0.99	0.99	8	1.00	1.00	8	1.00	1.00
DS_0100	5	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00	4	0.99	0.99	5	0.99	0.99	5	0.99	0.99
DS_0101	8	4	0.98	0.98	8	1.00	1.00	5	1.00	1.00	4	0.98	0.98	6	0.99	0.99	6	0.99	0.99
DS_0110	5	4	0.99	0.99	5	1.00	1.00	5	1.00	1.00	4	0.99	0.99	4	0.99	0.99	4	0.99	0.99
DS_0111	8	4	0.98	0.98	8	1.00	1.00	5	1.00	1.00	7	0.99	0.99	7	0.99	0.99	7	0.99	0.99
DS_0200	5	4	0.98	0.98	5	1.00	1.00	5	1.00	1.00	17	0.92	1.00	19	0.92	1.00	13	0.93	0.99
DS_0201	8	4	0.98	0.98	8	1.00	1.00	5	1.00	1.00	18	0.92	1.00	21	0.92	1.00	10	0.64	0.66
DS_0210	5	4	0.91	0.95	5	1.00	1.00	5	0.92	0.99	16	0.67	0.73	18	0.67	0.73	11	0.67	0.70
DS_0211	8	4	0.90	0.94	8	1.00	1.00	5	0.96	1.00	18	0.67	0.73	21	0.67	0.73	11	0.40	0.42
DS_1000	5	5	0.86	0.99	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1001	8	5	0.84	0.99	8	1.00	1.00	5	1.00	1.00	6	1.00	1.00	8	1.00	1.00	8	1.00	1.00
DS_1010	5	5	0.77	0.95	4	0.69	0.79	4	0.99	0.99	4	0.69	0.69	4	0.69	0.69	4	0.69	0.69
DS_1011	8	6	0.60	0.93	7	0.40	0.45	4	0.98	0.98	5	0.69	0.69	6	0.69	0.69	6	0.69	0.69
DS_1100	5	4	0.98	0.98	5	0.65	0.65	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1101	8	4	0.97	0.97	10	0.65	0.65	5	1.00	1.00	6	0.99	0.99	6	0.99	0.99	6	0.99	0.99
DS_1110	5	5	0.77	0.94	5	0.62	0.58	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00	5	1.00	1.00
DS_1111	8	6	0.67	0.96	10	0.33	0.33	5	1.00	1.00	5	0.99	0.99	6	0.99	0.99	6	0.99	0.99
DS_1200	5	5	0.85	0.99	6	0.52	0.52	5	1.00	1.00	17	0.92	1.00	18	0.92	1.00	11	0.95	0.99
DS_1201	8	6	0.72	0.99	10	0.52	0.52	5	1.00	1.00	19	0.92	1.00	21	0.92	1.00	10	0.39	0.41
DS_1210	5	5	0.78	0.95	8	0.41	0.41	5	0.86	0.99	16	0.25	0.27	17	0.25	0.27	13	0.25	0.26
DS_1211	8	5	0.78	0.95	8	0.25	0.25	5	0.86	0.99	17	0.24	0.27	18	0.24	0.27	14	0.25	0.26

Table D.9 Comparison of *k*-means, single-linkage, DBSCAN and NOM in terms of time (in seconds) for group 3 data sets

		K-means		**Single-linkage**		**DBSCAN**		**NC**		**Outlier Detection**		**NOM*****	
Data set	**#TC**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**	**#C**	**Time**
DS_0000	5	4	0.58	5	6.47	5	1.16	5	4729.13	5	2250.91	5	6986.76
DS_0001	8	4	0.45	8	6.13	8	1.14	8	3953.68	8	1491.87	8	5452.12
DS_0010	5	4	0.45	5	3.25	5	1.24	5	3834.32	5	637.01	5	4475.31
DS_0011	8	4	0.46	8	3.34	5	1.25	7	18660.97	8	624.74	8	19289.81
DS_0100	5	4	0.31	5	3.02	5	0.76	4	1917.27	5	578.95	5	2499.59
DS_0101	8	4	0.30	8	2.94	5	0.76	4	1931.07	6	584.58	6	2519.04
DS_0110	5	4	0.35	5	3.00	5	0.77	4	1903.47	4	587.85	4	2494.80
DS_0111	8	4	0.32	8	2.94	5	0.80	7	1931.58	7	583.35	7	2518.53
DS_0200	5	4	0.46	5	1.45	5	1.18	17	9442.06	19	1942.84	13	11405.16
DS_0201	8	4	0.46	8	1.46	5	1.18	18	24099.10	21	257.11	10	24388.43
DS_0210	5	4	0.46	5	1.45	5	1.16	16	3840.04	18	409.37	11	4270.61
DS_0211	8	4	0.47	8	1.46	5	1.21	18	3876.84	21	411.50	11	4320.92
DS_1000	5	5	0.44	5	3.28	5	1.12	5	3828.47	5	616.40	5	4449.02
DS_1001	8	5	0.47	8	3.21	5	1.14	6	3863.46	8	619.00	8	4486.67
DS_1010	5	5	0.48	4	3.34	4	1.12	4	3825.85	4	742.64	4	4572.34
DS_1011	8	6	0.52	7	3.26	4	1.18	5	3845.62	6	745.81	6	4595.30
DS_1100	5	4	0.37	5	1.87	5	0.77	5	1907.03	5	278.45	5	2187.67
DS_1101	8	4	0.31	10	1.78	5	0.78	6	4522.14	6	838.78	6	5363.20
DS_1110	5	5	0.33	5	1.79	5	0.78	5	4754.65	5	849.43	5	5606.34
DS_1111	8	6	0.40	10	1.80	5	0.77	5	5077.17	6	633.61	6	5713.03
DS_1200	5	5	0.46	6	1.44	5	1.27	17	6542.76	18	253.27	11	6818.25
DS_1201	8	6	0.46	10	1.41	5	1.23	19	3901.61	21	253.73	10	4191.51
DS_1210	5	5	0.56	8	1.39	5	1.17	16	3837.40	17	2373.50	13	6235.58
DS_1211	8	5	0.49	8	1.40	5	1.18	17	3876.83	18	5986.69	14	9889.20

*** Times for NOM include NC and Outlier Detection times.

APPENDIX E

EXTERNAL SHAPE GENERATION ALGORITHMS AND THE EXPERIMENTAL RESULTS

We present the pseudocodes of the external shape generation algorithms, namely DTC and IS algorithm in Section E.1. In Section E.2, we provide the experimental results of the DTC and IS algorithms in Tables E.1 through E.5.

E.1. The DTC and IS Algorithms

DTC Algorithm

Step 1. DT construction.

Step 2. Identification of the edges on the initial boundary of each cluster.

```
For $\forall s \in$ DT
    If an edge of triangle $s$ belongs to a single triangle only
        Add this edge to the set of boundary edges of the associated cluster.
    End if
    If triangle $s$ includes points from different clusters
        Add the edge between the two adjacent points that are in the same
        cluster to the set of boundary edges of the associated cluster.
    End if
End for
```

```
Step 3. Extraction of the non-convex parts of the boundaries.
Repeat
    Step 3.1. Edge selection
    Select the longest edge (e') on the boundary that has not been evaluated yet.

    Step 3.2. 2-degree check
    If both end points of e' have a degree higher than two

        Step 3.3. Edge deletion
        If both of the inner edges on the same triangle as e' are shorter than e'
            Delete e' from the set of boundary edges of the associated
            cluster, and add the two inner edges to the set.
        End if
    End if
Until all the edges in the sets of boundary edges have been evaluated and no edge is
deleted
```

IS Algorithm

```
Step 0. Initialization of elongation thresholds, sets A and TA.

Step 1. Construction of the set of artificial points A for D.

Step 2. Construction of DT for $D \cup A$.

Step 3. Identification of the edges on the initial boundary of each cluster.
For $\forall s \in$ DT
    If triangle s includes artificial points
        Add the original points to the set of boundary points of the associated
        cluster.
    End if
    If triangle s includes points from different clusters
```

```
            Add the original adjacent points that are in the same cluster to the set
            of boundary points of the associated cluster.
        End if
End for

Step 4. Identification of the non-convex parts of the boundary.
Repeat
        For $\forall s \in$ DT
            If simplex $s$, that includes original points only, includes ($d$-1)
            boundary points
                Simplex $s$ is a facet and calculate its elongation thresholds,
                $em_1$ and $em_2$.

                Step 4.1. Elongation check
                If $em_1 > threshold_1$ or $em_2 > threshold_2$
                    Add simplex $s$ to the set of elongated simplices.

                    Step 4.2. Insertion of an artificial point
                    If simplex $s$ is not a member of the set of blocked
                    simplices
                        Insert an artificial point at the center of gravity
                        of simplex $s$. Add the artificial point to the
                        temporary set of artificial points, TA.
                    End if
                End if
        End for

        Step 5. Construction of DT for D $\cup$ A $\cup$ TA.

        Step 6. Feasibility of the temporary artificial points in set TA.
        For $\forall i \in$ D
```

```
            If all the simplices that include point $i$ have no other original points
            from the same cluster with point $i$
                Find the nearest temporary artificial point to point $i$, say point
                $j$, and delete point $j$ from set TA. Add the associated simplex
                of point $j$ to the set of blocked simplices.
                Construct DT for $D \cup A \cup TA$.
            End if
        End for

        For $\forall i \in TA$
            If all the simplices that include point $i$ have no other artificial points
            except point $i$
                Delete point $i$ from set TA and add the associated simplex of
                point $i$ to the set of blocked simplices.
                Construct DT for $D \cup A \cup TA$.
            End if
        End for

        **Step 7.** Update set of artificial points.
        If $TA \neq \varnothing$
            Update set A, $A = A \cup TA$.
        End if
    Until $TA = \varnothing$
```

E.2. The Experimental Results of the DTC and IS Algorithms

Table E.1 DTC results of 2-dimensional group 1 data sets

Data set	**# of points in the data set**	**# of points found on the boundary**	**Time (sec.)**	**% reduction**
data_60	60	59	0.91	1.67
data_66	66	65	0.26	1.52
data-c-cv-nu-n_v2	73	54	0.31	26.03
data-c-cv-nu-n	76	57	0.34	25
data-c-cv-u-n	81	51	0.12	37.04
data-uc-cv-nu-n	127	106	1.12	16.54
data-oo_v2	140	36	0.78	74.29
data-oo	144	40	0.17	72.22
data-uc-cc-nu-n_v2	188	146	1.66	22.34
data-uc-cc-nu-n	191	149	1.67	21.99
data-c-cc-nu-n2_v2	192	170	1.77	11.46
data-c-cc-nu-n2	195	173	0.69	11.28
dataX_v2	200	110	2.08	45
dataX	202	112	2	44.55
data-c-cc-nu-n_v2	285	183	3.76	35.79
train2	287	206	1.77	28.22
data-c-cc-nu-n	289	187	1.22	35.29
train1_v1	306	169	1.48	44.77
train1	307	170	1.46	44.63
train3_v1	361	326	3.24	9.7
train3	397	338	2.02	14.86
data_circle	700	316	8.42	54.86
data_mix_uniform_normal	1000	604	10.67	39.6
data_circle_10_1_10_10	1100	427	19.86	61.18
data_circle_1_10_5_10	1100	489	22.19	55.55
data_circle_2_10_5_13	1200	531	15.16	55.75
data_circle_2_10_4_12	1200	657	18.78	45.25
data_circle_2_10_2_12	1200	544	22.55	54.67
data_circle_2_10_3_12	1200	600	26.34	50
data_circle_2_10_3_12	1200	600	26.35	50
data_circle_2_10_6_12	1200	683	33.54	43.08
data_circle_3_10_8_12	1300	626	26.46	51.85
data_circle_5_10_8_12	1500	522	17.15	65.2
data_circle1	1890	870	49.32	53.97
data_circle2	1890	680	50.82	64.02
data_circle_1_20_1_11	2100	567	13.93	73
data_circle_20_1_5_10	2100	786	35.11	62.57
data_circle_1_20_1_15	2100	697	37.84	66.81
data_circle3	2100	868	56.88	58.67
data_circle_1_20_1_13	2100	871	67.39	58.52
data_circle_1_20_1_19	2100	768	68.01	63.43
data_circle_1_20_1_17	2100	1002	87.07	52.29

Table E.2 IS results of 2-dimensional group 1 data sets

Data set	**# of points in the data set**	**# of points found on the boundary**	**Time (sec.)**	**% reduction**
data_60	60	55	0.98	8.33
data_66	66	64	0.45	3.03
data-c-cv-nu-n_v2	73	52	1.97	28.77
data-c-cv-nu-n	76	53	2.03	30.26
data-uc-cv-nu-n	126	78	7.30	38.10
data-oo_v2	140	24	1.30	82.86
data-uc-cc-nu-n_v2	188	134	28.84	28.72
data-uc-cc-nu-n	191	94	5.53	50.79
data-c-cc-nu-n2_v2	192	130	9.61	32.29
dataX_v2	200	58	0.38	71.00
dataX	202	60	0.38	70.30
data-c-cc-nu-n_v2	285	135	51.29	52.63
train2	285	199	86.85	30.18
train1_v1	302	143	44.89	52.65
train1	303	146	40.28	51.82
train3_v1	360	249	181.02	30.83
data_circle	700	239	321.55	65.86
data_mix_uniform_normal	1000	629	1685.66	37.10
data_circle_1_10_5_10	1100	355	2047.91	67.73
data_circle_2_10_2_12	1200	440	2342.84	63.33
data_circle_2_10_2_12	1200	440	2342.84	63.33
data_circle1	1890	672	9611.18	64.44
data_circle2	1890	466	4991.17	75.34
data_circle3	2098	710	14057.06	66.16
data_circle_20_1_5_10	2100	889	5957.65	57.67
data_circle_1_20_1_11	2100	656	3036.09	68.76
data_circle_1_20_1_13	2100	598	5262.716	71.52
data_circle_1_20_1_15	2100	519	1949.072	75.29

Table E.3 IS results of higher dimensional group 1 data sets

Data set	# of points in the data set	# of points found on the boundary	Time (sec.)	% reduction
iris	143	136	10.64	4.90
3d_dataset3	325	191	2.57	41.23
3d_dataset4	1523	703	12.59	53.84

Table E.4 IS results of group 2 data sets

Data set	# of points in the data set	# of points found on the boundary	Time (sec.)	% reduction
D_0000	2752	1996	1987.38	27.47
D_0001	2755	2001	5281.82	27.37
D_0010	2752	1995	4620.58	27.51
D_0011	2755	1990	4686.27	27.77
D_0100	1962	1432	3343.64	27.01
D_0101	1965	1449	4994.33	26.26
D_0110	1913	1409	2692.96	26.35
D_0111	1916	1412	2964.73	26.30
D_0200	2780	2199	11299.98	20.90
D_0201	2783	2276	13266.75	18.22
D_0210	2780	2189	14302.85	21.26
D_0211	2783	2191	16364.14	21.27
D_1000	2752	2328	4344.79	15.41
D_1001	2755	2326	4017.79	15.57
D_1010	2752	2540	2826.86	7.70
D_1011	2755	2542	2821.20	7.73
D_1100	1912	1438	1550.63	24.79
D_1101	1915	1432	2788.82	25.22
D_1110	1932	1544	5376.50	20.08
D_1111	1935	1483	1608.08	23.36
D_1200	2780	2343	9339.93	15.72
D_1201	2783	2349	8844.49	15.59
D_1210	2780	2251	48055.49	19.03
D_1211	2783	2254	55914.38	19.01

Table E.5 IS results of group 3 data sets

Data set	**# of points in the data set**	**# of points found on the boundary**	**Time (sec.)**	**% reduction**
DS_0000	894	739	87.28	17.34
DS_0001	903	732	80.07	18.94
DS_0010	894	805	474.61	9.96
DS_0011	903	809	483.82	10.41
DS_0100	709	586	73.46	17.35
DS_0101	712	587	51.17	17.56
DS_0110	708	573	50.47	19.07
DS_0111	711	580	50.74	18.42
DS_0200	894	766	257.71	14.32
DS_0201	897	775	365.76	13.6
DS_0210	894	761	216.58	14.88
DS_0211	897	764	253.42	14.83
DS_1000	894	741	83.68	17.11
DS_1001	897	746	154.73	16.83
DS_1010	894	743	183.59	16.89
DS_1011	897	743	219.74	17.17
DS_1100	708	582	50.72	17.8
DS_1101	711	581	50.89	18.28
DS_1110	715	604	51.56	15.52
DS_1111	718	610	98.94	15.04
DS_1200	894	752	41.72	15.88
DS_1201	897	772	220.52	13.94
DS_1210	894	751	502.43	16
DS_1211	897	752	465.22	16.16

APPENDIX F

EXPERIMENTAL RESULTS FOR THE ACO-C ALGORITHM

In this part, we present the experimental results of the ACO-C algorithm. Column headings in Tables F.1 through F.12 are explained as follows.

#P	: number of points in the original data set
#P-ADR	: number of points after data set reduction
#TC	: number of target clusters
#C after NC	: number of clusters after the NC algorithm
#SN	: number of nondominated solutions in SN
#ITA	: iteration number in which the target clustering is found where maximum number of iterations is set twice #P-ADR.

Table F.1 Experimental results for the factorial setting, S_000

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	3	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	323.06
data_66	66	65	4	4	4	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	414.20
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	318.01
data-c-cv-nu-n	76	57	6	4	3	0.54	1.00	0.54	1.00	0.73	1.00	61	1.11	397.80
data-uc-cv-nu-n	127	106	6	5	5	0.55	1.00	0.55	1.00	0.65	1.00	1	0.01	2298.17
data-uc-cc-nu-n_v2	188	146	3	3	3	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	5545.08
data-uc-cc-nu-n	191	149	6	4	5	0.48	0.98	0.48	0.99	0.58	0.99	NA*	NA*	5682.89
data-c-cc-nu-n2_v2	192	170	3	4	4	0.64	1.00	0.64	1.00	0.71	1.00	2	0.01	8694.53
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	7425.50
data-c-cc-nu-n_v2	285	183	3	3	3	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	10289.99
train2	287	206	4	6	5	0.68	1.00	0.68	1.00	0.83	1.00	265	1.30	16319.65
train1_v1	306	169	5	8	5	0.49	1.00	0.49	1.00	0.71	1.00	21	0.13	10463.90
train1	307	170	6	9	5	0.40	1.00	0.40	1.00	0.57	1.00	68	0.42	8484.69
train3_v1	361	326	5	6	3	0.63	1.00	0.63	1.00	0.78	1.00	5	0.02	45135.16
data_circle	700	316	2	14	1	0.68	0.68	1.00	1.00	0.80	0.80	NA*	NA*	26713.36

NA* : not available

Table F.2 Experimental results for the factorial setting, S_001

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	90.15
data_66	66	65	4	4	3	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	123.48
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	116.99
data-c-cv-nu-n	76	57	6	4	2	0.54	0.57	0.54	0.57	0.73	0.75	NA*	NA*	107.71
data-uc-cv-nu-n	127	106	6	5	4	0.55	1.00	0.55	1.00	0.65	1.00	1	0.01	656.29
data-uc-cc-nu-n_v2	188	146	3	3	3	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	1186.21
data-uc-cc-nu-n	191	149	6	4	3	0.48	0.98	0.48	0.98	0.58	0.99	NA*	NA*	1230.32
data-c-cc-nu-n2_v2	192	170	3	4	3	0.64	1.00	0.64	1.00	0.71	1.00	2	0.01	2356.01
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	5168.22
data-c-cc-nu-n_v2	285	183	3	3	2	0.54	0.57	0.54	0.57	0.73	0.75	NA*	NA*	2604.86
train2	287	206	4	6	5	0.68	1.00	0.68	1.00	0.83	1.00	48	0.24	4275.32
train1_v1	306	169	5	8	5	0.40	1.00	0.40	1.00	0.58	1.00	5	0.03	2387.17
train1	307	170	6	9	4	0.49	1.00	0.49	1.00	0.71	1.00	54	0.33	2523.96
train3_v1	361	326	5	6	3	0.48	1.00	0.48	1.00	0.60	1.00	27	0.09	11647.17
data_circle	700	316	2	14	2	0.70	0.70	1.00	1.00	0.81	0.81	NA*	NA*	17991.73

NA* : not available

Table F.3 Experimental results for the factorial setting, S_010

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	88.94
data_66	66	65	4	4	3	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	112.77
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	82.10
data-c-cv-nu-n	76	57	6	4	2	0.54	0.95	0.54	0.95	0.73	0.98	NA*	NA*	76.91
data-uc-cv-nu-n	127	106	6	5	3	0.55	1.00	0.55	1.00	0.65	1.00	1	0.01	519.55
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	947.03
data-uc-cc-nu-n	191	149	6	4	2	0.48	0.98	0.48	0.99	0.58	0.99	NA*	NA*	1067.49
data-c-cc-nu-n2_v2	192	170	3	4	3	0.64	1.00	0.64	1.00	0.71	1.00	4	0.02	1723.34
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	4515.18
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	1887.30
train2	287	206	4	6	2	0.68	1.00	0.68	1.00	0.83	1.00	2	0.01	2650.32
train1_v1	306	169	5	8	4	0.73	1.00	0.73	1.00	0.90	1.00	5	0.03	1446.23
train1	307	170	6	9	3	0.72	1.00	0.72	1.00	0.89	1.00	4	0.02	1519.38
train3_v1	361	326	5	6	5	0.66	1.00	0.66	1.00	0.80	1.00	60	0.20	9030.18
data_circle	700	316	2	14	1	0.67	0.67	1.00	1.00	0.69	0.79	NA*	NA*	10284.58

NA* : not available

Table F.4 Experimental results for the factorial setting, S_011

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	169.93
data_66	66	65	4	4	4	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	223.54
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	202.92
data-c-cv-nu-n	76	57	6	4	2	0.54	0.57	0.54	0.57	0.73	0.75	NA*	NA*	207.70
data-uc-cv-nu-n	127	106	6	5	5	0.55	1.00	0.55	1.00	0.65	1.00	1	0.01	1086.28
data-uc-cc-nu-n_v2	188	146	3	3	3	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	2532.77
data-uc-cc-nu-n	191	149	6	4	4	0.48	0.98	0.48	0.98	0.58	0.99	NA*	NA*	2407.90
data-c-cc-nu-n2_v2	192	170	3	4	3	0.64	1.00	0.64	1.00	0.71	1.00	3	0.02	4107.00
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	8907.86
data-c-cc-nu-n_v2	285	183	3	3	3	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	4583.99
train2	287	206	4	6	4	0.77	1.00	0.77	1.00	0.89	1.00	58	0.28	6861.84
train1_v1	306	169	5	8	5	0.49	1.00	0.49	1.00	0.71	1.00	46	0.29	4809.80
train1	307	170	6	9	4	0.73	1.00	0.73	1.00	0.90	1.00	55	0.34	4141.55
train3_v1	361	326	5	6	3	0.66	1.00	0.66	1.00	0.80	1.00	9	0.03	21712.58
data_circle	700	316	2	14	2	0.57	0.68	1.00	1.00	0.73	0.80	NA*	NA*	39350.91

NA* : not available

Table F.5 Experimental results for the factorial setting, S_100

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.81	1.00	0.81	1.00	0.91	1.00	1	0.02	187.78
data_66	66	65	4	4	3	0.67	1.00	0.67	1.00	0.83	1.00	1	0.02	272.99
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	199.95
data-c-cv-nu-n	76	57	6	4	2	0.54	0.97	0.54	0.97	0.73	0.99	NA*	NA*	254.39
data-uc-cv-nu-n	127	106	6	5	3	0.55	0.97	0.55	0.97	0.65	0.99	NA*	NA*	1605.44
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	3469.06
data-uc-cc-nu-n	191	149	6	4	3	0.48	0.98	0.48	0.98	0.58	0.99	NA*	NA*	4548.81
data-c-cc-nu-n2_v2	192	170	3	4	3	0.78	1.00	0.78	1.00	0.85	1.00	4	0.02	5469.30
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	4571.74
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	7003.74
train2	287	206	4	6	3	0.50	1.00	0.50	1.00	0.63	1.00	20	0.10	11954.62
train1_v1	306	169	5	8	4	0.40	1.00	0.40	1.00	0.58	1.00	85	0.53	5718.17
train1	307	170	6	9	4	0.49	1.00	0.49	1.00	0.70	1.00	31	0.19	6250.71
train3_v1	361	326	5	6	6	0.49	1.00	0.49	1.00	0.62	1.00	10	0.03	42888.72
data_circle	700	316	2	14	2	0.52	1.00	1.00	1.00	0.70	1.00	17	0.06	24927.89

NA* : not available

Table F.6 Experimental results for the factorial setting, S_101

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.81	1.00	0.81	1.00	0.91	1.00	1	0.02	187.78
data_66	66	65	4	4	3	0.67	1.00	0.67	1.00	0.83	1.00	1	0.02	272.99
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	199.95
data-c-cv-nu-n	76	57	6	4	2	0.90	0.97	0.90	0.97	0.97	0.99	NA*	NA*	254.39
data-uc-cv-nu-n	127	106	6	5	3	0.55	0.97	0.55	0.97	0.65	0.99	NA*	NA*	1605.44
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	3469.06
data-uc-cc-nu-n	191	149	6	4	3	0.48	0.49	0.48	0.49	0.58	0.60	NA*	NA*	4548.81
data-c-cc-nu-n2_v2	192	170	3	4	3	0.78	1.00	0.78	1.00	0.85	1.00	4	0.02	5469.30
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	4571.74
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	7003.74
train2	287	206	4	6	3	0.77	1.00	0.77	1.00	0.89	1.00	20	0.10	11954.62
train1_v1	306	169	5	8	4	0.40	1.00	0.40	1.00	0.58	1.00	85	0.53	5718.17
train1	307	170	6	9	4	0.40	1.00	0.40	1.00	0.57	1.00	31	0.19	6250.71
train3_v1	361	326	5	6	6	0.49	1.00	0.49	1.00	0.62	1.00	10	0.03	42888.72
data_circle	700	316	2	14	2	0.78	0.78	1.00	1.00	0.86	0.86	NA*	NA*	24927.89

NA* : not available

Table F.7 Experimental results for the factorial setting, S_110

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.81	1.00	0.81	1.00	0.91	1.00	1	0.02	392.01
data_66	66	65	4	4	3	0.67	1.00	0.67	1.00	0.83	1.00	1	0.02	576.61
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	408.02
data-c-cv-nu-n	76	57	6	4	2	0.54	0.97	0.54	0.97	0.73	0.99	NA*	NA*	508.57
data-uc-cv-nu-n	127	106	6	5	3	0.55	0.97	0.55	0.97	0.65	0.99	NA*	NA*	3272.53
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	6600.36
data-uc-cc-nu-n	191	149	6	4	3	0.48	0.98	0.48	0.98	0.58	0.99	NA*	NA*	9843.91
data-c-cc-nu-n2_v2	192	170	3	4	3	0.78	1.00	0.78	1.00	0.85	1.00	3	0.02	11459.56
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	9051.22
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	14158.51
train2	287	206	4	6	3	0.50	1.00	0.50	1.00	0.63	1.00	2	0.01	27192.27
train1_v1	306	169	5	8	4	0.49	1.00	0.49	1.00	0.71	1.00	10	0.06	12718.37
train1	307	170	6	9	4	0.40	1.00	0.40	1.00	0.57	1.00	35	0.21	12991.10
train3_v1	361	326	5	6	6	0.49	1.00	0.49	1.00	0.62	1.00	7	0.02	83777.01
data_circle	700	316	2	14	2	0.51	1.00	1.00	1.00	0.70	1.00	85	0.28	69943.25

NA* : not available

Table F.8 Experimental results for the factorial setting, S_111

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.81	1.00	0.81	1.00	0.91	1.00	1	0.02	179.62
data_66	66	65	4	4	3	0.67	1.00	0.67	1.00	0.83	1.00	1	0.02	224.70
data-c-cv-nu-n_v2	73	54	3	3	2	0.61	1.00	0.61	1.00	0.76	1.00	1	0.02	160.86
data-c-cv-nu-n	76	57	6	4	2	0.54	0.97	0.54	0.97	0.73	0.99	NA*	NA*	187.86
data-uc-cv-nu-n	127	106	6	5	3	0.55	0.97	0.55	0.97	0.65	0.99	NA*	NA*	975.42
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	2126.75
data-uc-cc-nu-n	191	149	6	4	4	0.48	1.00	0.48	1.00	0.58	1.00	52	0.36	2471.31
data-c-cc-nu-n2_v2	192	170	3	4	3	0.78	1.00	0.78	1.00	0.85	1.00	3	0.02	3305.57
dataX	202	112	4	4	2	0.33	1.00	1.00	1.00	0.68	1.00	1	0.00	8974.09
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	4252.89
train2	287	206	4	6	3	0.50	1.00	0.50	1.00	0.63	1.00	24	0.12	5718.25
train1_v1	306	169	5	8	4	0.40	1.00	0.40	1.00	0.58	1.00	5	0.03	3223.32
train1	307	170	6	9	4	0.49	1.00	0.49	1.00	0.70	1.00	16	0.10	3472.29
train3_v1	361	326	5	6	4	0.66	1.00	0.66	1.00	0.80	1.00	3	0.01	15906.19
data_circle	700	316	2	14	3	0.56	0.81	1.00	1.00	0.68	0.88	NA*	NA*	17987.49

NA* : not available

Table F.9 Experimental results with CERN for group 1 data sets

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	116.66
data_66	66	65	4	4	4	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	403.09
data-c-cv-nu-n_v2	73	54	3	3	2	0.60	1.00	0.60	1.00	0.75	1.00	1	0.02	877.75
data-c-cv-nu-n	76	57	6	4	4	0.34	1.00	0.34	1.00	0.36	1.00	61	1.07	1182.30
data-c-cv-u-n	81	51	5	5	4	0.45	1.00	0.45	1.00	0.48	1.00	1	0.02	1564.88
data-uc-cv-nu-n	127	106	6	5	5	0.55	1.00	0.55	1.00	0.65	1.00	1	0.01	1009.54
data-oo_v2	140	36	2	2	2	0.94	1.00	1.00	1.00	0.94	1.00	1	0.03	1343.72
data-oo	144	40	6	6	4	0.85	1.00	0.85	1.00	0.86	1.00	1	0.03	9219.90
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	10443.00
data-uc-cc-nu-n	191	149	6	4	5	0.48	0.98	0.48	0.99	0.58	0.99	NA*	NA*	28635.04
data-c-cc-nu-n2_v2	192	170	3	4	4	0.64	1.00	0.64	1.00	0.71	1.00	5	0.03	37629.45
data-c-cc-nu-n2	195	173	6	7	3	0.96	0.97	0.96	1.00	0.98	0.99	NA*	NA*	28905.86
dataX_v2	200	110	2	2	2	0.98	1.00	0.99	1.00	0.99	1.00	1	0.01	47231.89
dataX	202	112	4	4	3	0.96	1.00	0.96	1.00	0.98	1.00	1	0.01	58055.25
data-c-cc-nu-n_v2	285	183	3	3	3	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	60914.35
train2	287	206	4	6	4	0.77	1.00	0.77	1.00	0.89	1.00	45	0.22	52196.26
data-c-cc-nu-n	289	187	7	7	4	0.76	1.00	0.76	1.00	0.86	1.00	1	0.01	28679.22

NA* : not available

Table F.9 Experimental results with CERN for group 1 data sets (cont'd)

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
train1_v1	306	169	5	8	4	0.49	1.00	0.49	1.00	0.70	1.00	65	0.38	41655.94
train1	307	170	6	9	3	0.70	1.00	0.70	1.00	0.88	1.00	180	1.06	628316.00
train3_v1	361	326	5	6	5	0.52	1.00	0.52	1.00	0.65	1.00	50	0.15	63438.99
train3	397	338	36	17	3	0.59	0.89	0.86	0.90	0.87	0.96	NA*	NA*	376360.54
data_circle	700	316	2	14	2	0.68	0.80	1.00	1.00	0.79	0.88	NA*	NA*	415833.80
data_mix_uniform_normal	1000	604	2	38	5	0.58	0.62	1.00	1.00	0.84	0.90	NA*	NA*	413802.45
data_circle_1_10_5_10	1100	489	2	29	2	0.76	0.88	1.00	1.00	0.81	0.90	NA*	NA*	94285.89
data_circle_10_1_10_10	1100	427	2	22	2	0.74	0.83	1.00	1.00	0.77	0.89	NA*	NA*	10964.87
data_circle_2_10_2_12	1200	544	2	22	1	0.74	0.87	1.00	1.00	0.86	0.92	NA*	NA*	41374.86
data_circle_2_10_3_12	1200	600	2	30	1	0.92	0.92	1.00	1.00	0.97	0.97	NA*	NA*	11825.11
data_circle_2_10_4_12	1200	657	2	37	2	0.78	0.86	1.00	1.00	0.83	0.89	NA*	NA*	396369.99
data_circle_2_10_5_13	1200	531	2	30	2	0.73	0.89	1.00	1.00	0.84	0.95	NA*	NA*	634347.54
data_circle_2_10_6_12	1200	683	2	39	2	0.76	0.89	1.00	1.00	0.85	0.93	NA*	NA*	670313.00
data_circle_3_10_8_12	1300	626	2	26	3	0.51	0.91	0.79	0.98	0.60	0.96	NA*	NA*	511387.52
data_circle_5_10_8_12	1500	522	2	29	3	0.68	0.81	0.82	0.91	0.75	0.89	NA*	NA*	488377.63
data_circle1	1890	870	2	35	2	0.90	0.95	1.00	1.00	0.94	0.97	NA*	NA*	686963.26

NA* : not available

Table F.9 Experimental results with CERN for group 1 data sets (cont'd)

Data set	**#P**	**#P-ADR**	**# TC**	**#C after NC**	**# SN**	**min JI**	**max JI**	**min QJI**	**max QJI**	**min RI**	**max RI**	**# ITA**	**# ITA / #P-ADR**	**Time (sec.)**
data_circle2	1890	680	2	35	2	0.92	0.95	1.00	1.00	0.95	0.99	NA*	NA*	786454.59
data_circle_1_20_1_11	2100	567	2	49	3	0.82	0.93	1.00	1.00	0.83	0.94	NA*	NA*	678240.83
data_circle_1_20_1_13	2100	871	2	41	2	0.87	0.95	1.00	1.00	0.97	0.99	NA*	NA*	635587.57
data_circle_1_20_1_15	2100	697	2	47	3	0.73	0.92	1.00	1.00	0.91	0.95	NA*	NA*	774884.04
data_circle_1_20_1_17	2100	1002	2	57	3	0.75	0.94	1.00	1.00	0.88	0.95	NA*	NA*	946392.20
data_circle_1_20_1_19	2100	768	2	38	4	0.73	0.83	0.78	0.96	0.84	0.92	NA*	NA*	145991.85
data_circle_20_1_5_10	2100	786	2	53	5	0.61	0.81	1.00	1.00	0.75	0.87	NA*	NA*	305031.10
data_circle3	2100	868	2	41	2	0.76	0.87	1.00	1.00	0.88	0.95	NA*	NA*	854649.92
iris	143	136	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.01	10430.63
3d_dataset3	325	191	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.15	343233.30
3d_dataset4	1523	703	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.01	438031.45

NA* : not available

Table F.10 Experimental results with WCERN for group 1 data sets

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_60	60	59	3	3	2	0.52	1.00	0.52	1.00	0.64	1.00	1	0.02	169.70
data_66	66	65	4	4	3	0.43	1.00	0.43	1.00	0.55	1.00	1	0.02	238.96
data-c-cv-nu-n_v2	73	54	3	3	2	0.60	1.00	0.60	1.00	0.75	1.00	1	0.02	184.69
data-c-cv-nu-n	76	57	6	4	2	0.57	0.97	0.57	0.97	0.75	0.99	NA*	NA*	247.47
data-c-cv-u-n	81	51	5	5	4	0.45	1.00	0.45	1.00	0.48	1.00	1	0.02	134.58
data-uc-cv-nu-n	127	106	6	5	4	0.55	0.97	0.55	0.97	0.65	0.99	NA*	NA*	1563.85
data-oo_v2	140	36	2	2	2	0.94	1.00	1.00	1.00	0.94	1.00	1	0.03	23.18
data-oo	144	40	6	6	4	0.85	1.00	0.85	1.00	0.86	1.00	1	0.03	62.36
data-uc-cc-nu-n_v2	188	146	3	3	2	0.50	1.00	0.50	1.00	0.60	1.00	1	0.01	4300.71
data-uc-cc-nu-n	191	149	6	4	5	0.48	1.00	0.48	1.00	0.58	1.00	201	1.35	5028.95
data-c-cc-nu-n2_v2	192	170	3	4	3	0.78	1.00	0.78	1.00	0.85	1.00	1	0.01	7148.07
data-c-cc-nu-n2	195	173	6	7	4	0.76	0.98	0.76	1.00	0.84	0.99	50	0.29	9735.84
dataX_v2	200	110	2	2	2	0.98	1.00	0.99	1.00	0.99	1.00	1	0.01	1711.40
dataX	202	112	4	4	3	0.49	1.00	0.49	1.00	0.50	1.00	1	0.01	1647.37
data-c-cc-nu-n_v2	285	183	3	3	2	0.78	1.00	0.78	1.00	0.87	1.00	1	0.01	7869.58
train2	287	206	4	6	3	0.77	1.00	0.77	1.00	0.89	1.00	25	0.12	11256.55
data-c-cc-nu-n	289	187	7	7	3	0.46	1.00	0.46	1.00	0.47	1.00	1	0.01	10102.57

NA* : not available

Table F.10 Experimental results with WCERN for group 1 data sets (cont'd)

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
train1_v1	306	169	5	8	4	0.49	1.00	0.49	1.00	0.70	1.00	80	0.47	8098.40
train1	307	170	6	9	3	0.70	1.00	0.70	1.00	0.88	1.00	31	0.18	8504.68
train3_v1	361	326	5	6	5	0.50	1.00	0.50	1.00	0.62	1.00	10	0.03	51485.39
train3	397	338	36	17	3	0.55	0.85	0.66	0.85	0.85	0.95	NA*	NA*	73438.99
data_circle	700	316	2	14	2	0.46	1.00	1.00	1.00	0.67	1.00	17	0.05	38442.84
data_mix_uniform_normal	1000	604	2	38	5	0.58	0.92	1.00	1.00	0.66	0.97	NA*	NA*	404907.01
data_circle_1_10_5_10	1100	489	2	29	4	0.74	0.94	1.00	1.00	0.82	0.98	NA*	NA*	157090.19
data_circle_10_1_10_10	1100	427	2	22	6	0.85	0.81	1.00	1.00	0.88	0.90	NA*	NA*	134393.33
data_circle_2_10_2_12	1200	544	2	22	4	0.94	0.97	1.00	1.00	0.97	0.98	NA*	NA*	361383.09
data_circle_2_10_3_12	1200	600	2	30	2	0.97	1.00	1.00	1.00	0.98	1.00	620	1.03	338478.52
data_circle_2_10_4_12	1200	657	2	37	3	0.84	0.97	1.00	1.00	0.92	0.97	NA*	NA*	481845.00
data_circle_2_10_5_13	1200	531	2	30	3	0.88	0.95	1.00	1.00	0.95	0.98	NA*	NA*	251748.71
data_circle_2_10_6_12	1200	683	2	39	3	0.86	0.93	1.00	1.00	0.96	0.97	NA*	NA*	511722.92
data_circle_3_10_8_12	1300	626	2	26	3	0.61	0.72	0.61	0.73	0.78	0.88	NA*	NA*	281968.66
data_circle_5_10_8_12	1500	522	2	29	3	0.58	0.79	0.65	0.79	0.70	0.83	NA*	NA*	196187.05
data_circle1	1890	870	2	35	4	0.94	1.00	1.00	1.00	0.97	1.00	1287	1.48	1057098.04

NA*	: not available

Table F.10 Experimental results with WCERN for group 1 data sets (cont'd)

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
data_circle2	1890	680	2	35	4	0.78	0.96	1.00	1.00	0.90	0.98	NA*	NA*	592462.24
data_circle_1_20_1_11	2100	567	2	49	4	0.98	1.00	1.00	1.00	1.00	1.00	417	0.74	338156.83
data_circle_1_20_1_13	2100	871	2	41	3	0.81	1.00	1.00	1.00	0.85	1.00	645	0.74	1201467.39
data_circle_1_20_1_15	2100	697	2	47	2	0.97	1.00	1.00	1.00	0.98	1.00	417	0.60	719592.45
data_circle_1_20_1_17	2100	1002	2	57	2	0.81	0.98	1.00	1.00	0.87	0.99	NA*	NA*	897515.29
data_circle_1_20_1_19	2100	768	2	38	1	0.69	0.69	0.69	0.69	0.75	0.75	NA*	NA*	562401.06
data_circle_20_1_5_10	2100	786	2	53	4	0.73	0.85	1.00	1.00	0.80	0.90	NA*	NA*	634145.25
data_circle3	2100	868	2	41	1	1.00	1.00	1.00	1.00	1.00	1.00	1405	1.62	1081741.34
iris	143	136	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.01	197450.63
3d_dataset3	325	191	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.01	55341.45
3d_dataset4	1523	703	2	2	1	1.00	1.00	1.00	1.00	1.00	1.00	1	0.00	902841.56

NA* : not available

Table F.11 Experimental results with CERN for group 3 data sets

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
DS_0000	894	739	5	5	2	0.58	1.00	0.58	1.00	0.59	1.00	1	0.00	440597.04
DS_0001	903	732	8	8	2	0.44	1.00	0.44	1.00	0.49	1.00	1	0.00	431997.45
DS_0010	894	805	5	5	2	0.48	1.00	0.48	1.00	0.50	1.00	1	0.00	611567.42
DS_0011	903	809	8	7	1	1.00	1.00	1.00	1.00	1.00	1.00	15	0.02	826754.81
DS_0100	709	586	5	4	2	0.98	1.00	1.00	1.00	0.99	1.00	480	0.82	15136.24
DS_0101	712	587	8	4	3	0.98	1.00	1.00	1.00	0.99	1.00	230	0.39	105842.05
DS_0110	708	573	5	4	2	0.50	1.00	1.00	1.00	0.74	1.00	612	1.07	113241.92
DS_0111	711	580	8	7	4	0.39	1.00	1.00	1.00	0.62	1.00	802	1.38	106285.73
DS_0200	894	766	5	17	2	0.45	0.96	0.45	1.00	0.56	0.97	NA*	NA*	758576.74
DS_0201	897	775	8	18	4	0.46	0.87	0.46	0.96	0.63	0.92	NA*	NA*	816136.72
DS_0210	894	761	5	16	2	0.50	0.68	0.50	0.80	0.66	0.76	NA*	NA*	88602.66
DS_0211	897	764	8	18	3	0.53	0.61	0.53	0.75	0.70	0.70	NA*	NA*	566601.80

NA* : not available

Table F.12 Experimental results with WCERN for group 3 data sets

Data set	#P	#P-ADR	# TC	#C after NC	# SN	min JI	max JI	min QJI	max QJI	min RI	max RI	# ITA	# ITA / #P-ADR	Time (sec.)
DS_0000	894	739	5	5	3	0.58	1.00	0.58	1.00	0.59	1.00	1	0.00	413885.35
DS_0001	903	732	8	8	5	0.44	1.00	0.44	1.00	0.49	1.00	1	0.00	457816.59
DS_0010	894	805	5	5	3	0.67	1.00	0.67	1.00	0.88	1.00	1	0.00	588700.81
DS_0011	903	809	8	7	4	0.24	1.00	0.24	1.00	0.25	1.00	42	0.05	433819.53
DS_0100	709	586	5	4	3	0.50	1.00	0.50	1.00	0.75	1.00	489	0.83	107215.21
DS_0101	712	587	8	4	3	0.98	1.00	0.98	1.00	1.00	1.00	817	1.39	111630.17
DS_0110	708	573	5	4	2	0.50	1.00	0.50	1.00	0.75	1.00	315	0.55	102406.58
DS_0111	711	580	8	7	5	0.43	1.00	0.43	1.00	0.67	1.00	987	1.70	103166.29
DS_0200	894	766	5	17	2	0.62	0.91	0.62	0.98	0.75	0.96	NA*	NA*	480506.34
DS_0201	897	775	8	18	3	0.55	0.83	0.55	0.97	0.64	0.90	NA*	NA*	58580.51
DS_0210	894	761	5	16	3	0.41	0.76	0.41	0.90	0.61	0.84	NA*	NA*	675741.51
DS_0211	897	764	8	18	3	0.61	0.71	0.61	0.82	0.74	0.77	NA*	NA*	812144.05

NA* : not available

VITA

PERSONAL INFORMATION

Surname, Name:	İnkaya, Tülin
Nationality:	Turkish (T.R.)
Date and Place of Birth:	18 October 1980, İzmir
e-mail:	tinkaya@gmail.com

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	Uludağ University, Industrial Engineering	2005
BS	Uludağ University, Industrial Engineering	2002
High School	Bursa Anatolian High School	1998

WORK EXPERIENCE

Year	**Place**	**Enrollment**
2005-Present	METU	Research Assistant
2002-2005	Uludağ University	Research Assistant
2001	Robert Bosch TR	Intern
2000	Oyak Renault	Intern

FOREIGN LANGUAGES

Advanced English, Intermediate French

PUBLICATIONS

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

2. İnkaya T., Kayalıgil S., Özdemirel N.E., 2010. A new density-based clustering approach in graph theoretic context, *In: Proceedings of the Fourth European Conference on Data Mining*, 1-8.

3. Taşkaya Temizel, T., Mizani M. A., İnkaya T., and Yücebaş S. C., 2007. The Effect of Data Set Characteristics on the Choice of Clustering Validity Index Type. *In: Proceedings of the Twenty-second International Symposium on Computer and Information Sciences*, 1-6.

HOBBIES

Reading books, play sports (skiing, tennis, squash, aerobics), movies and travelling