Sanghamitra Bandyopadhyay
Sriparna Saha

Unsupervised Classification

Similarity Measures, Classical and Metaheuristic Approaches, and Applications

Springer

Unsupervised Classification

Sanghamitra Bandyopadhyay • Sriparna Saha

Unsupervised Classification

Similarity Measures, Classical and Metaheuristic Approaches, and Applications

Springer

Sanghamitra Bandyopadhyay
Machine Intelligence Unit
Indian Statistical Institute
Kolkata, West Bengal, India

Sriparna Saha
Dept. of Computer Science
Indian Institute of Technology
Patna, India

ISBN 978-3-642-32450-5 ISBN 978-3-642-32451-2 (eBook)
DOI 10.1007/978-3-642-32451-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012953901

ACM Computing Classification (1998): I.2, J.3, H.2, H.3

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To my parents Satyendra Nath Banerjee and Bandana Banerjee, my husband Prof. Ujjwal Maulik, and son Utsav
Sanghamitra Bandyopadhyay

To my husband Dr. Asif Ekbal, my daughter Spandana, and my parents Sankar Prasad Saha and Sila Saha
Sriparna Saha

Preface

Classification is an integral part of any pattern recognition system. Depending on whether a set of labeled training samples is available or not, classification can be either supervised or unsupervised. Clustering is an important unsupervised classification technique where a number of data points are grouped into clusters such that points belonging to the same cluster are similar in some sense and points belonging to different clusters are dissimilar in the same sense. Cluster analysis is a complex problem as a variety of similarity and/or dissimilarity measures exist in the literature without any universal definition. In a crisp clustering technique, each pattern is assigned to exactly one cluster, whereas in the case of fuzzy clustering, each pattern is given a membership degree to each class. Fuzzy clustering is inherently more suitable for handling imprecise and noisy data with overlapping clusters.

For partitioning a data set, one has to define a measure of similarity or proximity based on which cluster assignments are done. The measure of similarity is usually data dependent. It may be noted that, in general, one of the fundamental features of shapes and objects is symmetry, which is considered to be important for enhancing their recognition. Examples of symmetry abound in real life, such as the human face, human body, flowers, leaves, and jellyfish. As symmetry is so common, it may be interesting to exploit this property while clustering a data set. Based on this observation, in recent years a large number of symmetry-based similarity measures have been proposed. This book is focused on different issues related to clustering, with particular emphasis on symmetry-based and metaheuristic approaches.

The aim of a clustering technique is to find a suitable grouping of the input data set so that some criteria are optimized. Hence, the problem of clustering can be posed as an optimization problem. The objective to be optimized may represent different characteristics of the clusters, such as compactness, symmetrical compactness, separation between clusters, connectivity within a cluster, etc. A straightforward way to pose clustering as an optimization problem is to optimize some cluster validity index that reflects the goodness of the clustering solutions. All possible values of the chosen optimization criterion (validity index) define the complete search space. Traditional partitional clustering techniques, such as K-means and fuzzy C-means, employ a greedy search technique over the search space in order to optimize

the compactness of the clusters. Although these algorithms are computationally efficient, they suffer from some drawbacks. They often get stuck at some local optima depending on the choice of the initial cluster centers. They are not able to determine the appropriate number of clusters from data sets and/or are capable of detecting clusters of some specific shape only.

To overcome the problem of getting stuck at local optima, several metaheuristic optimization tools, such as genetic algorithms (GAs), simulated annealing (SA), differential evolution (DE), etc., have been widely used to reach the global optimum value of the chosen validity measure. These techniques perform multimodal search in complex landscapes and provide near-optimal solutions for the objective or fitness function of an optimization problem. They have applications in fields as diverse as pattern recognition, image processing, neural networks, machine learning, job shop scheduling, and very large-scale integration (VLSI design), to mention just a few.

The two fundamental questions that need to be addressed in any typical clustering scenario are: (i) how many clusters are actually present in the data and (ii) how real or good is the clustering itself. That is, whatever the clustering technique, one has to determine the number of clusters and also the validity of the clusters formed. The measure of validity of clusters should be such that it will be able to impose an ordering of the clusters in terms of their goodness. Several cluster validity indices have been proposed in the literature, e.g., the Davies-Bouldin (DB) index, Dunn's index, Xie-Beni (XB) index, I-index, CS-index, etc., to name just a few. In recent years, several symmetry-based cluster validity indices have also been developed which are able to detect any kind of symmetric cluster from data sets. In this book, we discuss in detail several existing well-known cluster validity indices as well as some recently proposed symmetry-based versions.

Conventional GA-based clustering techniques, as well as related approaches, use some validity measure for optimization. Most of the existing clustering algorithms are able to determine hyperspherical/ellipsoidal-shaped clusters depending on the distance norm used. In recent years, some symmetry-based clustering techniques have been developed which can determine the appropriate number of clusters and the appropriate partitioning from data sets having any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristic of symmetry. A major focus of this book is on GA-based clustering techniques, which use symmetry as a similarity measure. Some of these clustering techniques are also able to detect the number of clusters automatically.

In many real-life situations one may need to optimize several objectives simultaneously. These problems are known as multiobjective optimization problems (MOOPs). In this regard, a multitude of metaheuristic single-objective optimization techniques such as genetic algorithms, simulated annealing, differential evolution, and their multiobjective versions have been developed. In this book we discuss in detail some existing single- and multiobjective optimization techniques. Moreover, a newly developed multiobjective simulated annealing-based technique is elaborately described and its effectiveness for solving several benchmark test problems is shown.

In the realm of clustering, simultaneous optimization of multiple validity indices, capturing different characteristics of the data, is expected to provide improved robustness to different data properties. Hence, it is useful to apply some multiobjective optimization (MOO) technique to solve the clustering problem. In MOO, search is performed over a number of objective functions. In contrast to single-objective optimization, which yields a single best solution, in MOO the final solution set contains a number of Pareto-optimal solutions, none of which can be further improved with regard to any one objective without degrading another. In a part of this book some recently developed multiobjective clustering techniques are discussed in detail.

A major focus of this book is on several real-life applications of clustering techniques in domains such as remote sensing satellite images, magnetic resonance (MR) brain images, and face detection. Analysis of remote sensing satellite images has significant utility in different areas such as climate studies, assessment of forest resources, examining marine environments, etc. An important task in remote sensing applications is the classification of pixels in the images into homogeneous regions, each of which corresponds to some particular land cover type. This problem has often been modeled as a segmentation problem, and clustering methods have been used to solve it. However, since it is difficult to have a priori information about the number of clusters in satellite images, the clustering algorithms should be able to automatically determine this value. Moreover, in satellite images it is often the case that some regions occupy only a few pixels, while the neighboring regions are significantly large. Thus, automatically detecting regions or clusters of such widely varying sizes presents a challenge in designing segmentation algorithms. In this book, we explore the applications of some symmetry-based clustering algorithms to classify remote sensing imagery in order to demonstrate their effectiveness.

Automatically classifying brain tissues from magnetic resonance images (MRI) has a major role in research and clinical study of neurological pathology. Additionally, regional volume calculations may provide even more useful diagnostic information. Automatic segmentation of brain MR images is a complex problem. Clustering approaches have been widely used for segmentation of MR brain images. A part of this book is dedicated to the applications of some symmetry-based clustering algorithms to classify MR brain images in order to demonstrate their effectiveness.

In recent years the problem of human face recognition has gained huge popularity. There are wide applications of face recognition systems including secure access control and financial transactions. The first important step of fully automatic human face recognition is human face detection. Face detection is the technique to automatically determine the locations and sizes of faces in a given input image. In this book a procedure to detect human faces based on symmetry is also described in detail.

This book aims to provide a treatise on clustering in a metaheuristic framework, with extensive applications to real-life data sets. Chapter 1 provides an introduction to pattern recognition, machine learning, classification, clustering, and related areas, along with different applications of pattern recognition. Chapter 2 describes in detail existing metaheuristic-based single- and multiobjective optimization techniques. An elaborate discussion on a recently developed multiobjective simulated annealing-based technique, archived multiobjective simulated annealing (AMOSA), is also

provided in this chapter. The utility of this technique to solve several benchmark test problems is shown. Chapter 3 mainly describes the different types of similarity measures developed in the literature for handling binary, categorical, ordinal, and quantitative variables. It also contains a discussion on different normalization techniques. Chapter 4 gives a broad overview of the existing clustering techniques and their relative advantages and disadvantages. It also provides a detailed discussion of several recently developed single- and multiobjective metaheuristic clustering techniques. Chapter 5 presents symmetry-based distances and a genetic algorithm-based clustering technique that uses this symmetry-based distance for assignment of points to different clusters and for fitness computation. In Chap. 6, some symmetry-based cluster validity indices are described. Elaborate experimental results are also presented in this chapter. Application of these symmetry-based cluster validity indices to segment remote sensing satellite images is also presented. In Chap. 7, some automatic clustering techniques based on symmetry are presented. These techniques are able to determine the appropriate number of clusters and the appropriate partitioning from data sets having symmetric clusters. The effectiveness of these clustering techniques is shown for many artificial and real-life data sets, including MR brain image segmentation. Chapter 8 deals with an extension of the concept of point symmetry to line symmetry-based distances, and genetic algorithm-based clustering techniques using these distances. Results are presented for some artificial and real-life data sets. A technique using the line symmetry-based distance for face detection from images is also discussed in detail. Some multiobjective clustering techniques based on symmetry are described in detail in Chap. 9. Three different clustering techniques are discussed here. The first one assumes the number of clusters a priori. The second and third clustering techniques are able to detect the appropriate number of clusters from data sets automatically. The third one, apart from using the concept of symmetry, also uses the concept of connectivity. A method of measuring connectivity between two points in a cluster is described for this purpose. A connectivity-based cluster validity index is also discussed in this chapter. Extensive experimental results illustrating the greater effectiveness of the three multiobjective clustering techniques over the single-objective approaches are presented for several artificial and real-life data sets.

This book contains an in-depth discussion on clustering and its various facets. In particular, it concentrates on metaheuristic clustering using symmetry as a similarity measure with extensive real-life applications in data mining, satellite remote sensing, MR brain imaging, gene expression data, and face detection. It is, in this sense, a complete book that will be equally useful to the layman and beginner as to an advanced researcher in clustering, being valuable for several reasons.

It includes discussions on traditional as well as some recent symmetry-based similarity measurements. Existing well-known clustering techniques along with metaheuristic approaches are elaborately described. Moreover, some recent clustering techniques based on symmetry are described in detail. Multiobjective clustering is another emerging topic in unsupervised classification. A multiobjective data clustering technique is described elaborately in this book along with extensive experimental results. A chapter of this book is wholly devoted to discussing some existing and

new multiobjective clustering techniques. Extensive real-life applications in remote sensing satellite imaging, MR brain imaging, and face detection are also provided in the book. Theoretical analyses of some recent symmetry-based clustering techniques are included.

The book will be useful to graduate students and researchers in computer science, electrical engineering, system science, and information technology as both text and reference book for some parts of the curriculum. Theoretical and experimental researchers will benefit from the discussions in this book. Researchers and practitioners in industry and R & D laboratories working in fields such as pattern recognition, data mining, soft computing, remote sensing, and biomedical imaging will also benefit.

The authors gratefully acknowledge the initiative and support for the project provided by Mr. Ronan Nugent of Springer. Sanghamitra Bandyopadhyay acknowledges the support provided by the Swarnajayanti Project Grant (No. DST/SJF/ET-02/2006-07) of the Department of Science and Technology, Government of India. The authors acknowledge the help of Mr. Malay Bhattacharyya, senior research fellow of ISI Kolkata, for drawing some of the figures of Chap. 5. Sriparna Saha also acknowledges the help of her students Mridula, Abhay, and Utpal for proofreading the chapters.

Kolkata, India

Sanghamitra Bandyopadhyay
Sriparna Saha

Contents

1 Introduction 1
1.1 Introduction 1
1.2 Data Types with Examples 3
1.3 Pattern Recognition: Preliminaries 4
1.3.1 Data Acquisition 6
1.3.2 Feature Selection 7
1.3.3 Classification 7
1.3.4 Clustering 8
1.3.5 Distance Measures in Clustering 9
1.3.6 Model Selection 10
1.3.7 Model Order Selection 10
1.4 Robustness to Outliers and Missing Values 11
1.5 Fuzzy Set-Theoretic Approach: Relevance and Features 12
1.6 Applications of Pattern Recognition and Learning 13
1.7 Summary and Scope of the Book 14

2 Some Single- and Multiobjective Optimization Techniques 17
2.1 Introduction 17
2.2 Single-Objective Optimization Techniques 18
2.2.1 Overview of Genetic Algorithms 19
2.2.2 Simulated Annealing 23
2.3 Multiobjective Optimization 25
2.3.1 Multiobjective Optimization Problems 26
2.3.2 Various Methods to Solve MOOPs 28
2.3.3 Recent Multiobjective Evolutionary Algorithms 29
2.3.4 MOOPs and SA 33
2.4 An Archive-Based Multiobjective Simulated Annealing Technique: AMOSA 36
2.4.1 Introduction 36
2.4.2 Archived Multiobjective Simulated Annealing (AMOSA) . 37
2.4.3 Archive Initialization 38

2.4.4 Clustering the Archive Solutions . . . 38
2.4.5 Amount of Domination . . . 40
2.4.6 The Main AMOSA Process . . . 41
2.4.7 Complexity Analysis . . . 45
2.5 Simulation Results . . . 46
2.5.1 Comparison Measures . . . 47
2.5.2 Comparison of Binary-Encoded AMOSA with NSGA-II and PAES . . . 48
2.5.3 Comparison of Real-Coded AMOSA with the Algorithm of Smith et al. and Real-Coded NSGA-II . . . 51
2.5.4 Discussion on Annealing Schedule . . . 56
2.6 Discussion and Conclusions . . . 57

3 Similarity Measures . . . 59
3.1 Introduction . . . 59
3.2 Definitions . . . 60
3.2.1 Need for Measuring Similarity . . . 60
3.3 Similarity/Dissimilarity for Binary Variables . . . 61
3.4 Distance for Nominal/Categorical Variable . . . 63
3.4.1 Method 1: Each Category Is Represented by a Single Binary Variable [278] . . . 64
3.4.2 Method 2: Each Category Is Represented by Several Binary Variables [278] . . . 64
3.5 Distance for Ordinal Variables . . . 65
3.5.1 Normalized Rank Transformation . . . 66
3.5.2 Spearman Distance . . . 67
3.5.3 Footrule Distance . . . 67
3.6 Distance for Quantitative Variables . . . 68
3.6.1 Euclidean Distance . . . 68
3.6.2 Minkowski Distance of Order λ . . . 68
3.6.3 City Block Distance . . . 69
3.6.4 Chebyshev Distance . . . 69
3.6.5 Canberra Distance . . . 70
3.6.6 Bray-Curtis Distance . . . 70
3.6.7 Angular Separation . . . 70
3.6.8 Correlation Coefficient . . . 71
3.6.9 Mahalanobis Distance . . . 72
3.7 Normalization Methods . . . 72
3.8 Summary . . . 73

4 Clustering Algorithms . . . 75
4.1 Introduction . . . 75
4.2 Preliminaries . . . 76
4.2.1 Definition of Clustering . . . 76
4.2.2 Some Clustering Techniques . . . 76

4.3 Partitional Clustering Techniques . . . 77
4.3.1 K-Means Clustering Technique . . . 77
4.3.2 K-Medoid Clustering Technique . . . 79
4.3.3 Fuzzy C-Means Clustering Algorithm . . . 79
4.4 Distribution-Based Clustering Approach . . . 80
4.5 Hierarchical Clustering Techniques . . . 82
4.6 Density-Based Clustering Techniques . . . 83
4.7 Grid-Based Clustering Techniques . . . 85
4.8 Some Recent Clustering Techniques . . . 85
4.9 Some Evolutionary Approaches to Clustering . . . 87
4.9.1 Algorithms for a Fixed Value of the Number of Clusters . . . 88
4.9.2 Algorithms with Variable Number of Clusters . . . 89
4.10 MOO and Clustering . . . 90
4.11 Summary . . . 91

5 Point Symmetry-Based Distance Measures and Their Applications to Clustering . . . 93
5.1 Introduction . . . 93
5.2 Some Existing Symmetry-Based Distance Measures . . . 94
5.3 A New Definition of the Point Symmetry Distance [27] . . . 100
5.4 Some Properties of $d_{ps}(\overline{x}, \overline{c})$ . . . 102
5.5 Kd-Tree-Based Nearest Neighbor Computation . . . 103
5.6 GAPS: The Genetic Clustering Scheme with New PS-Distance [27] . . . 104
5.6.1 Chromosome Representation and Population Initialization . . . 104
5.6.2 Fitness Computation . . . 106
5.6.3 Selection . . . 107
5.6.4 Crossover . . . 107
5.6.5 Mutation . . . 108
5.6.6 Termination . . . 109
5.6.7 Complexity Analysis . . . 109
5.7 On the Convergence Property of GAPS . . . 110
5.7.1 Preliminaries . . . 110
5.7.2 Convergence Proof . . . 112
5.8 Experimental Results of GAPS . . . 113
5.8.1 Data Sets Used . . . 113
5.8.2 Implementation Results . . . 114
5.8.3 Summary . . . 121

6 A Validity Index Based on Symmetry: Application to Satellite Image Segmentation . . . 125
6.1 Introduction . . . 125
6.2 Some Existing Cluster Validity Indices . . . 126
6.2.1 BIC Index . . . 126
6.2.2 Calinski-Harabasz Index . . . 127

6.2.3 Silhouette Index . . . 127
6.2.4 DB Index . . . 127
6.2.5 Dunn's Index . . . 128
6.2.6 Xie-Beni Index . . . 128
6.2.7 PS Index . . . 129
6.2.8 I-Index . . . 129
6.2.9 CS-Index . . . 129
6.3 *Sym*-Index: The Symmetry-Based Cluster Validity Index . . . 130
6.3.1 The Cluster Validity Measure . . . 130
6.3.2 Mathematical Justification . . . 134
6.3.3 Interaction Between the Different Components of Sym-Index . . . 136
6.4 Experimental Results . . . 140
6.4.1 Data Sets . . . 140
6.4.2 Comparative Results . . . 140
6.4.3 Analysis of Results . . . 143
6.5 Incorporating d_{ps} in Some Existing Cluster Validity Indices . . . 147
6.6 Point Symmetry-Based Cluster Validity Indices . . . 148
6.6.1 Symmetry-Based Davies-Bouldin Index (*Sym-DB* Index) . . . 148
6.6.2 Symmetry-Based Dunn's Index (*Sym-Dunn* Index) . . . 149
6.6.3 Symmetry-Based Generalized Dunn's Index (*Sym-GDunn* Index) . . . 149
6.6.4 New Symmetry Distance-Based PS-Index (*Sym-PS* Index) . . . 150
6.6.5 Symmetry-Based Xie-Beni Index (*Sym-XB* Index) . . . 151
6.6.6 Symmetry-Based FS-Index (*Sym-FS* Index) . . . 151
6.6.7 Symmetry-Based K-Index (*Sym-K* Index) . . . 151
6.6.8 Symmetry-Based SV-Index (*Sym-SV* Index) . . . 152
6.7 Experimental Results . . . 152
6.7.1 Discussion of Results . . . 152
6.8 Application to Remote Sensing Imagery . . . 155
6.8.1 Simulated Circle Image (SCI) . . . 156
6.8.2 SPOT Image of Kolkata . . . 156
6.9 Discussion and Conclusions . . . 160

7 Symmetry-Based Automatic Clustering . . . 165
7.1 Introduction . . . 165
7.2 Some Existing Genetic Algorithm-Based Automatic Clustering Techniques . . . 166
7.3 Description of VGAPS . . . 168
7.3.1 Chromosome Representation and Population Initialization . . . 168
7.3.2 Fitness Computation . . . 168
7.3.3 Genetic Operations and Terminating Criterion . . . 168

7.4 On the Convergence Property of VGAPS . . . 171
7.4.1 To Check Whether the Mutation Matrix Is Positive . . . 172
7.4.2 Conditions on Selection . . . 173
7.5 Data Sets Used and Implementation Results . . . 174
7.5.1 Data Sets Used . . . 174
7.5.2 Results and Discussions . . . 174
7.6 Extending VGAPS to Fuzzy Clustering . . . 178
7.6.1 Fuzzy Symmetry-Based Cluster Validity Index . . . 178
7.6.2 Fuzzy-VGAPS Clustering . . . 179
7.7 Implementation Results and Comparative Study . . . 182
7.7.1 Discussion of Results . . . 184
7.7.2 Application to MR Brain Image Segmentation . . . 189
7.7.3 Experimental Results . . . 190
7.8 Summary . . . 194

8 Some Line Symmetry Distance-Based Clustering Techniques . . . 197
8.1 Introduction . . . 197
8.2 The Line Symmetry-Based Distance . . . 197
8.2.1 Definition . . . 198
8.3 GALSD: The Genetic Clustering Scheme with Line Symmetry-Based Distance . . . 199
8.3.1 String Representation and Population Initialization . . . 199
8.3.2 Fitness Computation . . . 199
8.3.3 Genetic Operators . . . 200
8.4 Experimental Results . . . 200
8.4.1 Data Sets Used . . . 200
8.4.2 Discussion of Results . . . 202
8.4.3 Computation Time . . . 203
8.5 Application to Face Recognition . . . 204
8.5.1 Human Face Detection Algorithm . . . 205
8.5.2 Experimental Results . . . 207
8.6 A Generalized Line Symmetry-Based Distance and Its Application to Data Clustering . . . 209
8.7 Implementation Results . . . 210
8.7.1 Data Sets Used . . . 211
8.7.2 Discussion of Results . . . 211
8.8 Discussion and Conclusions . . . 215

9 Use of Multiobjective Optimization for Data Clustering . . . 217
9.1 Introduction . . . 217
9.2 MOPS: Multiobjective Clustering Using Point Symmetry Distance . . . 219
9.2.1 Selection of a Solution from the Archive . . . 220
9.2.2 Experimental Results . . . 220
9.3 VAMOSA: Symmetry-Based Multiobjective Clustering Technique for Automatic Evolution of Clusters . . . 221

9.3.1 Data Sets Used for Experiment 222
9.3.2 Experimental Results 223
9.4 A Generalized Automatic Clustering Algorithm in a Multiobjective Framework 227
9.4.1 *GenClustMOO*: Multiobjective Clustering Technique . . . 228
9.4.2 Subcluster Merging for Objective Function Calculation . . 232
9.5 Experimental Results 233
9.5.1 Data Sets Used 233
9.5.2 Discussion of Results 235
9.6 Discussion and Conclusions 242

References 245

Index 259

Chapter 1
Introduction

1.1 Introduction

Classification is the task of assigning labels to a set of instances in such a way that instances with the same label share some common properties and logically belong to the same class. As an example, it may be required to retrieve those images from a database that contain a picture of a flower. Here, each image is an instance, and all images that contain a picture of a flower are labeled say 1, while those that do not are labeled say 2. Note that the labeling could have been different, e.g., '1' for the former and '0' for the latter, or 'F' for the former and 'NF' for the latter. The classification step is an integral part of any computational pattern recognition system. The more accurate this task, the better the recognition ability of the system will be.

Classification can be either supervised or unsupervised in nature. When the decision about the class "belongingness" of a query instance (unlabeled) is made using knowledge extracted from other labeled instances, the approach is called supervised. Here, the availability of a set of labeled instances, or supervisors, is assumed. This is the traditional way in which a child learns–through the presentation of a set of examples. When preclassified, labeled instances are not available, the task of classification is often referred to as unsupervised classification or clustering. Given a set of instances, clustering generally means finding groups in them such that objects belonging to the same group are more similar to each other than objects belonging to different groups.

Cluster analysis is a difficult problem because of the existence of multiple similarity and dissimilarity definitions which do not have any universal definition. Therefore, seeking an appropriate clustering of the data is experiment oriented. In crisp clustering, each pattern is assigned to exactly one cluster, whereas in the case of fuzzy clustering, each pattern is given a membership degree to each class. Fuzzy clustering is essentially more appropriate for partitioning data sets having overlapping clusters.

For partitioning a data set, one has to define a measure of similarity or proximity based on which cluster assignments are done. The measure of similarity is usually

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_1,

data dependent. It may be noted that, in general, symmetry can be treated as one of the essential attributes of shapes and objects. This is considered to be important for enhancing their recognition. Examples of symmetry abound in real life, such as the human face, human body, and jellyfish. Almost every interesting area around us consists of some generalized form of symmetry. As symmetry is so common, it may be interesting to exploit this property while clustering a data set. Based on this observation, in recent years, a large number of symmetry-based similarity measures have been proposed. This book is mainly focused on different issues related to clustering, with particular emphasis on symmetry-based and metaheuristic approaches.

In our everyday life, we make decisions consciously or unconsciously. These decisions can be very simple, such as selecting the color of a dress or deciding the menu for lunch, or may be as difficult as designing a missile or selecting a career. The former are easy to make, while the latter might take several years due to the level of complexity involved. The main goal of most kinds of decision-making is to optimize one or more criteria in order to achieve the desired result. In other words, problems related to optimization abound in real life. Development of optimization algorithms has therefore been a great challenge in computer science. The problem is compounded by the fact that in many situations one may need to optimize several objectives simultaneously. These specific problems are known as multiobjective optimization problems (MOOPs). In this regard, a multitude of metaheuristic single-objective optimization techniques such as genetic algorithms, simulated annealing, differential evolution, and their multiobjective versions have been developed.

Clustering is considered to be a difficult task as no unambiguous partitioning of the data exists for many data sets. Most of the existing clustering techniques are based on only one criterion which reflects a single measure of goodness of a partitioning. However, a single cluster quality measure is seldom equally applicable for different kinds of data sets with different characteristics. Hence, it may become necessary to simultaneously optimize several cluster quality measures that can capture different data characteristics. In order to achieve this, the problem of clustering a data set has been posed as one of multiobjective (MO) optimization in literature. Therefore, the application of sophisticated metaheuristic multiobjective optimization techniques seems appropriate and natural. This book also discusses some recent multiobjective clustering techniques.

Clustering is a commonly used unsupervised data mining tool that is useful for mining different kind of data sets including geoscientific data, biological data, world wide web data, multimedia data, etc. In this book, we explore the application of different clustering techniques for discovering knowledge from geoscientific data such as satellite images and biological data such as MR brain image data sets. Remote sensing satellite images have significant applications in different areas such as climate studies, assessment of forest resources, examining marine environments, etc. An important aspect of remote sensing applications is the classification of pixels in the images into homogeneous partitions, each of which corresponds to some particular land cover type. This problem has often been formulated as a partitioning problem, and different clustering algorithms have been employed to solve it. However, since it is problematic to know a priori the number of clusters present in satellite images, such clustering algorithms should be able to automatically determine

this value. Moreover, in satellite images it is often the case that some regions occupy only a few pixels, while the neighboring regions are significantly large. Thus, automatically detecting regions or clusters of such widely varying sizes presents a challenge in designing segmentation algorithms. In a part of this book, the problem of automatically partitioning satellite images into different land cover types is dealt with.

Automatically identifying brain tissues from magnetic resonance imaging (MRI) is a crucial task for research and clinical study of neurological pathology. Additionally, some useful diagnostic information can be obtained from regional volume calculations. This includes the quantization of gray and white matter volumes, which plays a significant role in different diseases including Alzheimer disease, Parkinson or Parkinson-related syndrome, white matter metabolic or inflammatory disease, congenital brain malformations or perinatal brain damage, or posttraumatic syndrome. Automatic partitioning of brain MR images, however, remains an open problem. Clustering techniques have been mostly used for partitioning these MR brain images. In this book the application of some recently developed clustering algorithms for classifying MR brain images is explored in order to demonstrate their effectiveness.

1.2 Data Types with Examples

Variables are primarily classified depending on their type, and use, being mainly categorized into four different types:

- Binary variable: A binary variable can only take binary values, i.e., 0 or 1. It can only take values 'True' or 'False'.
- Quantitative variable: The term 'Quantitative' comes from the word "quantity" indicating an exact amount. A quantitative variable is generally measured as a number or a value. As these variables have some numeric values, they are suitable for applications of some meaningful arithmetic operations. Examples of this type of variable are height, weight, age, blood pressure, salary, temperature, area, time, distance, etc.
- Categorical variable: This type of variable does not take any numeric value; rather it can take a value that is one of several possible categories. No numerical meaning is associated with categorical variables. Examples of this type of variable include: hair color, gender, field of study, college attended, subjects in a semester, political affiliation, status of disease infection, etc.

  Often, categorical variables can be converted into quantitative variables; for example, gender information can be coded as $0 =$ Male, $1 =$ Female, but note that the variable is still a categorical variable; it is not naturally measured as a number.
- Ordinal variable: This is a special type of categorical variable for which there exists an ordering of the categories. A good example of this type of variable is 'Grades of students'. 'AA' represents 'Excellent', 'AB' represents 'Good', 'BA'

represents 'Average', and 'BB' represents 'Poor'. Thus, these four categories 'AA', 'AB', 'BA', and 'BB' are four categories but there is an ordering among these four categories. In case of categorical variables such as 'Gender' there exists no ordering for the different categories.

1.3 Pattern Recognition: Preliminaries

Human beings are very adept at recognizing patterns, a task that they continually execute. In contrast, getting a machine to learn and recognize patterns can be extremely difficult and challenging [24]. It is generally conjectured that animal brains process sensory data obtained from different senses, apply rules that are embedded in the brain (which are usually thought to be learned starting from birth), and recognize patterns.

Pattern recognition and machine learning are two major areas of research in artificial intelligence [24, 86, 104, 201, 229, 279]. Different textual, pictorial, and nonnumerical information obtained from the interaction between science, technology, and society can be processed by these two fields. Research in machine learning is primarily engaged in designing some specialized algorithms to enable machines to behave like human beings. Using these algorithms, a given machine can identify some unknown samples by utilizing knowledge learned via inductive inference based on observing data that represent incomplete information about statistical phenomena. Pattern recognition is another subbranch of machine learning. Different pattern recognition algorithms are aimed at automatic identification of unknown complex patterns by a given machine, and thereby enabling them to make some intelligent decision. The most prominent outcome of these fields is the concept of ubiquitous computing systems [24].

Objects in the real world are captured using some measurements on them. For example, in the famous iris flower data, the petal length, petal width, sepal length, and sepal width of iris flowers are measured. Each flower can belong to one of three classes: Setosa, Versicolor, and Virginica [24]. Thus, a flower may be represented using a vector in $\mathbb{R}^4$ of the above four measurement values, and it may be labeled by one of the three classes. In other words, a flower may be mapped to a point in a four-dimensional space, and this point may have an associated class label. The task of any pattern recognition system essentially involves taking some measurements of the objects to be recognized, thereby mapping them to points in some space. Thereafter, if the system has some labeled points then it has to infer some rules about the "belongingness" of points to the classes. Given an unknown point, the system will then have to decide about its classification, based on the rules that it has already learned. On the other hand, if the system is not provided with any labeled information, then its task may be to find some relationship among the unlabeled points that it has collected [24].

Machine learning [24] is an area of artificial intelligence that deals with the task of making machines capable of learning from their environment. Evolution is probably the most powerful learning mechanism in existence. Over millions of years,

different species have evolved in such a way that they have become more suited to their environment. In biological evolution, changes occur at the genetic level through crossover and mutation. These changes are propagated to future generations, where the process gets repeated. Usually, these changes are such that they provide some survival advantage to an individual with respect to the local environment. Changes that are disadvantageous tend to decrease in frequency, while those that are beneficial are more likely to be preserved because they aid survival.

There exist a large number of popular single-objective metaheuristic optimization techniques including genetic algorithms [112], simulated annealing [159], evolutionary strategies [231], etc. Genetic algorithms (GAs) [112, 129] are randomized search and optimization techniques guided by the principles of evolution and natural genetics. GAs mimic some of the processes observed in natural evolution, which include operations such as selection, crossover, and mutation. They perform multimodal search in complex landscapes and provide near-optimal solutions for the objective or fitness function of an optimization problem. They are efficient, adaptive, and robust search processes with a large amount of implicit parallelism [112]. Genetic algorithms have diverse applications in solving problems requiring efficient and effective search, in business, scientific, and engineering circles [20, 24]. Genetic algorithms find plenty of applications in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics, and other fields.

Simulated annealing (SA) [159] belongs to the class of local search algorithms. It utilizes the principles of statistical mechanics, regarding the behavior of a large number of atoms at low temperature, to find minimal cost functions for large optimization problems by minimizing the associated energy. SA has been applied in diverse areas [22, 52, 191] by optimizing a single criterion.

In particular, many tasks involved in the process of recognizing a pattern need appropriate parameter selection and efficient search in complex and large spaces in order to attain optimal solutions. This not only makes the process computationally intensive, but also leads to the possibility of missing the exact solution. Therefore, the application of metaheuristic optimization techniques such as GAs or SA for solving certain problems of pattern recognition that require optimization of computation requirements, and robust, fast, and close approximate solution, seems appropriate and natural.

Automatic recognition of patterns by a given machine can be categorized as a two-stage task. In the first stage, the unvarying properties of a set of samples of a given class are studied and learned by a given machine. In the second stage, some unknown samples are categorized using this learned knowledge. A typical pattern recognition system comprises three steps (Fig. 1.1): *data acquisition, feature extraction*, and *classification*. In the data acquisition phase, sensors are used to collect data from a particular domain, which depends on the environment. The second phase is the feature selection/feature extraction stage, which extracts the important features from a given data set and retains those features which are most relevant. In a broader perspective, this stage significantly influences the entire recognition process. Finally, in the third or classification stage, the classifier uses the extracted

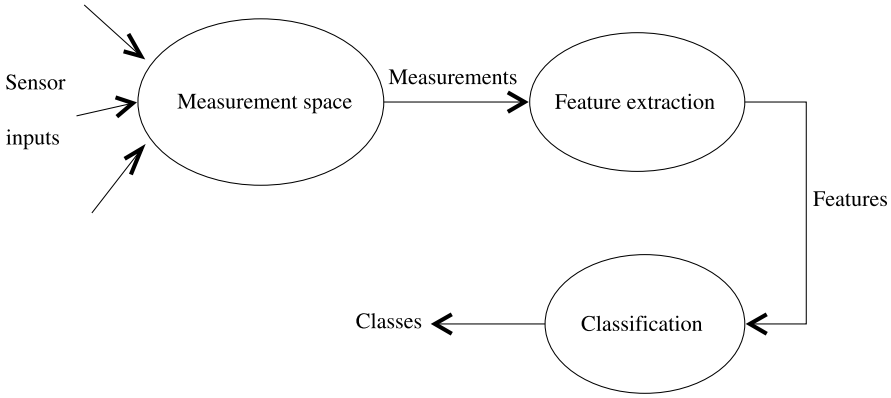


Fig. 1.1 A typical pattern recognition system; figure taken from [24]

features of the second stage to determine the class level of a particular unknown pattern. This phase basically establishes a transformation between the features and the classes. Therefore, pattern recognition can be described as a transformation from the measurement space M to the feature space F and finally to the decision space D, i.e.,

$$M \to F \to D.$$

Here the mapping $\delta : F \to D$ is the decision function, and the elements $d \in D$ are termed decisions.

In the following sections we describe, in brief, some tasks commonly undertaken in typical pattern recognition systems (some parts of the discussion are taken from [24]). These are data acquisition, feature selection, and some classification and clustering techniques.

1.3.1 Data Acquisition

Pattern recognition techniques are applicable to a wide range of domains, where the data may be qualitative, quantitative or both; they may be numerical, linguistic, pictorial or any combination thereof. Generally, the data structures that are used in pattern recognition systems are of two types: *object data vectors* and *relational data*. Object data, sets of numerical vectors of Q features, are represented in what follows as $\mathscr{Y} = \{Y_1, Y_2, \ldots, Y_t\}$, a set of t feature vectors in the Q-dimensional measurement space Ω_Y. The ith object, $i = 1, 2, \ldots, t$, observed in the process has vector Y_i as its numerical representation; y_{ij} is the jth ($j = 1, 2, \ldots, Q$) feature associated with the object i.

Relational data is a set of t^2 numerical relationships, say $\{r_{ii'}\}$, between pairs of objects. In other words, $r_{ii'}$ represents the extent to which object i and i' are related

in the sense of some binary relationship ρ. If the objects that are pairwise related by ρ are called $O = \{o_1, o_2, \ldots, o_t\}$, then $\rho : O \times O \to \mathbb{R}$.

Sometimes some preprocessing tasks such as noise reduction, filtering, encoding, and enhancement for extracting pattern vectors are involved in the data acquisition phase; for example, in case of satellite images, preprocessing is important for extracting some prominent features for recognition.

1.3.2 Feature Selection

Feature selection is a sub task of any automatic pattern recognition system. The primary objective of *feature selection* [31, 81] is to select some relevant features from a set of given features. Thus, it is capable of reducing the dimensionality of a given data set so that effective and easily computable algorithms can be devised for efficient classification. The problem of feature selection has two aspects: the formulation of a suitable criterion to evaluate the goodness of a feature and the selection of optimal subsets from the available features.

The major mathematical measures so far devised for estimation of feature quality are mostly statistical in nature, and can be broadly classified into two categories:

- Feature selection in the measurement space.
- Feature extraction in the transformed space.

The techniques in the first category generally reduce the dimensionality of the feature set by discarding redundant information carrying features. On the other hand, those in the second category utilize all the information contained in the pattern vectors, and map a higher-dimensional pattern vector to a lower-dimensional one.

Feature selection is the process of selecting a map of the form $X = f(Y)$, by which a sample Y $(= y_1, y_2, \ldots, y_Q)$ in a Q-dimensional measurement space Ω_Y is transformed into a point X $(= x_1, x_2, \ldots, x_N)$ in an N-dimensional feature space Ω_X, where $N < Q$.

The pioneering research on feature selection mostly deals with statistical tools. Later, the thrust of research is shifted to the development of various other approaches to feature selection, including fuzzy, neural, and genetic approaches [89, 214, 215, 235, 255, 284].

1.3.3 Classification

The problem of classification basically establishes a transformation $F \to D$ between the features and the classes (Fig. 1.1). In other words, it provides a partitioning of the feature space into regions, one region for each category of input. That is, it attempts to assign every data point in the entire feature space to

one of the possible (say K) classes. Different forms of this transformation include a Bayesian rule for computing a posteriori class probability, nearest neighbor rule, linear discriminant functions, perceptron rule, nearest prototype rule, etc. [6, 81, 85, 105, 106, 110, 113, 219, 281]. Classifiers are usually, but not always, designed with labeled data, in which case these problems are sometimes referred to as *supervised classification* (where the parameters of a classifier function D are learned). Some common examples of *supervised* pattern classification techniques are the nearest neighbor rule, Bayes' maximum-likelihood classifier, and the perceptron rule.

Let $C_1, C_2, \ldots, C_i, \ldots, C_K$ be the K possible classes in an N-dimensional feature space. Let $\mathbf{x} = [x_1, x_2, \ldots, x_j, \ldots, x_N]'$ be an unknown pattern vector. In a deterministic classification approach, it is assumed that there exists only one unambiguous pattern class corresponding to each of the unknown pattern vectors. If the pattern $\mathbf{x}$ is a member of class C_i, the *discriminant (decision) function* $D_i(\mathbf{x})$ associated with the class C_i, $i = 1, 2, \ldots, K$, must then possess the largest value. In other words, a classificatory decision would be as follows:

$$\text{Decide} \quad \mathbf{x} \in C_i, \quad \text{if } D_i(\mathbf{x}) > D_l(\mathbf{x}), \tag{1.1}$$

$(\forall i, l)$ in $(1, \ldots, K)$ and $i \neq l$. Ties are resolved arbitrarily. The decision boundary in the feature space between regions associated with the classes C_i and C_l would be governed by the expression

$$D_i(\mathbf{x}) - D_l(\mathbf{x}) = 0. \tag{1.2}$$

Many different forms satisfying Eq. 1.2 can be selected for $D_i(\mathbf{x})$. The functions that are often used are linear discriminant functions, quadratic discriminant functions, and polynomial discriminant functions. One commonly used piecewise linear discriminant function, which is used in the nearest neighbor (NN) classifier, involves distance as a similarity measure for classification.

1.3.4 Clustering

As already mentioned, when only unlabeled data are available, the classification methodology adopted is known as *unsupervised classification* or *clustering*. Many clustering algorithms are used as precursors to the design of a classifier in such situations. In clustering [24, 35, 81, 125, 143, 281] a set of data points are grouped into clusters such that points belonging to the same cluster are similar in some sense and points belonging to different clusters are dissimilar in the same sense. Clustering in N-dimensional Euclidean space $\mathbb{R}^N$ is the process of partitioning a given set of n points into a number, say K, of groups (or clusters) based on some similarity/dissimilarity metric. Let the set of n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be represented by the set S, and the K clusters be represented by $C_1, C_2, \ldots, C_K$. Then

$$C_i \neq \emptyset, \qquad \text{for } i = 1, \ldots, K,$$
$$C_i \bigcap C_j = \emptyset, \quad \text{for } i = 1, \ldots, K, \ j = 1, \ldots, K, \text{ and } i \neq j, \quad \text{and}$$
$$\bigcup_{i=1}^{K} C_i = S.$$

For this, it is necessary to first define a measure of similarity which will establish a rule for assigning patterns to the domain of a particular cluster center. One such measure of similarity may be the Euclidean distance D between two patterns x and z defined by $D = \|x - z\|$. The smaller the distance between x and z, the greater the similarity between the two, and vice versa.

Clustering techniques have been broadly categorized into partitional and hierarchical methods. One commonly used clustering technique in the partitional category is the K-means algorithm [281], where the number of clusters K is assumed to be known a priori. The K-means algorithm has been widely recommended after extensive studies dealing with comparative analysis of different clustering methods [84, 106, 196]. Popular techniques in the hierarchical category are the single linkage, complete linkage, and average linkage algorithms. A feature distinguishing the hierarchical clustering techniques from the partitional ones is that, while the former provide a valid clustering of the data at each iteration of the algorithm, the latter do not do so (they provide a valid clustering only on termination of the algorithm). Minimal spanning tree-based graph-theoretic clustering techniques are also quite popular in the pattern recognition community.

1.3.5 Distance Measures in Clustering

The main goal of clustering is to maximize both the homogeneity within each cluster and the heterogeneity among different clusters [24, 95, 134] irrespective of the type of clustering algorithm (partitional, hierarchical or overlapping). Alternatively, different objects that belong to the same cluster should be more similar to each other than objects belonging to different clusters. Distance measures are mostly used to quantify the similarity between two objects. The choice of distance function plays an important role in cluster analysis. Suppose that X denotes a set of n points, i.e., $X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$. Let d be the dimension of the data. Then x_{ij}, $i = 1, \ldots, n$ and $j = 1, \ldots, d$, denotes the value of jth dimension of the ith point. Let the distance between two data points $\overline{x}_i$ and $\overline{x}_j$ be denoted by $D(\overline{x}_i, \overline{x}_j)$. Then this distance function should satisfy the following properties:

1. $D(\overline{x}_i, \overline{x}_j) \geq 0 \ \forall \overline{x}_i, \overline{x}_j \in \mathbf{X}$, and $D(\overline{x}_i, \overline{x}_j) = 0$ only if $i = j$.
2. $D(\overline{x}_i, \overline{x}_j) = D(\overline{x}_j, \overline{x}_i) \ \forall \overline{x}_i, \overline{x}_j \in \mathbf{X}$.
3. $D(\overline{x}_i, \overline{x}_j) \leq D(\overline{x}_i, \overline{x}_l) + D(\overline{x}_l, \overline{x}_j) \ \forall \overline{x}_i, \overline{x}_j, \overline{x}_l \in \mathbf{X}$.

Several measures have been employed in the literature for clustering [24, 143, 296]. One commonly used measure of similarity is the Euclidean distance D between two patterns $\overline{x}$ and $\overline{z}$, defined by $D = \|\overline{x} - \overline{z}\|$. Smaller Euclidean distance

means better similarity, and vice versa. This measure has been used in the K-means clustering algorithm [143] in which hyperspherical clusters of almost equal sizes can be easily identified. This measure fails when clusters tend to develop along principal axes. In order to detect hyperellipsoidal-shaped clusters from data sets, the Mahalanobis distance from $\overline{x}$ to $\overline{m}$, $D(\overline{x}, \overline{m}) = (\overline{x} - \overline{m})^T \sum^{-1} (\overline{x} - \overline{m})$, is commonly used [185]. Here $\overline{x}$ represents an input pattern, the matrix $\sum$ is the covariance matrix of a pattern population constituting a particular cluster, and $\overline{m}$ is the mean vector of the vectors which are in the same cluster. A disadvantage of the Mahalanobis distance as a similarity measure is that one has to recompute the inverse of the sample covariance matrix every time a pattern changes its cluster domain [266]. This is a computationally expensive task.

Each measure has its own advantages and disadvantages that make it more or less suitable to a given domain or application area such as bioinformatics, text clustering or document categorization.

Symmetry is considered as an important feature in recognition and reconstruction of shapes and objects [12, 266]. Almost every interesting area around us exhibits some generalized form of symmetry. As symmetry is so common in the natural world, it can be assumed that some kind of symmetry exists in the clusters also. Based on this idea, some symmetry-based similarity measurements and clustering algorithms have been developed [58, 60, 61, 178, 266]. A detailed discussion on symmetry-based clustering techniques is provided in Chap. 5.

1.3.6 Model Selection

Model selection for a data set is basically selection of an appropriate clustering technique suitable for a particular data set. Clusters may be of different shapes. Model selection in clustering consists of two steps. In the first step, a proper clustering method for a particular data set has to be identified. Thereafter, the model order remains to be determined for the given data set in the second step.

Clusters may have different shapes such as:

- Hyperspherical-shaped clusters as shown in Fig. 5.9(b).
- Ring-shaped clusters as shown in Fig. 5.9(a).
- Linear clusters as shown in Fig. 9.9(b).
- Concentric clusters: Here clusters form some concentric spheres, as shown in Fig. 9.9(c).

1.3.7 Model Order Selection

Model order selection means selection of an appropriate number of clusters from data sets. The task of determining the number of clusters and also the validity of the clusters formed [24, 189] is generally addressed by providing several definitions of

validity indices. The measure of validity of clusters should be such that it will be able to impose an ordering on the clusters in terms of their goodness. In other words, if $U_1, U_2, \ldots, U_m$ are the m partitions of X, and the corresponding values of a validity measure are $V_1, V_2, \ldots, V_m$, then $V_{k1} \geq V_{k2} \geq \cdots \geq V_{km}$, $\forall ki \in 1, 2, \ldots, m$, $i = 1, 2, \ldots, m$ will indicate that $U_{k1} \uparrow \cdots \uparrow U_{km}$. Here '$U_i \uparrow U_j$' indicates that partition U_i is a better clustering than U_j. Note that a validity measure may also define a decreasing sequence instead of an increasing sequence of $V_{k1}, \ldots, V_{km}$. Several cluster validity indices have been proposed in the literature, e.g., the Davies-Bouldin (DB) index [73], Dunn's index [87], Xie-Beni (XB) index [295], I-index [189], CS-index [59], XB*-index [156], the index proposed in [155, 157], fuzzy cluster validity indices proposed in [290, 305] etc., to name just a few. A good review of cluster validity indices and their categorization can be found in [156]. Some of these indices have been found to be able to detect the correct partitioning for a given number of clusters, while some can determine the appropriate number of clusters as well. Milligan and Cooper [200] have provided a comparison of several validity indices for data sets containing distinct non-overlapping clusters while using only hierarchical clustering algorithms. Maulik and Bandyopadhyay [189] evaluated the performance of four validity indices, namely the Davies-Bouldin index [73], Dunn's index [87], Calinski-Harabasz index [189], and a recently developed index I, in conjunction with three different algorithms, viz. the well-known K-means [96], single linkage algorithm [96], and an SA-based clustering method [189].

1.4 Robustness to Outliers and Missing Values

Given a dataset, an outlier is a data point which is numerically distant from the rest of the data. Grubbs [118] has defined an outlier as follows: "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". Outliers can occur in any given data set due to some measurement error. In general in the data preprocessing step these outliers are discarded from the data set. Outliers are the most extreme observations; thus they are sometimes the sample maximum or sample minimum or both depending on their locations. Outliers can be included in the data set because of some error while measuring the data points using some probability distribution. In order to remove the outliers from the data set, first we need to identify outliers from a given data set. There are three different ways to detect outliers:

- Using an unsupervised approach: These approaches do not require any knowledge about the data set. This is very similar to unsupervised clustering. This technique considers the data as a static distribution and determines the most remote points, which are regarded as the potential outliers.
- Using a supervised approach: These approaches use some classification techniques to identify which points are normal and which points are abnormal/outliers. This approach needs some training, which are sometimes difficult to obtain.

- Using a semi supervised approach: These approaches use some semi supervised classification techniques. Unsupervised classification can be used to identify the normal data points, but supervised classification can be used to identify outliers.

In pattern recognition, missing data, or missing values, occur when no value is stored for a particular data point at a particular feature position or dimension. Missing values are unfortunate but occur frequently in many practical situations. These can have a significant effect on the decisions that can be made based on the data. For example, in case of medical studies, some tests may be performed for some patients but not others. Generally, missing values are specified in the input data in three different ways: leaving the corresponding column blank, putting a single period ('.') in the column without any numbers around it or by putting a question mark ('?') in the column.[1]

There are various ways of dealing with data sets having missing values. Some of them are listed below:

- Exclude the data row: The easiest way to deal with data having some missing values is to exclude those points from the analysis. If the number of available data points is large and the percentage of points with missing values is small, then this may be the best method. This is a fast technique and prevents any error from being introduced due to the missing values.
- Replace missing values with median/mode values: In this approach missing feature values are replaced by the median value of that particular feature over the entire data set. The mode (most frequent category) is used for replacement in case of categorical variables. Use of the median/mode introduces some error into the model for that variable, but it helps the non-missing values of the other features to contribute to the model.
- Treat missing data as just another category. Suppose there is a variable called Religion which is defined as follows: 1 = Hindu, 2 = Muslim, 3 = Christian, and 4 = Other. Suppose that, while taking a survey, some people fail to answer this question. Rather than just exclude these subjects, we can just add a fifth category, 5 = Missing Data (or no response).

1.5 Fuzzy Set-Theoretic Approach: Relevance and Features

Fuzzy set theory deals with the uncertainty present in a data set. Uncertainty can arise either implicitly or explicitly in each and every phase of a pattern recognition system [24]. It results from incomplete, imprecise or ambiguous input information, ill-defined and/or overlapping boundaries among classes or regions, and indefiniteness in defining/extracting features and relations among them. Any decision taken at a particular level will have an impact on all other higher-level activities. It is therefore required for a pattern recognition system to have sufficient provision for

[1] http://www.dtreg.com/MissingValues.htm

representing these uncertainties involved at every stage, so that the ultimate output of the system can be obtained with the least uncertainty.

Fuzzy sets were introduced in 1965 by Zadeh [301] as a way to represent vagueness in everyday life. Since this theory is a generalization of classical set theory, it has greater flexibility to capture various aspects of incompleteness, impreciseness or imperfection in information about a situation.

The relevance of fuzzy set theory in the realm of pattern recognition [37, 39, 149, 150, 215, 216, 220, 221] is adequately justified in:

- Representing input patterns as an array of membership values denoting the degree of possession of certain properties.
- Representing linguistically phrased input features for processing.
- Providing an estimate (representation) of missing information in terms of membership values.
- Representing multiclass membership of ambiguous patterns and in generating rules and inferences in linguistic form.
- Extracting ill-defined image regions, primitives, and properties and describing relations among them as fuzzy subsets.

1.6 Applications of Pattern Recognition and Learning

Pattern recognition research is mostly driven by the need to process data and information obtained from the interaction between humans, society, science, and technology. As already mentioned, in order to make machines more intelligent and human like, they must possess automatic pattern recognition and learning capabilities. Some of the typical application areas of such intelligent machines are:

- *Medicine*: Medical diagnosis, image analysis, disease classification.
- *Domestic systems*: Appliances.
- *Space science*: Mars rover control, camera tracking.
- *Computation biology*: Gene identification, protein modeling, rational drug design.
- *Biometry*: Face recognition, gesture recognition, emotion detection, optical character recognition.
- *Natural resource study and estimation*: Agriculture, forestry, geology, the environment.
- *Natural language processing*: Detection of named entities from texts, determining part-of-speech tags, event identification, machine translation, machine transliteration, emotion analysis, sentiment analysis.
- *Human-machine communication*: Automatic speech recognition and understanding, image processing, script recognition, signature analysis.
- *Security*: Security attack detection, e-security, steganography.
- *Defense*: Automatic target recognition, guidance and control.
- *Electrical engineering*: Design of analog and mixed-signal integrated circuits (ICs), design of some analog circuits.

- *Design automation*: Automatically driven cars and planes.
- *Vehicular*: Automobile, airplane, train, and boat controllers.
- *Stock market prediction*: Forecasting financial markets.
- *VLSI design*: VLSI design, layout, and test automation, placement and area optimization, noise reduction.
- *Remote sensing*: Detection of manmade objects and estimation of natural resources.
- *Networks*: Wireless sensor network, intrusion detection, network design and planning, network security.
- *Police and detective*: Crime and criminal detection from analysis of speech, handwriting, fingerprints, photographs.

1.7 Summary and Scope of the Book

Pattern recognition and learning are two important issues that need to be addressed for making machines that can emulate human behavior. Clustering is an important component of pattern recognition. It deals with classification of some given data points by considering only the data distribution. For this purpose no labeled information is required. In this chapter, an introduction to pattern recognition has been provided. The different tasks are described briefly.

In the past, several clustering algorithms have been developed. In this book some recent metaheuristic optimization techniques for solving clustering problems are discussed. Clustering can be considered as an optimization problem whose focus is to optimize some cluster quality measure. The problem of clustering requires appropriate parameter selection (e.g., model and model order) and efficient search in complex and large spaces in order to attain optimal solutions. Moreover, defining a single measure of optimality is often difficult in clustering, leading to the need to define multiple objectives that are to be simultaneously optimized. This not only makes the process computationally intensive, but also leads to the possibility of missing the exact solution. Therefore, the application of sophisticated metaheuristic optimization techniques, both single and multiobjective, that provide robust, fast, and close approximate solutions, seems appropriate and natural.

Chapter 2 provides a detailed discussion on optimization techniques. The different steps of two single-objective optimization techniques, i.e.,genetic algorithms (GAs) and simulated annealing (SA) are described in detail. The basic concepts behind multiobjective optimization are also elaborately described in this chapter. A discussion on multiobjective GAs is provided in a part of this chapter. Finally, a recently developed multiobjective simulated annealing-based technique, AMOSA, is described in detail.

Chapter 3 provides a detailed discussion on several similarity measurements used for clustering a given data. For different types of data points, different similarity measurements are available. Some primarily used measures of similarity for binary, categorical, ordinal, and quantitative variables are described in detail in this chapter.

Before application of some clustering or classification techniques, sometimes it is required to normalize the data. Different normalization techniques are also described in detail in this chapter.

Chapter 4 deals with the basic concepts related to clustering. The traditional clustering algorithms are described here. This is followed by descriptions of some evolutionary clustering techniques along with some multiobjective clustering techniques.

Chapter 5 deals with some symmetry-based clustering techniques. Some recently developed symmetry-based similarity measurements are described in detail. The Kd-tree approach [4] has been used to reduce the computational cost of symmetry-based distance calculation. Thereafter, the problem of clustering a data set is formulated as one of optimization of the total symmetry of a partitioning. Genetic algorithm is used to solve this optimization problem. This yields a new clustering technique named the genetic algorithm with point symmetry-based clustering technique (GAPS) [27] which is able to detect any type of clusters possessing the property of point symmetry. The global convergence property of the proposed GAPS clustering is also established. Detailed comparative results on artificial and real-life data sets are provided. Time complexities of these clustering algorithms are also described in detail.

Chapter 6 deals with cluster validity indices. The definition of cluster validity index is provided. This is followed by an elaborate description of some existing validity indices. Thereafter, some newly developed symmetry-based cluster validity indices are elaborately described. Experimental results are provided to support that the newly developed symmetry-based *Sym*-index is not only able to find the proper number of clusters from a given data set but is also able to detect the proper clustering algorithm suitable for that data set. Some mathematical analysis of this *Sym*-index is also provided. Elaborate experimental results are provided, comparing the performance of this newly developed index with several existing cluster validity indices. The concept of point symmetry is thereafter incorporated into several well-known cluster validity indices [244]. Point symmetry versions of eight cluster validity indices are developed to mimic eight existing cluster validity indices. These indices exploit the property of point symmetry to indicate both the appropriate number of clusters as well as the appropriate partitioning. The effectiveness of these indices in comparison with *Sym*-index and eight existing cluster validity indices are provided for two artificially generated and three real-life data sets. Results show that incorporation of point symmetry distance in the definitions of the existing eight cluster validity indices makes them more effective in determining the proper number of clusters and the appropriate partitioning from data sets having clusters of different shapes and sizes as long as they possess the property of point symmetry. Finally, the application of the newly proposed symmetry-based cluster validity index, *Sym*-index, and GAPS-clustering technique is described for image segmentation [243]. Its effectiveness, vis-à-vis, other well-known validity indices, is first established for segmenting an artificially generated image. Thereafter, it is used for classifying the different land covers in a multispectral satellite image.

Chapter 7 deals with some automatic clustering techniques which can automatically determine the appropriate number of clusters and the appropriate partitioning

from a data set having symmetrically shaped clusters [28]. The global convergence property of these clustering techniques is also established. Elaborate comparative results are provided to show the effectiveness of these automatic clustering techniques. Results establish the fact that this symmetry-based automatic clustering is well suited to detect the number of clusters and the proper partitioning from data sets having clusters of widely varying characteristics, irrespective of their convexity, overlap or size, as long as they possess the property of symmetry. The second part of this chapter describes a fuzzy automatic clustering technique. Its effectiveness is shown for automatically partitioning multispectral resonance images of the human brain.

Chapter 8 deals with some line symmetry-based clustering techniques. First the definitions of some line symmetry-based distances are provided. This is followed by the elaborate description of some line symmetry-based clustering techniques.

As already mentioned, for appropriate clustering it often becomes necessary to simultaneously optimize several cluster quality measures that can capture different data characteristics. In order to achieve this, in Chap. 9 the problem of clustering a data set into a fixed number of clusters is posed as one of multiobjective optimization (MOO), where search is performed over a number of objective functions [239]. The simulated annealing-based multiobjective optimization technique AMOSA is used as the underlying optimization technique. Several different variations of these MOO-based clustering algorithms are described. Elaborate experimental results are provided.

Chapter 2
Some Single- and Multiobjective Optimization Techniques

2.1 Introduction

Optimization deals with the study of those kinds of problems in which one has to minimize or maximize one or more objectives that are functions of some real or integer variables. This is executed in a systematic way by choosing the proper values of real or integer variables within an allowed set. Given a defined domain, the main goal of optimization is to study the means of obtaining the best value of some objective function. An optimization problem [210] can be defined as follows:

Given a function $f : S \rightarrow R$ from some set S to the set of real numbers, the aim is to determine an element $\overline{x}_0$ in S such that $f(\overline{x}_0) \leq f(\overline{x}), \forall \overline{x} \in S$ ("minimization") or such that $f(\overline{x}_0) \geq f(\overline{x}), \forall \overline{x} \in S$ ("maximization").

Here, S denotes a subset of the Euclidean space R^n which is a collection of entities such as constraints, equalities or inequalities. The members of S should satisfy these entities. S, the domain of f, is called the search space, and the elements of S are called candidate or feasible solutions. The function f is called an objective function/cost function/energy function. A feasible solution that optimizes the objective function is called an optimal solution.

Multiobjective optimization (MOO) [77] (multicriteria or multiattribute optimization) deals with the task of simultaneously optimizing two or more conflicting objectives with respect to a set of certain constraints. If the optimization of one objective leads to the automatic optimization of the other, it should not be considered as MOO problem. However, in many real-life situations we come across problems where an attempt to improve one objective leads to degradation of the other. Such problems belong to the class of MOO problems and appear in several fields including product and process design, network analysis, finance, aircraft design, bioinformatics, the oil and gas industry, automobile design, etc.

In this chapter, some existing single- and multiobjective optimization techniques are first described. Thereafter, a newly developed simulated annealing-based multiobjective optimization technique named the archived multiobjective simulated annealing-based optimization technique (AMOSA) [29] is elaborately presented.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_2,

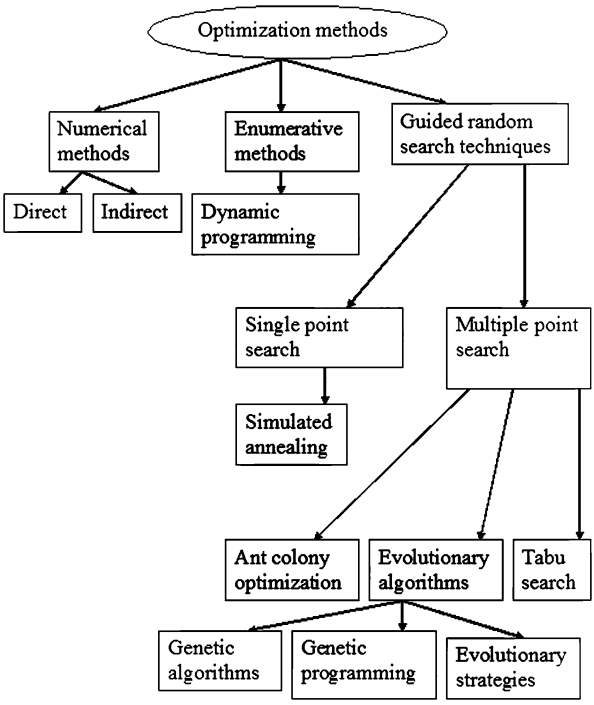


Fig. 2.1 The different search and optimization techniques

2.2 Single-Objective Optimization Techniques

Different optimization techniques that are found in the literature can be broadly classified into three categories (Fig. 2.1) [24, 112]:

- Calculus-based techniques.
- Enumerative techniques.
- Random techniques.

Numerical methods, also called calculus-based methods, use a set of necessary and sufficient conditions that must be satisfied by the solution of the optimization problem [24, 112]. They can be further subdivided into two categories, viz. direct and indirect methods. Direct search methods perform hill climbing in the function space by moving in a direction related to the local gradient. In indirect methods, the solution is sought by solving a set of equations resulting from setting the gradient of the objective function to zero. The calculus-based methods are local in scope and also assume the existence of derivatives. These constraints severely restrict their application to many real-life problems, although they can be very efficient in a small class of unimodal problems.

Enumerative techniques involve evaluating each and every point of the finite, or discretized infinite, search space in order to arrive at the optimal solution [24, 112]. Dynamic programming is a well-known example of enumerative search. It is obvious that enumerative techniques will break down on problems of even moderate

size and complexity because it may become simply impossible to search all the points in the space.

Guided random search techniques are based on enumerative methods, but they use additional information about the search space to guide the search to potential regions of the search space [24, 112]. These can be further divided into two categories, namely single-point search and multiple-point search, depending on whether it is searching just with one point or with several points at a time. Simulated annealing is a popular example of a single-point search technique that uses thermodynamic evolution to search for the minimum-energy states. Evolutionary algorithms such as genetic algorithms are popular examples of multiple-point search, where random choice is used as a tool to guide a highly explorative search through a coding of the parameter space. The guided random search methods are useful in problems where the search space is huge, multimodal, and discontinuous, and where a near-optimal solution is acceptable. These are robust schemes, and they usually provide near-optimal solutions across a wide spectrum of problems. In the remaining part of this chapter, we focus on such methods of optimization for both single and multiple objectives.

2.2.1 Overview of Genetic Algorithms

Genetic algorithms (GAs), which are efficient, adaptive, and robust search and optimization processes, use guided random choice as a tool for guiding the search in very large, complex, and multimodal search spaces. GAs are modeled on the principles of natural genetic systems, where the genetic information of each individual or potential solution is encoded in structures called *chromosomes*. They use some domain- or problem-dependent knowledge to direct the search to more promising areas; this is known as the *fitness function*. Each individual or chromosome has an associated fitness function, which indicates its degree of goodness with respect to the solution it represents. Various biologically inspired operators such as *selection, crossover*, and *mutation* are applied to the chromosomes to yield potentially better solutions.

Note that the classical gradient search techniques perform efficiently when the problems under consideration satisfy tight constraints. However, when the search space is discontinuous and/or huge in size, noisy, high dimensional, and multimodal, GAs have been found to consistently outperform both the gradient descent method and various forms of random search [24, 116]. The below discussion of GA is taken from [24].

2.2.1.1 Genetic Algorithms: Basic Principles and Features

Genetic algorithms (GAs) [75, 112, 197] are adaptive computational procedures modeled on the mechanics of natural genetic systems. They efficiently exploit historical information to speculate on new offspring with improved performance [112].

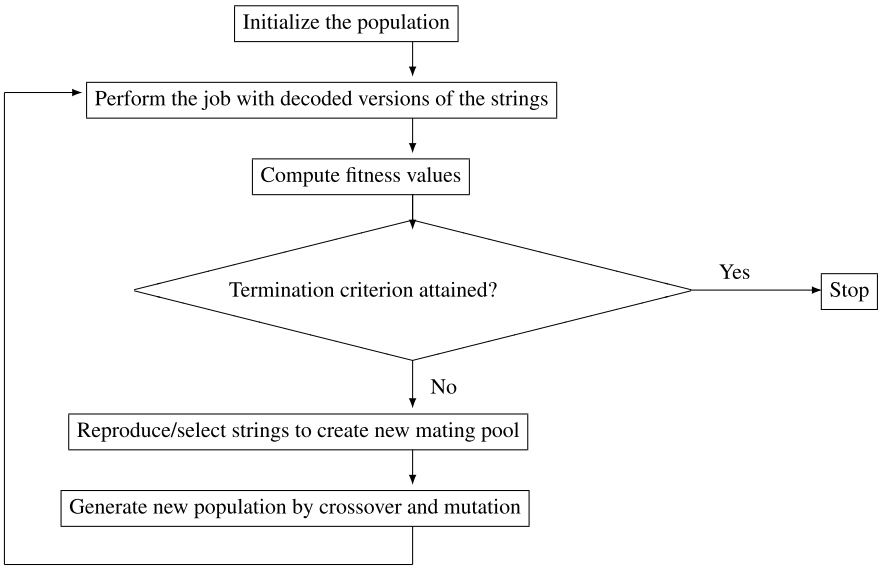


Fig. 2.2 Basic steps of a genetic algorithm

As mentioned before, GAs encode the parameters of the search space in structures called a *chromosomes* (or *strings*). They execute iteratively on a set of chromosomes, called *population*, with three basic operators: *selection/reproduction*, *crossover*, and *mutation*. GAs are different from most of the normal optimization and search procedures in four ways [112]:

- GAs work with a coding of the parameter set, not with the parameters themselves.
- GAs work simultaneously with multiple points, and not with a single point.
- GAs search via sampling (blind search) using only the payoff information.
- GAs search using stochastic operators, not deterministic rules, to generate new solutions.

Since a GA works simultaneously on a set of coded solutions, it has very little chance of getting stuck at a local optimum when used as an optimization technique. Again, the search space need not be continuous, and no auxiliary information, such as derivatives of the optimizing function, is required. Moreover, the resolution of the possible search space is increased by operating on coded (possible) solutions and not on the solutions themselves.

A schematic diagram of the basic structure of a genetic algorithm is shown in Fig. 2.2. The evolution starts from a set of chromosomes (representing a potential solution set for the function to be optimized) and proceeds from generation to generation through genetic operations. Replacement of an old population with a new one is known as a generation (or iteration) when the *generational replacement technique* (where all the members of the old population are replaced with the new ones) is used. Another population replacement technique, called *steady-state reproduc-*

tion, may be used, where one or more individuals are replaced at a time, instead of the whole population [75]. GAs require only a suitable objective function, which is a mapping from the chromosomal space to the solution space, in order to evaluate the suitability or *fitness* of the derived solutions.

A GA typically consists of the following components:

- A population of binary strings or coded possible solutions (biologically referred to as *chromosomes*).
- A mechanism to encode a possible solution (mostly as a binary string).
- An objective function and associated fitness evaluation techniques.
- A selection/reproduction procedure.
- Genetic operators (*crossover* and *mutation*).
- Probabilities to perform genetic operations.

These components are now briefly described.

Population To solve an optimization problem, GAs start with the chromosomal representation of a parameter set. The parameter set is to be coded as a finite-length string over an alphabet of finite length. Usually, the chromosomes are strings of 0s and 1s. For example, let $\{a_1, a_2, \ldots, a_p\}$ be a realization of the set of p parameters, and the binary representation of $a_1, a_2, \ldots, a_p$ be $10110, 00100, \ldots, 11001$, respectively. Then the string

$$10110\,00100\ \ldots\ 11001$$

is a chromosomal representation of the parameter set. It is evident that the number of different chromosomes (or strings) is 2^l, where l is the string length. Each chromosome actually refers to a coded possible solution. A set of such chromosomes in a generation is called a *population*, the size of which may be constant or may vary from one generation to another. A common practice is to choose the initial population randomly.

Encoding/Decoding Mechanism This is one of the primary tasks in GAs. It is the mechanism of converting the parameter values of a possible solution into strings, resulting in the chromosomal representation. If the solution of a problem depends on p parameters and if we want to encode each parameter with a string of length q, then the length, l, of each chromosome will be

$$l = p * q.$$

Decoding is the task of retrieving the parameter values from the chromosomes. It proceeds in a manner that is just the reverse of the encoding process.

One commonly used principle for coding is known as the *principle of minimum alphabet* [112]. It states that, for efficient coding, the smallest alphabet set that permits a natural expression of the problem should be chosen. In general, it has been found that the binary alphabet offers the maximum number of schemata per bit of information of any coding [112]. Hence, binary encoding is one of the commonly used strategies, although other techniques such as floating-point coding [79, 197] are also popular.

Objective Function and Associated Fitness Evaluation Techniques The fitness/objective function is chosen depending on the problem to be solved, in such a way that the strings (possible solutions) representing good points in the search space have high fitness values. This is the only information (also known as the payoff information) that GAs use while searching for possible solutions.

Selection/Reproduction Procedure The selection/reproduction process copies individual strings (called parent chromosomes) into a tentative new population (known as a mating pool) for genetic operations. The number of copies of an individual included in the next generation is usually taken to be directly proportional to its fitness value; thereby mimicking the natural selection procedure to some extent. This scheme is commonly called the *proportional selection scheme*. *Roulette wheel parent selection* [112] and *linear selection* [75] are two of the most frequently used selection procedures.

As proved in [236], one problem with *proportional selection* is that this procedure cannot guarantee asymptotic convergence to the global optima. There is no assurance that any improvement made up to a given generation will be retained in future generations. To overcome this, a commonly used strategy known as *elitist selection* [116] is adopted, thereby providing an *elitist GA* (EGA), where the best chromosome of the current generation is retained in the next generation.

Genetic operators are applied to parent chromosomes, and new chromosomes (also called offspring) are generated. The frequently used genetic operators are described below.

Crossover The main purpose of crossover is to exchange information between randomly selected parent chromosomes by recombining parts of their corresponding strings. It recombines genetic material of two parent chromosomes to produce offspring for the next generation. *Single-point crossover* is one of the most commonly used schemes. Here, first of all, the members of the reproduced strings in the mating pool are paired at random. Then an integer position k (known as the crossover point) is selected uniformly at random between 1 and $l-1$, where l is the string length greater than 1. Two new strings are created by swapping all characters from position $(k+1)$ to l. For example, let

$$a = 11000\ 10101\ 01000\ \ldots\ 01111\ 10001,$$

$$b = 10001\ 01110\ 11101\ \ldots\ 00110\ 10100$$

be two strings (parents) selected from the mating pool for crossover. Let the randomly generated crossover point be 11 (eleven). Then the newly produced offspring (swapping all characters after position 11) will be

$$a' = 11000\ 10101\ 01101\ \ldots\ 00110\ 10100,$$

$$b' = 10001\ 01110\ 11000\ \ldots\ 01111\ 10001.$$

Some other common crossover techniques are multiple-point crossover, shuffle-exchange crossover, and uniform crossover [75].

Mutation The main aim of mutation is to introduce genetic diversity into the population. Sometimes, this helps to regain information lost in earlier generations. In case of binary representation, it negates the bit value and is known as bit mutation. Like natural genetic systems, mutation in GAs is usually performed occasionally. A random bit position of a randomly selected string is replaced by another character from the alphabet. For example, let the third bit of string a, given above, be selected for mutation. Then the transformed string after mutation will be

$$11100\ 10101\ 01000\ \ldots\ 01111\ 10001.$$

High mutation rate can lead the genetic search to a random one. It may change the value of an important bit and thereby affect the fast convergence to a good solution. Moreover, it may slow down the process of convergence at the final stage of GAs.

Probabilities to Perform Genetic Operations Both the crossover and mutation operations are performed stochastically. The probability of crossover is chosen in a way so that recombination of potential strings (highly fit chromosomes) increases without any disruption. Generally, the crossover probability lies between 0.6 and 0.9 [75, 112]. Since mutation occurs occasionally, it is clear that the probability of performing a mutation operation will be very low. Typically the value lies between $1/l$ and 0.1 [75, 112].

As shown in Fig. 2.2, the cycle of selection, crossover, and mutation is repeated a number of times till one of the following occurs:

1. The average fitness value of a population becomes more or less constant over a specified number of generations.
2. A desired objective function value is attained by at least one string in the population.
3. The number of generations (or iterations) is greater than some threshold.

2.2.2 Simulated Annealing

Simulated annealing (SA) [159] is another popular search algorithm which utilizes the principles of statistical mechanics regarding the behavior of a large number of atoms at low temperature for finding minimal cost solutions to large optimization problems by minimizing the associated energy. In statistical mechanics, investigating the ground states or low-energy states of matter is of fundamental importance. These states are achieved at very low temperatures. However, it is not sufficient to lower the temperature alone, since this results in unstable states. In the annealing process, the temperature is first raised, then decreased gradually to a very low value (*Tmin*), while ensuring that one spends sufficient time at each temperature value. This process yields stable low-energy states. SA has been applied in diverse areas [22, 52, 191] by optimizing a single criterion. The below discussion of SA is taken from [26].

2.2.2.1 Basic Principle

In statistical mechanics, if a system is in thermal equilibrium, the probability $\pi_T(s)$ that the system is in state s, $s \in S$, S being the state space, at temperature T, is given by

$$\pi_T(s) = \frac{e^{\frac{-E(s)}{kT}}}{\sum_{w \in S} e^{\frac{-E(w)}{kT}}}, \tag{2.1}$$

where k is Boltzmann's constant and $E(s)$ is the energy of the system in state s.

Metropolis [195] developed a technique to simulate the behavior of a system in thermal equilibrium at temperature T as follows: Let the system be in state q at time t. Then the probability p that it will be in state s at time $t+1$ is given by the equation

$$p = \frac{\pi_T(s)}{\pi_T(q)} = e^{\frac{-(E(s)-E(q))}{kT}}. \tag{2.2}$$

If the energy of the system in state s is less than that in state q, then $p > 1$ and the state s is automatically accepted. Otherwise, it is accepted with probability p. Thus, it is also possible to attain higher energy values. It can be shown that, for $T \to \infty$, the probability that the system is in state s is given by $\pi_T(s)$ irrespective of the starting configuration [111].

2.2.2.2 Annealing Schedule

When dealing with a system of particles, it is important to investigate very low-energy states, which predominate at extremely low temperatures. To achieve such states, it is not sufficient to lower the temperature. An annealing schedule is used, where the temperature is first increased and then decreased gradually, spending enough time at each temperature in order to reach thermal equilibrium.

The annealing process of the Boltzmann machine is often used for this purpose, being a variant of the Metropolis algorithm. Here, at a given temperature T, the new state is chosen with probability

$$p_{qs} = \frac{1}{1 + e^{\frac{-(E(q,T)-E(s,T))}{T}}}. \tag{2.3}$$

2.2.2.3 Algorithm

In SAs, a configuration or a state represents a potential solution of the problem in hand. The objective value associated with the state is computed, which is analogous to the fitness computation in GA, and mapped to its energy. The state with the minimum energy value provides the solution to the problem. The initial state (say q) is generated randomly, and its energy value is computed. Keeping the initial

Fig. 2.3 Steps of simulated annealing

```
Begin
  Generate the initial state q
  T = T_max
  Let E(q, T) be the associated energy
  while (T ≥ T_min)
    for i = 1 to k
      Perturb q to yield s
      Let E(s, T) be the associated energy
      Set q ← s with probability 1/(1+e^{-(E(q,T)-E(s,T))/T})
    end for
    T = rT
  end while
Decode q to provide the solution of the problem.
End
```

temperature high (say $T = T_{max}$), a neighbor of the current state (say s) is generated. As an example, consider the states to be represented by binary strings. Then the operation of flipping a randomly selected bit can be used to generate a possible neighbor. The energy of the new state is computed, and it is accepted in favor of q with the probability p_{qs} mentioned earlier. This process is repeated a number of times (say k) keeping the temperature constant. Then the temperature is decreased using the equation $T = rT$, where $0 < r < 1$, and the k loops, as earlier, are executed. This process is continued till a minimum temperature (say T_{min}) is attained. The simulated annealing steps are shown in Fig. 2.3.

Simulated annealing has been successfully applied in various domains [74] including computer design [159, 160], image restoration and segmentation [259], contour detection [40, 52, 191], edge detection [171], combinatorial problems such as the traveling salesman problem [158], and artificial intelligence [22, 26]. It is, however, not always trivial to map an optimization problem into the simulated annealing framework. The difficulties come from constructing an objective function that encapsulates the essential properties of the problem and that can be efficiently evaluated. It is necessary to determine a concise description of the parameter configurations as well as an efficient method for generating configurations. Moreover, it is important to select an effective and efficient annealing schedule.

2.3 Multiobjective Optimization

In our everyday life, we make decisions consciously or unconsciously. These decisions can be very simple, such as selecting the color of a dress or deciding the menu for lunch, or may be as difficult as those involved in designing a missile or in selecting a career. The former decisions are easy to make, while the latter might take several years due to the level of complexity involved. The main goal of most kinds of decision-making is to optimize one or more criteria in order to achieve the desired result. In other words, problems related to optimization abound in real life.

The development of optimization algorithms has therefore been a great challenge in computer science. The problem is compounded by the fact that in many situations one may need to optimize several objectives simultaneously. These specific problems are known as multiobjective optimization problems (MOOPs). In this regard, a multitude of metaheuristic single-objective optimization techniques such as genetic algorithms, simulated annealing, differential evolution, and their multiobjective versions have been developed.

2.3.1 Multiobjective Optimization Problems

We encounter numerous real-life scenarios where multiple objectives need to be satisfied in the course of optimization. Finding a single solution in such cases is very difficult, if not impossible. In such problems, referred to as multiobjective optimization problems (MOOPs), it may also happen that optimizing one objective leads to some unacceptably low value of the other objective(s). For an in-depth discussion on MOOPs, the reader is referred to [62, 77]. Some definitions and basic concepts related to MOOPs are given below.

2.3.1.1 Formal Definition of MOOPs

Multiobjective optimization (MOO) can be formally stated as follows[62, 77]: Find the vector $\overline{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ of decision variables which will satisfy the m inequality constraints

$$g_i(\overline{x}) \geq 0, \quad i = 1, 2, \ldots, m, \tag{2.4}$$

the p equality constraints

$$h_i(\overline{x}) = 0, \quad i = 1, 2, \ldots, p, \tag{2.5}$$

and simultaneously optimize the M objective values

$$f_1(\overline{x}), \quad f_2(\overline{x}), \quad \ldots, \quad f_M(\overline{x}). \tag{2.6}$$

The constraints given in Eqs. 2.4 and 2.5 define the feasible region $\mathscr{F}$ which contains all the admissible solutions. Any solution outside this region is inadmissible since it violates one or more constraints. The vector $\overline{x}^*$ denotes an optimal solution in $\mathscr{F}$. In the context of multiobjective optimization, the difficulty lies in the definition of optimality, since it is only rarely that a situation can be found where a single vector $\overline{x}^*$ represents the optimum solution to all the M objective functions.

2.3.1.2 Dominance Relation and Pareto Optimality

An important concept of multiobjective optimization is that of domination. Below, a formal definition of domination is given in the context of maximization problems [77]. The definition is easily extended to minimization problems.

A solution $\overline{x}_i$ is said to dominate $\overline{x}_j$ if

$$\forall k \in 1, 2, \ldots, M, \quad f_k(\overline{x}_i) \geq f_k(\overline{x}_j) \quad \text{and}$$
$$\exists k \in 1, 2, \ldots, M \text{ such that } \quad f_k(\overline{x}_i) > f_k(\overline{x}_j).$$

This concept can be explained using a two-objective optimization problem that has five different solutions, as shown in Fig. 2.4 (example taken from [77]). Let us assume that the objective function f1 needs to be maximized while f2 needs to be minimized. Five solutions having different values of the objective functions are shown. Evidently, solution 1 dominates solution 2 since the former is better than the latter on both objectives. Again solution 5 dominates 1, but 5 and 3 do not dominate each other. Intuitively, we can say that, if a solution 'a' dominates another solution 'b', then the solution 'a' is better than 'b' in the parlance of multiobjective optimization. Thus, the concept of domination allows us to compare different solutions with multiple objectives. It may be noted that the dominance relation is irreflexive, asymmetric, and transitive in nature.

Assume a set of solutions P. The solutions of P that are not dominated by any other solution in P form the nondominated set [77]. The rank of a solution $\overline{x}$ in P is defined as the number of solutions in P that dominate $\overline{x}$ [77]. In Fig. 2.4, solutions 3 and 5 are in the nondominated set, and their ranks are 0. The non-dominated set of the entire search space S is the globally Pareto-optimal set [77]. Below, some properties of the dominance relation are mentioned.

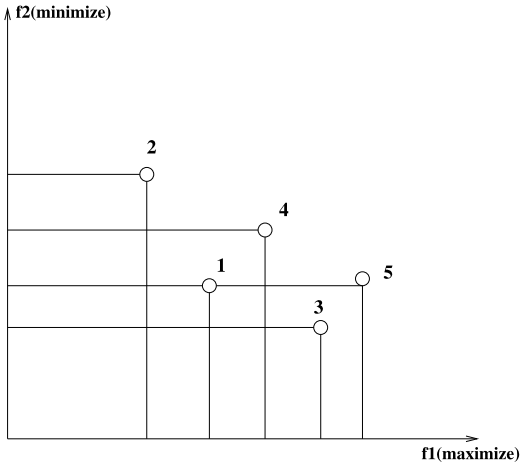


Fig. 2.4 Example of dominance and Pareto optimality

Properties of Dominance Relation [77] *Reflexive*: *The dominance relation is not reflexive, since a solution p does not dominate itself. The second condition of the definition is not satisfied in this case.*

Symmetric: *The dominance relation is not symmetric, because if a dominates b, this does not imply that b dominates a. Actually the opposite is true. Thus, the dominance relation is asymmetric.*

Antisymmetric: *Since the dominance relation is not symmetric, it cannot be antisymmetric.*

Transitive: *The dominance relation is transitive. This is because, if p dominates q and q dominates r, then p dominates r. Another interesting property that the dominance relation possesses is that, if a solution p does not dominate solution q, this does not imply that q dominates p.*

2.3.1.3 Performance Measures

In order to evaluate the performance of different MOO algorithms, several measures have been proposed in the literature. Some such measures are the convergence measure γ [78], *Purity* [25], *Spacing* [250], and *Minimal Spacing* [25]. The convergence measure γ indicates how close an obtained nondominated front is from the true Pareto-optimal front. The *Purity* of a MOO algorithm measures the fraction of the nondominated solutions that remain nondominated with respect to the solutions obtained using several other MOO techniques. *Spacing* and *Minimal Spacing* measure the diversity of the solutions in the final nondominated set. More details about these measures appear later on in this chapter while reporting the comparative performance of different MOO algorithms. Apart from these, there are many other measures in the literature. Details can be found in [63, 77].

2.3.2 Various Methods to Solve MOOPs

A large number of approaches exist in the literature to solve multiobjective optimization problems [62, 77, 193]. These are aggregating, population-based non-Pareto- and Pareto-based techniques. In case of aggregating techniques, different objectives are generally combined into one using weighting or a goal-based method. One of the techniques in the population-based non-Pareto approach is the vector evaluated genetic algorithm (VEGA). Here, different subpopulations are used for the different objectives. Pareto-based approaches include multiple objective GA (MOGA), non-dominated sorting GA (NSGA), and niched Pareto GA. Note that all these techniques are essentially nonelitist in nature. Some recent elitist techniques are NSGA-II [77], the strength Pareto evolutionary algorithm (SPEA) [309], and SPEA2 [308].

Simulated annealing (SA) performs reasonably well in solving single-objective optimization problems. However, its application for solving multiobjective problems has been limited, mainly because it finds a single solution in a single run

instead of a set of solutions. This appears to be a critical bottleneck in MOOPs. However, SA has been found to have some favorable characteristics for multimodal search. The advantage of SA stems from its good selection technique. Another reason behind the good performance of SA is the annealing (the gradual temperature reduction technique). However, the disadvantage of SA is the long annealing time. There are some algorithms, namely fast simulated annealing (FSA), very fast simulated re-annealing (VFSR), new simulated annealing (NSA), etc. [138, 276, 298], that take care of this crucial issue very effectively [206]. In this chapter a newly developed multiobjective version of the standard simulated annealing algorithm [29] is elaborately described. In this context the next section reviews some existing multiobjective simulated annealing-based techniques in detail.

2.3.3 Recent Multiobjective Evolutionary Algorithms

Population-based methods such as genetic algorithms can be easily extended to solve multiobjective optimization problems. There are different approaches for solving multiobjective optimization problems [24, 62, 77, 164]. (The below discussion is taken from [24].) They are categorized as follows:

- Aggregating approaches
  1. Weighted sum approach: Here, different objectives are combined using some weights w_i, $i = 1, \ldots, M$ (where M is the number of objectives). Then using these weights the objective functions are merged into a single function. Thus, the objective to be optimized is $\sum_{i=1}^{M} w_i \times f_i(\overline{x})$.
  2. Goal programming-based approach: Here, apart from the target vector, users are asked to fix targets T_i, $i = 1, \ldots, M$, or goals for each objective f_i. The purpose is then to minimize the deviation from the targets to the objectives, i.e., $\sum_{i=1}^{M} |f_i(\overline{x}) - T_i|$.
  3. Goal attainment-based approach: Here, the users are required to provide, along with the target vector, a weight vector w_i, $i = 1, \ldots, M$, relating the relative under- or overattainment of the desired goals.
  4. ε-Constraint approach: Here, the primary objective function is to be optimized considering the other objective functions as constraints bounded by some allowable levels ε_i.
- Population-based non-Pareto approaches
  1. Vector evaluated genetic algorithm (VEGA) [249]: This algorithm incorporates a special selection operator where a number of subpopulations are generated by applying proportional selection according to each objective function in turn. This was the first multiobjective genetic algorithm (MOGA) to handle multiple objective functions.
  2. Lexicographic ordering [114]: In this approach the objectives are ranked in order of importance by the user. The optimization is performed on these objectives according to this order.

3. Game theory-based approach: This approach considers that a player is associated with each objective.
4. Use of gender for identifying the objectives: In this approach, palmitic reproduction where several parents combine to produce a single offspring is allowed.
5. Use of the contact theorem: Here the fitness of an individual is set according to its relative distance with respect to the Pareto front.
6. Use of nongenerational GA: In this strategy, a multiobjective problem is transformed into a single-objective one through a set of appropriate transformations. The fitness of an individual is calculated incrementally. Genetic operators are applied to yield a single individual that replaces the worst individual in the population.

- Pareto-based nonelitist approaches

1. Multiple objective GA (MOGA) [101]: In this approach, an individual is assigned a rank corresponding to the number of individuals in the current population by which it is dominated plus 1. All nondominated individuals are ranked 1. The fitness of individuals with the same rank is averaged so that all of them are sampled at the same rate. A niche formation method is used to distribute the population over the Pareto-optimal region. This method has a very slow convergence rate, and there are some problems related to niche size parameters.
2. Niched Pareto GA (NPGA) [91]: Here, Pareto dominance-based tournament selection with a sample of the population is used to determine the winner between two candidate solutions. Around ten individuals are used to determine dominance, and the nondominated individual is selected. If both individuals are either dominated or nondominated, then the result of the tournament is decided through fitness sharing. This method again suffers from the problem of selecting an appropriate value of the niche size parameter.
3. Nondominated sorting GA (NSGA) [261]: In this approach, all nondominated individuals are classified into one category, with a dummy fitness value proportional to the population size. This group is then removed, and the remaining population is reclassified. The process is repeated until all the individuals in the entire population are classified. Stochastic remainder proportionate selection is used here. This method has a very high convergence rate, but it also suffers from problems related to the niche size parameter.

- Pareto-based elitist approaches

1. Strength Pareto evolutionary algorithm (SPEA) [309]: This algorithm implements elitism explicitly by maintaining an external population called an archive. Hence, at any generation t both the main population P_t of constant size N and the archive population P'_t (external) of maximum size N' exist. All nondominated solutions of P_t are stored in P'_t. For each individual in the archive P'_t, a strength value similar to the ranking value in MOGA is computed. The strength of an individual is proportional to the number of individuals dominated by it. The fitness of each individual in the current population P_t

is computed according to the strength of all the archived nondominated solutions that dominate it. Moreover, average linkage clustering is used to limit the size of the external population and also to increase the diversity in the population. This algorithm is well tested, and for diversity preservation no parameter is required. Its most limiting aspect is the use of clustering.

2. Strength Pareto evolutionary algorithm 2 (SPEA2) [308]: The SPEA algorithm has two potential weaknesses. Firstly, the fitness assignment is entirely based on the strength of the archive members. This results in individuals having the same fitness value in P_t, if the corresponding set of nondominated members in the archive P'_t is the same. In the worst case, if the archive contains a single member, then all the members of P_t will have same rank. In SPEA2, a technique is developed to avoid the situation where individuals dominated by the same archive members have the same fitness values. Here, both the main population and the archive are considered to determine the fitness values of the individuals. Secondly, the clustering technique used in SPEA to maintain diversity may loose outer and boundary solutions, which should be kept in the archive in order to obtain a good spread of nondominated solutions. In SPEA2, a different scheme has been adopted that prevents the loss of the boundary solutions during the updating of the archive. Diversity is maintained by using a density-based approach on the kth nearest neighbor. This method suffers from computationally expensive fitness and density calculations.
3. Pareto archived evolutionary strategy (PAES) [161]: Knowles and Corne [161] suggested a simple MOEA using a single-parent, single-child evolutionary algorithm which is similar to a $(1 + 1)$ evolutionary strategy. Here a binary representation and bitwise mutation are used while creating offspring. First the child is created, then the objective functions are computed. Thereafter, it is compared with respect to the parent. A child is accepted as the next parent if it dominates the parent, and the iteration continues. Otherwise, the child is discarded and a new mutated solution (a new solution) is generated from the parent if the parent dominates the child. However, it is also possible that the parent and the child are nondominating to each other. In such cases, both child and the parent are compared with an archive of best solutions found so far in order to find an appropriate choice. The child is compared with all members of the archive to check if it dominates any member of the archive. If it does, the child is accepted as the new parent and all the dominated solutions are eliminated from the archive. If the archive does not have any member that is dominated by the child, then both the parent and the child are checked for their nearness to the solutions of the archive. The child is accepted as a parent if it resides in a less crowded region in the parameter space. A copy of the child is also added to the archive. In order to implement the concept of crowding, the whole solution space is divided into d^M subspaces, where d is the depth parameter and M is the number of objective functions. The subspaces are updated dynamically.
4. Pareto envelope-based selection algorithm (PESA) [66]: In this approach, a smaller internal (or primary) population and a larger external (or secondary)

population are used. PESA uses the same hypergrid division of objective space adopted by PAES to maintain diversity. Its selection mechanism is based on the crowding measure used by the hypergrid. This same crowding measure is used to decide which solutions to introduce into the external population (i.e., the archive of nondominated individuals found along the evolutionary process).

5. Pareto envelope-based selection algorithm-II (PESA-II) [65]: PESA-II is the revised version of PESA in which region-based selection is proposed. In case of region-based selection, the unit of selection is a hyperbox instead of an individual. The procedure of selection is to select (using any of the traditional selection techniques) a hyperbox and then randomly select an individual within such hyperbox. The main advantage of this algorithm is that it is easy to implement and computationally very efficient. The performance of this algorithm depends on the gridsize. In order to draw grids in the objective space, prior information is needed for the objective space.
6. Elitist non-dominated sorting GA (NSGA-II) [78]: NSGA-II was proposed to eliminate the weaknesses of NSGA, especially its nonelitist nature and specification of the sharing parameter. Here, the individuals in a population undergo nondominated sorting as in NSGA, and individuals are given ranks based on this. A new selection technique, called crowded tournament selection, is proposed where selection is done based on crowding distance (representing the neighborhood density of a solution). To implement elitism, the parent and child population are combined and the nondominated individuals from the combined population are propagated to the next generation. NSGA-II is one of the widely used MOO algorithms, and is therefore described in more detail below [24].

   The non-dominated sorting method is an important characteristic of NSGA-II. This is performed as follows: Given a set of solutions S, the nondominated set of solutions $N \subseteq S$ is composed of those solutions of S which are not dominated by any other solution in S. To find the nondominated set, the following steps are carried out [24, 78]:
   - Step 1: Set $i = 1$ and initialize the nondominated set N to empty.
   - Step 2: For each solution $j \in S$ ($j \neq i$), if solution j dominates solution i then go to step 4.
   - Step 3: If $j < \|S\|$, set $j = j + 1$ and go to step 2. Otherwise, set $N = N \cup i$.
   - Step 4: Set $i = i + 1$. If $i \leq \|S\|$ then go to step 2. Otherwise, output N as the nondominated set.

   The nondominated sorting procedure first finds the nondominated set N from the given set of solutions S. Each solution belonging to N is given the rank 1. Next the same process is repeated on the set $S = S - N$ and the next set of nondominated solutions N' is found. Each solution of the set N' is given the rank 2. This procedure goes on until all the solutions in the initial set are given some rank, i.e., S becomes empty.

   A measure called crowding distance has been defined on the solutions of the nondominated front for diversity maintenance. The crowding distances for

the boundary solutions are set to maximum values (logically infinite). For each solution i among the remaining solutions, the crowding distance is computed as the average distance of the $(i+1)$th and $(i-1)$th solutions along all the objectives. The following are the steps for computing the crowding distance d_i of each point i in the nondominated front N [78]:

- For $i = 1, \ldots, N$, initialize $d_i = 0$.
- For each objective function f_k, $k = 1, \ldots, M$, do the following:
  - Sort the set N according to f_k in ascending order.
  - Set $d_1 = d_{\|N\|} = \infty$.
  - For $j = 2$ to $(\|N\| - 1)$, set $d_j = d_j + (f_{k(j+1)} - f_{k(j-1)})$.

In NSGA-II, a binary tournament selection operator is used based on the crowding distance. If two solutions a and b are compared during a tournament, then solution a wins the tournament if either:

- The rank of a is better (less) than the rank of b, i.e., a and b belong to two different nondominated fronts, or
- The ranks of a and b are the same (i.e., they belong to the same nondominated front) and a has higher crowding distance than b. This means that, if two solutions belong to the same nondominated front, the solution situated in the lesser crowded region is selected.

The following are the main steps of the NSGA-II algorithm [24, 78]:

- Initialize the population.
- While the termination criterion is not met, repeat the following:
  - Evaluate each solution in the population by computing m objective function values.
  - Rank the solutions in the population using nondominated sorting.
  - Perform selection using the crowding binary tournament selection operator.
  - Perform crossover and mutation (as in conventional GA) to generate the offspring population.
  - Combine the parent and child populations.
  - Replace the parent population by the best members (selected using nondominated sorting and the crowded comparison operator) of the combined population.
- Output the first nondominated front of the final population.

2.3.4 MOOPs and SA

As already mentioned, one of the major obstacles to use SA for MOOPs is that it produces only a single solution at a time. Since solving a multiobjective problem generally requires finding all the solutions at the same time, some researchers have thought of using multiple search agents at the same time. Good reviews of multiobjective simulated annealing techniques can be found in Chap. 9 of [63] and in [274]. A part of the following discussion is based on [63] and [274].

In most of the earlier attempts, a single objective function was constructed by combining the different objectives into one [68, 90, 123, 207, 252, 275, 285, 286]. In general, a weighted sum approach is used, where the objectives are combined as $\sum_{i=1}^{M} w_i f_i(\overline{x})$. Here $f_i(\overline{x})$, $1 \leq i \leq M$, are the M different objective functions defined on the solution vector $\overline{x}$, and the w_is are the corresponding weights. This composite objective is then used as the energy to be minimized in a scalar SA optimization method. Evidently, a major problem here is the appropriate selection of the weights. Some alternative approaches have also been used in this regard. For example, in [90], the sum of log $f_i(\overline{x})$ is taken. Additionally, the weighted summation essentially provides a convex combination of the different objectives. Solutions in the concave regions of the Pareto surface will never be considered at all.

Multiobjective simulated annealing with a composite energy clearly converges to the true Pareto front if the objectives have ratios given by w_i^{-1}. In [69], it is proved that part of the front will be inaccessible with fixed weights. In [147], several different schemes are explored for adapting the w_i during the annealing process to encourage exploration along the front. However, a proper choice of the w_i remains a challenging task.

An integration of the weighted sum approach and the concept of Pareto optimality was introduced in [68]. The algorithm, called Pareto simulated annealing (PSA), uses a population instead of a single solution at each iteration. The non-dominated vectors are stored in an external file, and quadtrees are used to store and retrieve them efficiently. Another new approach taken into consideration in PSA is that, when a new solution $\overline{x}'$ is generated in the neighborhood of $\overline{x}$ (where $\overline{x}'$ denotes the closest neighborhood solution of $\overline{x}$), the weights are incremented or decremented for those objectives depending upon whether $f(\overline{x})$ dominates $f(\overline{x}')$ or $f(\overline{x}')$ dominates $f(\overline{x})$. The main goal is to increase the probability of moving far from $f(\overline{x})$ as much as possible. The concept of parallelism is applicable to this approach as the computations required at each step can be parallelized. Experimental studies demonstrate that PSA generates more solutions on the true Pareto-optimal front and the solutions are also well distributed. PSA has been applied to solve various real-world problems.

SA is used with an energy function that transforms the MOO problem into a single-objective min-max problem in [56]. Here, the problem is to minimize the maximum deviations of each objective with respect to a set of user-defined goals. Suppapitnarm et al. [275] have used an approach where the non-dominated solutions are stored in an external file. This algorithm basically employs a single-point search. In order to be added to the external file, a new solution has to be nondominated with respect to all the solutions of the external file. This external population plays the role of a population. If the newly generated solution is archived, then it is selected as the new search starting point. Otherwise, the acceptance probability is given by $p = \prod_{i=1}^{k} \exp\{-\frac{f_i(\overline{x}) - f_i(\overline{x}')}{T_i}\}$, where k is the number of objective functions, T_i is the temperature associated with objective $f_i(\overline{x})$, and $\overline{x}$ and $\overline{x}'$ denote the current and new solution, respectively. A potential solution will be evaluated based on this acceptance criterion. It is treated as the new starting point of search once accepted, else the previous solution is considered again as the starting point. There

is also a strategy called "return to base" by which the currently accepted solution is replaced by some randomly chosen solution from the external file. This helps the SA to maintain diversity and avoid convergence to a local optimal front. However, authors have shown that their approach does not provide good results as compared with MOGA; its only advantage is its simplicity.

In [207] and [286] different nonlinear and stochastic composite energy functions have been investigated. In [207] six different criteria for energy difference calculation for MOSA are suggested and evaluated. These are (i) minimum cost criterion, (ii) maximum cost criterion, (iii) random cost criterion, (iv) self cost criterion, (v) average cost criterion, and (vi) fixed cost criterion. Since each run of the SA provides just a single solution, the algorithm attempts to evolve the set of Pareto-optimal (PO) solutions by using multiple SA runs. As a result of the independent runs, the diversity of the set of solutions suffer. After performing some comparative study, the authors conclude that the criteria that work best are the random, average, and fixed criteria [63]. For comparison with respect to some existing multiobjective evolutionary algorithms (MOEAs), the average criterion is considered. Authors have compared their approach with respect to the NPGA [91]. The proposed approach presents competitive performance but has some diversity problems. The use of niches is suggested to deal with this problem.

A modified multiobjective simulated annealing method named Pareto cost simulated annealing (PCSA) is proposed in [206]. Here, the cost of a state is computed by sampling either the neighborhood or the whole population. These techniques are very similar to the tournament selection of the NPGA with a small tournament size in the first case and the whole population size in the second case. PCSA was compared with the existing MOEA techniques, MOGA [101], NPGA [91], and NSGA [261] for 18 test problems. These problems are basically two-objective problems, having two to four variables. Authors have shown that, in 67 % of the cases, PCSA performs better than the MOEAs.

A multiobjective version of simulated annealing based on Pareto dominance is proposed by Suman [270–273]. An external archive and a scheme to handle constraints within the expression used to determine the probability of moving to a different state are proposed. A weight vector is calculated for the acceptance criterion. Each weight vector considers the number of constraints satisfied by a particular solution. In another work, five multiobjective extensions of SA, Suppapitnarm multiobjective simulated annealing (SMOSA) [271], Ulungu multiobjective simulated annealing (UMOSA) [272], Pareto simulated annealing (PSA) [273], weight-based multiobjective simulated annealing (WMOSA) [270] and Pareto dominant based multiobjective simulated annealing (PDMOSA) are compared for several constrained multiobjective optimization problems. The selection criterion adopted by SPEA [309] is used in PDMOSA. Here, solutions stored in the external archive take part in the selection process. Constraints are handled through the penalty function approach. Results show that PSA [273] provides the best qualitative results, whereas PDMOSA provides the best results with respect to diversity. Some more recent Pareto-dominance based multiobjective SA methods are proposed in [257, 258, 270].

In the Pareto domination-based multiobjective SAs developed relatively recently [257, 258, 270], the acceptance criterion between the current and a new solution has often been formulated in terms of the difference in the number of solutions that they dominate. One of the recently developed MOSA algorithms is by Smith et al. [257, 258]. Here, a dominance-based energy function is used. If the true Pareto front is available, then the energy of a particular solution $\overline{x}$ is calculated as the total number of solutions that dominate $\overline{x}$. However, as the true Pareto front is not available all the time, a proposal has been made to estimate the energy based on the current estimate of the Pareto front, F', which is the set of mutually nondominating solutions found thus far in the process. Then the energy of the current solution $\overline{x}$ is the total number of solutions in the estimated front which dominate $\overline{x}$. If $\|F'_{\overline{x}'}\|$ is the energy of the new solution $\overline{x}'$ and $\|F'_{\overline{x}}\|$ is the energy of the current solution $\overline{x}$, then the energy difference between the current and the proposed solution is calculated as $\delta E(\overline{x}', \overline{x}) = \frac{(\|F'_{\overline{x}'}\| - \|F'_{\overline{x}}\|)}{\|F'\|}$. Division by $\|F'\|$ ensures that δE is always less than unity and provides some robustness against fluctuations in the number of solutions in F'. If the size of F' is less than some threshold, then the attainment surface sampling method is adopted to increase the number of solutions on the final Pareto front. Authors have perturbed a decision variable with a random number generated from a Laplacian distribution. Two different sets of scaling factors, namely traversal scaling which generates moves to a nondominated proposal within a front and location scaling which locates a front closer to the original front, are kept. These scaling factors are updated with the iterations. A newly developed multiobjective SA termed archived multiobjective simulated annealing (AMOSA), which incorporates a concept of amount of dominance in order to determine the acceptance of a new solution as well as situation-specific acceptance probabilities, is described in detail in Sect. 2.4.

2.4 An Archive-Based Multiobjective Simulated Annealing Technique: AMOSA

2.4.1 Introduction

In Pareto domination-based multiobjective SAs developed so far, the acceptance criterion between the current and a new solution has been formulated in terms of the difference in the number of solutions that they dominate [257, 270]. Here, a newly developed multiobjective SA, referred to as archived multiobjective simulated annealing (AMOSA), is presented which incorporates a concept of amount of dominance in order to determine the acceptance of a new solution [29]. The Pareto-optimal (PO) solutions are stored in an archive. A complexity analysis of AMOSA is also provided. The performance of AMOSA has been compared with the two other well-known MOEA's, namely NSGA-II [78] and PAES [161] (described earlier) for several function optimization problems when binary encoding is used. The

comparison has been made in terms of several performance measures, namely *Convergence* [78], *Purity* [25, 140], *Spacing* [250], and *Minimal Spacing* [25]. Another measure called *displacement* [68, 141], which reflects both the proximity to and the coverage of the true PO front, is also used here for the purpose of comparison. This measure is especially useful for discontinuous fronts in order to estimate if the solution set is able to approximate all the subfronts. Many existing measures are unable to achieve this.

It may be noted that the multiobjective SA methods developed in [257, 270] are on lines similar to AMOSA. The concept of an archive or set of potentially PO solutions is also utilized in [257, 270] for storing the nondominated solutions. Instead of scalarizing the multiple objectives, a domination-based energy function is defined. However, there are notable differences. Firstly, while the number of solutions that dominate the new solution x determined the acceptance probability of x in the earlier attempts, in AMOSA this is based on the amount of domination of x with respect to the solutions in the archive and the current solution. In contrast to the works in [257, 270], where a single form of acceptance probability is considered, AMOSA deals with different forms of acceptance probabilities depending on the domination status, the choice of which are explained intuitively later on.

In [257] an unconstrained archive is maintained. Note that, theoretically, the number of Pareto-optimal solutions can be infinite. Since the ultimate purpose of a MOO algorithm is to provide the user with a set of solutions to choose from, it is necessary to limit the size of this set for it to be usable by the user. Though maintaining unconstrained archives as in [257] is novel and interesting, it is still necessary to finally reduce it to a manageable set. Limiting the size of the population (as in NSGA-II) or the archive (as in AMOSA) is an approach in this direction. Clustering appears to be a natural choice for reducing the loss of diversity, and this is incorporated in AMOSA. Clustering has also been used earlier in [309].

For comparing the performance of real-coded AMOSA with that of multiobjective SA (MOSA) [257], six three-objective test problems, namely, DTLZ1-DTLZ6 are used. Results demonstrate that the performance of AMOSA is comparable to, and often better than, that of MOSA in terms of *Purity*, *Convergence*, and *Minimal Spacing*. Comparison is also made with real-coded NSGA-II for the above-mentioned six problems, as well as for some 4-, 5-, 10-, and 15-objective test problems. Results show that the performance of AMOSA is superior to that of NSGA-II, especially for the test problems with many objective functions. This is an interesting and the most desirable feature of AMOSA, since Pareto ranking-based MOEAs, such as NSGA-II, [78] do not work well on many-objective optimization problems as pointed out in some recent studies [137, 139].

2.4.2 Archived Multiobjective Simulated Annealing (AMOSA)

AMOSA incorporates the concept of an *Archive* where the nondominated solutions seen so far are stored. In [99], the use of an unconstrained *Archive* size to reduce the

loss of diversity is discussed in detail. In this approach the archive size is kept limited, since finally only a small number of well-distributed Pareto-optimal solutions are needed. Two limits are kept on the size of the *Archive*: a hard or strict limit denoted by *HL*, and a soft limit denoted by *SL*. During the process, the nondominated solutions are stored in the *Archive* as and when they are generated until the size of the *Archive* increases to *SL*. Thereafter, if more nondominated solutions are generated, these are added to the *Archive*, the size of which is thereafter reduced to *HL* by applying clustering. The structure of the simulated annealing-based AMOSA is shown in Fig. 2.5. The parameters that need to be set a priori are mentioned below:

- *HL*: The maximum size of the *Archive* on termination. This set is equal to the maximum number of nondominated solutions required by the user.
- *SL*: The maximum size to which the *Archive* may be filled before clustering is used to reduce its size to *HL*.
- *Tmax*: Maximum (initial) temperature.
- *Tmin*: Minimal (final) temperature.
- *iter*: Number of iterations at each temperature.
- α: The cooling rate in SA.

The different steps of the algorithm are now explained in detail.

2.4.3 Archive Initialization

The algorithm begins with the initialization of a number $\gamma \times SL$ ($\gamma > 1$) of solutions. Each of these solutions is refined by using a simple hill-climbing technique, accepting a new solution only if it dominates the previous one. This is continued for a number of iterations. Thereafter, the nondominated solutions (*ND*) that are obtained are stored in the *Archive*, up to a maximum of *HL*. In case the number of nondominated solutions exceeds *HL*, clustering is applied to reduce the size to *HL* (the clustering procedure is explained below). This means that initially *Archive* contains a maximum of *HL* number of solutions.

In the initialization phase it is possible to get an *Archive* of size one. In MOSA [257], in such cases, other newly generated solutions which are dominated by the archival solution will be indistinguishable. In contrast, the amount of domination as incorporated in AMOSA will distinguish between "more dominated" and "less dominated" solutions. However, in the future, it is intended to use a more sophisticated scheme, in line with that adopted in MOSA.

2.4.4 Clustering the Archive Solutions

Clustering of the solutions in the *Archive* has been incorporated in AMOSA in order to explicitly enforce diversity of the nondominated solutions. In general, the size of

Set *Tmax*, *Tmin*, *HL*, *SL*, *iter*, α, temp=*Tmax*.
Initialize the *Archive*.
current-pt = random(*Archive*). /* randomly chosen solution from *Archive**/
while (temp > *Tmin*)
  for (i=0; i< iter; i++)
    new-pt=perturb(*current-pt*).
    Check the domination status of *new-pt* and *current-pt*.
    /* Code for different cases */
    if (*current-pt* dominates *new-pt*) /* Case 1*/
      $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k}(\Delta dom_{i,new\text{-}pt}) + \Delta dom_{current\text{-}pt,new\text{-}pt}\right)}{(k+1)}$.
      /* k=total-no-of points in the *Archive* which dominate *new-pt*, $k \geq 0$. */
      Set *new-pt* as *current-pt* with probability calculated using Eq. 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg}.

    if (*current-pt* and *new-pt* are nondominating to each other) /* Case 2*/
      Check the domination status of *new-pt* and points in the *Archive*.
      if (*new-pt* is dominated by k ($k \geq 1$) points in the *Archive*) /* Case 2(a)*/
        $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k} \Delta dom_{i,new\text{-}pt}\right)}{k}$.
        Set *new-pt* as *current-pt* with probability calculated using Eq. 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg}.
      if (*new-pt* is non-dominating w.r.t all the points in the *Archive*) /* Case 2(b)*/
        Set *new-pt* as *current-pt* and add *new-pt* to the *Archive*.
        if *Archive-size* > *SL*
          Cluster *Archive* to *HL* number of clusters.
      if (*new-pt* dominates k, ($k \geq 1$) points of the *Archive*) /* Case 2(c)*/
        Set *new-pt* as *current-pt* and add it to *Archive*.
        Remove all the k dominated points from the *Archive*.
    if (*new-pt* dominates *current-pt*) /* Case 3 */
      Check the domination status of *new-pt* and points in the *Archive*.
      if (*new-pt* is dominated by k ($k \geq 1$) points in the *Archive*) /* Case 3(a)*/
        Δdom_{min} = minimum of the difference of domination amounts between the *new-pt* and the k points
        $prob = \frac{1}{1+exp(-\Delta dom_{min})}$.
        Set point of the archive which corresponds to Δdom_{min} as *current-pt*
        with probability=*prob*.
        else set *new-pt* as *current-pt*
      if (*new-pt* is nondominating with respect to the points in the *Archive*) /* Case 3(b) */
        select the *new-pt* as the *current-pt* and add it to the *Archive*.
        if *current-pt* is in the *Archive*, remove it from *Archive*.
        else if *Archive-size*> *SL*.
          Cluster *Archive* to *HL* number of clusters.
      if (*new-pt* dominates k other points in the *Archive*) /* Case 3(c)*/
        Set *new-pt* as *current-pt* and add it to the *Archive*.
        Remove all the k dominated points from the *Archive*.
  End for
$temp = \alpha * temp$.
End while
if *Archive-size* > *SL*
  Cluster *Archive* to *HL* number of clusters.

Fig. 2.5 The AMOSA algorithm [29] (source code is available at: www.isical.ac.in/~sriparna_r)

the *Archive* is allowed to increase up to *SL* (>*HL*), after which the solutions are clustered for grouping the solutions into *HL* clusters. Allowing the *Archive* size to increase up to *SL* not only reduces excessive calls to clustering, but also enables the formation of more spread-out clusters and hence better diversity. Note that, in the initialization phase, clustering is executed once even if the number of solutions in the *Archive* is less than *SL*, as long as it is greater than *HL*. This enables it to start with at most *HL* nondominated solutions and reduces excessive calls to clustering in the initial stages of the AMOSA process.

For clustering, the well-known single linkage algorithm [143] has been used. Here, the distance between any two clusters corresponds to the length of the shortest link between them. This is similar to the clustering algorithm used in SPEA [309], except that they used the average linkage method [143]. After *HL* clusters are obtained, the member within each cluster whose average distance to the other members is the minimum is considered as the representative member of the cluster. A tie is resolved arbitrarily. The representative points of all the *HL* clusters are thereafter stored in the *Archive*.

2.4.5 Amount of Domination

As already mentioned, AMOSA uses the concept of amount of domination in computing the acceptance probability of a new solution. Given two solutions a and b, the amount of domination is defined as $\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{M} \frac{|f_i(a) - f_i(b)|}{R_i}$ where M, is the number of objectives and R_i is the range of the ith objective. Note that, in several cases, R_i may not be known a priori. In such situations, the solutions present in the *Archive* along with the new and the current solutions are used for computing it. The concept of $\Delta dom_{a,b}$ is illustrated pictorially in Fig. 2.6 for the two-objective case. $\Delta dom_{a,b}$ is used in AMOSA while computing the probability of acceptance of a newly generated solution.

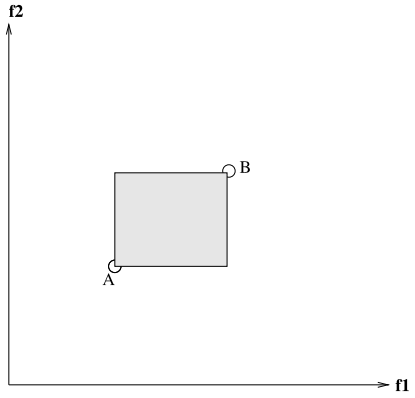


Fig. 2.6 Total amount of domination between the two solutions A and B = the area of the shaded rectangle

2.4.6 The Main AMOSA Process

One of the points, called *current-pt*, is randomly selected from *Archive* as the initial solution at temperature $temp = Tmax$. *current-pt* is perturbed to generate a new solution called *new-pt*. The domination status of *new-pt* is checked with respect to *current-pt* and solutions in the *Archive*.

Based on the domination status between *current-pt* and *new-pt*, three different cases may arise. These are enumerated below:

- Case 1: *current-pt* dominates *new-pt*, and k ($k \geq 0$) points from the *Archive* dominate *new-pt*.

  This situation is shown in Fig. 2.7. Figures 2.7(a) and 2.7(b) denote the situations where $k = 0$ and $k \geq 1$, respectively. (Note that all the figures correspond to a two-objective maximization problem.) In this case, *new-pt* is selected as *current-pt* with a probability calculated using the following equation (as in Eq. 2.3):

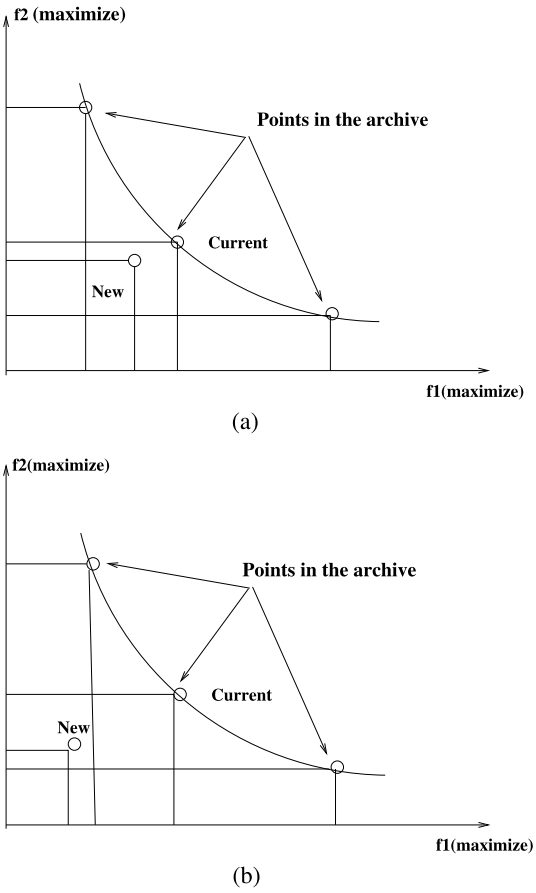


Fig. 2.7 Different cases when *New* is dominated by *Current*: (**a**) *New* is nondominating with respect to the solutions of *Archive* except *Current* if it is in the *Archive*; (**b**) Some solutions in the *Archive* dominate *New*

$$p_{qs} = \frac{1}{1 + e^{\frac{\Delta dom_{avg}}{T}}}, \tag{2.7}$$

where

$$\Delta dom_{avg} = \left(\sum_{i=1}^{k} (\Delta dom_{i,new\text{-}pt}) + \Delta dom_{current\text{-}pt,new\text{-}pt} \right) / (k+1).$$

Note that Δdom_{avg} denotes the average amount of domination of *new-pt* by $(k+1)$ points, namely, *current-pt* and k points of the *Archive*. Also, as k increases, Δdom_{avg} will increase, since here dominating points that are further away from the *new-pt* are contributing to its value.

Lemma *When $k = 0$, current-pt is on the archival front.*

Proof If this is not the case, then some point, say A, in the *Archive* dominates it. Since *current-pt* dominates *new-pt*, by transitivity, A will also dominate *new-pt*. However, it is considered that no other point in the *Archive* dominates *new-pt*, as $k = 0$. Hence proved. □

However, if $k \geq 1$, this may or may not be true.

- Case 2: *current-pt* and *new-pt* are non-dominating with respect to each other.
  Now, based on the domination status of *new-pt* and members of *Archive*, the following three situations may arise:

  1. *new-pt* is dominated by k points in the *Archive*, where $k \geq 1$. This situation is shown in Fig. 2.8(a). *new-pt* is selected as *current-pt* with a probability calculated using the following equation (as in Eq. 2.3):

  $$p_{qs} = \frac{1}{1 + e^{\frac{\Delta dom_{avg}}{T}}}, \tag{2.8}$$

  where

  $$\Delta dom_{avg} = \sum_{i=1}^{k} (\Delta dom_{i,new\text{-}pt}) / k.$$

  Note that here *current-pt* may or may not be on the archival front.
  2. *new-pt* is nondominating with respect to the other points in the *Archive* as well. In this case *new-pt* is on the same front as the *Archive*, as shown in Fig. 2.8(b). So *new-pt* is selected as *current-pt* and added to the *Archive*. In case the *Archive* becomes overfull (i.e., *SL* is exceeded), clustering is performed to reduce the number of points to *HL*.
  3. *new-pt* dominates k ($k \geq 1$) points of the *Archive*. This situation is shown in Fig. 2.8(c). Again, *new-pt* is selected as *current-pt* and added to the *Archive*. All the k dominated points are removed from the *Archive*. Note that here too *current-pt* may or may not be on the archival front.

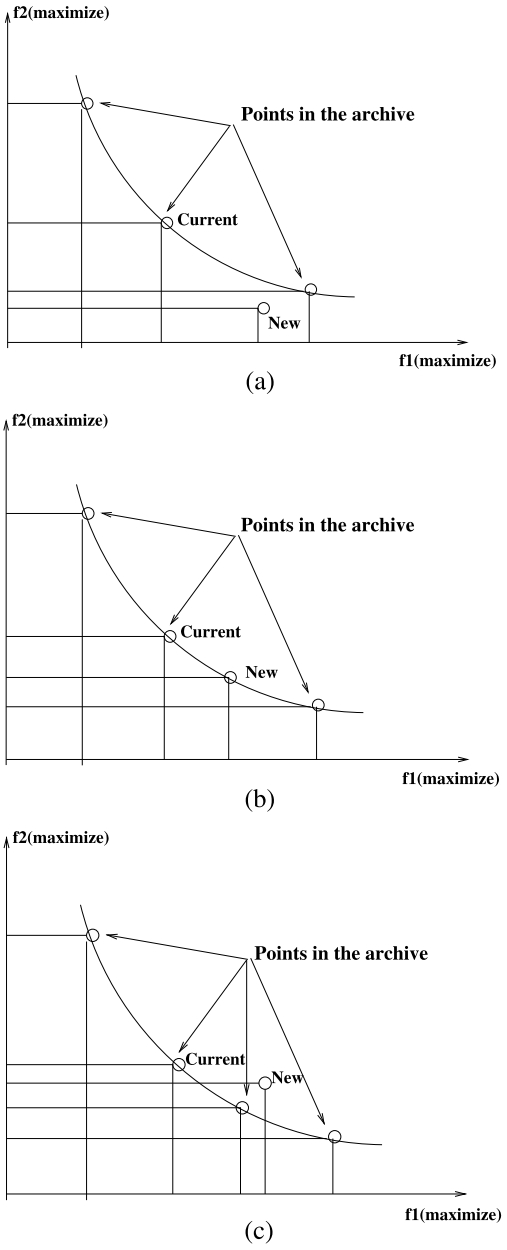


Fig. 2.8 Different cases when *New* and *Current* are nondominating: (**a**) Some solutions in *Archive* dominate *New*; (**b**) *New* is nondominating with respect to all the solutions of *Archive*; (**c**) *New* dominates k ($k \geq 1$) solutions in the *Archive*

- Case 3: *new-pt* dominates *current-pt*

  Now, based on the domination status of *new-pt* and members of *Archive*, the following three situations may arise:

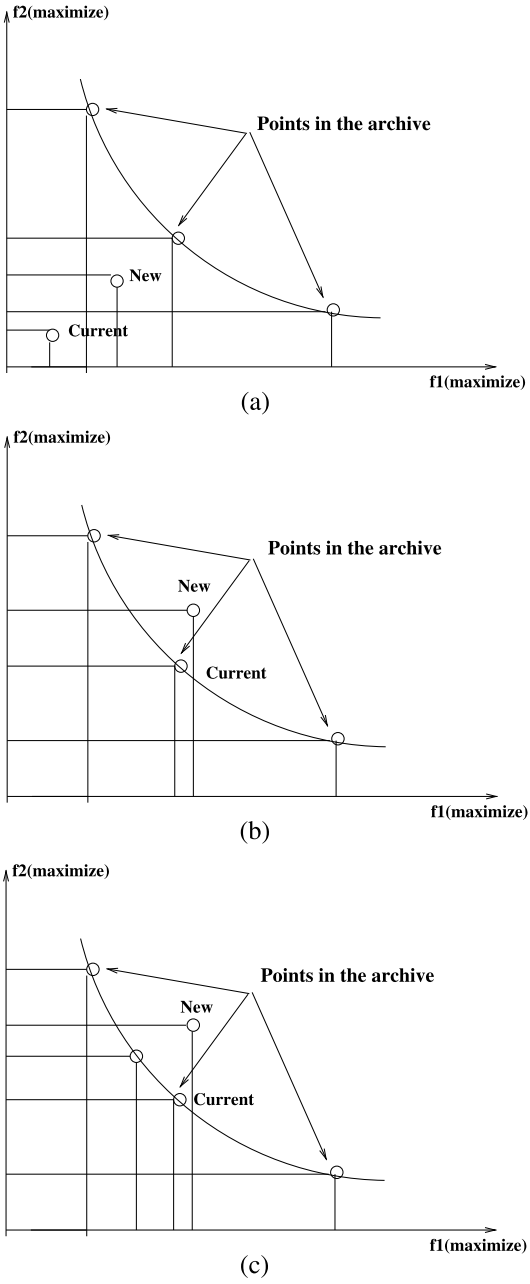


Fig. 2.9 Different cases when *New* dominates *Current*: (**a**) *New* is dominated by some solutions in *Archive*; (**b**) *New* is nondominating with respect to the solutions in the *Archive* except *Current*, if it is in the *Archive*; (**c**) *New* dominates some solutions of *Archive* other than *Current*

1. *new-pt* dominates *current-pt*, but k ($k \geq 1$) points in the *Archive* dominate this *new-pt*. Note that this situation [shown in Fig. 2.9(a)] may arise only if the *current-pt* is not a member of the *Archive*. Here, the minimum of

the difference of domination amounts between *new-pt* and the k points, denoted by Δdom_{min}, of the *Archive* is computed. The point from the *Archive* which corresponds to the minimum difference is selected as *current-pt* with $prob = \frac{1}{1+exp(-\Delta dom_{min})}$. Otherwise, *new-pt* is selected as *current-pt*. Note that, according to the SA paradigm, *new-pt* should have been selected with probability 1. However, due to the presence of *Archive*, it is evident that solutions still better than *new-pt* exist. Therefore, *new-pt* and the dominating points in the *Archive* that are closest to the *new-pt* (corresponding to Δdom_{min}) compete for acceptance. This may be considered a form of informed reseeding of the annealer only if the *Archive* point is accepted, but with a solution closest to the one which would otherwise have survived in the normal SA paradigm.

2. *new-pt* is nondominating with respect to the points in the *Archive* except *current-pt* if it belongs to the *Archive*. This situation is shown in Fig. 2.9(b). Thus, *new-pt*, which is now accepted as *current-pt*, can be considered as a new nondominated solution that must be stored in the *Archive*. Hence, *new-pt* is added to the *Archive*. If *current-pt* is in the *Archive*, then it is removed. Otherwise, if the number of points in the *Archive* becomes more than *SL*, clustering is performed to reduce the number of points to *HL*. Note that here *current-pt* may or may not be on the archival front.
3. *new-pt* also dominates k ($k \geq 1$) other points in the *Archive* [see Fig. 2.9(c)]. Hence, *new-pt* is selected as *current-pt* and added to the *Archive*, while all the dominated points of the *Archive* are removed. Note that here *current-pt* may or may not be on the archival front.

The above process is repeated *iter* times for each temperature (*temp*). The temperature is reduced to $\alpha \times temp$, using the cooling rate of α, till the minimum temperature, *Tmin*, is attained. The process thereafter stops, and the *Archive* contains the final nondominated solutions.

Note that in AMOSA, as in other versions of multiobjective SA algorithms, there is a possibility that a new solution worse than the current solution may be selected. In most other MOEAs, e.g., NSGA-II and PAES, if a choice needs to be made between two solutions $\overline{x}$ and $\overline{y}$, and if $\overline{x}$ dominates $\overline{y}$, then $\overline{x}$ is always selected. It may be noted that, in single-objective EAs or SA, usually a worse solution also has a nonzero chance of surviving in subsequent generations; this leads to a reduced possibility of getting stuck at suboptimal regions. However, most of the MOEAs have been so designed that this characteristic is lost. The present simulated annealing-based algorithm provides a way of incorporating this feature.

2.4.7 Complexity Analysis

The complexity analysis of AMOSA is provided in this section. The basic operations and their worst-case complexities are as follows:

1. Archive initialization: $O(SL)$.
2. Procedure to check the domination status of any two solutions: $O(M)$, $M =$ number of objectives.
3. Procedure to check the domination status between a particular solution and the *Archive* members: $O(M \times SL)$.
4. Single linkage clustering: $O(SL^2 \times \log(SL))$ [283].
5. Clustering procedure is executed

   - Once after initialization if $|ND| > HL$
   - After each $(SL - HL)$ number of iterations.
   - At the end if final $|Archive| > HL$

   So the maximum number of times the clustering procedure is called is $(TotalIter/(SL - HL)) + 2$.

   Therefore, the total complexity due to the clustering procedure is $O((TotalIter/(SL - HL)) \times SL^2 \times \log(SL))$.

The total complexity of AMOSA becomes

$$(SL + M + M \times SL) \times (TotalIter) + \frac{TotalIter}{SL - HL} \times SL^2 \times \log(SL). \tag{2.9}$$

Let $SL = \beta \times HL$, where $\beta \geq 2$, and $HL = N$, where N is the population size in NSGA-II and archive size in PAES. Therefore, overall complexity of AMOSA becomes

$$(TotalIter) \times \left(\beta \times N + M + M \times \beta \times N + \left(\beta^2/(\beta - 1)\right) \times N \times \log(\beta N)\right) \tag{2.10}$$

or

$$O\left(TotalIter \times N \times \left(M + \log(N)\right)\right). \tag{2.11}$$

Note that the total complexity of NSGA-II is $O(TotalIter \times M \times N^2)$ and that of PAES is $O(TotalIter \times M \times N)$. The NSGA-II complexity depends on the complexity of non-dominated procedure. With the best procedure, the complexity is $O(TotalIter \times M \times N \times \log(N))$.

2.5 Simulation Results

The experimental results of AMOSA as compared with several other methods are reported in detail in [29]. Some of these are described in this section. The comparison metrics used for the experiments are described first. The performance analysis of both the binary-coded AMOSA and the real-coded AMOSA are provided thereafter.

2.5.1 Comparison Measures

In multiobjective optimization, there are basically two functionalities that an MOO strategy must achieve regarding the obtained solution set [77]: It should converge as close to the true PO front as possible, and it should maintain as diverse a solution set as possible.

The first condition clearly ensures that the obtained solutions are near optimal, and the second condition ensures that a wide range of trade-off solutions is obtained. Clearly, these two tasks cannot be measured with one performance measure adequately. A number of performance measures have been suggested in the past. Here mainly three such performance measures are used. The first measure is the *Convergence* measure γ [78]. It measures the extent of convergence of the obtained solution set to a known set of PO solutions.The lower the value of γ, the better the convergence of the obtained solution set to the true PO front. The second measure, called *Purity* [25, 140], is used to compare the solutions obtained using different MOO strategies. It calculates the fraction of solutions from a particular method that remains nondominating when the final front solutions obtained from all the algorithms are combined. A value near to 1 (0) indicates better (poorer) performance. The third measure, named *Spacing*, was first proposed by Schott [250]. It reflects the uniformity of the solutions over the nondominated front. It is shown in [25] that this measure will fail to give an adequate result in some situations. In order to overcome the above limitations, a modified measure, named *Minimal Spacing*, is proposed in [25]. Smaller values of *Spacing* and *Minimal Spacing* indicate better performance.

It may be noted that, if an algorithm is able to approximate only a portion of the true PO front, not its full extent, none of the existing measures will be able to reflect this. In case of a discontinuous PO front, this problem becomes severe when an algorithm totally misses a subfront. Here, a performance measure which is very similar to the measure used in [68] and [141], named *displacement*, is used to overcome this limitation. It measures how far the obtained solution set is from a known set of PO solutions. In order to compute the *displacement* measure, a set P^* consisting of uniformly spaced solutions from the true PO front in the objective space is found (as is done while calculating γ). Then, *displacement* is calculated as

$$displacement = \frac{1}{|P^*|} \times \sum_{i=1}^{|P^*|} \Big(min_{j=1}^{|Q|} d(i, j) \Big), \tag{2.12}$$

where Q is the obtained set of final solutions, and $d(i, j)$ is the Euclidean distance between the ith solution of P^* and jth solution of Q. The lower the value of this measure, the better the convergence to and extent of coverage of the true PO front.

2.5.2 Comparison of Binary-Encoded AMOSA with NSGA-II and PAES

Firstly, the binary-encoded AMOSA is compared with the binary-coded NSGA-II and PAES algorithms. For AMOSA, binary mutation is used. Seven test problems have been considered for the comparison. These are SCH1 and SCH2 [77], Deb1 and Deb4 [76], ZDT1, ZDT2, and ZDT6 [77]. All the algorithms were executed ten times per problem, and the results reported are the average values obtained for the ten runs. In NSGA-II, the crossover probability (p_c) is kept equal to 0.9. For PAES, the depth value d is set equal to 5. For AMOSA, the cooling rate α is kept equal to 0.8. The number of bits assigned to encode each decision variable depends on the problem. For example, in ZDT1, ZDT2, and ZDT6, which all are 30-variable problems, 10 bits are used to encode each variable; for SCH1 and SCH2, which are single-variable problems, and for Deb1 and Deb4, which are two-variable problems, 20 bits are used to encode each decision variable. In all the approaches, binary mutation applied with a probability of $p_m = 1/l$, where l is the string length, is used as the perturbation operation. The values of *Tmax* (maximum value of the temperature), *Tmin* (minimum value of the temperature), and *iter* (number of iterations at each temperature) are chosen so that the total number of fitness evaluations of the three algorithms becomes approximately equal. For PAES and NSGA-II, identical parameter settings as suggested in the original studies have been used. Here the population size in NSGA-II, and archive sizes in AMOSA and PAES, are set to 100. The maximum number of iterations for NSGA-II and PAES are 500 and 50,000, respectively. For AMOSA, $Tmax = 200$, $Tmin = 10^{-7}$ and, $iter = 500$. The parameter values were determined after extensive sensitivity studies, which are omitted here to restrict the size of the chapter.

2.5.2.1 Discussion of the Results

The *Convergence* and *Purity* values obtained using the three algorithms are presented in Table 2.1. AMOSA performs best for ZDT1, ZDT2, ZDT6, and Deb1 in terms of γ. For SCH1, all three perform equally well. NSGA-II performs well for SCH2 and Deb4. Interestingly, for all the functions, AMOSA is found to provide a greater number of overall nondominated solutions than NSGA-II. (This is evident from the quantities in parentheses in Table 2.1, where x/y indicates that on average the algorithm produced y solutions of which x remained good even when solutions from other MOO strategies are combined.) AMOSA took 10 seconds to provide the first PO solution, compared with 32 seconds for NSGA-II in case of ZDT1. From Table 2.1 it is also clear that AMOSA and PAES provide a greater number of distinct solutions than NSGA-II.

Table 2.2 shows the *Spacing* and *Minimal Spacing* measurements. AMOSA provides the best values of *Spacing* in most cases, while PAES performs the worst. This

Table 2.1 *Convergence* and *Purity* measures on the test functions for binary encoding

Test problem	Convergence			Purity		
	AMOSA	PAES	NSGA-II	AMOSA	PAES	NSGA-II
SCH1	0.0016	0.0016	0.0016	0.9950 (99.5/100)	0.9850 (98.5/100)	1 (94/94)
SCH2	0.0031	0.0015	0.0025	0.9950 (99.5/100)	0.9670 (96.7/100)	0.9974 (97/97.3)
ZDT1	0.0019	0.0025	0.0046	0.8350 (83.5/100)	0.6535 (65.4/100)	0.970 (68.64/70.6)
ZDT2	0.0028	0.0048	0.0390	0.8845 (88.5/100)	0.4050 (38.5/94.9)	0.7421 (56.4/76)
ZDT6	0.0026	0.0053	0.0036	1 (100/100)	0.9949 (98.8/99.3)	0.9880 (66.5/67.3)
Deb1	0.0046	0.0539	0.0432	0.91 (91/100)	0.718 (71.8/100)	0.77 (71/92)
Deb4	0.0026	0.0025	0.0022	0.98 (98/100)	0.9522 (95.2/100)	0.985 (88.7/90)

Table 2.2 *Spacing* and *Minimal spacing* measures on the test functions for binary encoding

Test problem	Spacing			Minimal spacing		
	AMOSA	PAES	NSGA-II	AMOSA	PAES	NSGA-II
SCH1	0.0167	0.0519	0.0235	0.0078	0.0530	0.0125
SCH2	0.0239	0.5289	0.0495	*N.A.*	*N.A.*	*N.A.*
ZDT1	0.0097	0.0264	0.0084	0.0156	0.0265	0.0147
ZDT2	0.0083	0.0205	0.0079	0.0151	0.0370	0.0130
ZDT6	0.0051	0.0399	0.0089	0.0130	0.0340	0.0162
Deb1	0.0166	0.0848	0.0475	0.0159	0.0424	0.0116
Deb4	0.0053	0.0253	0.0089	*N.A.*	*N.A.*	*N.A.*

Table 2.3 New measure *displacement* on the test functions for binary encoding

Algorithm	SCH2	Deb4	ZDT1	ZDT2	ZDT6
AMOSA	0.0230	0.0047	0.0057	0.0058	0.0029
PAES	0.6660	0.0153	0.0082	0.0176	0.0048
NSGA-II	0.0240	0.0050	0.0157	0.0096	0.0046

is also evident from Figs. 2.10 and 2.11, which show the final PO fronts of SCH2 and Deb4 obtained by the three methods for the purpose of illustration. With respect to *Minimal Spacing* the performance of AMOSA and NSGA-II is comparable.

Table 2.3 presents the value of *displacement* for five problems, two with discontinuous and three with continuous PO fronts. AMOSA performs the best in almost

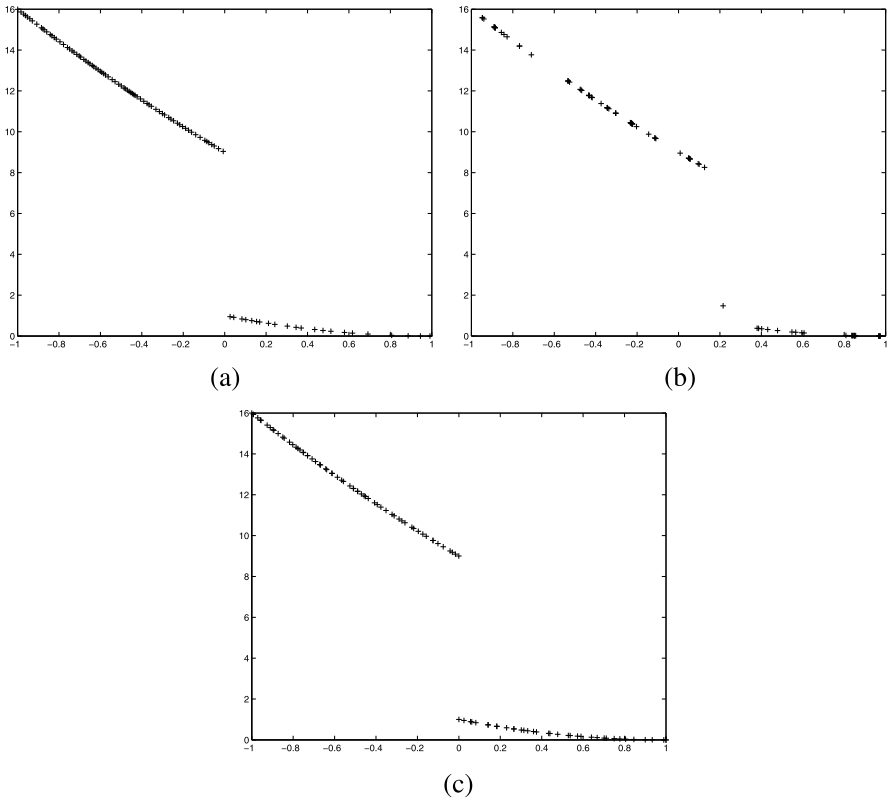


Fig. 2.10 The final nondominated front for SCH2 obtained by: (**a**) AMOSA, (**b**) PAES, and (**c**) NSGA-II

all the cases. The utility of the new measure is evident in particular for Deb4, where PAES performs quite poorly (see Fig. 2.11). Interestingly, the *Convergence* value for PAES (Table 2.1) is very good here, though the *displacement* correctly reflects that the PO front has been represented very poorly.

Table 2.4 presents the time taken by the algorithms for the different test functions. It is seen that PAES takes less time in six of the seven problems because of its lower complexity. AMOSA takes less time than NSGA-II in 30-variable problems such as ZDT1 and ZDT2, and the 10-variable problem ZDT6. But for the single- and two-variable problems SCH1, SCH2, Deb1 and Deb4, AMOSA takes more time than NSGA-II. This may be due to the complexity of its clustering procedure. Generally, for single- or two-variable problems, this procedure dominates the crossover and ranking procedures of NSGA-II. But for 30-variable problems the scenario is reversed. This is because of the increased complexity of ranking and crossover (due to increased string length) in NSGA-II.

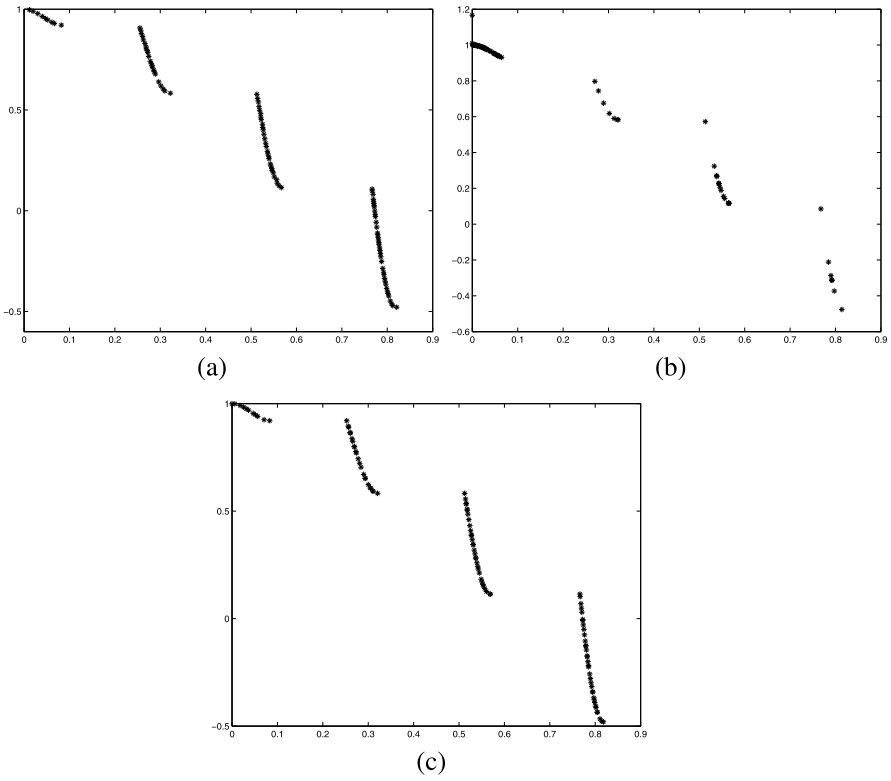


Fig. 2.11 The final nondominated front for Deb4 obtained by: (**a**) AMOSA, (**b**) PAES, and (**c**) NSGA-II

2.5.3 Comparison of Real-Coded AMOSA with the Algorithm of Smith et al. and Real-Coded NSGA-II

In the real-coded version of AMOSA [29] (source code available at: www.isical.ac.in/~sriparna_r), mutation is done as suggested in [257]. Here, a new string is generated from the old string $\overline{x}$ by perturbing only one parameter or decision variable of $\overline{x}$. The parameter to be perturbed is chosen at random and perturbed with a random variable ε drawn from a Laplacian distribution, $p(\varepsilon) \propto e^{-\frac{|\varepsilon-\mu|}{\delta}}$, where the scaling factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. A fixed scaling factor equal to 0.1 is used for mutation. The initial temperature is selected by the procedure mentioned in [257]; that is, the initial temperature, *Tmax*, is calculated by using a short 'burn-in' period during which all solutions are accepted and setting the temperature equal to the average positive change of energy divided by ln(2). Here, *Tmin* is kept equal to 10^{-5} and the temperature is adjusted according to $T_k = \alpha^k Tmax$, where α is set equal to 0.8. For NSGA-II, population size is kept equal to 100 and the total number of generations is set such

Table 2.4 Time taken by different programs (in seconds) for binary encoding

Algorithm	SCH1	SCH2	Deb1	Deb4	ZDT1	ZDT2	ZDT6
AMOSA	15	14.5	20	20	58	56	12
PAES	6	5	5	15	17	18	16
NSGA-II	11	11	14	14	77	60	21

that the total number of function evaluations of AMOSA and NSGA-II are the same. For AMOSA the archive size is set equal to 100. (However, in a part of the investigations, the archive size is kept unlimited as in [257]. The results are compared with those obtained by MOSA [257] and provided in [3].) AMOSA is executed for a total of 5,000, 1,000, 15,000, 5,000, 1,000, 5,000 and 9,000 run lengths, respectively, for DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ5, and DTLZ6. The total number of iterations, $iter$, per temperature is set accordingly. Real-coded NSGA-II (code obtained from KANGAL site: http://www.iitk.ac.in/kangal/codes.html) is executed. For NSGA-II the following parameter setting is used: probability of crossover = 0.99, probability of mutation = $(1/l)$, where l is the string length, distribution index for the crossover operation = 10, distribution index for the mutation operation = 100.

In MOSA [257], authors have used unconstrained archive size. Note that the archive size of AMOSA and the population size of NSGA-II are both 100. For the purpose of comparison with MOSA, which has an unlimited archive [257], the clustering procedure (adopted for AMOSA) is used to reduce the number of solutions of [3] to 100. Comparison is performed in terms of *Purity*, *Convergence*, and *Minimal Spacing*. Table 2.5 presents the *Purity*, *Convergence*, and *Minimal Spacing* measurements for problems DTLZ1-DTLZ6 obtained after application of AMOSA, MOSA, and NSGA-II. It can be seen from this table that AMOSA performs the best in terms of *Purity* and *Convergence* for DTLZ1, DTLZ3, DTLZ5, and DTLZ6. In DTLZ2 and DTLZ4, the performance of MOSA is marginally better than that of AMOSA. NSGA-II performs the worst of all. Table 2.5 presents the *Minimal Spacing* values obtained by the three algorithms for DTLZ1-DTLZ6. AMOSA performs the best in all the cases.

As mentioned earlier, for comparison with the performance of MOSA (by considering the results reported in [3]), a version of AMOSA without clustering and with unconstrained archive is executed. The results reported here are averages over 10 runs. Table 2.6 presents the corresponding *Purity*, *Convergence*, and *Minimal Spacing* values. Again AMOSA performs much better than MOSA for all the test problems, except DTLZ4. For DTLZ4, MOSA performs better than AMOSA in terms of both *Purity* and *Convergence* values. Figure 2.12 shows the final Pareto-optimal front obtained by AMOSA and MOSA for DTLZ1-DTLZ3, while Fig. 2.13 shows the same for DTLZ5 and DTLZ6. As can be seen from the figures, AMOSA appears to be able to better approximate the front with more dense-solutions as compared with MOSA.

It was mentioned in [307] that, for a particular test problem, almost 40 % of the solutions provided by an algorithm with truncation of the archive are domi-

Table 2.5 *Convergence*, *Purity*, and *Minimal Spacing* measures on the three-objective test functions while *Archive* is bounded to 100

Test problem	Convergence			Purity			Minimal spacing		
	AMOSA	MOSA	NSGA-II	AMOSA	MOSA	NSGA-II	AMOSA	MOSA	NSGA-II
DTLZ1	0.01235	0.159	13.695	0.857 (85.7/100)	0.56 (28.35/75)	0.378 (55.7/100)	0.0107	0.1529	0.2119
DTLZ2	0.014	0.01215	0.165	0.937 (93.37/100)	0.9637 (96.37/100)	0.23 (23.25/100)	0.0969	0.1069	0.1236
DTLZ3	0.0167	0.71	20.19	0.98 (93/95)	0.84 (84.99/100)	0.232 (23.2/70.6)	0.1015	0.152	0.14084
DTLZ4	0.28	0.21	0.45	0.833 (60/72)	0.97 (97/100)	0.7 (70/100)	0.20	0.242	0.318
DTLZ5	0.00044	0.0044	0.1036	1 (97/97)	0.638 (53.37/83.6)	0.05 (5/100)	0.0159	0.0579	0.128
DTLZ6	0.043	0.3722	0.329	0.9212 (92.12/100)	0.7175 (71.75/100)	0.505 (50.5/100)	0.1148	0.127	0.1266

Table 2.6 *Convergence*, *Purity*, and *Minimal Spacing* measures on the three-objective test functions by AMOSA and MOSA while *Archive* is unbounded

Test problem	Convergence		Purity		Minimal spacing	
	AMOSA	MOSA	AMOSA	MOSA	AMOSA	MOSA
DTLZ1	0.010	0.1275	0.99 (1253.87/1262.62)	0.189 (54.87/289.62)	0.064	0.083.84
DTLZ2	0.0073	0.0089	0.96 (1074.8/1116.3)	0.94 (225/239.2)	0.07598	0.09595
DTLZ3	0.013	0.025	0.858 (1212/1412.8)	0.81 (1719/2003.9)	0.0399	0.05
DTLZ4	0.032	0.024	0.8845 (88.5/100)	0.4050 (38.5/94.9)	0.1536	0.089
DTLZ5	0.0025	0.0047	0.92 (298/323.66)	0.684 (58.5/85.5)	0.018	0.05826
DTLZ6	0.0403	0.208	0.9979 (738.25/739.75)	0.287 (55.75/194.25)	0.0465	0.0111

nated by solutions provided by an algorithm without archive truncation. However, the experiments conducted in [29] did not adequately justify this finding. Let the set of solutions of AMOSA with and without clustering be denoted by S_c and S_{wc}, respectively. For DTLZ1 it was found that 12.6 % of S_c were dominated by S_{wc}, while 4 % of S_{wc} were dominated by S_c. For DTLZ2, 5.1 % of S_{wc} were dominated by S_c while 5.4 % of S_c were dominated by S_{wc}. For DTLZ3, 22.38 % of S_{wc} were dominated by S_c while 0.53 % of S_c were dominated by S_{wc}. For DTLZ4, all the members of S_{wc} and S_c were the same since AMOSA without clustering did not provide more than 100 solutions. For DTLZ5, 10.4 % of S_{wc} were dominated by S_c while 0.5 % of S_c were dominated by S_{wc}. For DTLZ6, all the members of S_{wc} and S_c were nondominating with respect to each other.

To investigate the performance on a four-objective problem, AMOSA and NSGA-II were applied to the 13-variable DTLZ2 test problem [29]. This is referred to as DTLZ2_4. The problem has a spherical Pareto front in four dimensions given by the equation: $f_1^2 + f_2^2 + f_3^2 + f_4^2 = 1$ with $f_i \in [0, 1]$ for $i = 1$ to 4. Both algorithms were applied for a total of 30,000 function evaluations (for NSGA-II popsize 100 and number of generations 300) and the *Purity*, *Convergence*, and *Minimal Spacing* values are presented in Table 2.7. AMOSA performs much better than NSGA-II in terms of all three measures.

AMOSA and NSGA-II were also compared for DTLZ1_5 (9-variable 5-objective version of the test problem DTLZ1), DTLZ1_10 (14-variable 10-objective version of DTLZ1), and DTLZ1_15 (19-variable 15-objective version of DTLZ1). The three problems have a spherical Pareto front in their respective dimensions given by the equation $\sum_{i=1}^{M} f_i = 0.5$, where M is the total number of objective functions. Both algorithms were executed for a total of 100,000 function evaluations for these three

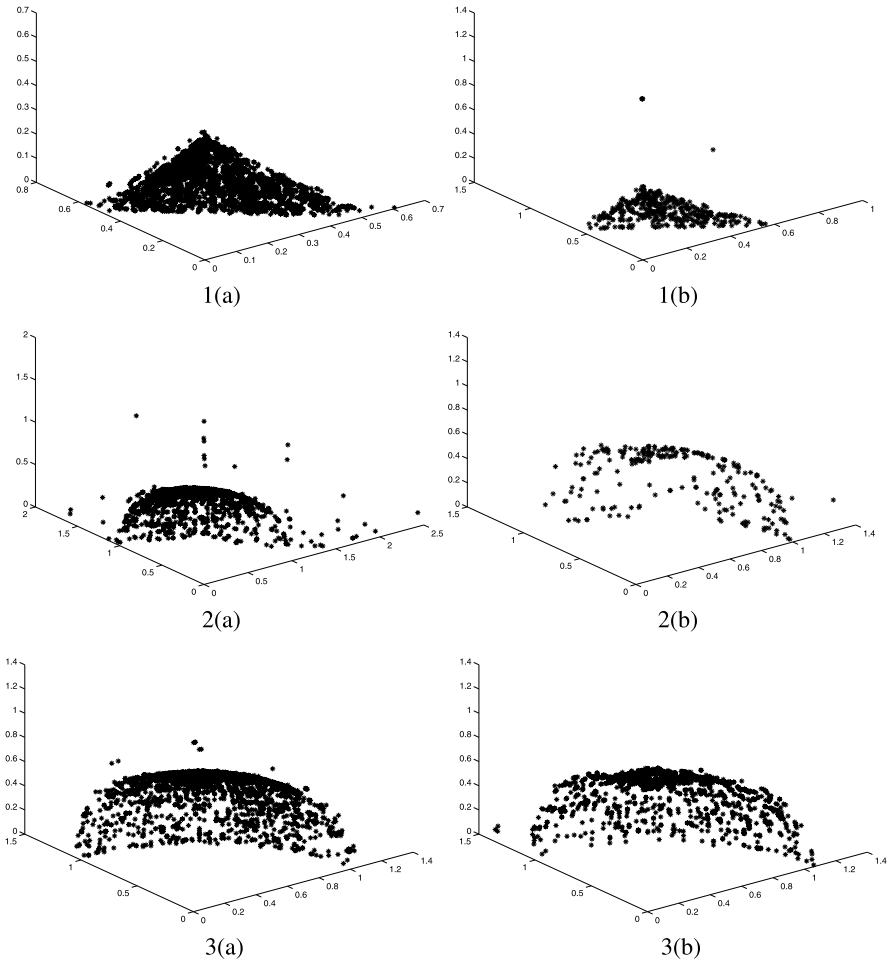


Fig. 2.12 The final nondominated front obtained by (**a**) AMOSA and (**b**) MOSA for the test problems (**1**) DTLZ1, (**2**) DTLZ2, and (**3**) DTLZ3

test problems (for NSGA-II popsize 200, number of generations 500), and the corresponding *Purity*, *Convergence*, and *Minimal Spacing* values are presented in Table 2.7. The *Convergence* values indicate that NSGA-II does not converge to the true PO front where as AMOSA reaches the true PO front for all three cases. The *Purity* measure also indicates this. The results on many-objective optimization problems show that AMOSA performs much better than NSGA-II. These results support the fact that Pareto ranking-based MOEAs such as NSGA-II do not work well on many-objective optimization problems, as pointed out in some recent studies [137, 139].

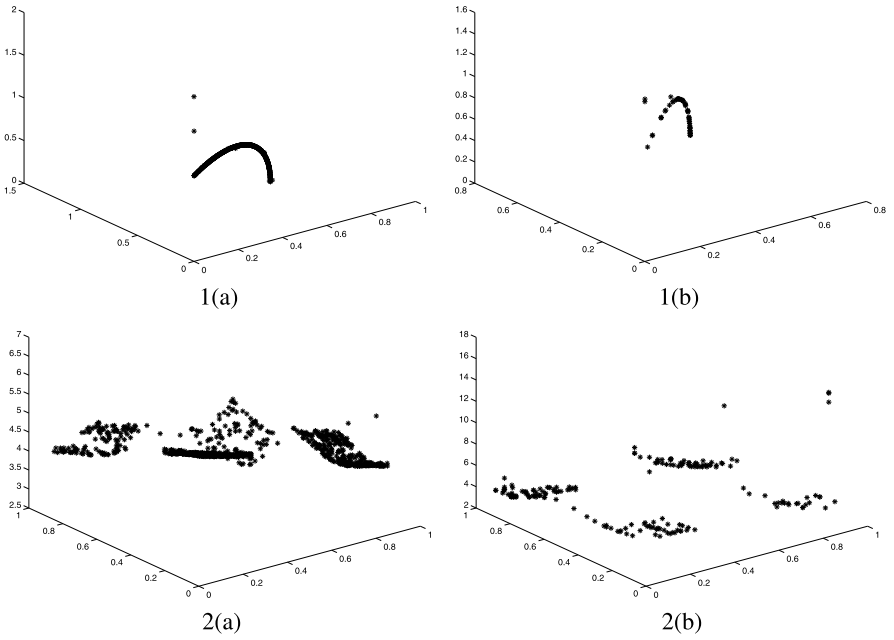


Fig. 2.13 The final nondominated front obtained by (**a**) AMOSA and (**b**) MOSA for the test problems (**1**) DTLZ5 and (**2**) DTLZ6

Table 2.7 *Convergence*, *Purity*, and *Minimal Spacing* measures on the DTLZ2_4, DTLZ1_5, DTLZ1_10, and DTLZ1_15 test functions by AMOSA and NSGA-II

Test problem	Convergence		Purity		Minimal spacing	
	AMOSA	NSGA-II	AMOSA	NSGA-II	AMOSA	NSGA-II
DTLZ2_4	0.2982	0.4563	0.9875 (98.75/100)	0.903 (90.3/100)	0.1876	0.22
DTLZ1_5	0.0234	306.917	1	0	0.1078	0.165
DTLZ1_10	0.0779	355.957	1	0	0.1056	0.2616
DTLZ1_15	0.193	357.77	1	0	0.1	0.271

2.5.4 Discussion on Annealing Schedule

The annealing schedule of an SA algorithm consists of (i) the initial value of temperature (*Tmax*), (ii) the cooling schedule, (iii) the number of iterations to be performed at each temperature, and (iv) the stopping criterion to terminate the algorithm. The initial value of the temperature should be so chosen that it allows the SA to perform a random walk over the landscape. Some methods to select the initial temperature

are given in detail in [274]. In AMOSA [29], as in [257], the initial temperature is set to achieve an initial acceptance rate of approximately 50 % on derogatory proposals. This is described in Sect. 2.5.3.

The functional form of the change in temperature required in SA is determined by the cooling schedule. The most frequently used decrement rule, also used in this chapter, is the geometric schedule given by $T_{k+1} = \alpha \times T_k$, where α $(0 < \alpha < 1)$ denotes the cooling factor. Typically the value of α is chosen in the range between 0.5 and 0.99. This cooling schedule has the advantage of being very simple. Some other cooling schedules available in the literature are logarithmic, Cauchy, and exponential. More details about these schedules are available in [274]. The cooling schedule should be so chosen that it is able to strike a good balance between exploration and exploitation of the search space.

The third component of an annealing schedule is the number of iterations performed at each temperature. It should be so chosen that the system is sufficiently close to the stationary distribution at that temperature. As suggested in [274], the value of the number of iterations should be chosen depending on the nature of the problem. Several criteria for termination of an SA process have been developed. In some of them, the total number of iterations that the SA procedure must execute is given, whereas in some others, the minimum value of the temperature is specified. Detailed discussion on this issue can be found in [274].

2.6 Discussion and Conclusions

Some existing single- and multiobjective optimization techniques are described in this chapter. This includes details of a newly developed simulated annealing-based multiobjective optimization technique, AMOSA [29]. In contrast to most other MOO algorithms, AMOSA selects dominated solutions with a probability that is dependent on the amount of domination, measured in terms of the hypervolume between the two solutions in the objective space. The results of binary-coded AMOSA are compared with those of two existing well-known multiobjective optimization algorithms, NSGA-II (binary-coded) [78] and PAES [161], for a suite of seven two-objective test problems having different complexity levels. In a part of the investigation, comparison of the real-coded version of AMOSA is conducted against a very recent multiobjective simulated annealing algorithm, MOSA [257], and real-coded NSGA-II for six three-objective test problems. Real-coded AMOSA is also compared with real-coded NSGA-II for some 4-, 5-, 10- and 15-objective test problems. Several different comparison measures such as *Convergence*, *Purity*, *Minimal Spacing*, and *Spacing*, and the time taken are used for the purpose of comparison. In this regard, a measure called *displacement* has also been used, which is able to reflect whether a front is close to the PO front as well as its extent of coverage. A complexity analysis of AMOSA is performed. It is found that its complexity is greater than that of PAES but less than that of NSGA-II.

It is seen from the given results that the performance of AMOSA is better than that of MOSA and NSGA-II in a majority of the cases, while PAES performs poorly

in general. AMOSA is found to provide more distinct solutions than NSGA-II in each run for all the problems; this is a desirable feature in MOO. AMOSA is less time consuming than NSGA-II for complex problems such as ZDT1, ZDT2, and ZDT6. Moreover, for problems with many objectives, the performance of AMOSA is found to be much better than that of NSGA-II. This is an interesting and appealing feature of AMOSA, since Pareto ranking-based MOEAs, such as NSGA-II [78], do not work well on many-objective optimization problems, as pointed out in some recent studies [137, 139]. An interesting feature of AMOSA, as in other versions of multiobjective SA algorithms, is that it has a nonzero probability of allowing a dominated solution to be chosen as the current solution in favor of a dominating solution. This makes the problem less greedy in nature, thereby leading to better performance for complex and/or deceptive problems. Note that it may be possible to incorporate this feature as well as the concept of amount of domination into other MOO algorithms in order to improve the performance.

In order to compute similarities/dissimilarities between different objects, several similarity/distance measurements are developed in the literature. These similarity measurements have wide application in data clustering. Depending on the similarity measurements, points can be grouped into different clusters. In the next chapter we discuss different similarity/distance measurements reported in the literature. These definitions of distance measures suitable for handling binary, categorical, ordinal, and quantitative variables are described in detail.

Chapter 3
Similarity Measures

3.1 Introduction

The concepts of similarity and distance are crucial in many scientific applications. Similarity and distance measurements are mostly needed to compute the similarities/distances between different objects, an essential requirement in almost all pattern recognition applications including clustering, classification, feature selection, outlier detection, regression, and search. There exist a large number of similarity measurements in the literature; thus, selecting one most appropriate similarity measurement for a particular task is a crucial issue. The success or failure of any pattern recognition technique largely depends on the choice of the similarity measurement. The similarity measurements vary depending on the data types used. This chapter provides a detailed discussion on a large number of similarity measurements existing in the literature. An excellent tutorial on similarity measurement is available in Ref. [278]. A large portion of this chapter is based on the discussion presented in [278] and is included here for the sake of completeness.

Similarity and dissimilarity between objects can be measured based on several features. Thus, feature selection is required before computing the similarity/distance between objects. The similarity/distance between two variables varies depending on the values of the selected features. A general approach is to find the similarity/distance for each feature, and thereafter the total similarity/distance is determined by aggregating over these individual similarities/distances. Similarity measures can be broadly divided into several categories [46, 97, 186, 278, 293]:

1. Similarity measures for binary variables.
2. Similarity measures for categorical variables.
3. Similarity measures for ordinal variables.
4. Similarity measures for quantitative variables.
5. Dissimilarity between two groups of variables.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_3,

3.2 Definitions

In order to define some similarity/dissimilarity measures, first some features from the data sets have to be extracted. Thereafter, the relationships between different features of different objects are used to compute the similarity among them. Note that the measure of similarity will largely depend on the scale of the measurements and also on the data types.

Similarity primarily quantifies the relationship between different features of different objects. Suppose there are two objects i and j. Let the similarity between these two objects i and j be denoted by s_{ij}. The value of s_{ij} largely depends on the scale of the measurements and also on the data types. The *distance* or *dissimilarity*, on the other hand, measures the difference between two points based on their attribute values. Distance/*dissimilarity* can also be viewed as the disorder between two objects. If the normalized distance and similarity between two objects i and j are d_{ij} and s_{ij}, respectively, then they are related as below:

$$s_{ij} = 1 - d_{ij}.$$

Distance is a quantitative value, and generally it satisfies the following conditions [278]:

- $d_{ij} \geq 0$; the distance between two objects should be always positive or zero.
- $d_{ii} = 0$; distance is zero if and only if it is measured with respect to itself.
- $d_{ij} = d_{ji}$; distance is symmetric.
- $d_{ij} \leq d_{ik} + d_{jk}$; distance should satisfy the triangular inequality.

A distance measure is called a *metric* if it satisfies all the above four conditions. Hence, not all distances are metrics but all metrics are distances.

3.2.1 Need for Measuring Similarity

Similarity measurement between two data points/objects is very important in order to distinguish between different objects [278].

The main motivations for measuring similarity are the following:

- Objects can be distinguished from one to another.
- Any clustering algorithm can be applied to group them based on this similarity measurement (for example, using K-means clustering).
- After clustering the objects form several groups. Thus it is easier to understand the characteristics of each group.
- The characteristics/behavior of the clusters can be explained.
- Clustering of data objects may also help in organizing and retrieving the information in a more organized way.
- Similarity measurement is essential for performing classification. A new object can be classified into the group by using a similarity measurement.

- Similarity measurement can also be used to predict the behavior of a new object based on this grouping information.
- The structure within the data set can be explored by using this similarity measurement.
- Many other data mining techniques such as clustering, classification, and feature selection can be applied to particular data using this similarity measurement.
- The data can be simplified by using the relationship extracted after application of similarity measurement. These simplified data can be used by different data mining techniques in a smooth manner.
- Thus, decision-making and planning become easier after knowing the structure and prediction of the data.

The following sections describe ways of measuring similarity when the variables are of different types.

3.3 Similarity/Dissimilarity for Binary Variables

Binary variables can only take values 0 or 1/yes or no/true or false/positive or negative, etc. In order to measure the similarity or dissimilarity (distance) between two objects/points, each represented by a vector of binary variables, first the total number of occurrences of each value is counted. Below is an example where the distance between two binary variables is calculated [278].

Let there be two instances, H_1 and H_2, from the class "Human". Suppose there are three features "Height $>$ 5.2 ft", "Weight $>$ 60 kg" and "Male or Female". Then, if H_1 has the height of 5.8 ft, weight of 58 kg, and gender is Male, then H_1 is represented as "$H_1 = (1, 0, 1)$". If H_2 has height of 5.0 ft, weight of 50 kg, and gender is Female, then H_2 is represented as " $H_2 = (0, 0, 0)$". Here, both H_1 and H_2 are three-dimensional objects because each object is represented by three variables. Suppose:

$$
\begin{aligned}
m_{00} &= \text{total number of features having 0s in both objects,} \\
m_{01} &= \text{total number of features which have 0s for the } i\text{th object} \\
&\quad \text{and 1s for the } j\text{th object,} \\
m_{10} &= \text{total number of features which have 1s for the } i\text{th object} \\
&\quad \text{and 0s for the } j\text{th object,} \\
m_{11} &= \text{total number of features having 1s in both objects.}
\end{aligned}
$$

Then, the total number of features$(F) = m_{00} + m_{01} + m_{10} + m_{11}$.

Table 3.1 illustrates the concepts of m_{00}, m_{01}, m_{10}, and m_{11}.

For the above examples of H_1 and H_2, $m_{00} = 1$, $m_{01} = 0$, $m_{10} = 2$, and $m_{11} = 0$.

There exist a large number of binary distance measurements, among which the following are the most popular [278]:

Table 3.1 Confusion matrix

H_1	H_2	
	0	1
0	m_{00}	m_{01}
1	m_{10}	m_{11}

- Simple matching distance (d_{ij}) [14, 278]: This distance function is used to compute the dissimilarity between binary variables when the frequency of occurrence of 0 and 1 values are the same in the two variables; For example, the simple matching distance can be used to measure the dissimilarity between two 'gender' variables which have equal information for male and female classes. It is defined as follows:

$$d_{ij} = \frac{m_{01} + m_{10}}{F}.$$

  Here, F denotes the total number of features; For example, the simple matching distance between H_1 and H_2 is $\frac{0+2}{3} = \frac{2}{3} = 0.67$. Here, $m_{01} + m_{10}$ also provides the Hamming distance between two objects.
- Jaccard's distance [142, 278]: Jaccard's coefficient is a similarity measurement, whereas Jaccard's distance is a distance measurement. These measures are mostly used for those binary variables where the values 0 and 1 do not have equal frequency of occurrence. Let us take the example of a library. In a library there is a large collection of books. Thus, there are a greater number of books than a student can take. Suppose the task is to count the total number of similar books taken by two students. Then, it would be a time-consuming task to calculate the total number of not common books taken by two students, as done in case of computing the simple matching distance. Rather, it would be easier to calculate the total number of common books taken by two students. Jaccard's coefficient takes care of this. It is defined as follows:

$$s_{ij} = \frac{m_{11}}{m_{11} + m_{01} + m_{10}}.$$

  The Jaccard distance is computed as:

$$d_{ij} = 1 - s_{ij} = \frac{m_{01} + m_{10}}{m_{01} + m_{10} + m_{00}}.$$

  For the above example the Jaccard coefficient between H_1 and H_2 is 0, and the Jaccard distance is $\frac{2}{3}$. Jaccard's coefficient can also be generalized to apply to nonbinary variables. In that case, it is computed based on set theory. Suppose there are two sets A and B. Then, Jaccard's coefficient between these two sets is computed as follows:

$$s_{AB} = \left| \frac{A \cap B}{A \cup B} \right|.$$

Let us consider the following two sets: $A = \{2, 3, 4, 5\}$ and $B = \{3, 4, 5, 6\}$; so here $A \cup B = \{2, 3, 4, 5, 6\}$ and $A \cap B = \{3, 4, 5\}$. Then, $s_{AB} = \frac{3}{5} = 0.6$.

- Hamming distance [120, 277, 278]: The Hamming distance between two binary vectors is equal to the number of positions where they have distinct digits. This distance is defined as

$$d_{ij} = m_{01} + m_{10}.$$

Again consider the two binary variables $H1$ and $H2$. The Hamming distance between them is 2. Note that, the simple matching distance is equal to the Hamming distance divided by the word length (= total number of variables).

3.4 Distance for Nominal/Categorical Variable

Nominal/categorical variables are those which cannot be measured in a quantitative way [278]; rather, they represent some data categories. In case of nominal or categorical variables, numbers are only used to represent different categories; For example, gender is a nominal variable with value of 1 = male and 2 = female. The main characteristic of nominal or categorical variable is that categories should be labeled "consistently". The order of the labels is not important, but consistency is very important. As an example, gender labeling can be changed to 10 = female, 15 = male, without affecting the representation logic in any way.

However, this labeling should be used consistently while using some categorical variables. One can generate his/her own labeling as long as consistency is preserved. To calculate the distance between two objects having categorical features, first the number of possible categories for each feature has to be counted. In case of two categories, distance measures for binary variables such as simple matching, Jaccard's or Hamming distance can be used. If the number of category is more than two, transformation of these categories into a set of binary variables is needed. There are two methods to transform a categorical variable (with number of categories greater than 2) into binary variables:

1. Each category is represented by a binary variable.
2. Each category is represented by several binary variables.

Use of the above two methods produces different distance measures. Calculation of the distance function is based on the original variables. First the total number of binary variables needed to represent values of a categorical variable is determined. The distance between two categorical objects is then calculated as the ratio of the number of unmatched and the total number of binary variables used for the representation. If p = number of variables having 0s for the ith object and 1s for the jth object and q = number of variables having 1s for the ith object and 0s for the jth object, then

$$d_{ij} = \frac{p + q}{\text{total No. of binary variables to represent categorical variables}}.$$

3.4.1 Method 1: Each Category Is Represented by a Single Binary Variable [278]

In this procedure each category can be represented by a binary variable [278]. The distance between two objects is then calculated as the ratio of the number of unmatched to the total number of binary variables used to represent these categorical variables.

Suppose there are two variables: Color and Gender. Gender has two values: 0 = male and 1 = female. Color has three choices: white, blue, and red. Suppose there are three subjects: Ram (male) wears a red shirt, Rekha (female) wears a white shirt, and Mithi (female) wears a blue shirt. Each value of Color is assigned a binary variable. Let us set the first coordinate as Gender, and the second coordinate as Color (red, white, blue). Then, the feature vectors corresponding to the three objects, Ram, Rekha, and Mithi, are: Ram = (0, (1, 0, 0)), Rekha = (1, (0, 1, 0)), and Mithi = (1, (0, 0, 1)).

To compute the distance between objects, it is first computed for each coordinate. Suppose the Hamming distance is used (= length of different digits). Then:

- Distance(Ram, Rekha) is $(1, 2)$, and the overall distance for the two variables is $1 + 2 = 3$.
- Distance(Ram, Mithi) is $(1, 2)$, and the overall distance for the two variables is $1 + 2 = 3$.
- Distance(Rekha, Mithi) is $(0, 2)$, and the overall distance for the two variables is $0 + 2 = 2$.

The distance between two categorical objects is then calculated as the ratio of the number of unmatched and total number of binary variables used for the representation:

- Distance(Ram, Rekha) is $(1/1, 2/3)$, and the average distance for the two variables is $(1 + 2/3)/2 = 5/6 = 0.83$.
- Distance(Ram, Mithi) is $(1/1, 2/3)$, and the average distance for the two variables is $(1 + 2/3)/2 = 0.83$.
- Distance(Rekha, Mithi) is $(0/1, 2/3)$, and the average distance for the two variables is $(0 + 2/3)/2 = 0.33$.

3.4.2 Method 2: Each Category Is Represented by Several Binary Variables [278]

Suppose there are a total of c categories, then, a total number dv of binary variables are used to represent these categories, where $c < 2^{dv}$. Thus, $dv = \lceil \frac{\log c}{\log 2} \rceil$; For example, in the previous example in order to represent the color of shirts, two binary variables are needed because $\lceil \frac{\log 3}{\log 2} \rceil = \lceil 1.5 \rceil = 2$. Then, the three colors may be

represented as follows: red color by 00, white by 01, and blue by 10. Here again the assignment to dummy variables is somewhat arbitrary but consistent.

Let us consider the previous example, where there are two categorical variables Gender and Color; Gender has two values: 0 = male and 1 = female. Color has three choices: red, white, and blue. In order to represent Gender, a binary dummy variable is used, whereas in order to represent color two dummy binary variables are used. Then, Ram, Rekha, and Mithi are represented as follows: Ram = (0, (0, 0)), Rekha = (1, (0, 1)), Mithi = (1, (1, 0)).

To compute the distance between objects, it must be calculated for each original variable.

Suppose the Hamming distance is used. Then:

- Distance(Ram, Rekha) is (1, 1), and the overall distance is $1 + 1 = 2$.
- Distance(Ram, Mithi) is (1, 1), and the overall distance is $1 + 1 = 2$.
- Distance(Mithi, Rekha) is (0, 2), and the overall distance is $0 + 2 = 2$.

The distance between two categorical objects is the ratio of the number of unmatched and total number of binary variables used for their representation:

- Distance(Ram, Rekha) is $(1/1, 1/2)$, and the average distance for the two variables is $(1 + 1/2)/2 = 3/4$.
- Distance(Ram, Mithi) is $(1/1, 1/2)$, and the average distance for the two variables is $(1 + 1/2)/2 = 3/4$.
- Distance(Mithi, Rekha) is $(0/1, 2/2)$, and the average distance for the two variables is $(0 + 1)/2 = 1/2$.

3.5 Distance for Ordinal Variables

Ordinal variables generally provide an ordering of a given data set in terms of degrees. These variables are primarily used to indicate the relative ordering/ranking of some data points [127, 278]. Thus, the quantitative difference between these variables is not important, but their order is important; For example, consider the 'letter grades' which are used to provide marks to students. AA would be ranked higher than AB, and BB is higher than CC. Here again ranking is important, not the precise distance between AA and AB. Sometimes, numbers can also be used to represent ordinal variables. Below are some examples of ordinal scale:

- Relative grading: AA = excellent, AB = good, BB = average, DD = poor.
- Rank of priority: 1 = best, higher value indicates low priority.
- Rating of satisfaction: 1 = very dissatisfied, 100 = very satisfied.

In order to compute the distances/dissimilarities between ordinal variables, the following methods are mostly used: normalized rank transformation, Spearman distance, footrule distance, Kendall distance, Cayley distance, Hamming distance, Chebyshev/Maximum distance, and Minkowski distance. Below, some of these distances are described.

3.5.1 Normalized Rank Transformation

Here, using some normalization techniques [278], the given ordinal variables are first converted into quantitative variables [127, 237]. Thereafter, the distance between quantitative variables can be calculated by using any method described in Sect. 3.6. In order to determine the distance between two ordinal variables, these variables are first transformed into a ratio scale by applying the following steps:

- The ordinal value is converted into rank ($r = 1$ to R).
- The rank is normalized to a standardized value of zero to one [0, 1] by using the following formula:

$$x = \frac{r-1}{R-1}.$$

- The distances between these ordinal variables can be computed by considering the ordinal value as a quantitative variable. Any distance measure for quantitative variables can be used.

This distance can only be applied if an ordinal variable can be normalized as a quantitative variable. If such types of conversion need to be avoided, then other distance measures such as the Spearman distance, Kendall distance, Cayley distance, and Hamming distance for ordinal variables or the Ulam distance, Chebyshev/Maximum distance can be used.

Let us consider a conference paper review system; each reviewer has to review some papers; for each paper there is a questionnaire asking level of acceptance in terms of appropriateness, clarity, originality, soundness, and meaningful comparison. Each five acceptance criterion has five values: $-2 =$ very dissatisfied, $-1 =$ dissatisfied, $0 =$ average, $1 =$ good, $2 =$ excellent. Suppose the scores of two reviewers for a given paper are as follows:

	Appropriateness	Clarity	Originality	Soundness	Meaningful comparison
Rev1	0	−1	0	1	0
Rev2	1	0	1	0	0

Suppose that the aim is to measure the distance/dissimilarity between these two reviewers according to the answers. First, the ordinal scale is transformed into a ratio scale. The original index ($i = -2$ to 2) is ordered and converted into a rank ($r = 1$ to 5). The highest rank is $R = 5$. Then, the rank is normalized to a value [0, 1]. For instance, in position 1 of Rev1, there is $i = 0$, which converted to rank becomes $r = 3$, and the normalized rank is $\frac{3-1}{5-1} = 2/4 = 0.5$. The following conversions hold:

- Original index: -2 -1 0 1 2.
- Converted to rank: 1 2 3 4 5.
- Normalized rank: 0 $\frac{1}{4}$ $\frac{2}{4}$ $\frac{3}{4}$ 1.

Using the normalized rank as the new values, the coordinates of Rev1 become $(\frac{2}{4}\ \frac{1}{4}\ \frac{2}{4}\ \frac{3}{4}\ \frac{2}{4})$ and Rev2 become $= (\frac{3}{4}\ \frac{2}{4}\ \frac{3}{4}\ \frac{2}{4}\ \frac{2}{4})$. The Euclidean distance between Rev1 and Rev2 is

$$d_{12} = \sqrt{\left(\frac{2}{4} - \frac{3}{4}\right)^2 + \left(\frac{1}{4} - \frac{2}{4}\right)^2 + \left(\frac{2}{4} - \frac{3}{4}\right)^2 + \left(\frac{3}{4} - \frac{2}{4}\right)^2 + \left(\frac{2}{4} - \frac{2}{4}\right)^2} = 0.5.$$

3.5.2 Spearman Distance

The Spearman distance [127, 237, 278] is the square of the Euclidean distance between two ordinal vectors, computed as:

$$d_{ij} = \sum_{p=1}^{n} (x_{ip} - x_{jp})^2,$$

where n denotes the total number of ordinal features.

Suppose two persons P1 and P2 are asked to provide their preference on color. Let the choice of P1 be [Blue, White, Red] and the choice of P2 be [Red, White, Blue]. Suppose the pattern vector is [Blue, White, Red]; then the following rank vectors are obtained $A = [1, 2, 3]$ and $B = [3, 2, 1]$. Thus, the two vectors are represented as two points in a three-dimensional space. Point P1 has coordinates (1, 2, 3), and point P2 has coordinates (3, 2, 1).

Then, the Spearman distance of preference between P1 and P2 is $d_{12} = (1 - 3)^2 + (2 - 2)^2 + (3 - 1)^2 = 4 + 0 + 4 = 8$.

3.5.3 Footrule Distance

The footrule distance [127, 237, 278] is the summation of the absolute differences between the feature values of two ordinal vectors. The equation for this distance is very similar to that of city block distance or Manhattan distance for quantitative variables. Another name for this distance is the Spearman footrule distance. This distance is defined as:

$$d_{ij} = \sum_{k=1}^{n} |x_{ik} - x_{jk}|.$$

For, the previous example the rank vectors of P1 and P2 are (1, 2, 3) and (3, 2, 1), respectively. Then, the footrule distance between these two points is

$$d_{12} = 2 + 0 + 2 = 4.$$

3.6 Distance for Quantitative Variables

Quantitative variables can be measured on a numeric scale [278]. Thus, they are different from categorical variables, which primarily represent some categories, as well as from ordinal variables, which represent ordering of variables. These quantitative variables are measured as a number. Thus, any arithmetic operations can be applied to quantitative variables. Examples of quantitative variables are height, weight, age, amount of money, GPA, salary, temperature, area, etc.

Suppose the cost, weight, and time taken need to be measured. Let us use three quantitative variables to distinguish one object from another. The three features are cost (rounded down to thousands of rupees), time (in hours) and weight (in kg). Suppose our findings are Machine 1: (Rs. 1200, 4 hours, 10 kg); Machine 2: (Rs. 1700, 3 hours, 15 kg).

The two objects can be represented as points in three dimensions. Machine 1 has coordinates (1200, 4, 10) and Machine 2 has coordinates (1700, 3, 15). The dissimilarity (or similarity) between these two objects is then based on these coordinates.

3.6.1 Euclidean Distance

The most commonly used distance measure for quantitative variables is the Euclidean distance [277, 278]. In general, the term 'distance' refers to Euclidean distance. This primarily measures the root of the square differences between the coordinates of a pair of objects.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}.$$

For example, if there are two quantitative variables $A = (2, 3, 4)$ and $B = (5, 9, 10)$, then the Euclidean distance between them is

$$d_{AB} = \sqrt{\left\{(2-5)^2 + (3-9)^2 + (4-10)^2\right\}} = \sqrt{\{9 + 36 + 36\}} = 9.$$

Euclidean distance is a special case of Minkowski distance with $\lambda = 2$.

3.6.2 Minkowski Distance of Order λ

This is again a general distance measure. Depending on the value of the variable λ, this distance reduces to several different distance functions. When $\lambda = 1$, the Minkowski distance [277, 278] reduces to the city block distance; when $\lambda = 2$, the Minkowski distance reduces to the Euclidean distance; when $\lambda = \propto$, the Minkowski

distance reduces to the Chebyshev distance. The Minkowski distance can be used for both ordinal and quantitative variables.

It is defined as follows:

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{n}(x_{ik} - x_{jk})^{\lambda}}.$$

For example, if there are two quantitative variables $A = (2, 3, 4)$ and $B = (5, 9, 10)$, then the Minkowski distance of order 3 between them is

$$d_{AB} = \sqrt[3]{\{(2-5)^3 + (3-9)^3 + (4-10)^3\}} = -7.714.$$

3.6.3 City Block Distance

Other names of this distance are the Manhattan distance,/ boxcar distance, and/ absolute value distance [277, 278]. It primarily measures the absolute differences between the coordinates of a pair of objects, calculated as

$$d_{ij} = \sum_{k=1}^{n} |x_{ik} - x_{jk}|.$$

Consider the above example. Here, the city block distance between objects A and B is $d_{AB} = |2-5| + |3-9| + |4-10| = 3 + 6 + 6 = 15$. The city block distance is a special case of Minkowski distance with $\lambda = 1$.

3.6.4 Chebyshev Distance

Chebyshev/Tchebyschev distance [277, 278] is again applicable for both ordinal and quantitative variables. Another name for this distance is the maximum value distance. It calculates the maximum of the absolute differences between the features of a pair of objects. The formula for this distance is:

$$d_{ij} = \max_{k} |x_{ik} - x_{jk}|.$$

Consider the two points $A = (2, 3, 4)$ and $B = (5, 9, 10)$. Then, the Chebyshev distance between A and B is $d_{AB} = \max\{|2-5|, |3-9|, |4-10|\} = \max\{3, 6, 6\} = 6$. The Chebyshev distance is a special case of the Minkowski distance with $\lambda = \infty$ (taking the limit).

3.6.5 Canberra Distance

The Canberra distance [82, 173, 278] measures the sum of the absolute fractional differences between the features of a pair of objects. Each term of the fractional difference ranges between 0 and 1. Note that, if both features are zeros, then it is assumed that $\frac{0}{0} = 0$. This distance is very sensitive to a small change when both coordinates are near to zero. It is calculated as

$$d_{ij} = \sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|}.$$

For example, for the two vectors $A = (2, 3, 4)$ and $B = (5, 9, 10)$, the Canberra distance between these two points is $d_{AB} = \frac{3}{7} + \frac{6}{12} + \frac{6}{14} = 0.4286 + 0.5 + 0.4286 = 1.3571$.

3.6.6 Bray-Curtis Distance

The Bray-Curtis distance [82, 173, 278] is sometimes also called the Sorensen distance. This is a distance measure commonly used in botany, ecology, and environmental sciences. When all the coordinates are positive, this distance function takes a value between zero and one. A distance of zero corresponds to exactly similar coordinates. The Bray-curtis distance is undefined if both objects have zero coordinates. This distance function is calculated using the absolute differences divided by the summation.

$$d_{ij} = \frac{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}{\sum_{k=1}^{n} (x_{ik} + x_{jk})}.$$

For two vectors $A = (2, 3, 4)$ and $B = (5, 9, 10)$, the Bray-Curtis distance between these two points is $d_{AB} = \frac{|2-5|+|3-9|+|4-10|}{(2+5)+(3+9)+(4+10)} = \frac{3+6+6}{7+12+14} = \frac{15}{33} = 0.45$.

3.6.7 Angular Separation

This is sometimes termed the coefficient of correlation [82, 173, 278]. It primarily measures the cosine angle between two vectors. This is a similarity measurement taking values in the range $[-1, +1]$ rather than a distance measure. When two vectors are similar, the angular separation will take higher values. Again in order to define angular separation, it is assumed that $\frac{0}{0} = 0$. The angular separation between two vectors is defined as:

$$s_{ij} = \frac{\sum_{k=1}^{n} (x_{ik} \times x_{jk})}{\sqrt{(\sum_{k=1}^{n} x_{ik}^2 \times \sum_{r=1}^{n} x_{jr}^2)}}.$$

For two vectors $A = (2, 3, 4)$ and $B = (5, 9, 10)$, the angular separation similarity between these two points is $s_{AB} = \frac{2*5+3*9+4*10}{\sqrt{(2^2+3^2+4^2)(5^2+9^2+10^2)}} = \frac{77}{77.29} = 0.99624$.

The origin of the name of this distance is as follows: The cosine angle between two vectors is represented as the dot product divided by the lengths of the two vectors:

$$\cos\theta = \frac{AB}{|A|.|B|}.$$

The length of a vector A, which is also termed the modulus, is the root of the square of its coordinates, $|A| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$. Thus,

$$\cos\theta = \frac{a_1b_1 + a_2b_2 + \cdots + a_nb_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}\sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}}.$$

Changing the notation of vectors into coordinates, the angular separation formula become:

$$s_{ij} = \frac{\sum_{k=1}^{n} x_{ik}x_{jk}}{\sqrt{(\sum_{k=1}^{n} x_{ik}^2 \sum_{r=1}^{n} x_{jr}^2)}}.$$

3.6.8 Correlation Coefficient

The correlation coefficient [82, 173, 278], which is another similarity measurement rather than a distance measurement, is also termed the linear correlation coefficient or/Pearson correlation coefficient. It is a special case of angular separation taking values in the range $[-1, +1]$. It is angular separation standardized by centering the coordinates on its mean value, defined as

$$s_{ij} = \frac{\sum_{k=1}^{n}(x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j)}{\sqrt{(\sum_{k=1}^{n}(x_{ik} - \overline{x}_i)^2 \sum_{r=1}^{n}(x_{jr} - \overline{x}_j)^2)}},$$

where $\overline{x}_i = \frac{1}{n}\sum_{k=1}^{n} \overline{x}_{ik}$ and $\overline{x}_j = \frac{1}{n}\sum_{k=1}^{n} \overline{x}_{jk}$. Here it has been assumed that $\frac{0}{0} = 0$.

For two vectors $A = (2, 3, 4)$ and $B = (5, 9, 10)$, the correlation coefficient between these two points is defined, using $\overline{A} = \frac{2+3+4}{3} = \frac{9}{3} = 3$ and $\overline{B} = \frac{5+9+10}{3} = \frac{24}{3} = 8$, as

$$\begin{aligned} s_{AB} &= \frac{(2-3)\times(5-8) + (3-3)\times(9-8) + (4-3)\times(10-8)}{\sqrt{((2-3)^2 + (3-3)^2 + (4-3)^2)\times((5-8)^2 + (9-8)^2 + (10-8)^2)}} \\ &= \frac{3+0+2}{\sqrt{(1+0+1)\times(9+1+4)}} \\ &= \frac{5}{\sqrt{28}} = \frac{5}{5.29} = 0.9452. \end{aligned} \tag{3.1}$$

3.6.9 Mahalanobis Distance

Another important distance measure is the Mahalanobis distance [82, 277, 278], also termed as the quadratic distance. It generally determines the difference/distance between two groups of objects. Suppose there are two groups with means $\overline{x}_i$ and $\overline{x}_j$, the Mahalanobis distance is calculated by the following formula:

$$d_{ij} = \left((\overline{x}_i - \overline{x}_j)^T \times S^{-1} \times (\overline{x}_i - \overline{x}_j)\right)^{\frac{1}{2}}.$$

The points within each group should have the same number of features, but the number of points in each group may vary (i.e., the number of columns in each group should be the same, but the number of rows can vary).

3.7 Normalization Methods

The process of transforming the value of a feature/attribute into a fixed range, usually 0 to 1, is generally referred to as normalization [278]. Here, a brief discussion on how to transform values of feature/attribute into the range from 0 to 1 or [0, 1] is provided.

Let us assume that there is a feature/attribute which takes values in the range $[f_{max}$ to $f_{min}]$. Suppose it is required to transform this value into the range [0, 1]. Let the original feature be denoted by f and the normalized feature be denoted by δ. There are several ways to normalize a particular feature value. First, all the negative values are converted to positive, and then each number is divided by some value which is larger than the numerator. Below some such techniques for transforming the original feature value into a normalized value are discussed in brief [278].

- If the range of the given feature vector is known a priori then the feature can be normalized by using the following equation:

$$\delta = \frac{f - f_{min}}{f_{max} - f_{min}}, \tag{3.2}$$

  where f_{min} denotes the minimum value of this feature vector and f_{max} denotes its maximum value. By using the above transformation one can normalize the value in to the range [0, 1]. If $f_{min} = f$, then $\delta = 0$; if $f = f_{max}$, then $\delta = 1$. If for a given data set $f_{min} = 0$, then Eq. 3.2 can be simplified to: $\delta = \frac{f}{f_{max}}$.
- Suppose the maximum value of a particular feature is not known but it can be assumed that the feature will always take the value 0 or some positive value. If there are a total of n possible values for that particular feature, then normalization of the ith feature can be done as follows:

$$\delta_i = \frac{f}{\sum_{i=1}^{n} f_i}. \tag{3.3}$$

Note that normalization using Eq. 3.3 will provide much lower values than normalization using Eq. 3.2, because $f_{max} \leq (\sum_{i=1}^{n} f_i)$.

- If the maximum value of a particular feature is not known and it also takes negative values, then normalization can be done using the following equation:

$$\delta_i = \frac{|f|}{\sum_{i=1}^{n} |f_i|}. \tag{3.4}$$

- Normalization of negative values: For data sets with positive or zero values, the above-discussed normalization techniques work fine. However, if the data set contains some negative values, then these numbers first have to be shifted by adding the absolute value of the minimum of these numbers such that the most negative one will become zero and all other numbers become positive. After that, any of the above-mentioned techniques can be applied to normalize the data.

  Suppose our data set is $\{-6, -9, 0, 6, 7\}$. The minimum of these numbers is -9. Now the minimum of all these numbers, $|-9| = 9$, has to be added to all five numbers. Then, the modified numbers are $\{-6+9, -9+9, 0+9, 6+9, 7+9\} = \{3, 0, 9, 15, 16\}$. Now, any of the above-mentioned techniques can be applied to this data set to normalize it.
- z-Score normalization: This is a statistical normalization technique. Here the assumption is that the data follows a normal distribution. By using the below-mentioned transformation, any data following a normal distribution can be converted into another normal distribution with mean zero and variance $= 1$. The standard deviation of the ith attribute is

$$Z = \frac{X - u}{s}. \tag{3.5}$$

  Here, the original data set is X, and the transformed data set is Z. s is the calculated standard deviation, and u is the mean of the data.

3.8 Summary

In this chapter we have discussed in detail several distance measures available in the literature. We have provided the definitions of similarity and distance measures. Different distance measures for binary variables, categorical variables, ordinal variables, and quantitative variables are discussed. Sufficient examples are provided to make them understood. Finally, the last part of the chapter contains a discussion on normalization. Different normalization techniques are elaborately explained. A good review of different similarity measures is available in [278], on which a large part of this chapter is based.

Clustering [143, 281] is an important problem in data mining and pattern recognition. It has applications in a large number of fields. In the next chapter, some existing approaches to clustering are discussed in detail.

Chapter 4
Clustering Algorithms

4.1 Introduction

Computational pattern recognition can be viewed as a twofold task, comprising learning the invariant properties of a set of samples characterizing a class, and of deciding that a new sample is a possible member of the class by noting that it has properties common to those of the set of samples [24]. The latter classification task can be either supervised or unsupervised depending on the availability of labeled patterns. Clustering is an important unsupervised classification technique where a number of patterns, usually vectors in a multidimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. Cluster analysis is a difficult problem due to the variety of ways of measuring the similarity and dissimilarity concepts, which do not have any universal definition. A good review of clustering can be found in [145].

For partitioning a data set, one has to define a measure of similarity or proximity based on which cluster assignments can be done. The measure of similarity is usually data dependent. It may be noted that, in general, one of the fundamental features of shapes and objects is symmetry, which is considered to be important for enhancing their recognition [12]. As symmetry is commonly found in the natural world, it may be interesting to exploit this property while clustering a data set [27, 28, 58, 242, 243, 266].

The problem of clustering requires appropriate parameter selection (e.g., model and model order) and efficient search in complex and large spaces in order to attain optimal solutions. Moreover, defining a single measure of optimality is often difficult in clustering, leading to the need to define multiple objectives that are to be simultaneously optimized. This not only makes the process computationally intensive, but also leads to the possibility of missing the exact solution. Therefore, the application of sophisticated metaheuristic optimization techniques, both single, and multiobjective, that provide robust, fast, and close approximate solutions, seems appropriate and natural.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_4,

In this chapter we discuss some of the popularly used clustering algorithms. First, a discussion of traditional clustering algorithms including partitional clustering techniques, hierarchical clustering techniques, density-based clustering techniques, and grid-based clustering techniques is provided. Thereafter, we describe some recently developed clustering algorithms. This is followed by a discussion on evolutionary approaches to clustering. In the last section of the chapter we discuss different multiobjective approaches for solving clustering problems.

4.2 Preliminaries

4.2.1 Definition of Clustering

Clustering [143, 281], also known as *unsupervised classification*, is an important problem in data mining and pattern recognition. It has applications in a large number of fields. In clustering, a set of unlabeled patterns, usually vectors in a multidimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. Mathematically, clustering partitions the input space into K regions based on some similarity/dissimilarity metric, where the value of K may or may not be known a priori. The aim of any clustering technique is to evolve a partition matrix $U(X)$ of the given data set X (consisting of, say, n patterns, $X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$) such that

$$\sum_{j=1}^{n} u_{kj} \geq 1 \quad \text{for } k = 1, \ldots, K,$$

$$\sum_{k=1}^{K} u_{kj} = 1 \quad \text{for } j = 1, \ldots, n, \quad \text{and}$$

$$\sum_{k=1}^{K} \sum_{j=1}^{n} u_{kj} = n.$$

The partition matrix $U(X)$ of size $K \times n$ may be represented as $U = [u_{kj}]$, $k = 1, \ldots, K$ and $j = 1, \ldots, n$, where u_{kj} is the membership of pattern $\overline{x}_j$ to cluster C_k. In crisp partitioning $u_{kj} = 1$ if $\overline{x}_j \in C_k$, otherwise $u_{kj} = 0$.

4.2.2 Some Clustering Techniques

There exist a large number of clustering techniques in the literature [145]. Traditional clustering algorithms can be classified into four classes [143]: *hierarchical*,

partitional, *density based*, and *grid based*. Some examples of *hierarchical* clustering methods are the single linkage clustering algorithm, average linkage clustering algorithm, and complete linkage clustering algorithm. The *K*-means clustering algorithm is an example of a *partitional* method [281]. The *DBSCAN* [93] clustering algorithm is an example of a *density-based* clustering technique. *STING* [291] is an example of a *grid*-based clustering technique.

4.3 Partitional Clustering Techniques

The partitioning clustering methods generally produce a number of clusters, where each point belongs to a particular cluster. A cluster is generally represented by a cluster center or centroid which may be viewed as a summary description of all the points in that particular cluster. The definition of the cluster center/centroid depends on the type of points which are being clustered; For example, in case of real-valued points, the cluster center is often defined as the arithmetic mean of the attribute vectors for all objects within the cluster. In case of clustering of documents, a centroid may be the list of those keywords that occur in some minimum number of documents within the cluster. If the number of clusters is large, the centroids can be further clustered to produce a hierarchy within a data set. Some popular partitional clustering techniques are the *K*-means algorithm, *K*-medoids clustering, fuzzy *C*-means clustering, and expectation maximization clustering. In the following subsections, brief descriptions of these algorithms (some portions are based on the discussions in [24]) are provided.

4.3.1 K-Means Clustering Technique

The *K*-means algorithm [96, 143] is an iterative clustering technique that evolves K crisp, compact, and hyperspherical clusters in a data set such that the measure

$$J = \sum_{j=1}^{n} \sum_{k=1}^{K} u_{kj} \times \|\overline{x}_j - \overline{z}_k\|^2 \tag{4.1}$$

is minimized. Here, u_{kj} is equal to 1 if the jth point belongs to cluster k, and 0 otherwise; $\overline{z}_k$ denotes center of the cluster k, and $\overline{x}_j$ denotes the jth point of the data. In *K*-means, cluster centers are first initialized to K randomly chosen points from the data set. The initial partitioning is formed using the minimum distance criterion. The cluster centers are subsequently updated to the means of the respective clusters. The process of partitioning followed by updating centers is repeated until one of the following becomes true: (a) the cluster centers do not change in subsequent iterations, (b) the J value becomes smaller than a threshold, or (c) the maximum number

Step 1: Choose K cluster centers $z_1, z_2, \ldots, z_k$ randomly from the n points $x_1, x_2, \ldots, x_n$.
Step 2: Assign point x_i, $i = 1, 2, \ldots, n$ to cluster C_j, $j \in 1, 2, \ldots, k$ iff
$\|x_i - z_j\| < \|x_i - z_p\|, \quad p = 1, 2, \ldots, K, \quad$ and $\quad j \neq p$
Ties are resolved arbitrarily.
Step 3: Compute new cluster centers $z_1^*, z_2^*, \ldots, z_k^*$ as follows:
$z_i^* = \frac{\sum_{x_j \in C_i} x_j}{n_i}, \quad i = 1, 2, \ldots, K$
where n_i is the number of elements belonging to cluster C_i.
Step 4: If $z_i^* = z_i$, $i = 1, 2, \ldots, K$, then terminate. Otherwise, $z_i = z_i^*$, $i = 1, 2, \ldots, K$ and continue from step 2.

Fig. 4.1 The K-means algorithm

of iterations have been exhausted. The different steps of the K-means algorithm are enumerated in Fig. 4.1.

In general, if the process does not terminate in step 4 normally, then it is executed for a maximum fixed number of iterations. The rules for updating the cluster centers are obtained by differentiating J with respect to the centers, and equating the differential to zero. This analysis is as follows, the essential purpose being to minimize J.

$$\frac{dJ}{d\overline{z}_k} = 2\sum_{j=1}^{n} u_{kj}(\overline{x}_j - \overline{z}_k)(-1) = 0, \quad k = 1, 2, \ldots, K, \tag{4.2}$$

$$\sum_{j=1}^{n} u_{kj}\overline{x}_j - \overline{z}_k \sum_{j=1}^{n} u_{kj} = 0, \tag{4.3}$$

$$\overline{z}_k = \frac{\sum_{j=1}^{n} u_{kj}\overline{x}_j}{\sum_{j=1}^{n} u_{kj}}. \tag{4.4}$$

Note that in crisp clustering $\sum_{j=1}^{n} u_{kj}$ is nothing but the number of elements belonging to cluster k, i.e., n_k. Hence, the update equation boils down to the following:

$$\overline{z}_i^* = \frac{\sum_{\overline{x}_j \in C_i} \overline{x}_j}{n_i}. \tag{4.5}$$

In order to ensure that this indeed provides a minima of J, the second derivative,

$$\frac{d^2 J}{d\overline{z}_k^2} = 2\sum_{j=1}^{n} u_{kj},$$

should be positive. Note that the right-hand side of the above equation is positive, which indicates that using the update equation will indeed result in a minimum value of J.

The K-means algorithm has several limitations. It has been shown in [251] that the K-means algorithm may converge to values that are not optimal, depending on the choice of the initial cluster centers. Also, global solutions of large problems

cannot be found with a reasonable amount of computation effort [260]. Moreover, this technique assumes that clusters are of hyperspherical shape and more or less equal in size. Finally, K-means is not robust to outliers. It is because of these factors that several other methods for clustering have been developed.

4.3.2 K-Medoid Clustering Technique

The K-medoid clustering algorithm [279] is another popular partitional clustering algorithm. This is an extension of the K-means algorithm where the concept of medoid (a representative data point from the data set) is used instead of the cluster mean. A medoid of a set of points is the point whose average dissimilarity to all the data points is minimal, i.e., it is the most centrally located point in the set.

Similar to K-means, K-medoid also tries to minimize the total squared error of the whole data set, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. As the K-medoid clustering technique minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, it is more robust to noise and outliers as compared with the K-means clustering technique. The K-medoid clustering technique clusters the data set of n objects into number K of clusters where the value of K is known.

The steps of the K-medoid clustering technique closely follow those of K-means. The basic steps are listed below:

- *Initialize*: Number K of points are randomly selected from the set of n points and initialized as the K medoids.
- *Assignment step*: Each data point is assigned to the closest medoid (here the distance measure depends on the user requirement and type of data).
- *Update step*: Calculate the total squared error for the new partitioning.

Repeat steps 2 and 3 until there is no change in the assignments.

4.3.3 Fuzzy C-Means Clustering Algorithm

Fuzzy C-means [279] is a widely used technique that uses the principles of fuzzy sets to evolve a fuzzy partition matrix for a given data set. The set of all $c \times n$, where c is the number of clusters, nondegenerate constrained fuzzy partition matrices, denoted by M_{fcn}, is defined as

$$M_{fcn} = \left\{ U \in R^{c \times n} \Big| \sum_{i=1}^{c} u_{ik} = 1,\ \sum_{k=1}^{n} u_{ik} > 0, \right.$$
$$\left. \forall i \text{ and } u_{ik} \in [0,1];\ 1 \le i \le c;\ 1 \le k \le n \right\}.$$

The minimizing criterion used to define good clusters for fuzzy C-means partitions is the FCM function, defined as

$$J_\mu(U, Z) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^\mu D^2(\overline{z}_i, \overline{x}_k). \tag{4.6}$$

Here, $U \in M_{fcn}$ is a fuzzy partition matrix; $\mu \in [1, \infty]$ is the weighting exponent on each fuzzy membership; $Z = [\overline{z}_1, \ldots, \overline{z}_c]$ represents c cluster centers; $\overline{z}_i \in R^N$; and $D(\overline{z}_i, \overline{x}_k)$ is the distance of $\overline{x}_k$ from the ith cluster center. The fuzzy C-means theorem [37] states that, if $D(\overline{z}_i, \overline{x}_k) > 0$, for all i and k, then (U, Z) may minimize J_μ, only if $\mu > 1$ and

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D(\overline{z}_i, \overline{x}_k)}{D(\overline{z}_j, \overline{x}_k)}\right)^{\frac{2}{\mu-1}}}, \quad \text{for } 1 \le i \le c,\ 1 \le k \le n \tag{4.7}$$

and

$$\overline{z}_i = \frac{\sum_{k=1}^{n} (u_{ik})^\mu \overline{x}_k}{\sum_{k=1}^{n} (u_{ik})^\mu}, \quad 1 \le i \le c. \tag{4.8}$$

A common strategy for generating the approximate solutions of the minimization problem in Eq. 4.6 is by iterating through Eqs. 4.7 and 4.8 (also known as the Picard iteration technique). A detailed description of the FCM algorithm can be found in [37].

Although in fuzzy clustering the final output is a crisp partitioning, the user can utilize the information contained in the partition matrix [23]. The FCM algorithm shares the problems of the K-means algorithm in that it also gets stuck at local optima depending on the choice of the initial clusters, and requires the number of clusters to be specified a priori.

4.4 Distribution-Based Clustering Approach

This category of clustering algorithms assumes that the clusters follow some specific distribution. The expectation maximization (EM) [80] algorithm is a prominent example of this category.

The expectation maximization clustering technique [80] is based on mixture models. It is an iterative approach aiming to determine the parameters of the probability distribution which have the maximum likelihood of their attributes. The basic concept of this algorithm is to divide the n number of data points into K number of clusters so that the total probabilities of occurrence of all the clusters is maximized. The inputs to the algorithm are: the data set (X), the total number of clusters (K), the accepted convergence error (E), and the maximum number of iterations. Two steps are executed in each iteration. The first is the E-step (expectation), where, the probability of each point belonging to each cluster is calculated. The second one is

the M-step (maximization), which re-estimates the parameter vector of the probability distribution of each class after the re-allocation [80]. The algorithm terminates after completing a maximum number of iterations or when the parameters of the distribution converge. This statistical clustering technique assumes that given clusters follow a Gaussian probability distribution (normal distribution); thus, if clusters do not follow this distribution, the EM algorithm will fail in providing the appropriate partitioning (the below summary of the algorithm is inspired by the description given in [1]).

- Initialization: Let there be a total K number of clusters. Each cluster j is represented by a parameter vector (Θ), composed by the mean (μ_j) and by the covariance matrix (P_j). These parameters characterize the features of the Gaussian probability distribution (normal). They are used to distinguish between the observed and unobserved entities of the data set X.

$$\theta(t) = \mu_j(t), P_j(t), \quad j = 1, \ldots, K.$$

  Initially, i.e., at ($t = 0$), the EM algorithm randomly generates the initial values of mean (μ_j) and of covariance matrix (P_j). Thereafter, successive iterations of the EM algorithm aim to approximate the parameter vector (Θ) of the real distribution. An alternative way to initialize EM is by using clusters obtained by a hierarchical clustering technique.
- E-step: This step calculates the probability of each data point belonging to each cluster ($P(C_j|x_k)$). Each data point is represented by a feature vector (x_k). The membership of the points of each cluster is calculated by the likelihood of each feature of that point in comparison with the features of the other points of cluster C_j.

$$P(C_j|x) = \frac{|\sum_j(t)|^{-\frac{1}{2}} \exp^{n_j} P_j(t)}{\sum_{k=1}^{M} |\sum_j(t)|^{-\frac{1}{2}} \exp^{n_j} P_k(t)}.$$

- M-step: This step calculates the parameters of the probability distribution of each cluster for the next step. First, the mean (μ_j) of class j is calculated using the mean of all data points based on the relevance degree of each data point.

$$\mu_j(t+1) = \frac{\sum_{k=1}^{N} P(C_j|x_k)x_k}{\sum_{k=1}^{N} P(C_j|x_k)}.$$

  Thereafter, Bayes theorem is utilized to compute the covariance matrix for the next iteration. This states that $P(A|B) = P(B|A) * P(A)P(B)$. Now, based on

[1] http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/Expectation_Maximization_%28EM%29

the conditional probabilities of the class occurrence, the following holds:

$$\sum_j (t+1) = \frac{\sum_{k=1}^{N} P(C_j|x_k)(x_k - \mu_j(t))(x_k - \mu_j(t))}{\sum_{k=1}^{N} P(C_j|x_k)}.$$

The probability of occurrence of each class is now calculated using the mean of probabilities (C_j) based on the relevance degree of each point from the class.

$$P_j(t+1) = \frac{1}{N} \sum_{k=1}^{N} P(C_j|x_k).$$

The features characterize the parameter vector Θ which represents the probability distribution of each class. This vector Θ is used in the next iteration.

4.5 Hierarchical Clustering Techniques

These clustering techniques build a hierarchy of clusters [279]. There can be two types of hierarchical clustering techniques:

- Agglomerative: This is a "bottom-up" approach where each data point starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top-down" approach; it starts by assuming that all data points are in one cluster, and splits are performed recursively as one moves down the hierarchy.

These merges and splits to generate some new clusters are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

Linkage Clustering Algorithms The single linkage clustering technique is a non-iterative method based on a local connectivity criterion [145, 279]. Instead of using one single data point $\overline{x}$ in a data set, the single linkage algorithm processes sets of n^2 relationships, say $\{r_{jk}\}$, between pairs of objects represented by the data. The value of r_{jk} represents the extent to which the object j and k are related in the sense of some binary relation ρ. It starts by considering each point in a cluster of its own. Single linkage computes the distance between two clusters C_i and C_j as

$$\delta_{SL}(C_i, C_j) = \min_{\overline{x} \in C_i, \overline{y} \in C_j} \left\{ d(\overline{x}, \overline{y}) \right\}, \tag{4.9}$$

where $d(\overline{x}, \overline{y})$ is some distance measure defined between objects $\overline{x}$ and $\overline{y}$. Based on these distances, it merges two clusters whose distance is the minimum and then replaces these two clusters by the merged cluster. The distances of the merged cluster from the other clusters are recomputed. The process continues until the desired number of clusters (K) is found. The advantages of this clustering technique are as follows: (1) it is independent of the shape of the clusters, and (2) it works with

both numeric and categorical attributes. The disadvantages of this approach are its computational complexity and inability to handle overlapping clusters.

In case of the average linkage clustering technique, the distance between two clusters C_i and C_j is defined as

$$\delta_{SL}(C_i, C_j) = \frac{\sum_{\overline{x} \in C_i, \overline{y} \in C_j} \{d(\overline{x}, \overline{y})\}}{|A| \times |B|}. \tag{4.10}$$

Other steps of this clustering technique are similar to those of the single linkage clustering algorithm.

In case of the complete linkage clustering technique, the distance between two clusters C_i and C_j is defined as

$$\delta_{SL}(C_i, C_j) = \max_{\overline{x} \in C_i, \overline{y} \in C_j} \{d(\overline{x}, \overline{y})\}. \tag{4.11}$$

Other steps of this clustering technique are also similar to those of the single linkage clustering algorithm. Complete linkage generally determines compact clusters of approximately equal sizes, but is very much sensitive to outliers. A major disadvantage of single linkage is that it tends to form long chains of clusters. Average linkage is a compromise between the complete linkage and single linkage clustering techniques.

4.6 Density-Based Clustering Techniques

Density-based clustering algorithms [92, 93] are another kind of clustering technique which basically applies a local cluster criterion. Clusters are considered as regions in the data space where points form some dense area, and which are separated by regions of low point density (noise). These dense regions may sometimes form some arbitrary shapes, and points can be randomly distributed within this region. Thus, these density-based algorithms can easily identify clusters of any shape, but they rely on the condition that points within a particular cluster form a condensed region.[2] For many data mining applications, finding the outliers, i.e., the rare events, is more important and useful than finding the common cases, e.g., detecting criminal activities in E-commerce.

A large number of density-based clustering techniques exist in the literature:

- Density-based spatial clustering of applications with noise (DBSCAN) [93]: The concept of density-reachability for d-dimensional points is used in the algorithm DBSCAN. It is able to detect arbitrary shapes as long as clusters form some dense regions. The time complexity of this algorithm reduces to $O(n \log n)$ if region queries are handled using some spatial index structures, i.e., at least in moderately dimensional spaces. As mentioned earlier, in DBSCAN, clusters are defined

[2] http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/

using the concept of density-reachability,[3] which is defined as follows: a point q is directly density-reachable from a point p if p is within distance ε from q and if there are a sufficient number of points surrounding p so that a cluster can be formed including p and q. Note that this density-reachability relation is not symmetric; this means if p is density reachable from q that does not mean that q is also directly density-reachable from p. q is said to be density-reachable from p if there is a sequence $p_1, \ldots, p_n$ of points with $p_1 = p$ and $p_n = q$ where each $p_i + 1$ is directly density-reachable from p_i. The relation "density-reachability" is also not symmetric because p can be density-reachable from q but it can happen that q lies on the edge of a cluster, having insufficiently many neighbors to count as a true cluster element. Density-connectedness is defined as follows: two points p and q are density-connected if there are a set of points $o_1, o_2, \ldots, o_n$ such that o_1 and p are density-reachable, o_2 and o_1 are density-reachable, o_3 and o_2 are density-reachable, ..., and finally o_n and q are density-reachable. A cluster identified by the DBSCAN algorithm must satisfy the following two criteria:

- All the points within a particular cluster should be mutually density-connected.
- If a point is density-connected to any point of the cluster, then the point is included in that cluster structure.

Two parameters are associated with DBSCAN: ε (eps) (the distance required to calculate direct density reachability), and the minimum number of points required to form a cluster (minPts). The seed point is a randomly selected unvisited point. The computation of DBSCAN starts from this point. This point's ε-neighborhood is formed, and the size of this neighborhood is checked. If it has sufficient number of points, a cluster is formed consisting of all the points within the neighborhood. If the neighborhood is not large enough then the point is marked as an outlier. It is to be noted that this point might be found in a sufficiently sized ε-environment of a different point at a later time and hence be made part of a cluster. Once a point is included in a cluster, its ε-neighborhood is also included in that cluster using the concept of density connectivity. This procedure continues until no new points are added to a cluster. Thereafter the algorithm starts from another seed point. The algorithm will terminate if there are no seed points left.

- Generalized density-based spatial clustering of applications with noise (GDBSCAN) [92, 248]: This is a generalized version of the DBSCAN algorithm which extends the definition of point density. Thus, it can be applied to clusters having different shapes, including two-dimensional polygons.
- Ordering points to identify the clustering structure (OPTICS) [7]: This is a hierarchical density-based clustering technique. Recall that DBSCAN detects clusters of user-defined density. OPTICS is able to produce a hierarchical density-based structure of the given data set. The graph determined by using this algorithm is called the reachability plot showing the clusters of different densities as well as hierarchical clusters.

[3] http://en.wikipedia.org/wiki/DBSCAN

4.7 Grid-Based Clustering Techniques

Like density-based clustering techniques, grid-based clustering approaches are also very popular for determining clusters in a large multidimensional space. Here again clusters are regarded as denser regions than their surroundings [108].

The computational complexities of most of the existing clustering techniques depend linearly on the size of the data set. The main advantage of the grid-based clustering techniques is their ability to handle large data sets. The fundamental difference between grid-based clustering approach and density-based clustering is that these clustering algorithms do not deal with the data points but with the surrounding space. In general, there are five basic steps in a typical grid-based clustering algorithm [108, 115]:

- Generating the grid structure. This can be done by partitioning the data space into a finite number of grids.
- The grid density of each cell is calculated based on the total number of points within that grid.
- The grids are sorted according to their densities.
- Cluster centers are calculated.
- Neighbor grids are traversed.

One commonly used grid-based clustering technique is statistical information grid-based clustering method (STING) [291]. This clustering technique is mainly used to cluster spatial databases. Several kinds of spatial queries can be answered by using this clustering technique. The spatial data is first divided into grids. These grids are represented by a hierarchical structure. The root of the hierarchy is assumed to be at level 1, and its children are at subsequent levels. A cell at level i contains the union of the areas of all its children at level $i + 1$. In case of STING, each cell is assumed to have four children. Thus, here each child represents one quadrant area of its parent cell. STING is only applicable for two-dimensional spatial space. Some extended works can be found in [292].

4.8 Some Recent Clustering Techniques

In recent years many different types of clustering algorithms have been developed. The primary ones are partitional, hierarchical, and graph-theoretic methods. Typical examples of the three methods are the well-known K-means algorithm, single linkage, and the minimal spanning tree (MST)-based algorithms respectively [35, 85, 96, 143, 281].

The K-means algorithm performs poorly in the presence of outliers, because outliers change the computation of the centroids and thereby the resultant partitioning. In order to eliminate this problem, medoids, the most centrally located objects, are used in place of centroids to represent the clusters. The resultant algorithm is termed as the K-medoid algorithm. One of the earliest developed K-medoid algorithm was

the partitioning around medoid (PAM) [153]. The K number of clusters are first formed by randomly choosing a representative point, called the medoid, for each cluster. Thereafter, these medoids are modified by analyzing all possible pairs of points. PAM is not suitable for large datasets or large numbers of clusters. In order to handle large data sets, the clustering large applications (CLARA) algorithm was proposed by the same authors [153]. CLARA uses data sampling to choose a subset of real data, and thereafter PAM is used to determine medoids from this data set. Multiple number of samples are drawn, and the best clustering result of these samples is reported. Thus, CLARA is able to handle larger data sets than PAM. However, the performance of CLARA depends on the sampling technique used. If all the samplings do not represent the data set correctly, then best medoids will never be identified. Thus, CLARA may fail in such situations in providing the best partitioning. Ng and Han [208] proposed the CLARANS algorithm, which merges both PAM and CLARA by searching only a subset of the data set. However, CLARANS has the advantage that it does not restrain itself to a given sample at any given time, but draws data randomly at each step of the algorithm. Two other spatial data mining techniques named the spatial dominant approach, SD (CLARANS), and the nonspatial dominant approach, NSD (CLARANS), have been developed. Efficient spatial access methods such as R^* tree were used in [209] to make CLARANS applicable to large data sets. A disadvantage of CLARANS is that it can only detect equisized and convex clusters. DBSCAN [93], a density-based clustering technique, is able to detect any type of nonconvex and nonuniformly sized clusters. Balanced iterative reducing and clustering using hierarchies (BIRTH), proposed by Zhang et al. [302], is another algorithm for clustering large data sets. Two new concepts are used. The first is the clustering feature, and the second is the clustering feature tree (CFtree). This CFtree helps BIRTH to handle large databases efficiently with good speed by summarizing cluster representations. Discussion on several other clustering algorithms can be found in [145].

In the recent past several clustering algorithms have been developed to improve the performance of the K-means algorithm. In Ref. [166], Kövesi et al. presented a stochastic K-means algorithm to improve the clustering results of K-means. Kanungo et al. [152] presented an improved K-means algorithm based on the Kd-tree data structure [4]. This technique not only performs faster than K-means but also preserves the same clustering result as the K-means algorithm. In [177], the global K-means algorithm is presented. This works in an incremental manner by adding one cluster center at a time by using a deterministic global search procedure consisting of N number of executions of the K-means algorithm. Here, N equals the total number of data points present in the data set. In [54], an algorithm based on K-means, namely, split and merge approach based circular K-means algorithm (SMCK-means), is proposed for circular invariant clustering of vectors. Fuzzy C-shells clustering (FCS) methods have been proposed in [72, 168], being well established for detecting and representing hyperspherical (especially ellipsoidal) clusters. Some more clustering algorithms can be found in Refs. [122, 126–294]. Other algorithms have been developed that are appropriate for large data sets [45, 302]. In [297], a fuzzy clustering algorithm is proposed in order to identify some smooth

curves from unordered noisy data. In [198], an algorithm is developed to determine high-quality polygonal contours from connected objects in binary images. Some level set methods are introduced in [299] to recognize density peaks and valleys in a density landscape for data clustering. The method uses advancing contours to identify cluster structures. A generic iterative clustering scheme is proposed in [211]. This algorithm along with some reweighting schemes theoretically can provide some improved performance over "classical" clustering techniques. A new overlapping clustering algorithm, which clusters n objects into K clusters that may overlap with each other, is developed in [55]. A modification of fuzzy C-means algorithm [36] is proposed in [32], in which distances between cluster centers and the data are determined by the density of the data itself. A few typical examples of non-convex clusters are shown to be correctly identified by this algorithm. However, this method is unable to identify highly overlapping clusters.

Neural networks, for example, competitive learning networks [162], self-organizing feature maps (SOFM) [163], and adaptive resonance theory (ART) networks [50, 51], have also often been utilized for clustering data sets. Hyperspherical clusters can be suitably identified by both SOFM and ART [128]. A two-layer network utilizing a regularized Mahalanobis distance to identify hyperellipsoidal clusters was proposed in [185]. A clustering method based on a self-organizing feature map (SOFM) with quadratic neurons is developed in [267]. The network is updated in an unsupervised manner to cluster data into hyperellipsoidal-shaped or hyperspherical-shaped clusters based on the underlying structure of the data set, but it will fail for straight line or shell-type clusters [267]. In [268], an ART-based clustering algorithm is used to cluster well-separated and non-overlapping clusters with arbitrary shapes, but it will fail for data sets where clusters are not at all well separated. All the algorithms mentioned above have their own merits and disadvantages.

To identify clusters of different geometric shapes, several clustering algorithms with different distance measures have been proposed in the literature [71, 72, 109, 119, 130, 184]. These algorithms were utilized to detect compact clusters [109], straight lines [71, 109], ring-shaped clusters [184], or contours with polygonal boundaries [72, 130]. However, these algorithms fail to detect clusters of other shapes. In [15] a clustering technique is proposed which can automatically detect any number of well-separated clusters which may be of any shape, convex and/or non-convex, but it fails for overlapping clusters. Many fuzzy clustering techniques can be found in [30, 42, 43, 57]. Some other recent clustering techniques can be found in [8–11].

4.9 Some Evolutionary Approaches to Clustering

Clustering can be treated as a particular kind of *NP*-hard grouping problem [98] from an optimization perspective. Evolutionary algorithms are metaheuristics that are generally used for solving *NP*-hard problems. The ability of these algorithms

to provide near-optimal solutions in reasonable time stimulates their use in solving clustering problems [17, 134].

4.9.1 Algorithms for a Fixed Value of the Number of Clusters

Several research papers use evolutionary algorithms to solve clustering problems with fixed number of clusters (K), such as Bandyopadhyay and Maulik [19], Castro and Murray [94], Fränti et al. [102], Krishna and Murty [167], Krovi [169], Kuncheva and Bezdek [170], Lu et al. [180, 181], Lucasius et al. [182], Maulik and Bandyopadhyay [188], Merz and Zell [194], Murthy and Chowdhury [205], and Sheng and Liu [253]. A good review of these clustering techniques is available in [134].

Genetic algorithms have been used for determining global optimal solutions to clustering problems [41]. However, it has been pointed out in [167] that these GA-based methods are computationally inefficient as they use either complex crossover operators or computationally hard fitness functions. To avoid these problems, Krishna and Murty [167] have proposed a new GA-based algorithm (GKA). This algorithm uses the K-means algorithm instead of crossover to reach the locally optimal partitions and a biased mutation to widen the search space to reach the global optimum. Theoretical proof has been provided to show that this algorithm converges to the global optimum, but experimental results are shown only for small data sets (four-dimensional data with $n = 50$, and a two-dimensional data with $n = 59$, where n is the size of the data set), and for small number of clusters ($K \leq 10$). They have noted that, as the number of clusters increases, the size of the search space increases exponentially and the problem of finding the global solution becomes more intricate. This problem occurs because of employing only the mutation operator to widen the search space. It has to be applied several times to search the whole space, but this grows exponentially with n and K. This shows that GKA is not suitable for partitioning large data sets with huge number of clusters.

GAs have been used for the selection of the initial centers in the K-means algorithm in [13]. For a d-dimensional problem, a candidate set of K centers are selected from the vertices of d-dimensional hyperboxes formed by dividing each dimension into $(2^B - 1)$ segments. A chromosome is a bit string of length KdB formed by the collection of K B-bit strings for each of the d dimensions. Thus, the resulting search space is of size 2^{KdB}. Results are shown for small data sets, but note that for large data sets with high number of dimensions, as the search space increases exponentially, this algorithm becomes computationally intractable.

Maulik and Bandyopadhyay [188] have used center based encoding in chromosomes while performing clustering using GA. GA is used to evolve the appropriate set of cluster centers. The crossover operators [188] exchange randomly selected substrings. Experimental results showed the effectiveness of the proposed approach. Using this same representations they have also used variable-length strings to vary the number of clusters over a range [20].

Laszlo and Mukherjee [175] have proposed a new genetic algorithm-based K-means clustering technique. Here, centers of the K-means algorithm are evolved using GA to determine appropriate partitions for a range of values around a specified K. Here, a hyper-quadtree constructed on the entire data set is used to represent the cluster centers. Thereafter, an initial population of good centers are generated using this representation. A new crossover operator which is responsible for selectively passing good subsets of neighboring centers from parents to offspring by swapping subtrees is also developed here. Experimental results show that the proposed technique is wellsuited for large data sets as well as small ones.

4.9.2 Algorithms with Variable Number of Clusters

Evolutionary algorithms which automatically determine the number of clusters (K) present in a data set are described in the works by Cole [64], Cowgill et al. [67], Bandyopadhyay and Maulik [18, 20], Hruschka and Ebecken [135], Hruschka et al. [131–133], Ma et al. [183], and Alves et al. [256], but all the above-mentioned algorithms use Euclidean distance for computing similarity measures. Thus, none of these algorithms are able to detect clusters having shapes other than hyperspheres.

In genetic clustering with unknown K values (GCUK-clustering) [20], a variable string length genetic algorithm (VGA) [129] was applied with real parameter representation as the underlying search tool. The chromosome encodes the centers of a number of clusters, whose value may vary. Modified versions of crossover and mutation are used. The Davies-Bouldin cluster validity index [73] is utilized for computing the fitness of the chromosomes.

In hybrid niching genetic Algorithm (HNGA) , a weighted sum validity function (WSVF), which is a weighted sum of several normalized cluster validity functions, is used for optimization to automatically evolve the proper number of clusters and the appropriate partitioning of the data set. Within the HNGA, a niching method is developed to prevent premature convergence during the search. Additionally, in order to improve the computational efficiency, a hybridization between the niching method with the computationally attractive K-means is made. Here, the WSVF is defined as $WSVF = \sum_{i=1}^{m} w_i f_i(x)$, where m is the number of component functions specifically, $m = 6$ is used here. The w_i are the non-negative weighting coefficients representing the relative importance of the functions such that $\sum_{i=1}^{m} w_i = 1$, and the $f_i(x)$ are component functions (as used in [254]) corresponding to 1/(DB-index [73]), Silhouette (SIL)-index [153], Dunn-index [87], generalized Dunn-index [38], CH-index [47], and I-index [189], respectively. Here, weighting coefficients are chosen as $w_1 = w_2 = \cdots = w_m = 1/m$.

In [176] an algorithm for evolutionary clustering with self-adaptive genetic operators (ECSAGO) is developed. This algorithm is based on the unsupervised niche clustering (UNC) and hybrid adaptive evolutionary (HAEA) algorithms. The UNC is a genetic clustering algorithm which is robust to noise and can determine the appropriate number of clusters from data sets automatically [176]. HAEA is a parameter adaptation technique that automatically learns the rates of its genetic operators

at the same time that the individuals are evolved in an evolutionary algorithm (EA) [176]. In ECSAGO, real encoding and real genetic operators are used.

4.10 MOO and Clustering

Clustering is considered to be a difficult task as no unambiguous partitioning of the data exists for many data sets. Most of the existing clustering techniques are based on only one criterion which reflects a single measure of goodness of a partitioning. However, a single cluster quality measure is seldom equally applicable for different kinds of data sets with different characteristics. Hence, it may become necessary to simultaneously optimize several cluster quality measures that can capture different data characteristics. In order to achieve this the problem of clustering a data set has been posed as one of multiobjective optimization in literature [192, 193, 203, 204].

In Ref. [121], a multiobjective clustering technique called multiobjective clustering with automatic K-determination (MOCK) is developed which outperforms several single-objective clustering algorithms, a modern ensemble technique, and two other methods of model selection. Two cluster quality measures, one measuring the total Euclidean compactness of the obtained partitioning and the other measuring the total "connectedness" of the obtained partitioning are optimized simultaneously. Although the objectives of [121] are very useful, it can only handle clusters having either hyperspherical shape or "connected" but well-separated structures. It fails for data sets having overlapping clusters which do not contain any hyperspherical shape, e.g., the data sets in Figs. 5.9(a) and 5.9(b). Moreover, MOCK uses locus-based adjacency representation as proposed in Ref. [218]. As a result, when the number of data points is too large the string length becomes high too and convergence becomes slow. Here, a "Gap" statistic [174] is used to select a single solution from the set of final Pareto-optimal solutions obtained by MOCK. In [187], a scalable data clustering algorithm is developed for web mining based on MOCK. Here, a scalable automatic K-determination scheme is developed to automatically determine the number of clusters with a lower computational cost than the original version. The proposed technique is capable of reducing the size of the Pareto-optimal front. Thus, the appropriate number of clusters can be determined easily.

An EA for MOO clustering is proposed in [165]. Here, two objectives are minimized simultaneously. These are total intracluster variation (computed over all the clusters) and the number of clusters. These two objectives are conflicting with each other. Using MOO, the EA manages to provide a set of non-dominated solutions. Here, for each different number of clusters, the algorithm manages to provide the smallest possible total intra cluster variance. Users can then make a more informed choice about the solution to be used in practice.

A multiobjective evolutionary algorithm for fuzzy clustering has been proposed in [21, 23]. Here again, two objectives are simultaneously optimized. The first one is the objective function optimized in the fuzzy C-means algorithm [37], and the other is the well-known Xie-Beni index [295]. This is defined as a ratio between a

global measure of intracluster variation and a local measure of cluster separation. The separation between clusters is measured using the distance between the two closest clusters. Although the numerator of the second objective is similar to the first objective, the denominator measures a qualitative aspect of the clustering, which is eventually not captured by the first objective. The minimum value of the second objective corresponds to the partitioning where all clusters have an intra cluster variation as small as possible and the two closest clusters are as far away from each other as possible. This proposed algorithm is extended for categorical data clustering in [204].

In [192, 203], a novel approach that combines a recently proposed multiobjective fuzzy clustering method [21, 23] with a support vector machine (SVM) classifier to yield improved solutions is proposed. The multiobjective technique is first used to generate a set of nondominated solutions on the final Pareto-optimal front. The solutions on the final Pareto-optimal front are then used to find some high-confidence points using a fuzzy voting technique. The SVM classifier is thereafter trained by these high-confidence points. Finally, the remaining points are classified using the trained classifier. Results demonstrating the effectiveness of the proposed technique are provided for remote sensing data in [21] and gene expression data in [23].

Another multiobjective evolutionary clustering algorithm with two objectives is proposed in [233]. The first objective function is again a kind of measure of average intracluster variation computed over all clusters. Rather than using the total summation across clusters, its average value across clusters was used. This is done in order to produce a normalized value of the measure taking into account the number of clusters, which varies across different individuals in the evolutionary algorithm's population. The second objective measures the inter cluster distance, which is the average distance between a pair of clusters computed over all pairs of clusters.

4.11 Summary

In this chapter first we discussed some well-known traditional clustering techniques. In recent years several evolutionary-based clustering algorithms have been developed. These algorithms primarily employ the search capability of genetic algorithms to obtain the proper partitioning of a given data set. Such type of algorithms are also described in brief in this chapter. Genetic algorithm-based clustering techniques only optimize a single cluster quality measure at a time. For some data sets having different types of clusters it is necessary to optimize more than one cluster quality measurement. Thus, multiobjective optimization is used nowadays to solve the problem of clustering a data set. In the last part of the chapter a short description is given on these existing MOO-based clustering techniques.

As mentioned earlier, symmetry can be considered as an important distance measure for clustering a data set. Based on this idea, some symmetry-based similarity measurements and clustering algorithms have been developed in [58, 60, 61,

178, 266]. In the next chapter we describe the existing symmetry-based distances in detail. A newly developed symmetry-based distance is also discussed in detail. Some symmetry-based clustering algorithms are also elaborately described in the next chapter.

Chapter 5
Point Symmetry-Based Distance Measures and Their Applications to Clustering

5.1 Introduction

For partitioning a data set, one has to define a measure of similarity or proximity based on which cluster assignments can be done. The measure of similarity is usually data dependent. It may be noted that, in general, one of the fundamental features of shapes and objects is symmetry, which is considered to be important for enhancing their recognition [12]. As symmetry is commonly found in the natural world, it may be interesting to exploit this property while clustering a data set [27, 28, 58, 243, 266]. In the real world there are many objects which contain some form of symmetry; the human face, jellyfish, the human body, stars, etc. are some examples of symmetry. Symmetry mainly conveys balance and reflects perfection or beauty. As symmetry represents the well-defined concept of balance or "pattern self-similarity", it has been extensively used to describe many processes/objects in geometry and physics. Thus, we can assume that some kind of symmetry exists in the cluster structure also. Based on this concept, several different symmetry-based similarity measures/distances have been proposed in the literature.

In this chapter some symmetry-based similarity measurements are discussed in detail. The existing clustering techniques based on symmetry are also described. A newly developed point symmetry-based distance (PS-distance) [27] is also described, which incorporates both the Euclidean distance as well as a measure of symmetry. K-means is a widely used clustering algorithm. However, K-means is known to get stuck at suboptimal solutions depending on the choice of the initial cluster centers. In order to overcome this limitation, genetic algorithms have been used for solving the underlying optimization algorithm [188]. In view of the advantages of the GA-based clustering method [188] over the standard K-means, the former has been used in this work. In the presented GA with the point symmetry distance (GAPS) clustering technique, the assignment of points to different clusters is done based on the point symmetry distance rather the Euclidean distance. This enables this algorithm to detect both convex and non-convex clusters of any shape and size as long as the clusters have some symmetry property. The convergence of the GAPS clustering technique is also established in the present chapter.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_5, © Springer-Verlag Berlin Heidelberg 2013

5.2 Some Existing Symmetry-Based Distance Measures

Symmetry can be considered as a fundamental feature of shapes and objects. It is an important feature which helps to enhance recognition and reconstruction of shapes and objects. A good paper devoted to symmetry, describing symmetry as a continuous feature, is [300]. Many interesting objects around us exhibit some generalized form of symmetry. In [300] authors have shown that symmetry should be considered as a continuous feature rather than a binary feature; For example, take Fig. 5.1. Note that, even when a particular object is totally symmetrical, it may lose its perfect symmetry when projected onto an image plane or the retina due to digitization, occlusion, etc. In the example shown in Fig. 5.1, different shapes are shown to have different amounts of symmetry. Thus, in [300] symmetry is considered as a continuous feature rather than a binary feature. Based on this observation the authors then defined a new measure of symmetry which quantifies all types of symmetries of objects. Some definitions of symmetry are also mentioned in that paper [300]. These are provided below:

An n-dimensional object is said to have mirror symmetry *if it is invariant under a reflection about a hyperplane of dimension* $(n-1)$ *passing through the center of mass of the object. Using this definition, a 2D object is said to be mirror symmetric if it is invariant under a reflection about a line (called the axis of* mirror symmetry*) and a 3D object is said to be mirror symmetric if it is invariant under a reflection about a plane.*

A 2D object is said to have rotational symmetry *of order n if it is invariant under rotation of* $\frac{2\Pi}{n}$ *radians about the center of mass of the object.*

A 3D object is said to have rotational symmetry *of order n if it is invariant under rotation of* $\frac{2\Pi}{n}$ *radians about a line passing through the center of mass of the object (denoted the rotational symmetry axis).*

Rotational symmetry of order n is denoted by C_n-symmetry.

Radial symmetry is the symmetry of a 2D object having both mirror symmetry and C_n-symmetry *(note that such objects have* $2n$ *axes of mirror symmetry). Radial symmetry of order n is denoted by* D_n-symmetry. *Circular symmetry is* C_∞-symmetry. Figure 5.2 shows different forms of symmetry.

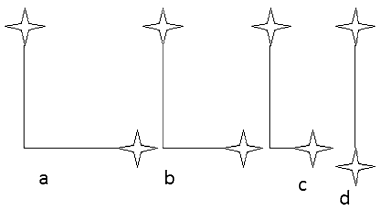


Fig. 5.1 Understanding continuous symmetry. (**a**) A shape is called "perfectly symmetric" (the oblique mirror axis passing through the vertex). (**b**) Shortening one arm, the shape is called "almost" symmetric. (**c**) With further shortening of the arm, the shape is called "less" symmetric. (**d**) When the arm is eliminated, the shape is again "perfectly symmetric" (with a mirror axis perpendicular to the existing arm)

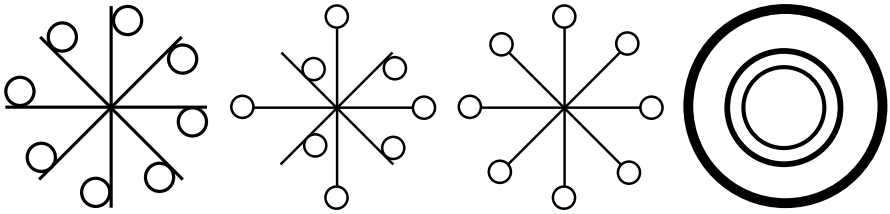


Fig. 5.2 Example of symmetries: (**a**) C_8-*symmetry*, (**b**) *mirror symmetry*, (**c**) D_8-*symmetry* (radial symmetry of order 8), and (**d**) *circular symmetry* (C_∞-*symmetry*)

The symmetry distance defined in [300] quantifies the minimum effort required to turn a given shape into a symmetric shape. This effort is measured by the means of the squared distances when each point is moved from its location in the original shape to its location in the symmetric shape. The distance is formally defined as:

Let Ω denote the space of all shapes of a given dimension. Here, each shape P *is represented by a sequence of m objects P_i, where $i = 0, \ldots, m-1$. A metric d on this space is defined as follows:*

$$d : \Omega \times \Omega \Rightarrow R, \tag{5.1}$$

$$d(P, Q) = d(P_i, Q_i) = \frac{1}{m} \sum_{i=0}^{m-1} \| P_i - Q_i \|^2. \tag{5.2}$$

The distance function between every two shapes in Ω can be defined using this metric.

The symmetry transform (ST) of a shape P is the symmetric shape closest to P in terms of the metric d.

The symmetry distance (SD) of a shape P is defined as the distance between P and its symmetry transform:

$$SD = d\big(P, ST(P)\big).$$

The SD of a shape $P = P_i$, $i = 0, \ldots, (m-1)$ is evaluated by finding the symmetry transform $\widehat{P}$ of P (shown in Fig. 5.3) and computing

$$SD = \frac{1}{m} \sum_{i=0}^{m-1} \| P_i - \widehat{P}_i \|^2.$$

Here, normalization of the original shape is done prior to the transformation.

Many clustering algorithms have been also developed in the literature to utilize the symmetry property of the data set to identify clusters therein. In [213], authors first developed a new definition of measuring circular symmetry. Then, a clustering algorithm is developed which extracts some sub and sup clusters from a data set

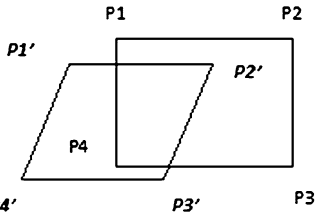


Fig. 5.3 The symmetry transform of $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4\}$ is $\{\mathbf{P}'_1, \mathbf{P}'_2, \mathbf{P}'_3, \mathbf{P}'_4\}$. $SD = \frac{\sum_{i=0}^{4} \|\mathbf{P}_i - \mathbf{P}'_i\|}{4}$

based on this measure of circular symmetry. These sub and sup clusters are then iteratively merged or split to form the final clusters, which can be of any shape, convex or nonconvex.

A point symmetry distance was proposed by Su and Chou in [266]. This is defined as follows: Given N patterns, $\overline{x}_j$, $j = 1, \ldots, N$, and a reference vector $\overline{c}$ (e.g., a cluster centroid), the point symmetry distance (PS-distance) between a pattern $\overline{x}_j$ and the reference vector $\overline{c}$ is defined as

$$d_s(\overline{x}_j, \overline{c}) = \min_{i=1,\ldots,N \text{ and } i \neq j} \frac{\|(\overline{x}_j - \overline{c}) + (\overline{x}_i - \overline{c})\|}{\|(\overline{x}_j - \overline{c})\| + \|(\overline{x}_i - \overline{c})\|}, \tag{5.3}$$

where the denominator term is used to normalize the distance so as to make it insensible to the Euclidean distances $\|\overline{x}_j - \overline{c}\|$ and $\|\overline{x}_i - \overline{c}\|$. It may be noted that the numerator of Eq. 5.3 is actually the distance between the mirror image point of $\overline{x}_j$ with respect to $\overline{c}$ and its nearest neighbor in the data set. If the right-hand term of the above equation is minimized when $\overline{x}_i = \overline{x}_{j^*}$, then the pattern $\overline{x}_{j^*}$ is denoted as the symmetrical pattern relative to $\overline{x}_j$ with respect to $\overline{c}$. Here it can be easily seen that the above equation is minimized when the pattern $\overline{x}_i = (2 \times \overline{c} - \overline{x}_j)$, i.e., the mirror image point of $\overline{x}_j$, exists in the data set (i.e., $d_s(\overline{x}_j, \overline{c}) = 0$). This idea of point symmetry is very simple and intuitive. Based on this point symmetry-based distance, Su and Chou have proposed a clustering algorithm called SBKM clustering which mimics the K-means algorithm but assigns the patterns to a particular cluster depending on the symmetry-based distance d_s rather than Euclidean distance [266], only when d_s is greater than some user-specified threshold θ. Otherwise, assignment is done according to the Euclidean distance, as in normal K-means. The algorithm is discussed in detail in Fig. 5.4, and the process of cluster assignment is clearly stated in step 3 of the figure.

It is evident from Eq. 5.3 that this similarity measure can be useful to detect clusters which have symmetrical shapes, but this clustering algorithm will fail for data sets where clusters themselves are symmetrical with respect to some intermediate point. Note that minimization of $d_s(\overline{x}_j, \overline{c})$ means minimization of its numerator and maximization of its denominator. In effect, if a point $\overline{x}_j$ is almost equally symmetrical with respect to two centroids $\overline{c}_1$ and $\overline{c}_2$, it will be assigned to the cluster that is the farthest. This is intuitively unappealing. In the example shown in Fig. 5.5, there are three clusters which are well separated. The centers of the clusters are denoted by $\overline{c}_1$, $\overline{c}_2$, and $\overline{c}_3$, respectively. Let us take the point $\overline{x}$. After application of the K-means algorithm, point $\overline{x}$ is assigned to cluster 1, but when SBKM is applied on the result of the K-means algorithm, the following will happen: The symmetrical

Step 1: Initialization: Randomly choose K data points from the data set to initialize K cluster centroids, $\overline{c}_1, \overline{c}_2, \ldots, \overline{c}_K$.
Step 2: Coarse-tuning: Use K-means algorithm to update the K cluster centroids.
After the K cluster centroids converge or some terminating criterion is satisfied, go to next step.
Step 3: Fine-tuning: For each data point $\overline{x}$ compute, $k^* = \text{argmin}_{k=1,\ldots,K}\, d_s(\overline{x}, \overline{c}_k)$, where $d_s(\overline{x}, \overline{c}_k)$ is computed using Eq. 5.3.
If $d_s(\overline{x}, \overline{c}_{k^*}) < \theta$ /*θ is a user specified parameter*/ assign $\overline{x}$ to the k^*th cluster.
else, compute $k^* = \text{argmin}_{k=1,\ldots,K}\, d_e(\overline{x}, \overline{c}_k)$, where $d_e(\overline{x}, \overline{c}_k)$ is the Euclidean distance between $\overline{x}$ and the cluster centroid $\overline{c}_k$.
assign $\overline{x}$ to the k^*th cluster
Step 4: Updating: Compute the new centroids of the K clusters as follows:
$\overline{c}_k(t+1) = \frac{\sum_{\overline{x}_i \in S_k(t)} \overline{x}_i}{N_k}, \quad k = 1, \ldots, K,$
where $S_k(t)$ is the set of elements that are assigned to the kth cluster at time t and $N_k = |S_k|$.
Step 5: Continuation: If no point changes category, or the number of iterations has reached a specified maximum number then stop, else go to step 3.

Fig. 5.4 Steps of the SBKM algorithm

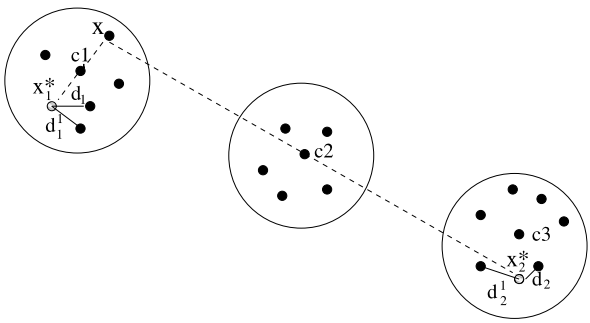


Fig. 5.5 Example of a data set having some symmetrical interclusters

point of $\overline{x}$ with respect to $\overline{c}_1$ is $\overline{x}_1$, since it is the first nearest neighbor of the point $\overline{x}_1^* = (2 \times \overline{c}_1 - \overline{x})$, the mirror image point of $\overline{x}$. Let the Euclidean distance between $\overline{x}_1^*$ and $\overline{x}_1$ be d_1. Therefore, the symmetrical distance of $\overline{x}$ with respect to $\overline{c}_1$ is

$$d_s(\overline{x}, \overline{c}_1) = \frac{d_1}{d_e(\overline{x}, \overline{c}_1) + d_e(\overline{x}_1, \overline{c}_1)}, \tag{5.4}$$

where $d_e(\overline{x}, \overline{c}_1)$, and $d_e(\overline{x}_1, \overline{c}_1)$ are the Euclidean distances of $\overline{x}$ and $\overline{x}_1$ from $\overline{c}_1$, respectively. Similarly, the symmetrical point of $\overline{x}$ with respect to $\overline{c}_2$ is $\overline{x}_2$, and the symmetrical distance of $\overline{x}$ with respect to $\overline{c}_2$ becomes

$$d_s(\overline{x}, \overline{c}_2) = \frac{d_2}{d_e(\overline{x}, \overline{c}_2) + d_e(\overline{x}_2, \overline{c}_2)}. \tag{5.5}$$

Let $d_2 < d_1$, and obviously $(d_e(\overline{x}, \overline{c}_2) + d_e(\overline{x}_2, \overline{c}_2)) \gg (d_e(\overline{x}, \overline{c}_1) + d_e(\overline{x}_1, \overline{c}_1))$. Therefore, $d_s(\overline{x}, \overline{c}_1) \gg d_s(\overline{x}, \overline{c}_2)$ and $\overline{x}$ is assigned to $\overline{c}_2$. This will happen for the other points also, finally resulting in merging of the three clusters after application of SBKM.

Chou et al. have noted the above-mentioned limitation of the measure proposed in [266], and have suggested a modified measure d_c in [58] that is defined as follows:

$$d_c(\overline{x}_j, \overline{c}) = d_s(\overline{x}_j, \overline{c}) \times d_e(\overline{x}_j, \overline{c}), \tag{5.6}$$

where $d_s(\overline{x}_j, \overline{c})$ is the point symmetry (PS) distance of $\overline{x}_j$ with respect to $\overline{c}$, and $d_e(\overline{x}_j, \overline{c})$ denotes the Euclidean distance between $\overline{x}_j$ and $\overline{c}$. No experimental results are provided in [58] corresponding to this new measure. A little thought will show that even this modification will not work for the situation shown in Fig. 5.5. Let $\overline{x}_j^*$ be the symmetrical point of $\overline{x}_j$ with respect to $\overline{c}$. Therefore, Eq. 5.6 is simplified to

$$d_c(\overline{x}_j, \overline{c}) = \frac{d_{symm}(\overline{x}_j, \overline{c})}{d_e(\overline{x}_j, \overline{c}) + d_e(\overline{x}_j^*, \overline{c})} d_e(\overline{x}_j, \overline{c}), \tag{5.7}$$

where $d_{symm}(\overline{x}_j, \overline{c}) = \|(\overline{x}_j - \overline{c}) + (\overline{x}_j^* - \overline{c})\|$. It can be also noted that $d_e(\overline{x}_j, \overline{c}) \approx d_e(\overline{x}_j^*, \overline{c})$. Therefore, Eq. 5.7 is simplified to

$$d_c(\overline{x}_j, \overline{c}) \propto d_{symm}(\overline{x}_j, \overline{c}). \tag{5.8}$$

As a result, there is no impact of Euclidean distance; only the symmetrical distance plays an important role in assignment of points to different clusters. Moreover, if the term $d_s(\overline{x}_j, \overline{c})$ becomes 0, then there will be no effect of the Euclidean distance. The clustering algorithm based on this modified measure is referred to as the Mod-SBKM algorithm. It has been shown experimentally that Mod-SBKM with this measure will also fail for several data sets considered in Sect. 5.8.1 of this chapter.

The most limiting aspect of the measures suggested in Ref. [266] and Ref. [58] is that, in cases where K-means provides reasonably good clusters, application of the fine-tuning phase (see the algorithm in Fig. 5.4) will destroy this structure. Another limitation of the SBKM is that it requires prior specification of a parameter θ, based on which assignment of points to clusters is done either on the basis of the PS-distance or the Euclidean distance. Su and Chou chose θ equal to 0.18. However, it has been observed that clustering performance is significantly affected by the choice of θ, and its best value is dependent on the data characteristics. No guidelines for the choice of θ are provided in [266].

In the following section an alternative definition of the PS-based distance [27] that can overcome the limitations of both measures d_s and d_c is described.

In [178], Lin et al. proposed a new distance measure based on central symmetry and then used it to define a modified version of the K-means algorithm. The proposed central distance is defined as follows: Suppose there are N objects, $\overline{x}_i$, $i = 1, 2, \ldots, N$, then the central symmetry between the point $\overline{x}_i$ and a particular cluster center $\overline{c}$ is defined as follows:

$$d(\overline{x}_i, \overline{c}) = \big(\cos(\alpha) + k\big) \times \frac{\max_{j=1,\ldots,N, j\neq i} d(m_{ic}, m_{jc})}{m_{ic} + m_{jc}},$$

where $m_{ic} = \|\overline{x}_i - \overline{c}\|$ and $\cos(\alpha)$ is the angle formed by the vectors $m_{ic} = (\overline{x}_i - \overline{c})$ and $m_{jc} = (\overline{x}_j - \overline{c})$. Thus, $\cos(\alpha)$ can be calculated as follows:

$$\cos(\alpha) = \frac{m_{ic}^2 + m_{jc}^2 - m_{ij}^2}{2 m_{ic} m_{jc}}.$$

Here, authors have kept $k = 2$, but the proper value of k depends on the application domain. Then, this distance function is used to fine-tune the clustering solutions provided by the K-means algorithm. However, this distance function is not able to detect symmetrical interclusters [28] well.

In a recent work by Chung and Lin [60], a new point symmetry similarity level (SSL) operator to determine the amount of symmetry between two points with respect to a particular center has been proposed. The novel SSL operator is developed to calculate the symmetry level between the data point $\overline{p}_i$ and the data point $\overline{p}_j$ with respect to the cluster center $\overline{c}_k$. When compared with the previous point symmetry based distance developed by Su and Chou [266], the proposed SSL operator can not only measure the orientation symmetry between $\overline{p}_i$ and $\overline{p}_j$ with respect to $\overline{c}_k$ as in the PS-distance proposed by Su and Chou [266], but can also measure the distance symmetry between the line segment $\overline{p_i c_k}$ and the line segment $\overline{c_k p_j}$. In addition, a single constraint is suggested to handle the case of symmetrical interclusters. Further, two speedup strategies are presented to reduce the computational time required in the proposed modified symmetry distance-based K-means clustering technique (MPSK). In order to speed up the computation of the SSL operator, a two-phase speedup strategy is presented. The proposed MPSK algorithm includes the coarse tuning step. A speed-up strategy is also presented to improve the code vector activity detection approach [199] such that the coarse-tuning step can be performed in a faster way. Experimental results show that the proposed MPSK algorithm performs much better than the SBKM algorithm for symmetrical interclusters, but as the fine-tuning phase of the proposed algorithm relies on the partitioning provided by the K-means algorithm, it often gets struck at suboptimal solutions.

In their next work, Chung and Lin [61] proposed a new clustering algorithm for handling data sets with line symmetry while preserving the advantages of the previous MPSK approach. The proposed line symmetry-based K-means algorithm (LSK) can handle data sets with point symmetry property, line symmetry property, or both. Given a data set, first K-means algorithm is applied to get the initial partitioning. Thereafter, the concept of centroid moment [136] is applied to detect the symmetrical line of each cluster identified by the K-means algorithm. Lastly, the symmetry

similarity level (SSL) operator is modified and extended to measure the line symmetry level between two points. Utilization of the modified SSL (MSSL) operator and the symmetrical line of each cluster enable the proposed LSK algorithm to determine line symmetric clusters from data sets.

5.3 A New Definition of the Point Symmetry Distance [27]

As discussed in Sect. 5.2, both symmetry-based distances, d_s and d_c, will fail when the clusters themselves are symmetrical with respect to some intermediate point. In order to overcome this limitation, a new point symmetry (PS) distance was proposed in [27]. The remaining part of this chapter deals with the detailed description of the PS-distance referred to as $d_{ps}(\overline{x}, \overline{c})$, associated with point $\overline{x}$ with respect to a center $\overline{c}$ as developed in Ref. [27]. The point symmetry distance is defined as follows: Let a point be $\overline{x}$. The symmetrical (reflected) point of $\overline{x}$ with respect to a particular center $\overline{c}$ is $2 \times \overline{c} - \overline{x}$. Let us denote this by $\overline{x}^*$. Let *knear* unique nearest neighbors of $\overline{x}^*$ be at Euclidean distances d_i, $i = 1, 2, \ldots,$ *knear*, such that the d_i are all distinct. Then,

$$d_{ps}(\overline{x}, \overline{c}) = d_{sym}(\overline{x}, \overline{c}) \times d_e(\overline{x}, \overline{c}) \tag{5.9}$$

$$= \frac{\sum_{i=1}^{knear} d_i}{knear} \times d_e(\overline{x}, \overline{c}), \tag{5.10}$$

where $d_e(\overline{x}, \overline{c})$ is the Euclidean distance between the point $\overline{x}$ and $\overline{c}$, and $d_{sym}(\overline{x}, \overline{c})$ is a symmetry measure of $\overline{x}$ with respect to $\overline{c}$. It can be seen from Eq. 5.10 that *knear* cannot be chosen equal to 1, since if $\overline{x}^*$ exists in the data set then $d_{ps}(\overline{x}, \overline{c}) = 0$ and hence there will be no impact of the Euclidean distance. On the contrary, large values of *knear* may not be suitable because this may underestimate the amount of symmetry of a point with respect to a particular cluster center. Here *knear* is chosen equal to 2, though its proper choice is an important issue that needs to be addressed in the future.

The concept of point symmetry-based distance is further illustrated by Fig. 5.6. Here, a particular point is $\overline{x}$. The cluster center is denoted by $\overline{c}$. Then, the reflected point of $\overline{x}$ with respect to $\overline{c}$ is $\overline{x}^*$, i.e., $\overline{x}^* = 2 \times \overline{c} - \overline{x}$. The two nearest neighbors of $\overline{x}^*$ are at Euclidean distances of d_1 and d_2, respectively. Then, the point symmetry-based distance between $\overline{x}$ and $\overline{c}$ is calculated as $d_{ps}(\overline{x}, \overline{c}) = \frac{d_1+d_2}{2} \times d_e(\overline{x}, \overline{c})$.

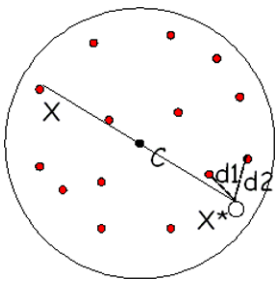


Fig. 5.6 Example of point symmetry distance

Example 5.1 Suppose a two-dimensional data set consists of six points: $(2, 3)$, $(4, 5)$, $(4, 2)$, $(3, 5)$, $(2, 5)$, and $(3, 4)$. The center of the data set is C. Here, $C = (cx, cy)$ is defined as follows: $cx = \frac{2+4+4+3+2+3}{6} = \frac{18}{6} = 3$ and $cy = \frac{3+5+2+5+5+4}{6} = \frac{24}{6} = 4$. So, $C = (3, 4)$. Now let $X = (2, 5)$. The mirror image point/reflected point of X with respect to C is $2 \times C - X$, which is calculated as $2 \times (3, 4) - (2, 5) = (4, 3)$. Thus, $X^* = (4, 3)$. The first and second nearest neighbors of X^* are $(4, 2)$ and $(3, 4)$, respectively. Thus, $d_1 = \sqrt{((4-4)^2 + (3-2)^2)} = \sqrt{1} = 1$ and $d_2 = \sqrt{((4-3)^2 + (3-4)^2)} = \sqrt{2}$, then $d_{sym}(X, C) = \frac{d_1+d_2}{2} = \frac{1+\sqrt{2}}{2} = 1.2071$ and $d_e(X, C) = \sqrt{2}$. Thus, $d_{ps}(X, C) = 1.2071 \times \sqrt{2} = 1.7071$.

The basic differences between the PS-based distances in [266] and [58], and the point symmetry distance, $d_{ps}(\overline{x}, \overline{c})$, are as follows:

1. Here, since the average distance between $\overline{x}^*$ and its *knear* unique nearest neighbors have been taken, this term will never be equal to 0, and the effect of $d_e(\overline{x}, \overline{c})$, the Euclidean distance, will always be considered. This will reduce the problems discussed in Fig. 5.5. Note that, if only the nearest neighbor of $\overline{x}^*$ is considered as in d_s of [266], and this happens to coincide with $\overline{x}^*$, then this term will be 0, making the distance insensitive to $d_e(\overline{x}, \overline{c})$. This in turn would indicate that if a point is marginally more symmetrical to a far-off cluster than to a very close one, it would be assigned to the farthest cluster. This often leads to undesirable results for d_s as demonstrated in Sect. 5.2.
2. Considering the *knear* nearest neighbors in the computation of d_{ps} makes it more robust and noise resistant. From an intuitive point of view, if this term is less, then the likelihood that $\overline{x}$ is symmetrical with respect to $\overline{c}$ increases. This is not the case when only the first nearest neighbor is considered, which could mislead the method in noisy situations.
3. Consideration of both the symmetry component ($\frac{\sum_{i=1}^{knear} d_i}{knear}$) and the Euclidean distance $d_e(\overline{x}, \overline{c})$ in the computation of d_{ps} helps to strike a better balance between these two. Thus, even if a point is marginally more symmetrical to a far-off cluster than to a closer one, it will not necessarily be assigned to the former (as happened for the distances in [58, 266]). This will depend on certain conditions discussed in detail in the next section.
4. A rough guideline for the choice of θ, the threshold value on d_{ps}, which is used for symmetry-based cluster assignment, is also provided. It is to be noted that, if a point is indeed symmetric with respect to some cluster center, then the symmetrical distance computed in the above way will be small. Let d_{NN}^{max} be the maximum nearest neighbor distance in the data set, that is

$$d_{NN}^{max} = \max_{i=1,\ldots N} d_{NN}(\overline{x}_i), \tag{5.11}$$

where $d_{NN}(\overline{x}_i)$ is the nearest neighbor distance of $\overline{x}_i$. Ideally, a point $\overline{x}$ is exactly symmetrical with respect to some $\overline{c}$ if $d_1 = 0$. However, considering the uncertainty of the location of a point as a sphere of radius $d_{NN}^{max}/2$ around it, d_1 and d_2

are bounded as $d_1 \leq \frac{d_{NN}^{max}}{2}$ and $d_2 \leq \frac{3 \times d_{NN}^{max}}{2}$, resulting in

$$\frac{d_1 + d_2}{2} \leq d_{NN}^{max}.$$

Thus, the threshold θ is kept equal to d_{NN}^{max}, making its computation automatic and without user intervention.

5.4 Some Properties of $d_{ps}(\overline{x}, \overline{c})$

It is to be noted that $d_{ps}(\overline{x}, \overline{c})$ is not a metric. It is a way of measuring the amount of point symmetry between a point and a cluster center, rather than the distance like any Minkowski distance.

In this section, some properties of $d_{ps}(\overline{x}, \overline{c})$ are established [28]. For this purpose, some terms are also defined.

Definition 5.1 The Euclidean distance ratio (EDR) property is defined as follows:

Let $\overline{x}$ be a data point, $\overline{c}_1$ and $\overline{c}_2$ be two cluster centers, and Δ be a distance measure. Here, a point is assigned to cluster i according to the following assignment rule: $i = \text{argmin}_v \Delta(\overline{x}, \overline{c}_v)$, where $\overline{c}_v$ is the center of cluster v. Let $\Delta_1 = \Delta(\overline{x}, \overline{c}_1)$, $\Delta_2 = \Delta(\overline{x}, \overline{c}_2)$, $d_{e_1} = d_e(\overline{x}, \overline{c}_1)$ and $d_{e_2} = d_e(\overline{x}, \overline{c}_2)$. Then Δ is said to satisfy the EDR property if and only if, for $\frac{\Delta_1}{\Delta_2} < \frac{d_{e2}}{d_{e1}}$, point $\overline{x}$ is assigned to $\overline{c}_1$; otherwise, it is assigned to $\overline{c}_2$.

Observation 5.1 *The symmetry measure satisfies the Euclidean distance ratio property.*

Proof Let us assume that there are two clusters, having cluster centers $\overline{c}_1$ and $\overline{c}_2$. Let $\overline{x}$ be a particular data point. Let the *knear* nearest neighbors of the reflected point of $\overline{x}$ with respect to center $\overline{c}_1$ and $\overline{c}_2$ be at distances of $d_i^{(1)}$ and $d_i^{(2)}$, respectively, for $i = 1, \ldots, knear$. Then, $d_{ps}(\overline{x}, \overline{c}_1) = d_{sym}(\overline{x}, \overline{c}_1) \times d_{e1} = \frac{\sum_{i=1}^{knear} d_i^{(1)}}{knear} \times d_{e1}$, and similarly $d_{ps}(\overline{x}, \overline{c}_2) = d_{sym}(\overline{x}, \overline{c}_2) \times d_{e2} = \frac{\sum_{i=1}^{knear} d_i^{(2)}}{knear} \times d_{e2}$, where d_{e1} and d_{e2} are the Euclidean distances between $\overline{x}$, $\overline{c}_1$ and $\overline{x}$, $\overline{c}_2$, respectively. Now, in order to preserve the EDR property, given that

$$\frac{d_{sym}(\overline{x}, \overline{c}_1)}{d_{sym}(\overline{x}, \overline{c}_2)} < \frac{d_{e2}}{d_{e1}}, \tag{5.12}$$

the point $\overline{x}$ is assigned to center $\overline{c}_1$. Point $\overline{x}$ is assigned to the cluster of $\overline{c}_1$ if $d_{ps}(\overline{x}, \overline{c}_1) < d_{ps}(\overline{x}, \overline{c}_2)$. This indicates that $\frac{\sum_{i=1}^{knear} d_i^{(1)}}{knear} \times d_{e1} < \frac{\sum_{i=1}^{knear} d_i^{(2)}}{knear} \times d_{e2} \rightarrow \frac{\frac{\sum_{i=1}^{knear} d_i^{(1)}}{knear}}{\frac{\sum_{i=1}^{knear} d_i^{(2)}}{knear}} < \frac{d_{e2}}{d_{e1}} \rightarrow \frac{d_{sym}(\overline{x}, \overline{c}_1)}{d_{sym}(\overline{x}, \overline{c}_2)} < \frac{d_{e2}}{d_{e1}}$. It therefore becomes evident that d_{sym} satisfies the EDR property defined in Definition 5.1. □

It may be noted that, for data sets having convex clusters of different densities, the point symmetry-based distance will detect the appropriate clustering as long as the condition in Eq. 5.12 holds well for the points.

Definition 5.2 If two clusters are symmetrical to each other with respect to a third cluster center, then these clusters are called "symmetrical interclusters".

Observation 5.2 *The d_{ps} measure is able to detect symmetrical interclusters properly as long as* $\frac{d_{sym}(\overline{x},\overline{c}_1)}{d_{sym}(\overline{x},\overline{c}_2)} < \frac{d_e(\overline{x},\overline{c}_2)}{d_e(\overline{x},\overline{c}_1)}$.

Proof Let us assume that there are three clusters, having cluster centers $\overline{c}_1$, $\overline{c}_2$, and $\overline{c}_3$. Let cluster 1 and cluster 3 be symmetrical to each other with respect to the second cluster center. Thus, clusters 1 and 3 are symmetrical interclusters. Let $\overline{x}$ be a particular data point in cluster 1. For this data point, $\frac{d_{sym}(\overline{x},\overline{c}_1)}{d_{sym}(\overline{x},\overline{c}_2)} < \frac{d_e(\overline{x},\overline{c}_2)}{d_e(\overline{x},\overline{c}_1)}$ is satisfied. This means that $\frac{d_{sym}(\overline{x},\overline{c}_1)}{d_{sym}(\overline{x},\overline{c}_2)} < \frac{d_e(\overline{x},\overline{c}_2)}{d_e(\overline{x},\overline{c}_1)} \Rightarrow d_{sym}(\overline{x},\overline{c}_1) \times d_e(\overline{x},\overline{c}_1) < d_{sym}(\overline{x},\overline{c}_2) \times d_e(\overline{x},\overline{c}_2) \Rightarrow d_{ps}(\overline{x},\overline{c}_1) < d_{ps}(\overline{x},\overline{c}_2)$. Thus, point $\overline{x}$ will be assigned to the cluster of $\overline{c}_1$. This will happen for all points of cluster 1. Similarly, points which should belong to cluster 3 will form cluster 3. Thus, the d_{ps} measure is able to detect the symmetrical interclusters properly. □

The above observation is also evident from Fig. 5.5, in which the first and the third clusters are "symmetrical interclusters" with respect to the middle one. As explained in the above example, although there exists a symmetrical point of $\overline{x}$ with respect to cluster center $\overline{c}_2$, but $\overline{x}$ is assigned to the first cluster as the newly developed d_{ps} distance satisfies the EDR property. As a result, the three clusters present in Fig. 5.5 are identified properly. Thus it is proved that the point symmetry-based distance is able to detect symmetrical interclusters properly.

It is evident that the symmetrical distance computation is very time consuming because it involves the computation of the nearest neighbors. Computation of $d_{ps}(\overline{x}_i,\overline{c})$ is of complexity $O(nd)$, where d is the dimension of the data set and n is the total number of points present in the data set. Hence, for K clusters, the time complexity of computing point symmetry distance between all points to different clusters is $O(n^2Kd)$. In order to reduce this computational complexity, an approximate nearest neighbor search using the Kd-tree approach is adopted in this chapter.

5.5 Kd-Tree-Based Nearest Neighbor Computation

A space-partitioning data structure used for arranging points in K dimensional space is the *Kd*-tree or K-dimensional tree [4]. The splitting planes used by a *Kd*-tree are only those which are perpendicular to one of the coordinate axes. In case of a nearest neighbor problem, a set of data points in d-dimensional space is given.

These points are preprocessed into a data structure, so that, given any query point q, the nearest or generally k nearest points of p to q can be reported efficiently. ANN (approximate nearest neighbor) is a library developed in C++.[1] The use of data structures and algorithms improvised for finding precise as well as comparative nearest neighbors in arbitrary high-dimensional space are supported by ANN. In [27], ANN is used to find d_1 and d_2 in Eq. 5.10 efficiently. The ANN library includes various data structures established using Kd-trees and box-decomposition trees. It also uses different types of search schemes. The Kd-tree data structure has been used in this chapter.

The function performing the k-nearest neighbor search in ANN is given a query point q, a nonnegative integer k, an array of point indices nn_{idx}, and an array of distances *dists*. Both arrays are assumed to contain at least k elements. This procedure computes the k nearest neighbors of q in the point set, and stores the indices of the nearest neighbors in the array nn_{idx}. Optionally a real value $\varepsilon \geq 0$ may be supplied. If so, then the ith nearest neighbor is a $(1+\varepsilon)$ approximation to the true ith nearest neighbor; that is, the true distance to this point may exceed the true distance to the real ith nearest neighbor of q by a factor of $(1+\varepsilon)$. If ε is omitted then the nearest neighbors will be computed exactly.

For computing $d_{ps}(\overline{x}, \overline{c})$ in Eq. 5.10, d_1 and d_2, the first two nearest neighbors of $\overline{x}^*$ (where $\overline{x}^* = 2 * \overline{c} - \overline{x}$), need to be computed. This is a computationally intensive task that can be speeded up by using the *Kd*-tree-based nearest neighbor search. Here, the exact nearest neighbor is computed, so ε is set equal to 0. The query point q in ANN is set equal to $\overline{x}^*$, and k is set to 2.

The next section describes a genetic algorithm-based clustering technique that uses a measure of cluster symmetry, computed using $d_{ps}(\overline{x}, \overline{c})$, for optimization.

5.6 GAPS: The Genetic Clustering Scheme with New PS-Distance [27]

A genetic algorithm-based clustering technique which uses the point symmetry-based distance is described in this section. The method is referred to as GAPS clustering [27]. Here, the number of clusters is assumed to be known a priori, and $d_{ps}(\overline{x}, \overline{c})$ is used to compute a clustering metric, which is optimized by a genetic algorithm. The basic steps of GAPS, which closely follow those of the conventional GA, are described in Figs. 5.7 and 5.8.

5.6.1 Chromosome Representation and Population Initialization

In GAPS, center-based chromosome encoding is used. Each string is a sequence of real numbers representing K cluster centers. The K cluster centers encoded in

[1] http://www.cs.umd.edu/~mount/ANN/

Begin
1. $t = 0$
2. initialize population $P(t)$ /* $Popsize = |P|$ */
3. for $i = 1$ to $Popsize$
   call clustering_PS() procedure for $P(i)$ and
   stores the inverse of the result in $M[i]$
   /* $M[i]$ stores the fitness of chromosome $P[i]$ */
4. $t = t + 1$
5. if termination criterion achieved go to step 10
6. select (P)
7. crossover (P)
8. mutate (P)
9. go to step 3
10. output best chromosome and stop
End

Fig. 5.7 Basic steps of GAPS

Procedure: clustering_PS()
Assignment of data points:
1. For all data points $\overline{x}_i$, $1 \leq i \leq n$, compute
   $k^* = \text{argmin}_{k=1,\ldots,K}\, d_{ps}(\overline{x}_i, \overline{c}_k)$
2. If $(d_{ps}(\overline{x}_i, \overline{c}_{k^*})/d_e(\overline{x}_i, \overline{c}_k) < \theta)$
   /*$d_e(\overline{x}_i, \overline{c}_k)$ is the Euclidean distance between the point $\overline{x}_i$ and cluster centroid $\overline{c}_k$*/
   assign the data point $\overline{x}_i$ to the k^*th cluster.
3. Otherwise, the data point is assigned to the k^* cluster where
   $k^* = \text{argmin}_{k=1,\ldots,K}\, d_e(\overline{x}, \overline{c}_k)$

Clustering metric calculation:
   $Clustering\_metric = \sum_{i=1}^{K} \sum_{j=1}^{n_i} d_{ps}(\overline{x}_j^i, \overline{c}_k)$
   where n_i is the total number of points in cluster i and
   $\overline{x}_j^i$ is the jth point of the ith cluster.

Update of centers:
   Compute the new centroids of the K clusters as follows:
   $\overline{c}_k(t+1) = \frac{\sum_{i \in S_k(t)} \overline{x}_i}{N_k}$,
   where $k = 1, \ldots, K$ and $S_k(t)$ is the set of elements that are assigned to the kth cluster at generation t and $N_k = |S_k|$.

Fig. 5.8 Clustering procedure: Clustering-PS()

each chromosome are initialized to K randomly chosen points from the data set. This process is repeated for each of the *Popsize* chromosomes in the population, where *Popsize* is the size of the population. Thereafter, five iterations of the K-means algorithm are executed with the set of centers encoded in each chromosome. The resultant centers are used to replace the centers in the corresponding chromosomes. This makes the centers separated initially.

Example 5.2 Let the space be two dimensional and the number of clusters considered be three. Thus, the chromosome

51.6 72.3 18.3 15.7 29.2 34.5

represents the three cluster centers (51.6, 72.3), (18.3, 15.7), and (29.2, 34.5). Note that each real number in the chromosome is an indivisible gene.

5.6.2 Fitness Computation

In order to compute the fitness of the chromosomes, the clustering procedure, clustering-PS() (as shown in Fig. 5.8), is called. Here a point $\overline{x}_i$, $1 \leq i \leq n$, is assigned to cluster k iff $d_{ps}(\overline{x}_i, \overline{c}_k) \leq d_{ps}(\overline{x}_i, \overline{c}_j)$, $j = 1, \ldots, K$, $j \neq k$, and $d_{sym}(\overline{x}_i, \overline{c}_k) = (d_{ps}(\overline{x}_i, \overline{c}_k)/d_e(\overline{x}_i, \overline{c}_k)) \leq \theta$. For $(d_{ps}(\overline{x}_i, \overline{c}_k)/d_e(\overline{x}_i, \overline{c}_k)) > \theta$, point $\overline{x}_i$ is assigned to some cluster m iff $d_e(\overline{x}_i, \overline{c}_m) \leq d_e(\overline{x}_i, \overline{c}_j)$, $j = 1, 2 \ldots K$, $j \neq m$. In other words, point $\overline{x}_i$ is assigned to that cluster with respect to whose center its d_{ps} is minimum, provided the corresponding d_{sym} value is less than some threshold θ. Otherwise, assignment is done based on the minimum Euclidean distance criterion as normally used in [20] or the K-means algorithm. The reason for doing such an assignment is as follows: In the intermediate stages of the algorithm, when the centers are not yet properly evolved, the minimum d_{ps} value for a point is expected to be quite large, since the point might not be symmetric with respect to any center. In such cases, using Euclidean distance for cluster assignment appears to be intuitively more appropriate. In contrast, when d_{ps} values are reasonably small, cluster assignment based on symmetry becomes more meaningful.

The value of θ is kept equal to the maximum nearest neighbor distance among all the points in the data set, as explained in Sect. 5.3. Thus, the computation of θ is automatic and does not require user intervention. After the assignments are done, the cluster centers encoded in the chromosome are replaced by the mean points of the respective clusters. Subsequently, for each chromosome *clustering_metric*, M is calculated as defined below:

$$M = \sum_{i=1}^{K} \sum_{j=1}^{n_i} d_{ps}\left(\overline{x}_j^i, \overline{c}_k\right), \tag{5.13}$$

where n_i is the number of points assigned to cluster i, and $\overline{x}_j^i$ denotes the jth point of the ith cluster. The fitness function of that chromosome, $F(s_i)$, is then defined as the inverse of M, i.e.,

$$F(s_i) = \frac{1}{M}. \tag{5.14}$$

This fitness function, $F(s_i)$, will be maximized by using a genetic algorithm. (Note that there could be other ways of defining the fitness function.)

Example 5.3 Let the data set consist of nine points. $(1, 2)$, $(2, 4)$, $(2, 5)$, $(3, 6)$, $(3, 9)$, $(4, 1)$, $(4, 5)$, $(4, 9)$, and $(5, 2)$. Let, initial cluster centers be $C1 = (2, 3)$, $C2 = (3, 5)$, and $C3 = (4, 7)$. The nearest neighbor distances of the nine data points are $\sqrt{5}$, 1, 1, $\sqrt{2}$, 1, $\sqrt{2}$, $\sqrt{2}$, 1, and $\sqrt{2}$, respectively. Then, $d_{NN}^{max} = \max\{\sqrt{5}, 1, 1, \sqrt{2}, 1, \sqrt{2}, \sqrt{2}, 1, \sqrt{2}\} = \sqrt{5} = 2.236068$. Here, $d_{ps}(x1, C1) = 1.7071$. Then, $\text{argmin}_{i=1}^{3} d_{ps}(x1, C_i) = 1$, and the corresponding $d_{sym}(x1, C1) = 1.2071 < d_{NN}^{max}$. So $x1$ is assigned to the cluster of $C1$ using the PS-distance. Similarly, $d_{ps}(x9, C1) = 9.2155$, $d_{ps}(x9, C2) = 9.1301$, $d_{ps}(x9, C3) = 15.7108$, and $\text{argmin}_{i=1}^{3} d_{ps}(x9, Ci) = 2$, but $(d_{sym}(x9, C2) = 2.5322) > d_{NN}^{max}$. Thus, $x9$ can not be assigned using the point symmetry-based distance. It will be assigned using the minimum Euclidean distance-based criterion. $x9$ is assigned to the jth cluster, where $j = \text{argmin}_{i=1}^{3} d_e(x9, Ci) = 1$. Thus, $x9$ will be assigned to first cluster based on the Euclidean distance.

In this way, points $(1, 2)$, $(4, 1)$, and $(5, 2)$ will form cluster 1, points $(2, 4)$, $(2, 5)$, $(3, 6)$, and $(4, 5)$ will form cluster 2, and points $(3, 9)$ and $(4, 9)$ will form cluster 3. Then, new centers will be formed as follows: $C1 = (\frac{1+4+5}{3}, \frac{2+1+2}{3}) = (\frac{10}{3}, \frac{5}{3}) = (3.33, 1.67)$, $C2 = (\frac{2+2+3+4}{4}, \frac{4+5+6+5}{4}) = (\frac{11}{4}, \frac{20}{4}) = (2.75, 5)$, $C3 = (\frac{3+4}{2}, \frac{9+9}{2}) = (\frac{7}{2}, 9) = (3.50, 9)$.

5.6.3 Selection

Roulette wheel selection [112] is used to implement the proportional selection strategy.

5.6.4 Crossover

Here, the normal single point crossover [129] is used. The crossover probability is selected adaptively as in [262]. The expressions for the crossover probabilities are computed as follows: Let f_{max} be the maximum fitness value of the current population, $\overline{f}$ be the average fitness value of the population, and f' be the larger of the fitness values of the solutions to be crossed. Then, the probability of crossover, μ_c, is calculated as

$$\mu_c = k_1 \times \frac{(f_{max} - f')}{(f_{max} - \overline{f})}, \quad \text{if } f' > \overline{f}, \tag{5.15}$$

$$\mu_c = k_3, \quad \text{if } f' \leq \overline{f}. \tag{5.16}$$

Here, as in [262], the values of k_1 and k_3 are kept equal to 1.0. Note that, when $f_{max} = \overline{f}$, then $f' = f_{max}$ and μ_c will be equal to k_3. The aim behind this adaptation is to achieve a trade-off between exploration and exploitation in a different manner.

The value of μ_c is increased when the better of the two chromosomes to be crossed is itself quite poor. In contrast, when it is a good solution, μ_c is low so as to reduce the likelihood of disrupting a good solution by crossover.

Example 5.4 Let a particular population contain ten chromosomes having fitness values 10.2, 6.7, 14.3, 11.9, 6.9, 8.7, 9.1, 9.5, 10.2, and 10.5, respectively. Then, the maximum fitness value of this population is 14.3. The average fitness value of the population is $\frac{10.2+6.7+14.3+11.9+6.9+8.7+9.1+9.5+10.2+10.5}{10} = 9.8$. Let the two chromosomes to be crossed be chromosome 2 and chromosome 3. Then, $f' = \max\{6.7, 14.3\} = 14.3$. Here, $\overline{f} = 9.8$ and $f_{max} = 14.3$. Now, $f' > \overline{f}$, so the probability of crossover is calculated as : $\mu_c = 1.0 \times \frac{14.3-14.3}{14.3-9.8} = 0$. Thus, the probability of crossover in this case is 0.

5.6.5 Mutation

Each chromosome undergoes mutation with probability μ_m. The mutation probability is also selected adaptively for each chromosome as in [262]. The expression for the mutation probability, μ_m, is:

$$\mu_m = k_2 \times \frac{(f_{max} - f)}{(f_{max} - \overline{f})} \quad \text{if } f > \overline{f}, \tag{5.17}$$

$$\mu_m = k_4 \quad \text{if } f \leq \overline{f}. \tag{5.18}$$

Here, the values of k_2 and k_4 are kept equal to 0.5. This adaptive mutation helps the GA to come out of local optima. When, GA converges to a local optimum, i.e., when $f_{max} - \overline{f}$ decreases, μ_c and μ_m will both be increased. As a result, the GA will come out of the local optimum. This will also happen for the global optimum and may result in disruption of near-optimal solutions. As a result, GA will never converge to the global optimum. However, as μ_c and μ_m get lower values for high-fitness solutions and higher values for low-fitness solutions, the high-fitness solutions aid in the convergence of the GA, while the low-fitness solutions prevent the GA from getting stuck at local optima. The use of elitism will also keep the best solution intact. For a solution with the maximum fitness value, μ_c and μ_m are both zero. The highest scoring individual of a population is copied intact to the next generation. Usage of both elitism and selection may cause exponential growth of solutions in the population, compelling the GA to converge prematurely. To overcome this above problem, a default mutation rate (of 0.02) is kept for every solution in the GAPS.

Example 5.5 Recall Example 5.4. Let chromosome 2 be selected for mutation. Here, $f = 6.7$, $f_{max} = 14.3$, and $\overline{f} = 9.8$. Then, the probability of mutation (μ_m) is calculated as: As $f < \overline{f}$, $\mu_m = k_4 = 0.5$. If chromosome 4 is selected for mutation, then $f = 11.9$. Thus, $f > \overline{f}$. Then, μ_m is calculated as: $\mu_m = k_2 \times \frac{f_{max}-f}{f_{max}-\overline{f}} =$

$0.5 \times \frac{14.3-11.9}{11.9-9.8} = 0.5 \times \frac{2.4}{2.1} = 0.5 \times \frac{8}{7} = \frac{4}{7} = 0.57$. Thus, here probability of mutation is 0.57.

Here, each position in a chromosome is mutated with probability μ_m in the following way: The value is replaced with a random variable drawn from a Laplacian distribution, $p(\varepsilon) \propto e^{-\frac{|\varepsilon-\mu|}{\delta}}$, where the scaling factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. The scaling factor δ is chosen equal to 2. The old value at the position is replaced with the newly generated value.

5.6.6 Termination

In GAPS, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the clustering problem. Elitism has been implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus, on termination, this location contains the centers of the final clusters.

5.6.7 Complexity Analysis

Below the complexity of GAPS clustering is analyzed.

1. As discussed in Sect. 5.5, the *Kd*-tree data structure has been used in order to find the nearest neighbor of a particular point. The construction of a *Kd*-tree requires $O(n \log n)$ time and $O(n)$ space [4].
2. Initialization of the GA needs *Popsize* $\times$ *stringlength* time, where *Popsize* and *stringlength* indicate the population size and the length of each chromosome in the GA, respectively. Note that, for K clusters in d-dimensional space, *stringlength* will become $K \times d$.
3. Fitness is computed by calling the *clustering_PS* procedure.
   a. In order to assign each point to a cluster, the minimum symmetrical distance of that point with respect to all clusters is calculated. For this purpose the *Kd*-tree-based nearest neighbor search is used. If the points are roughly uniformly distributed, then the expected case complexity is $O(c^d + \log n)$, where c is a constant depending on the dimension and the point distribution. This is $O(\log n)$ if the dimension d is a constant [34]. Friedman et al. [103] also reported $O(\log n)$ expected time for finding the nearest neighbor. So, in order to find the minimal symmetrical distance of a particular point, $O(K \log n)$ time is needed.

      For n points, the total complexity becomes $O(Kn \log n)$.
   b. The complexity for updating the centers is $O(K)$.

So the overall complexity for fitness evaluation is $= O(Popsize \times Kn \log n)$.
4. The selection step of the GA requires $O(Popsize \times stringlength)$ time.
5. Mutation and crossover require $O(Popsize \times stringlength)$ time each.

So, in general, the overall time complexity becomes $O(Kn \log n \times Popsize)$ per generation. For maximum *maxgen* number of generations, the overall complexity becomes $O(Kn \log n \times Popsize \times maxgen)$. As *Popsize* is constant, the overall complexity of GAPS clustering is $O(nK \log n \times maxgen)$.

5.7 On the Convergence Property of GAPS

Using finite Markov chain theory, it has been proved that the canonical genetic algorithms converge to the global optimum [236]. In [167], it has also been proved along the lines of [236] that the genetic K-means algorithm also converges to the global optimum of the square-error (SE), depending on some conditions on its parameters. Here, the global convergence of GAPS clustering to minimize symmetrical compactness is proved along similar lines by deriving some conditions on the parameters of GAPS clustering that ensure the global convergence.

5.7.1 Preliminaries

Consider the process $\{\mathscr{P}(t)\}$, $t \geq 0$, where $\mathscr{P}(t)$ represents the population maintained by GAPS at the tth generation. The state space, $\mathscr{S}$, of this process refers to the space of all possible populations. The states of this state space can be numbered from 1 to $|\mathscr{S}|$. Moreover, the state space is restricted to the populations containing valid strings, i.e., strings representing different partitions with K non-empty clusters. Therefore, according to the definition of GAPS, $\mathscr{P}(t+1)$ can be completely determined by $\mathscr{P}(t)$ as follows:

$$\begin{aligned} &Pr\big\{\mathscr{P}(t) = p_t \big| \mathscr{P}(t-1) = p_{t-1}, \ldots, \mathscr{P}(0) = p_0\big\} \\ &\quad = Pr\big\{\mathscr{P}(t) = p_t \big| \mathscr{P}(t-1) = p_{t-1}\big\}. \end{aligned}$$

Hence, $\{\mathscr{P}(t)\}$, $t \geq 0$ is a Markov chain. The transition probabilities are independent of the time instant, i.e., if

$$p_{ij}(t) = Pr\big\{\mathscr{P}(t) = p_j \big| \mathscr{P}(t-1) = p_i\big\},$$

then $p_{ij}(s) = p_{ij}(t)$ for all $p_i, p_j \in \mathscr{S}$ and for all $s, t \geq 1$. Thus, it can be concluded that $\{\mathscr{P}(t)\}$, $t \geq 0$ is a time-homogeneous finite Markov chain. Let $\mathbf{P} = (p_{ij})$ be the *transition matrix* of the process $\{\mathscr{P}(t)\}$, $t \geq 0$. The entries of the matrix $\mathbf{P}$ satisfy $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|\mathscr{S}|} p_{ij} = 1$, $\forall i \in \mathscr{S}$. Satisfaction of the above-mentioned condition is the required qualification of a stochastic matrix. Since $\mathbf{P}$ satisfies this, it can

be considered as a *stochastic matrix*. A few terms are defined below for use in the rest of this section.

A square matrix $\mathbf{A}_{m\times m}$ is said to be positive if $a_{ij} > 0, \forall i, j \in \{1, 2, \ldots, m\}$, and is said to be *primitive* if there exists a positive integer k such that $\mathbf{A}^k$ gives a positive value. A *column-allowable* matrix is a square matrix with at least one positive entry in each column.

The requirement of the following theorem is that the matrix $\mathbf{P}$ should be a primitive matrix. So, firstly investigation is necessary to find the conditions on the operators needed for the matrix $\mathbf{P}$ to be primitive. The probabilistic changes of the chromosome within the population caused by the operators used in GAPS are captured by the transition matrix $\mathbf{P}$, which can be decomposed in a natural way into a product of stochastic matrices

$$\mathbf{P} = \mathbf{K} \times \mathbf{C} \times \mathbf{M} \times \mathbf{S}, \tag{5.19}$$

where $\mathbf{K}$, $\mathbf{C}$, $\mathbf{M}$, and $\mathbf{S}$ describe the intermediate transitions caused by K-means such as the update center, crossover, mutation, and selection operators, respectively. It is easy to consider that all these matrices are stochastic matrices.

Proposition 5.1 *Stochastic matrices form a group under matrix multiplication.*

Thus, for the two stochastic matrices $\mathbf{K}$ and $\mathbf{C}$, by Proposition 5.1, $\mathbf{C}' = \mathbf{K} \times \mathbf{C}$ is also a stochastic matrix. Therefore, Eq. 5.19 can be written as

$$\mathbf{P} = \mathbf{C}' \times \mathbf{M} \times \mathbf{S}, \tag{5.20}$$

where $\mathbf{C}'$, $\mathbf{M}$, and $\mathbf{S}$ are all stochastic matrices.

Proposition 5.2 *Let* $\mathbf{C}'$, $\mathbf{M}$, *and* $\mathbf{S}$ *be stochastic matrices, where* $\mathbf{M}$ *is positive and* $\mathbf{S}$ *is column-allowable. Then the product* $\mathbf{C}' \times \mathbf{M} \times \mathbf{S}$ *is positive.*

A positive matrix is always a primitive matrix. Therefore, to make the matrix $\mathbf{P}$ primitive, the product of $\mathbf{C}' \times \mathbf{M} \times \mathbf{S}$ has to be positive. For the above-mentioned product of the stochastic matrices to be positive, $\mathbf{M}$ needs to be positive and $\mathbf{S}$ needs to be column-allowable.

To Check Whether the Mutation Matrix is Positive The matrix $\mathbf{M}$ is positive if any valid string $s \in \mathscr{S}$ can be obtained from any another valid string after application of the corresponding mutation operator. In the GAPS clustering technique, during the mutation operation, a particular center is modified by a value generated using a Laplacian distribution. Hence, there is a non-zero probability of generating any valid position from any other valid position, while the probability of generating a value near the old value is greater. This implies that the above-defined mutation operation can change any valid string to any other valid string with some nonzero probability. Hence, the transition matrix $\mathbf{M}$ corresponding to the above mutation operator is positive.

Conditions on Selection The probability of survival of a string in the current population depends on the fitness value of the string; so is the transition matrix due to selection, **S**. In GAPS, the fitness function of each chromosome is defined as in Eq. 5.14. So, the fitness value of each chromosome in the population is strictly positive. Therefore, the probability that the selection does not alter the present state, s_{ii}, can be bounded as follows:

$$s_{ii} \geq \frac{F(s_1)}{\sum_{l=1}^{P} F(s_l)} \times \frac{F(s_2)}{\sum_{l=1}^{P} F(s_l)} \times \cdots \times \frac{F(s_P)}{\sum_{l=1}^{P} F(s_l)} \tag{5.21}$$

$$= \frac{\prod_{l=1}^{P} F(s_l)}{(\sum_{l=1}^{P} F(s_l))^P} > 0 \quad \forall i \in S. \tag{5.22}$$

Here, s_l represents the lth string of the current population and $F(s_l)$ is the fitness value associated with the lth string. Even though this bound changes with the generation, it is always strictly positive; hence the selection matrix **S** is *column-allowable*.

5.7.2 Convergence Proof

Theorem *Let $X(t) = F(s^*(t))$, where $s^*(t)$ is the string with maximum fitness value encountered during the evolution of GAPS clustering till the time instant t. Let the mutation operator be the same as defined in Sect. 5.6.5, and the fitness function be as defined in Eq. 5.14. Then*

$$\lim_{t \to \infty} Pr\{X(t) = \mathscr{S}^*\} = 1, \tag{5.23}$$

where $\mathscr{S}^* = \max\{F(i) | i \in \mathscr{T}\}$, $\mathscr{T}$ is the set of all legal/valid strings.

Proof According to the proof provided in Ref. [236], Theorem 6, a canonical GA whose transition matrix is primitive and which maintains the best solution found over time converges to the global optimum in the sense given in Eq. 5.23. It is proved in Proposition 2 that the transition matrix of GAPS clustering with the mutation operator defined in Sect. 5.6.5 is positive. Since every positive matrix is primitive, thus the transition matrix of GAPS is also primitive. It may be noted that GAPS clustering uses an elitist model of GA, i.e., it maintains the best solution obtained up to the present time. Thus, the theorem follows from Ref. [236, Theorem 6]. □

The above theorem implies that $X(t)$, the maximum fitness value of the strings found by GAPS clustering till the instant t, converges to the global optimum S^*, with probability 1 when t goes to infinity.

5.8 Experimental Results of GAPS

5.8.1 Data Sets Used

A short description of the data sets used for the experiments is provided below.

1. Artificial data sets: Three artificial data sets are used.
   a. *Sym_3_2*: This data set is a combination of ring-shaped, compact, and linear clusters shown in Fig. 5.9(a). The total number of points in it is 350.
   b. *AD_5_2*: This data set consists of 250 data points distributed over five spherically shaped clusters as shown in Fig. 5.9(b). The clusters present here are highly overlapping, each consisting of 50 data points.
   c. *Bensaid_3_2*: This is a two-dimensional data set consisting of 49 points distributed in three clusters as shown in Fig. 5.9(c). This data set, used in Ref. [33], consists of two small clusters (one has 6 elements and the other has 3) separated by a large (40 element) cluster.
2. Real-life data sets: The three real-life data sets were obtained from [2].

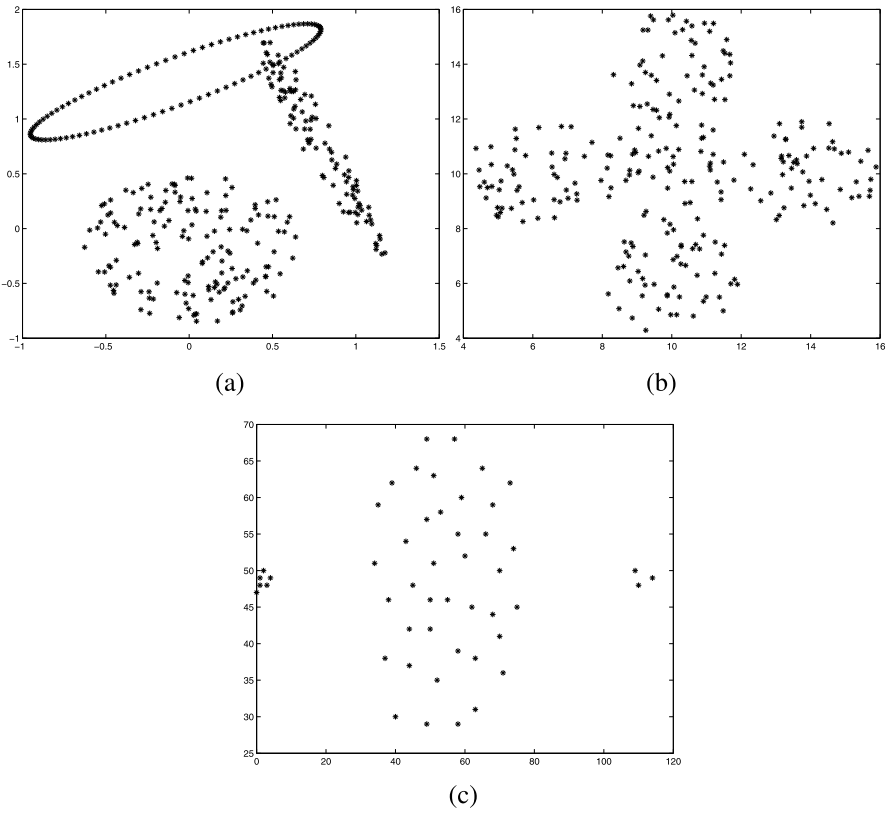


Fig. 5.9 (**a**) *Sym_3_2*, (**b**) *AD_5_2*, (**c**) *Bensaid_2_2*

a. *Iris*: The *Iris* data set consists of 150 data points distributed over three clusters. Each cluster consists of 50 points. This data set represents different categories of irises characterized by four feature values [100]. It has three classes: Setosa, Versicolor, and Virginica. It is known that two classes (Versicolor and Virginica) have a large amount of overlap while the class Setosa is linearly separable from the other two.
b. *Cancer*: This *Wisconsin Breast Cancer* data set consists of 683 sample points. Each pattern has nine features corresponding to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. There are two categories in the data: malignant and benign. The two classes are known to be linearly separable.
c. *LungCancer*: This data consists of 32 instances having 56 features each. The data describes three types of pathological lung cancer.

5.8.2 Implementation Results

The experimental results comparing the performance of the GAPS, SBKM [266], Mod-SBKM [58], K-means algorithm, genetic algorithm-based K-means clustering technique (GAK-means) [188], average linkage clustering technique (AL) [145], and expectation maximization clustering technique (EM) [145] are provided for the three artificial and three real-life data sets. For, SBKM algorithm, θ is set equal to 0.18, as suggested in [266]. For Mod-SBKM, θ is chosen as 0.5. In contrast, for the newly developed GAPS clustering, the value of θ is determined from the data set as discussed in Sect. 5.3.

For GAPS, the crossover probability, μ_c, and mutation probability, μ_m, are determined adaptively as described in Sects. 5.6.4 and 5.6.5, respectively. The population size, P, is set equal to 100. The total number of generations is kept equal to 50. Executing it further did not improve the performance.

In order to compare the obtained clustering results quantitatively, the *Minkowski scores* [146] are reported for each algorithm. This is a measure of the quality of a solution given the true clustering. Let T be the "true" solution and S the solution which will be measured. Denote by n_{11} the number of pairs of elements that are in the same cluster in both S and T. Denote by n_{01} the number of pairs that are in the same cluster only in S, and by n_{10} the number of pairs that are in the same cluster in T. The *Minkowski score* (MS) is then defined:

$$\text{MS(T, S)} = \sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}. \tag{5.24}$$

For MS, the optimum score is 0, with lower scores being "better". Here, each algorithm is executed five times on each data set. The best MS scores obtained for these five runs are reported in Table 5.1 for all data sets.

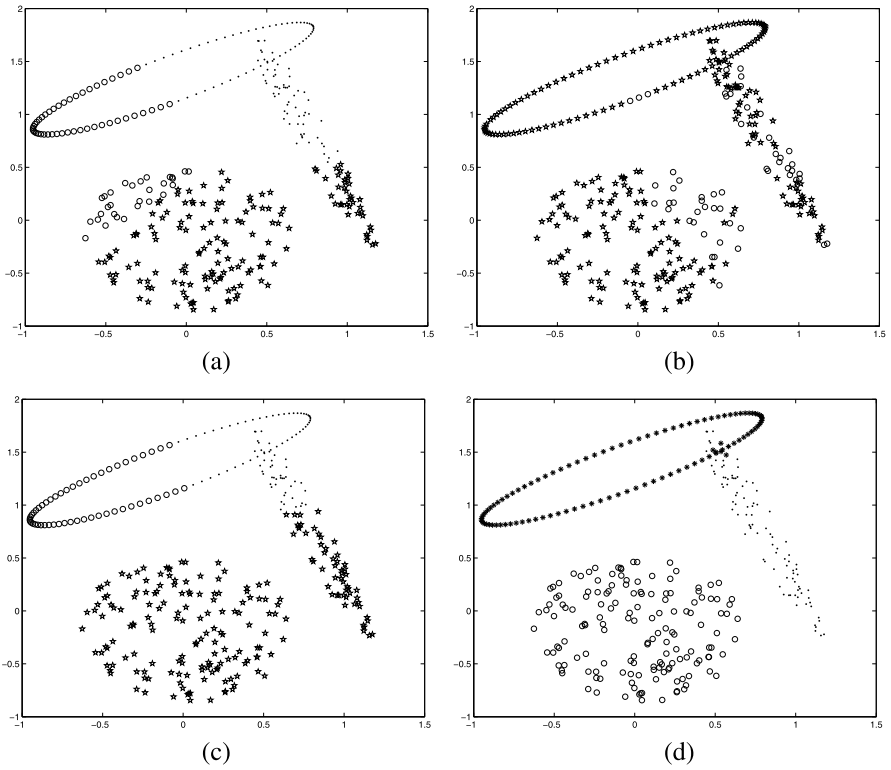


Fig. 5.10 Clustering of *Sym_3_2* for $K = 3$ after application of (**a**) K-means, (**b**) SBKM, (**c**) Mod-SBKM, and (**d**) GAPS

1. *Sym_3_2*: The clusters present in this data set are internally symmetrical, but the clusters themselves are not symmetrical with respect to any intermediate point.

   The final results obtained after application of K-means, SBKM, Mod-SBKM, GAPS, GAK-means, AL, and EM for *Sym_3_2* are shown in Figs. 5.10(a), 5.10(b), 5.10(c), 5.10(d), 5.11(a), 5.11(b), and 5.11(c), respectively. Both SBKM and Mod-SBKM (Figs. 5.10(b) and 5.10(c)) are not able to detect the appropriate partitioning from this data set. In contrast, GAPS (Fig. 5.10(d)) groups the points that are lying inside the ring, but actually belong to the elongated elliptic cluster, with those of the ring. In the absence of any class information about the points, such a grouping is not really surprising. The K-means, GAK-means, AL, and EM clustering techniques are found to fail in providing the proper clustering. These results are also evident from the MS values attained by the seven clustering techniques as reported in Table 5.1. It is to be noted that for the above data set K-means is unable to provide the correct clustering. However, the use of PS-distance in GAPS enables it to detect such clusters properly.

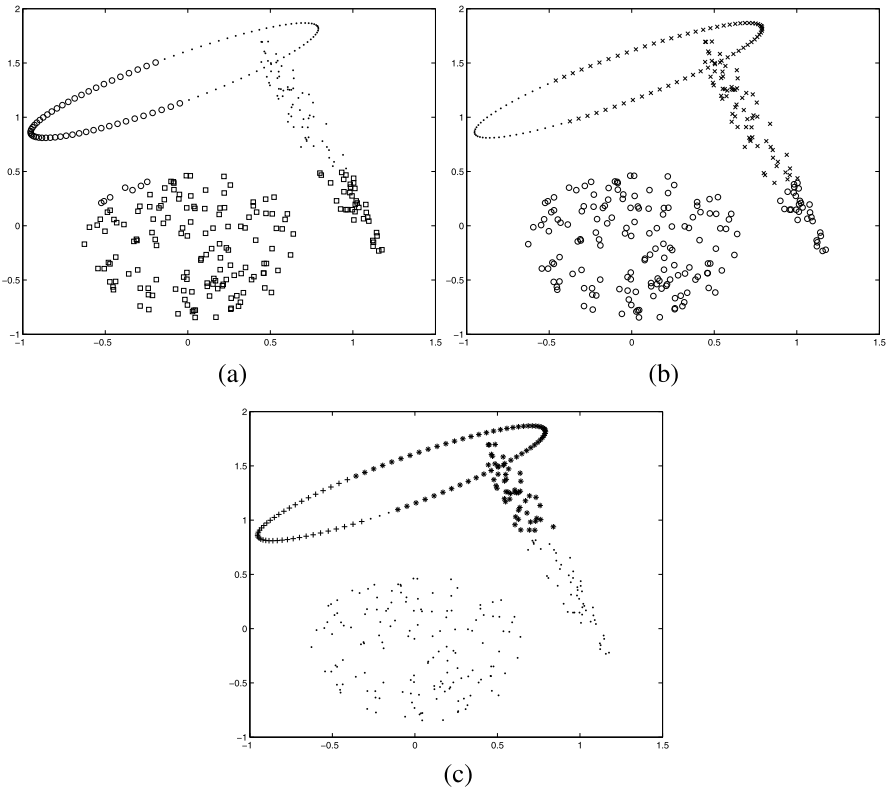


Fig. 5.11 Clustered *Sym_3_2* for $K = 3$ after application of (**a**) GAK-means clustering technique, (**b**) average linkage clustering technique, and (**c**) expectation maximization clustering technique

Table 5.1 Best *Minkowski score* obtained by the seven algorithms for all data sets. Here, EM and AL denote the 'expectation maximization' and 'average linkage' clustering techniques, respectively. Smaller values of MS indicate better partitioning

Data set	K-means	SBKM	Mod-SBKM	GAK-means	EM	AL	GAPS
Sym_3_2	0.91	1.28	0.85	0.83	0.58	0.80	**0.12**
AD_5_2	**0.25**	1.33	0.72	0.47	0.50	0.44	0.51
Bensaid_3_2	0.68	0.87	0.39	0.72	**0.0**	**0.0**	**0.0**
Iris	0.68	0.96	0.65	0.60	0.60	0.70	**0.59**
Cancer	0.37	0.37	0.37	**0.36**	0.43	0.45	**0.36**
LungCancer	1.46	1.09	1.09	1.24	0.96	0.96	**0.89**

2. *AD_5_2*: The clusters present in this data set, used earlier in [20], are internally symmetrical, and clusters are also symmetrical with respect to some intermediate point. Figures 5.12(a), 5.12(b), 5.12(c), 5.12(d), 5.13(a), 5.13(b), and 5.13(c) show the clustering results for K-means, SBKM, Mod-SBKM, GAPS,

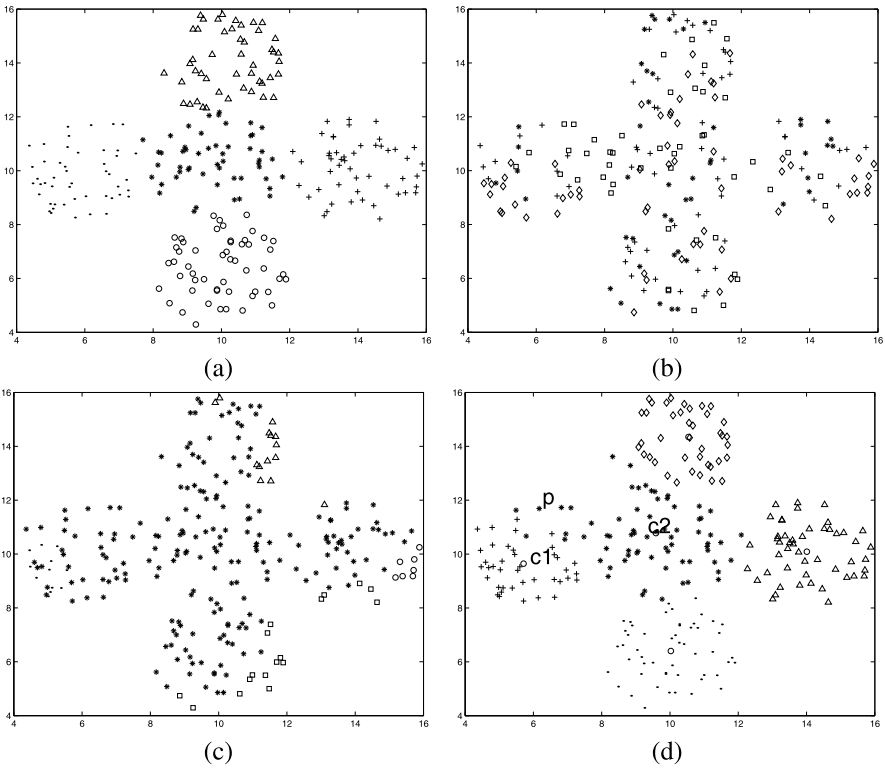


Fig. 5.12 Clustering of *AD_5_2* for $K = 5$ after application of (**a**) K-means, (**b**) SBKM, (**c**) Mod-SBKM, and (**d**) GAPS

GAK-means, AL, and EM, respectively. As is evident, K-means performs the best for this data. Although GAPS is also able to detect the clusters reasonably well, it is found to somewhat over approximate the central cluster (which extends to the left). The reason is as follows: Let us take a point p which actually belongs to the left cluster but after application of GAPS as is included in the central cluster (shown in Fig. 5.12(d)). It can be seen from the figure that the point p is more symmetrical with respect to the central cluster, $c2$. Here, even though $d_e(p, c2)$ is greater than $d_e(p, c1)$, due to the lower symmetry with respect to $c1$, p is assigned to the central cluster. Both SBKM and Mod-SBKM fail in detecting the proper clustering here because data points are more symmetrical with respect to some other cluster center than the actual cluster center (because of the limitations in the definitions of d_s and d_c). The point to be noted here is that the SBKM method destroys the proper clustering achieved by the K-means method. This is demonstrated in Figs. 5.14(a)–(d) which show how the cluster centers move with iterations during the application of SBKM. Evidently, because of the presence of symmetrical interclusters, SBKM is trying to bring all the cluster centers to the center of the whole

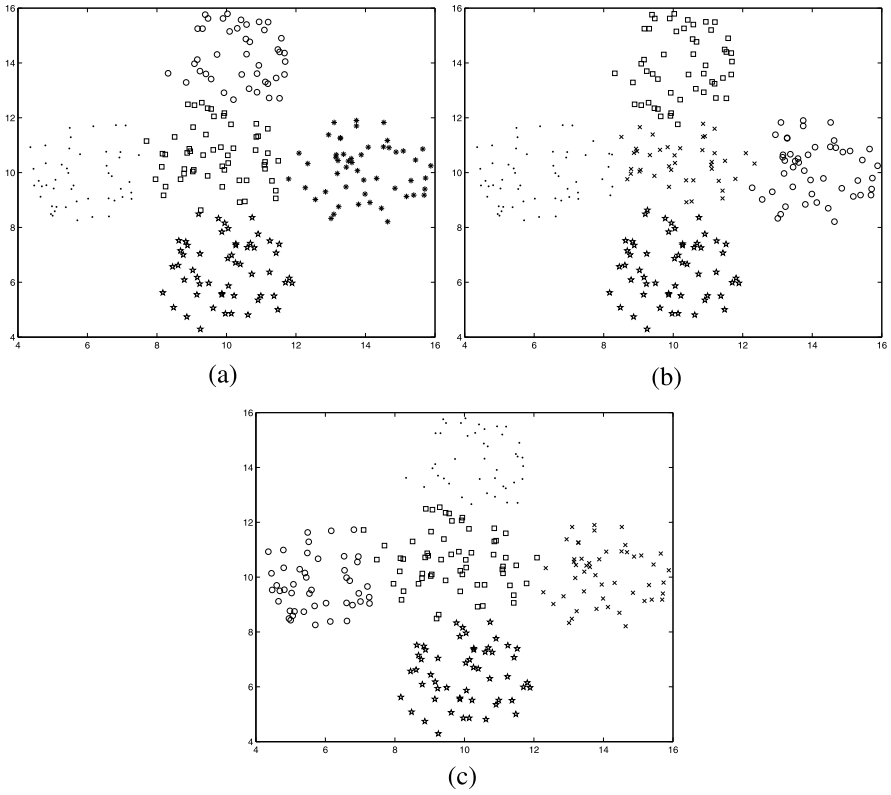


Fig. 5.13 Clustered *AD_5_2* for $K = 5$ after application of (**a**) GAK-means clustering technique, (**b**) average linkage clustering technique, and (**c**) expectation maximization clustering technique

data set, thereby providing very poor performance. The GAK-means and average linkage clustering techniques perform moderately for this data set (refer to Table 5.1).

3. *Bensaid_3_2*: This data set, used in [33], is taken in order to show that the GAPS clustering algorithm is able to find the proper clustering from a data set where clusters are of significantly different sizes. (Note that clusters of widely varying sizes may be present in several real-life domains, e.g., medical images, satellite images, fraud detection.) Figures 5.15(a), 5.15(b), 5.15(c), 5.15(d), and 5.15(e) show the partitionings obtained by *K*-means, SBKM, Mod-SBKM, GAPS/AL/EM, and GAK-means, respectively. GAPS along with the AL and EM clustering techniques are able to find the proper partitioning, while the other four algorithms fail (refer to Table 5.1).
4. Real-life data sets: This category consists of three real-life data sets: *Iris*, *Cancer*, and *LungCancer*. These data sets are obtained from [2]. Statistical analysis of variance (ANOVA) [5] is performed for the real-life data sets on the

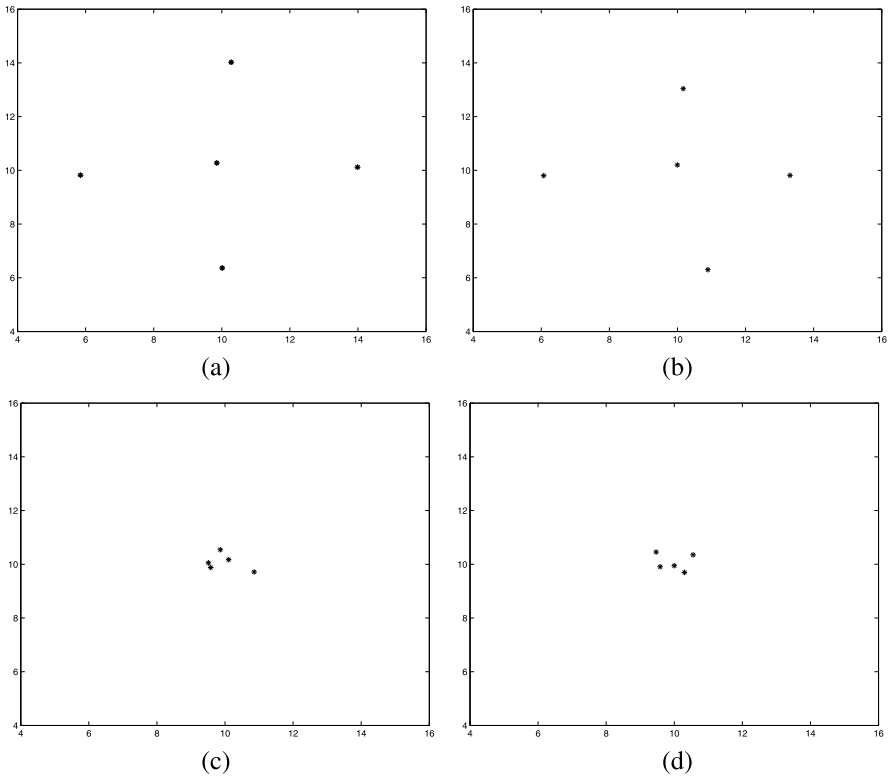


Fig. 5.14 Change of the cluster centers obtained by SBKM on *AD_5_ 2* after (**a**) application of the *K*-means algorithm, (**b**) 1 iteration, (**c**) 10 iterations, (**d**) 20 iterations

combined MS values of the seven algorithms when each is executed five times. ANOVA results are reported in detail for *Iris* (Table 5.2) for the purpose of illustration.

a. *Iris*: As seen from Table 5.1, the MS scores of GAPS is the best for *Iris*, followed by GAK-means and EM. However, it can be seen from Table 5.2 that the differences in the means of the MS scores of GAPS with GAK-means and EM are not significant, indicating their similar performance. The performance of the SBKM algorithm is found to be the poorest.
b. *Cancer*: As can be seen from Table 5.1, the performance of *K*-means, SBKM, Mod-SBKM, GAK-means, and GAPS are similar. ANOVA tests show that the differences in mean MS scores of GAPS with respect to these four algorithms are not statistically significant. The results indicate that the two clusters are convex as well as highly symmetrical.
c. *LungCancer*: The MS scores, reported in Table 5.1, demonstrate the superior performance of the GAPS clustering technique. ANOVA statistical analysis

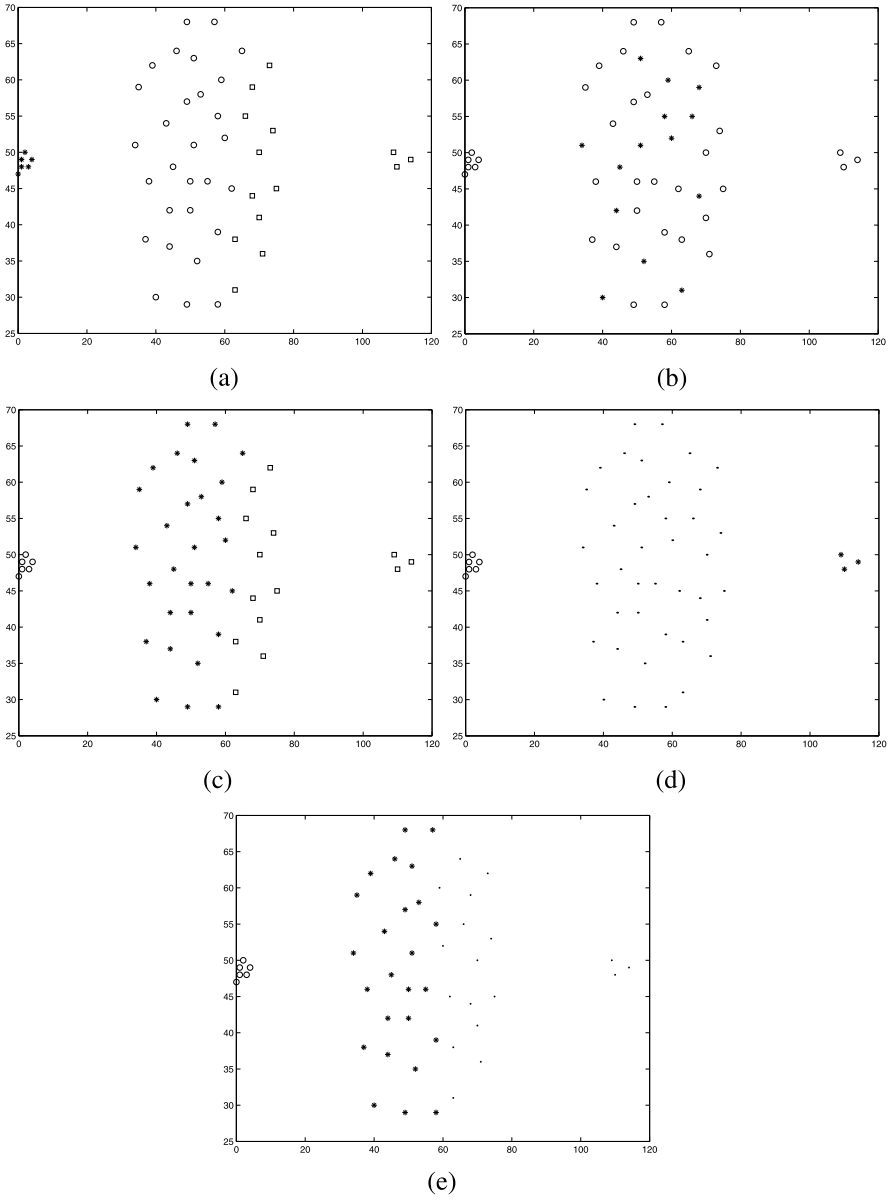


Fig. 5.15 Clustered *Bensaid_3_2* for $K = 3$ after application of (**a**) K-means, (**b**) SBKM, (**c**) Mod-SBKM, (**d**) GAPS/average linkage/expectation maximization clustering techniques, and (**e**) GAK-means clustering technique

is also done here. The analysis shows that the mean MS differences of all the algorithms are statistically significant.

Table 5.2 Estimated marginal means and pairwise comparisons of different algorithms on *Minkowski score* obtained by statistical analysis of variance (ANOVA) testing for *Iris* data (GAPS: GP, K-means: KM, Mod-SBKM: MSB, SBKM: SB, GAK-means: GAK, expectation maximization: EM, average linkage: AL). Smaller values of MS indicate better partitioning

Algo. name (I)	Comparing algo. (J)	Mean difference ($I - J$)	Significance value
GP	KM	$-9 \times 10^{-2} \pm 0.00163$	<0.01
	MSB	$-6.0 \times 10^{-2} \pm 0.004$	<0.01
	SB	-0.37 ± 0.023	<0.01
	GAK	$-1.00 \times 10^{-2} \pm 0.0001$	0.54
	EM	$-1.00 \times 10^{-2} \pm 0.02$	0.59
	AL	-0.11 ± 0.015	<0.01
KM	GP	$9 \times 10^{-2} \pm 0.00163$	<0.01
	MSB	$3.73 \times 10^{-2} \pm 0.004$	<0.01
	SB	-0.27864 ± 0.01	<0.01
	GAK	$8.0 \times 10^{-2} \pm 0.021$	<0.01
	AL	$-2.0 \times 10^{-2} \pm 0.015$	0.45
	EM	0.08 ± 0.025	<0.01
MSB	GP	$6.0 \times 10^{-2} \pm 0.004$	<0.01
	KM	$-3.73 \times 10^{-2} \pm 0.004$	<0.01
	SB	-0.3132 ± 0.0012	<0.01
	GAK	$5.0 \times 10^{-2} \pm 0.005$	<0.01
	AL	$-5.0 \times 10^{-2} \pm 0.0041$	<0.01
	EM	0.05 ± 0.003	<0.01
SB	GP	0.37 ± 0.023	<0.01
	KM	0.27864 ± 0.01	<0.01
	MSB	0.3132 ± 0.0012	<0.01
	GAK	0.36 ± 0.021	<0.01
	AL	0.26 ± 0.018	<0.01
	EM	0.36 ± 0.001	<0.01
GAK	GP	$1.00 \times 10^{-2} \pm 0.0001$	0.54
	KM	$-8.0 \times 10^{-2} \pm 0.021$	<0.01
	SB	-0.36 ± 0.021	<0.01
	MSB	$-5.0 \times 10^{-2} \pm 0.005$	<0.01
	AL	0.28 ± 0.02	<0.01
	EM	0.00 ± 0.001	1.00

5.8.3 Summary

It can be seen from the above results that GAPS is able to find the proper clustering where SBKM and Mod-SBKM succeed while K-means fails, as well as where

Table 5.2 (Continued)

Algo. name (I)	Comparing algo. (J)	Mean difference ($I - J$)	Significance value
EM	GP	$1.00 \times 10^{-2} \pm 0.02$	0.59
	KM	-0.08 ± 0.025	<0.01
	SB	-0.36 ± 0.001	<0.01
	MSB	-0.05 ± 0.003	<0.01
	GAK	0.00 ± 0.001	1.00
	AL	-0.10 ± 0.02	<0.01
AL	GP	0.11 ± 0.015	<0.01
	KM	$2.0 \times 10^{-2} \pm 0.015$	0.45
	SB	-0.26 ± 0.018	<0.01
	MSB	$5.0 \times 10^{-2} \pm 0.0041$	<0.01
	GAK	-0.28 ± 0.02	<0.01
	EM	0.10 ± 0.02	<0.01

K-means succeeds while SBKM and Mod-SBKM fail. The results on *Bensaid_3_2* show that GAPS is able to detect symmetric clusters irrespective of their sizes where SBKM, Mod-SBKM, and K-means all fail. The superiority of GAPS is also established on three real-life data sets. These real-life data sets are of different characteristics with the number of dimensions varying from 4 to 56. Results on six artificial and real-life data sets establish the fact that GAPS is well suited to detect clusters of widely varying characteristics. The improved performance of GAPS can be attributed to the fact that, in the new point symmetry-based distance, d_{ps}, there is an impact of both the symmetrical distance as well as the Euclidean distance. This was lacking in the earlier definitions of the point symmetry-based distances [58, 266], which resulted in some serious problems as discussed in Sect. 5.2 and displayed pictorially in Fig. 5.5.

K-means, SBKM, and Mod-SBKM are based on local search and hence may often get stuck at local optima depending on the choice of the initial cluster centers. The use of a genetic algorithm in GAPS in order to minimize the total symmetrical distance overcomes this problem. Moreover, the use of adaptive mutation and crossover probabilities helps GAPS to converge faster. GAPS also performs better than GAK-means clustering technique, which, being based on the principles of the K-means clustering technique, can only detect hyperspherical-shaped clusters. Thus, it also fails for data sets of type *Sym_3_2*, etc. AL is able to detect proper partitioning for well-separated clusters only, while EM is able to detect clusters that are normally distributed. The experimental results on a wide variety of data sets show that GAPS is able to detect any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristic of symmetry. In other words, GAPS can detect clusters of different shapes and sizes that form a superset of the types captured by most of the other methods considered. Based on this observation, and the fact that the property of symmetry is widely evident

in many real-life situations, application of GAPS in most clustering tasks seems justified.

The clustering techniques discussed in this chapter assumes the number of clusters present in a data set a priori. However, in real-life, this assumption does not hold true. It is very difficult to know the actual number of clusters present in a data set. In order to determine the number of clusters present in a data set automatically, cluster validity indices are proposed. These indices basically measure the goodness of a particular partitioning. In the next chapter we discuss several cluster validity indices. Some symmetry-based cluster validity indices have also been proposed in the recent past. A detailed introduction to these indices is also provided in the next chapter. Elaborate experimental results are presented to illustrate the comparative performance analyses of the existing indices.

Chapter 6
A Validity Index Based on Symmetry: Application to Satellite Image Segmentation

6.1 Introduction

The three fundamental questions that need to be addressed in any typical clustering scenario are: (i) What is the model of a data set or what is a good clustering technique suitable for a given data set (ii) What is the model order of the data i.e., how many clusters are actually present in the data and (iii) How real or good is the clustering itself. It is well-known to the pattern recognition community that different algorithms are applicable for data with different characteristics; For example, while K-means [96] is widely used for hyperspherical, convex, equisized clusters, it is known to fail where the clusters are not hyperspherical and also significantly unequal in size. Similarly, average (or single) linkage clustering algorithms [96] work well for non-overlapping clusters of any shape, but fail if the clusters overlap. The Gaussian Mixture models [44] are considered to be the appropriate partitioning techniques for data sets whose type of distribution (e.g., Gaussian) is known. Clustering methods such as SBKM [266], Mod-SBKM [58], and GAPS exploit the symmetry property within the clusters. These methods are found to be superior to several other techniques when the clusters offer a symmetric structure. Thus, given a wide choice of methods, determining an appropriate clustering algorithm invokes a challenge.

The tasks of determining the number of clusters and also the validity of the clusters formed [189] are generally addressed by providing several definitions of validity indices. Several cluster validity indices have been proposed in the literature. These include the Davies-Bouldin (DB) index [189], Dunn's index [87], Xie-Beni (XB) index [295], I-index [189], CS-index [59], etc., to name just a few. Some of these indices have been found to be able to detect the correct partitioning for a given number of clusters, while some can determine the appropriate number of clusters as well. However, the effectiveness of these indices in determining the proper clustering algorithm has seldom been studied. The present chapter describes such an investigation that was reported in [28, 240, 243, 244].

Most of the validity measures usually assume a certain geometrical structure in the shapes of all the clusters. But if different clusters possess different structural

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_6,

properties, these indices are often found to fail. In this chapter a symmetry-based cluster validity index named *Sym*-index [28] is defined which uses the new distance d_{ps}. This index is able to determine the appropriate clustering method, the proper partitioning, and the correct number of clusters from data sets having any type of clusters irrespective of their shape, size or convexity as long as they possess the property of point symmetry.

The newly defined d_{ps} has been incorporated in the definitions of several cluster validity indices to develop symmetry-based versions of these indices [244]. The performance of these symmetry-based validity indices has been compared with existing original cluster validity indices and with *Sym*-index, showing the effectiveness of the latter. It also demonstrates the utility of incorporating the measure of symmetry in each index. This study is described in a part of this chapter.

6.2 Some Existing Cluster Validity Indices

A large number of cluster validity indices exist in the literature. Cluster validity indices can be of two types: external and internal. External validity indices use the true partitioning (provided by user) information while validating a particular partition. But in unsupervised classification, it is often difficult to generate such information. Thus, external validity indices are seldom used to validate partitionings. Some common examples of such indices are *Minkowski scores*, *F-measures*, etc. Internal validity indices rely on the intrinsic structure of the data. Most of the internal validity indices quantify how good a particular partitioning is in terms of the compactness and separation between clusters:

- Compactness: The proximity among the cluster elements can be measured using this. Variance is a commonly used measure of compactness.
- Separability: In order to differentiate between two clusters, this measure is used. One commonly used measure of separability is the distance between two cluster centers. This measure is easy to compute and can detect hyperspherical-shaped clusters well.

Among the internal cluster validity indices some well-known cluster validity indices are BIC-index [230], CH-index [47], Silhouette-index [234], DB-index [73], Dunn-index [87], XB-index [295], PS-index [58], and *I*-index [189].

6.2.1 BIC Index

The Bayesian information criterion (BIC) [230], which basically determines which probability-based mixture is the most appropriate, is based on Baye's theorem [230]. This index is developed to remove the problem of overfitting. It is defined as follows:

$$BIC = -\ln(L) + v \times \ln(n). \tag{6.1}$$

Here, n is the number of objects, L is the likelihood of the parameters generating the data in the model, and v is the number of free parameters in the Gaussian model. Thus, the BIC index is devised in such a way that it takes into account both the fit of the model to the data and the complexity of the model. Smaller value of BIC indicates a better model.

6.2.2 Calinski-Harabasz Index

This index [47] is computed by

$$CH = \frac{trace(S_B)}{trace(S_w)} \times \frac{n_p - 1}{n_p - K}, \tag{6.2}$$

where (S_B) is the between-cluster scatter matrix and (S_w) is the internal scatter matrix, n_p is the number of clustered samples, and K is the number of clusters. Higher value of CH-index [47] indicates a better partitioning.

6.2.3 Silhouette Index

Silhouette index [23, 234] is a cluster validity index that is used to judge the quality of any clustering solution. Suppose a represents the average distance of a point from the other points of the cluster to which the point is assigned, and b represents the minimum of the average distances of the point from the points of the other clusters. Now the silhouette width s of the point is defined as:

$$s = \frac{(b - a)}{\max\{a, b\}}. \tag{6.3}$$

Silhouette index *Sil* is the average Silhouette width of all the data points and it reflects the compactness and separation of clusters. The value of Silhouette index varies from -1 to 1 and higher value indicates better clustering result.

6.2.4 DB Index

This index [18, 73] is a function of the ratio of the sum of *within-cluster scatter* to *between-cluster separation*. The scatter within the ith cluster is computed as

$$S_{i,q} = \left\{ \frac{\sum_{\overline{x} \in X_i} |\overline{x} - \overline{c}_i|^q}{|X_i|} \right\}^{\frac{1}{q}}, \tag{6.4}$$

and the distance between cluster X_i and X_j is defined as

$$d_{i,j,t} = \|\overline{c}_i - \overline{c}_j\|_t.$$

$S_{i,q}$ is the qth moment of the $|X_i|$ points in cluster X_i with respect to their mean $\overline{c}_i$ and is a measure of the dispersion of the points in the cluster. Generally, $S_{i,1}$ is used, which is the average Euclidean distance of the vectors in class i to the centroid of class i. $d_{i,j,t}$ is the Minkowski distance of order t between the centroids $\overline{c}_i$ and $\overline{c}_j$ that characterize clusters X_i and X_j. Subsequently, one computes

$$R_{i,q,t} = \max_{j, j \neq i} \frac{S_{i,q} + S_{j,q}}{d_{i,j,t}}. \tag{6.5}$$

The Davies-Bouldin (DB) index is then defined as

$$DB = \frac{1}{K} \sum_{i=1}^{K} R_{i,q,t}. \tag{6.6}$$

Here K denotes the number of clusters. The objective is to minimize the DB index for achieving the proper clustering.

6.2.5 Dunn's Index

Let S and T be two nonempty subsets of R^N [18, 87]. Then the diameter $\triangle$ of S and set distance δ between S and T are

$$\triangle(S) = \max_{\overline{x}, \overline{y} \in S} d(\overline{x}, \overline{y})$$

and

$$\delta(S, T) = \min_{\overline{x} \in S, \overline{y} \in T} d(\overline{x}, \overline{y}),$$

where $d(\overline{x}, \overline{y})$ is the distance between points $\overline{x}$ and $\overline{y}$. For any partition, Dunn [87] defined the following index:

$$Dunn = \min_{1 \leq i \leq K} \left\{ \min_{1 \leq j \leq K, j \neq i} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \leq k \leq K} \triangle(C_k)} \right\} \right\}.$$

Larger values of Dunn correspond to good clusters, and the number of clusters that maximizes the Dunn index is taken as the optimal number of clusters.

6.2.6 Xie-Beni Index

Xie and Beni [295] proposed a cluster validity index (XB) that again focused on two properties: compactness and separation. According to the definition of the XB-index, the numerator measures the compactness of the obtained fuzzy partition while

the denominator measures the separation between clusters. In general, a good partition has a small value for the compactness, and well-separated centers will produce a high value for the separation. Hence, the most desirable partition is obtained by minimizing the XB-index for $k = 1, \ldots, K_{max}$.

$$XB = \frac{\sum_{i=1}^{K} \sum_{j=1}^{n} \mu_{ij}^2 \|\overline{x}_j - \overline{c}_i\|^2}{n(\min_{i \neq k} \|\overline{c}_i - \overline{c}_k\|^2)}.$$

6.2.7 PS Index

Consider a partition of the data set $X = \overline{x}_j$, $j = 1, 2, \ldots, n$ and the center of each cluster $\overline{c}_i$ $(i = 1, \ldots, K)$. The cluster validity index, PS-index [58], is defined as

$$PS = \frac{1}{K} \sum_{i=1}^{K} \left\{ \frac{1}{n_i} \sum_{j \in S_i} \frac{d_c(\overline{x}_j, \overline{c}_i)}{\min_{m,n=1,\ldots,K, m \neq n}(d_e(\overline{c}_m, \overline{c}_n))} \right\}, \tag{6.7}$$

where S_i is the set whose elements are the data points assigned to the ith cluster and n_i is the number of elements in S_i. $d_c(\overline{x}_j, \overline{c}_i)$ is computed using Eq. 5.6. The most desirable partition is obtained by minimizing the PS-index.

6.2.8 I-Index

The I-index [189] is defined as follows:

$$I = \left(\frac{1}{K} \times \frac{E_1}{E_k} \times D_k \right)^2, \tag{6.8}$$

where E_1 is constant for a given data set. Here, $E_k = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|\overline{x}_i - \overline{c}_k\|^2$ and $D_k = \max_{i,j=1}^{K} \|\overline{c}_i - \overline{c}_j\|^2$, where n_i is the number of points in cluster C_i and $\overline{c}_k$ is the center of the kth cluster. The best partitioning occurs at the maximum value of the function.

6.2.9 CS-Index

The CS-index [59] is defined as follows:

$$CS = \frac{\sum_{i=1}^{K} [\frac{1}{N_i} \sum_{\overline{x}_i \in X_i} \max_{\overline{x}_q \in X_i} d(\overline{x}_i, \overline{x}_q)]}{\sum_{i=1}^{K} [\min_{j \in K, j \neq i} d(\overline{c}_i, \overline{c}_q)]}, \tag{6.9}$$

where $c_i, i = 1, \ldots, K$ are the cluster centers. According to Chou et al., the CS measure is more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures, the price being paid in terms of high computational load with increasing K and n.

6.3 *Sym*-Index: The Symmetry-Based Cluster Validity Index

In this section a new cluster validity index called *Sym*-index [243, 244] is described that is based on d_{ps}. An explanation of the interaction among the different components of the index is mentioned so that it can indicate the proper partitioning of the data. A theoretical analysis of the index is also provided.

6.3.1 The Cluster Validity Measure

6.3.1.1 Motivation

Consider a crisp partition of the data set $X = \{\overline{x}_j : j = 1, 2, \ldots, n\}$. The center of each cluster $\overline{c}_i$ can be computed by using $\overline{c}_i = \frac{\sum_{j=1}^{n_i} \overline{x}_j^i}{n_i}$, where n_i $(i = 1, 2, \ldots, K)$ is the number of points in cluster i and $\overline{x}_j^i$ is the jth point of cluster i.

Definition 6.1 The total symmetrical deviation of a particular cluster i is given by $E_i = \sum_{j=1}^{n_i} d_{ps}^*(\overline{x}_j^i, \overline{c}_i)$.

$d_{ps}^*(\overline{x}_j^i, \overline{c}_i)$ is computed by Eq. 5.10 with some constraint. Here, first *knear* nearest neighbors of $\overline{x}_j^* = 2 \times \overline{c}_i - \overline{x}_j^i$ will be searched among the points which are already in cluster i; i.e., now the *knear* nearest neighbors of the reflected point $\overline{x}_j^*$ of the point $\overline{x}_j^i$ with respect to $\overline{c}_i$ and $\overline{x}_j^i$ should belong to the ith cluster.

Definition 6.2 Total compactness of the partitioning in terms of symmetry is denoted by $\mathscr{E}_K$ and is given by

$$\mathscr{E}_K = \sum_{i=1}^{K} E_i. \tag{6.10}$$

Definition 6.3 D_K is called the separation of the crisp K-partitioning, where D_K is the maximum distance between any two cluster centroids, i.e.,

$$D_K = \max_{i,j=1}^{K} \|\overline{c}_i - \overline{c}_j\|. \tag{6.11}$$

For a good partitioning, the total compactness of the partitioning in terms of symmetry should be minimized while the cluster separation in terms of maximum distance

between any two cluster centers should be maximized. Motivated by these observations the *Sym*-index [28, 243] has been formulated as described below.

6.3.1.2 Formulation of *Sym*-Index

The newly developed point symmetry distance is used to define a cluster validity function which measures the overall average symmetry with respect to the cluster centers [28]. This is inspired by the I-index developed in [189]. The new cluster validity function *Sym* is defined as

$$Sym(K) = \left(\frac{1}{K} \times \frac{1}{\mathscr{E}_K} \times D_K \right), \tag{6.12}$$

where K is the number of clusters. The objective is to maximize this index in order to obtain the actual number of clusters and the proper partitioning.

6.3.1.3 Explanation

As formulated in Eq. 6.12, the *Sym*-index is a composition of three factors: $1/K$, $1/\mathscr{E}_K$, and D_K. The first factor increases as K decreases; as *Sym*-index needs to be maximized for optimal clustering, this factor prefers to decrease the value of K. The second factor is a measure of the total within-cluster symmetry. For clusters which have good symmetrical structures, the $\mathscr{E}_K$ value is lower. Note that, as K increases, in general, the clusters tend to become more symmetric. Moreover, as $d_e(\overline{x}, \overline{c})$ in Eq. 5.10 also decreases, $\mathscr{E}_K$ decreases, resulting in an increase in the value of the *Sym*-index. Since the *Sym*-index needs to be maximized, it will prefer to increase the value of K. Finally the third factor, D_K, measuring the maximum separation between a pair of clusters, increases with the value of K. Note that the value of D_K is bounded by the maximum separation between a pair of points in the data set. As these three factors are complementary in nature, they are expected to compete with and balance each other critically for determining the proper partitioning.

The use of D_K, as a measure of separation, requires further elaboration [247]. Instead of using the maximum separation between two clusters, several other alternatives could have been used; For example, if D_K was the sum of pairwise inter-cluster distances in a K-cluster structure, then it would increase greatly with increasing value of K. This might lead to the formation of a maximum possible number of clusters equal to the number of elements in the data set. If D_K was the average inter cluster distance, it would decrease at each step with K instead of being increased. So, this will only leave us with the minimum possible number of clusters. The minimum distance between two clusters may be another choice for D_K. However, this measure would also decrease significantly with increase in the number of clusters. So this would lead to a structure where the loosely connected sub structures remain as they were, where in fact a separation is expected. Thus, maximum separability

may not be attained. In contrast, if we consider the maximum inter cluster separation, then we see that this tends to increase significantly until we reach the maximum separation among compact clusters, when it becomes almost constant. The upper bound of this value, which is equal to the maximum separation between two points, is only attainable when we have two extreme data elements as two single-element clusters. But the terminating condition is reached well before this situation. This is the reason why we try to improve the maximum distance between two maximally separated clusters.

It may appear that consideration of D_K, as defined in Eq. 6.11, may lead to an undesirable clustering where two maximally separated clusters (say A and B) have been found, while another cluster C (which can be divided into more than one cluster) may lie in between. However, in such situations, considering only D_K, C cannot be divided into its component clusters. We show in the following paragraphs that in such cases, the other two factors become dominant and are able to provide the requisite clustering. This follows the line of reasoning provided in [212].

If there is any intermediate divisible cluster(s) between the extreme ones, this fact is taken into account by the total symmetrical distance of all the clusters and the number of clusters. It is seen that, when division is possible in the intermediate cluster, the second factor overrides the effect of the first one, and the reverse is true for an indivisible cluster. In order to show it analytically, we consider spherically approximated clusters. We assume that each cluster may be approximated by a hyper sphere having uniform distribution of elements. If a d-dimensional data set is considered and $r_1, r_2, \ldots, r_d$ are the radii of a cluster along these directions, then $r = (r_1 + r_2 + \cdots + r_d)/d$ is considered to be the radius of the spherically approximated cluster. Let the exact reflected point of every point with respect to the cluster center exist in the data set, i.e., the spherical cluster is totally symmetrical. Then, the total symmetrical distance of all the points with respect to the cluster center is given by $\sum(\sum_{i=2}^{knear} \frac{d_i}{knear} \times d_e)$, since here $d_1 = 0$. Assuming that the $\sum_{i=2}^{knear} \frac{d_i}{knear}$ values of all the points are almost the same, say α, (as the cluster is fully symmetrical) then the Euclidean distance is the only factor playing an important role here. Thus, for a spherically approximated cluster with radius r and n number of elements, its total within-cluster symmetrical distance is $\sim n\alpha\frac{r}{2}$.

If such a cluster of radius r having n number of elements is divided into two equal halves with radius $r/2$ and $n/2$ number of elements in each, then, total symmetrical distance of these two newly formed clusters will become $n\alpha r/4$. Note that, if we try to divide a compact symmetrical cluster into two almost equal halves (which are also compact symmetrical clusters), then its total symmetrical distance will be approximately halved.

Let $\mathscr{E}_K$ denote the sum of the within-cluster symmetrical distances of a K-cluster configuration and P_K denote the corresponding *Sym*-index value. Now let us consider a K cluster configuration where $K-1$ number of clusters are of radius r having n elements in each, and one intermediate cluster is of radius $2r$ with $2n$ elements. The sum of the within-cluster symmetrical distances for this configuration is $\mathscr{E}_K = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2n\alpha r = \frac{n\alpha r}{2}(K+3)$.

For this configuration, the larger intermediate cluster will be the natural candidate for division at the next step; hence, D_K will remain the same even after division.

(Let us assume again that division will produce two clusters of equal sizes with equal symmetry.) Now, two cases may arise. The candidate cluster may have a clear tendency for division, or it may be a compact symmetrical one. In the former case we have $\mathscr{E}_{K+1} = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + n\alpha\frac{r}{2} + n\alpha\frac{r}{2} = n\alpha\frac{r}{2}(K+1)$. Thus, $\frac{\mathscr{E}_K}{\mathscr{E}_{K+1}} = \frac{K+3}{K+1}$ and

$$\frac{P_K}{P_{K+1}} = \frac{(K+1)(K+1)}{K(K+3)} = \frac{K^2+2K+1}{K^2+3K},$$

since D_K is the same in both the K and $K+1$ cluster configurations. The factor is less than 1 for all $K > 1$; i.e., a division is suggested.

In the latter case, after division, the radius of each of the resultant clusters will be $(r + (d-1)2r)/d = ((2d-1)r)/d$. So, we have $\mathscr{E}_{K+1} = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2(\frac{2d-1}{d} \times \frac{n\alpha r}{2})$.

Thus, $\frac{\mathscr{E}_K}{\mathscr{E}_{K+1}} = \frac{K+3}{(K-1)+2(2d-1)/d}$ and

$$\frac{P_K}{P_{K+1}} = \frac{(K+1)(K-1)+2(2d-1)/d}{K(K+3)} = \frac{(K^2-1)+2(K+1)(2d-1)/d}{K^2+3K},$$

which is greater than 1 for all K (and $d > 1$); i.e., division is not suggested. These observations are also verified by our experimental results.

So, it is seen that, for small values of K, the second and third factors play an important role in revealing the maximum attainable separation and getting compact symmetrical clusters. As K grows, the effect of these two factors is overcome by the first factor. If, in any case, the maximum separation is reached before the desired symmetrical compactness, then the first and the second factors interact to produce the desired symmetrical compactness.

Instead of K in the denominator, one can use several other forms such as K^p $(p > 1)$, $\log(K)$, $\exp(K)$, etc. Let us assume that $Sym_p(K) = \frac{D_K}{\mathscr{E}_K \times K^p}$, where $p > 1$. For the case where the candidate cluster may have a clear tendency for division,

$$\frac{P_K}{P_{K+1}} = \frac{(K+1)(K+1)^p}{K^p(K+3)} = \frac{(K+1)^{p+1}}{K^{p+1}+3K^p} > 1,$$

since D_K is the same in both K and $K+1$ cluster configurations. This factor is greater than 1 for all $K > 1$; i.e., a division is not suggested. So, incorporating K^p in place of K in the denominator would pose an obstacle in dividing large compact clusters into subparts where a division is indeed suggested.

Now, if the form of *Sym*-index is $Sym_{exp}(K) = \frac{D_K}{\mathscr{E}_K \times \exp(K)}$, then as $\exp(K)$ increases rapidly with increase in the value of K, it will dominate over the other two factors (as these two factors are linearly increasing). With increase in the value of K, the value of Sym_{exp}-index will decrease. Thus, there will be a tendency to restrict K to small values even if the number of clusters is large.

If the normalizing factor is $\log(K)$, i.e., $Sym_{\log}(K) = \frac{D_K}{\mathscr{E}_K \times \log(K)}$, then for the first case

$$\frac{P_{K+1}}{P_K} = \frac{\frac{n\alpha r}{2}(K+3)\log(K)}{\frac{n\alpha r}{2}(K+1)\log(K+1)} = \frac{(K+3)\log(K)}{(K+1)\log(K+1)}.$$

Now, as $\frac{(K+3)}{(K+1)} > \frac{\log(K)}{\log(K+1)}$, the right-hand side term is >1 and the division is always suggested. Now, for the second case,

$$\frac{P_{(K+1)}}{P_K} = \frac{(K+3)\log(K)}{(K-1+2(2d-1)/d)\log(K+1)} = \frac{(K+3)\log(K)}{(K+3-2/d)\log(K+1)}.$$

Now, for $d = 2$,

$$\frac{P_{(K+1)}}{P_K} = \frac{(K+3)\log(K)}{(K+1)\log(K+1)} > 1,$$

thus the division is suggested, which is not desirable for this case.

If there is no such normalizing factor in the definition of *Sym*-index, i.e., $Sym(K) = \frac{D_K}{\mathcal{O}_K}$, then in the first case, where the candidate cluster may have a clear tendency for division,

$$\frac{P_K}{P_{K+1}} = \frac{(K+3)}{(K+1)} > 1,$$

i.e., the division is always suggested. Now, for the second case, where the candidate cluster may not have a clear tendency for division, if the intermediate cluster has a radius of $\frac{r+(d-1)2r}{d}$, then

$$\frac{P_{K+1}}{P_K} = \frac{\sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2n\alpha r}{\sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2\frac{(2d-1)}{d} \times \frac{n\alpha r}{2}} = \frac{K+3}{(K+3-\frac{2}{d})} > 1;$$

i.e., division is suggested, breaking compact clusters into subparts where no division is needed. Thus, this form of *Sym*-index is not suitable.

6.3.2 Mathematical Justification

In this section, the new validity index is mathematically justified by establishing its relationship to the well-known validity measure proposed by Dunn [87] for hard partitions [243]. It is inspired by a proof of optimality of the Xie-Beni index [295].

Uniqueness and Global Optimality of the *K*-Partition The separation index D_1 is a hard K-partition cluster validity criterion. It is known that, if $D_1 > 1$, unique, compact, and separated hard clusters have been found [87]. Here, we have shown that, if the optimal solution D_1 becomes sufficiently large, the validity function *Sym* will be large, which means that a unique K-partition has been found. The proof of this is as follows:

Theorem 6.1 *For any $K = 2, \ldots, n-1$, let Sym be the overall Sym-index value of any hard partition, and D_1 be the separation index of the corresponding partition. Then, we have*

$$Sym \geq \frac{D_1}{n \times K \times 0.5 \times knear \times d_{NN}^{max}},$$

where n is the total number of data points, K is the total number of clusters and knear is the number of nearest neighbors considered while computing d_{ps} as defined in Eq. 5.10. d_{NN}^{max} is the maximum nearest neighbor distance in the data set; that is,

$$d_{NN}^{max} = \max_{i=1,\ldots,N} d_{NN}(\overline{x}_i), \tag{6.13}$$

where $d_{NN}(\overline{x}_i)$ is the nearest neighbor distance of $\overline{x}_i$.

Proof Let the hard K-partition be an optimal partition of the data set $X = \{\overline{x}_j;\ j = 1, 2, \ldots, n\}$ with $\overline{c}_i$ $(i = 1, 2, \ldots, K)$ being the centroids of each class u_i. The total symmetrical variation $\mathscr{E}_K$ of the optimal hard K-partition is defined in Eq. 6.10. Thus,

$$\mathscr{E}_K = \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} d_{ps}(\overline{x}_j, \overline{c}_i) = \sum_{i}^{K} \sum_{\overline{x}_j \in u_i} \frac{\sum_{ii=1}^{knear} d_{ii}}{knear} d_e(\overline{x}_j, \overline{c}_i). \tag{6.14}$$

Assuming that $\overline{x}_j^*$ (the symmetrical point of $\overline{x}_j$ with respect to cluster center $\overline{c}_i$) lies within the data space, it may be noted that $d_1 \leq \frac{d_{NN}^{max}}{2}, d_2 \leq \frac{3d_{NN}^{max}}{2}, \ldots, d_i \leq \frac{(2i-1)d_{NN}^{max}}{2}$, where d_i is the ith nearest neighbor of $\overline{x}_j^*$. Considering the term $\frac{\sum_{ii=1}^{knear} d_{ii}}{knear}$, we can write

$$\frac{\sum_{ii=1}^{knear} d_{ii}}{knear} \leq \frac{d_{NN}^{max}}{2knear} \left(\sum_{ii=1}^{knear} (2 \times ii - 1) \right). \tag{6.15}$$

The right-hand side of the inequality may be written as

$$\frac{d_{NN}^{max}}{2 \times knear} \times \frac{(knear \times (2 \times 1 + (knear - 1)2))}{2} = \frac{knear \times d_{NN}^{max}}{2}. \tag{6.16}$$

So, combining Eqs. 6.14, 6.15, and 6.16, we can write,

$$\mathscr{E}_K \leq \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} 0.5 \times knear \times d_{NN}^{max} \times d_e(\overline{x}_j, \overline{c}_i)$$

$$\leq 0.5 \times knear \times d_{NN}^{max} \sum_{i=}^{K} \sum_{x_j \in u_i} d_e(\overline{x}_j, \overline{c}_i).$$

Suppose that the centroid $\overline{c}_i$ is inside the boundary of cluster i, for $i = 1$ to K. Then $d_e(\overline{x}_j, \overline{c}_i) \leq dia(u_i)$, for $\overline{x}_j \in u_i$, where $dia(u_i) = \max_{\overline{x}_k, \overline{x}_j \in u_i} d_e(\overline{x}_k, \overline{x}_j)$. We thus have

$$\mathscr{E}_K \leq 0.5 \times knear \times d_{NN}^{max} \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} dia(u_i) \leq 0.5 \times knear \times d_{NN}^{max} \sum_{i=1}^{K} n_i dia(u_i)$$

$$\leq 0.5 \times knear \times d_{NN}^{max} \times n \times \max_i dia(u_i).$$

Here, n_i denotes the total number of data points in cluster i. So,

$$\frac{1}{\mathscr{E}_K} \geq \frac{1}{0.5 \times knear \times d_{NN}^{max} \times n \times \max_i dia(u_i)}.$$

We also have that

$$\min_{i,j,i\neq j} dis(u_i, u_j) \leq D_K, \quad \text{where } dis(u_i, u_j) = \min_{\overline{x}_i \in u_i, \overline{x}_j \in u_j} d_e(\overline{x}_i, \overline{x}_j)$$

and D_K is as defined in Eq. 6.11. Thus,

$$Sym(K) = \frac{D_K}{K \times \mathscr{E}_K} \geq \frac{\min_{i,j} dis(u_i, u_j)}{K \times 0.5 \times knear \times d_{NN}^{max} \times n \times \max_i dia(u_i)},$$

i.e.,

$$Sym(K) \geq \frac{\min_{1\leq i\leq K}\{\min_{i+1\leq j\leq K-1}\{\frac{dis(u_i,u_j)}{\max_{1\leq k\leq K} dia(u_k)}\}\}}{K \times 0.5 \times knear \times d_{NN}^{max} \times n}. \tag{6.17}$$

The separation index D_1 (Dunn [87]) is defined as

$$D_1 = \min_{1\leq i\leq K}\left\{\min_{i+1\leq j\leq K-1}\left\{\frac{dis(u_i, u_j)}{\max_{1\leq k\leq K} dia(u_k)}\right\}\right\}. \tag{6.18}$$

So, combining Eqs. 6.17 and 6.18, we get

$$Sym(K) \geq \frac{D_1}{K \times 0.5 \times knear \times d_{NN}^{max} \times n}. \tag{6.19}$$

Since the denominator of the right-hand side is constant for given K, *Sym* increases as D_1 grows without bound. As mentioned earlier, it has been proved by Dunn [87] that if $D_1 > 1$ the hard K-partition is unique. Thus, if the data set has a distinct substructure and the partitioning algorithm has found it, then the corresponding *Sym*-index value will be lower bounded by Eq. 6.19. □

6.3.3 Interaction Between the Different Components of Sym-Index

In order to show how the different components of *Sym*-index compete with each other to determine the proper model order from a data set, the variations of different components of *Sym*-index with the number of clusters are shown pictorially for two artificially generated data sets. The description of these two data sets is given below:

- *Normal_2_4*: Shown in Fig. 6.1(a). It consists of 200 points in 2-d space and has four clusters.

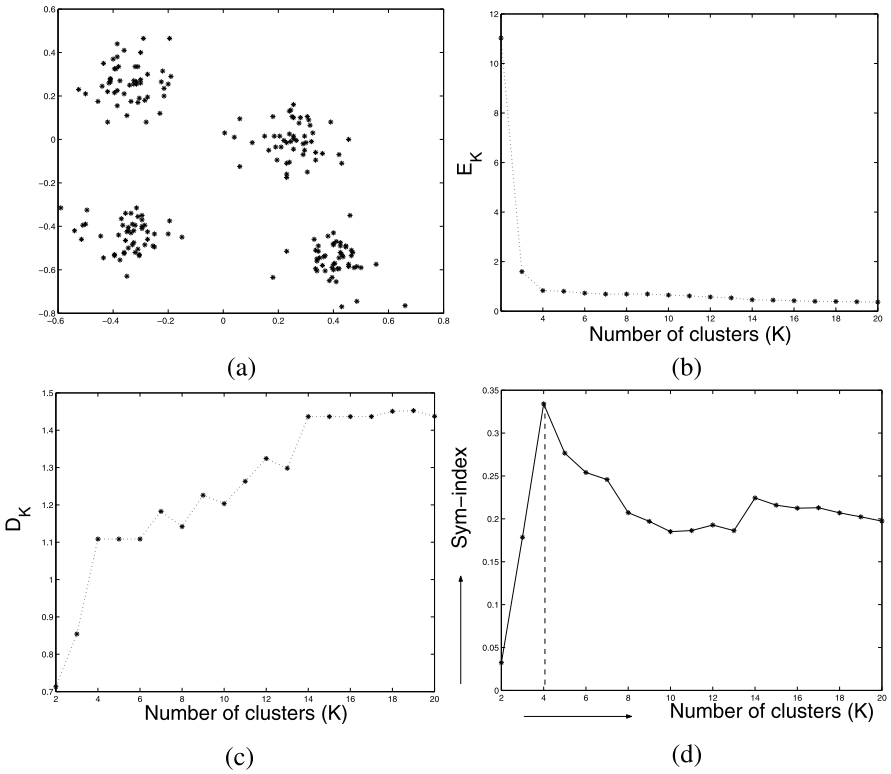


Fig. 6.1 (**a**) Normal_2_4. (**b**) Variation of $\mathscr{E}_K$ value with number of clusters. (**c**) Variation of D_K value with number of clusters. (**d**) Variation of *Sym*-index value with number of clusters

- *Normal_2_10*: Shown in Fig. 6.2(a). It consists of 500 points in 2-d space and has ten clusters.

The variations of the values of different components of the *Sym*-index with the number of clusters for the above two data sets are shown in Figs. 6.1 and 6.2, respectively. Instead of K in the denominator of *Sym*-index, some other possibilities are using K^2, $\exp(K)$, and $\log(K)$. Let us refer to the corresponding indices as Sym_2-index, Sym_{exp}-index, and Sym_{log}-index, respectively. The variations of *Sym*-index, Sym_2-index, Sym_{exp}-index and Sym_{log}-index with the number of clusters for the *Normal_2_10* data set are also shown in Fig. 6.2. It is clearly seen from these figures that the *Sym*-index with K in the denominator performs best compared with the other three.

The PS-index [58], which is also based on a point symmetry-based distance [58], is unable to identify the proper number of clusters from data sets like Normal_2_3 (shown in Fig. 6.3(a)). The variation of the value of the PS-index with the number of clusters is shown in Fig. 6.3(g). It is easy to see that the PS-index obtains its minimum value for $K = 2$. This is because d_{min} attains its maximum value for $K = 2$. Thus, the PS-index prefers the merging of three clusters into two clusters. The vari-

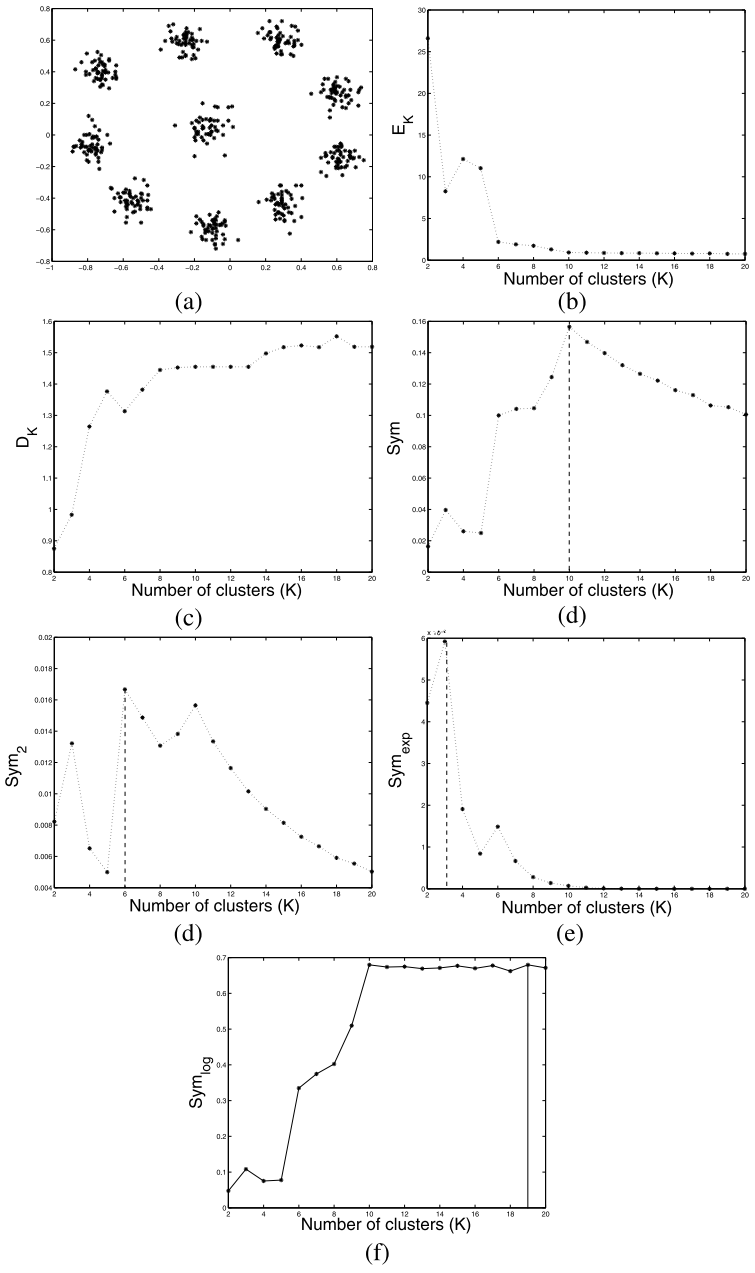


Fig. 6.2 (**a**) Normal_2_10. (**b**) Variation of $\mathscr{E}_K$ value with number of clusters. (**c**) Variation of D_K value with number of clusters. (**d**) Variation of *Sym*-index value with number of clusters. (**e**) Variation of Sym_2-index where in the denominator K is replaced by K^2 with number of clusters. (**f**) Variation of Sym_{exp}-index where in the denominator K is replaced by $\exp(K)$ with number of clusters. (**g**) Variation of Sym_{log}-index where in the denominator K is replaced by $\log(K)$ with number of clusters

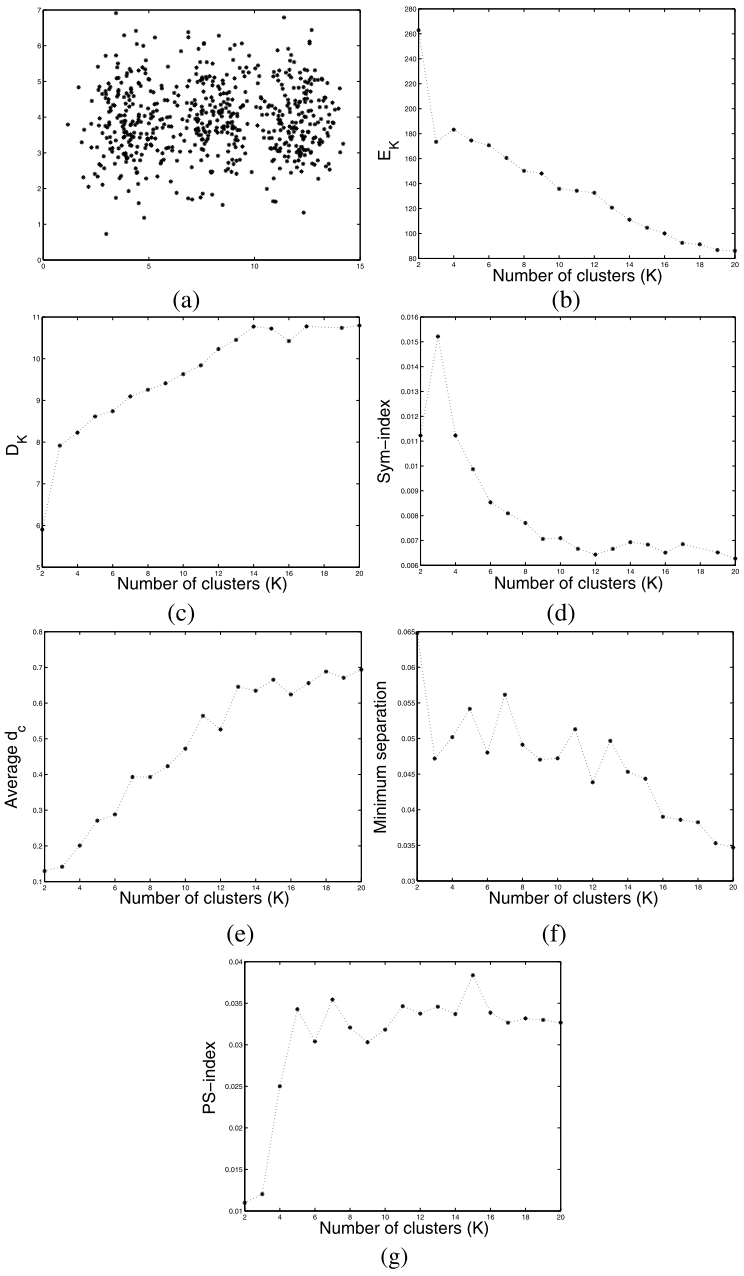


Fig. 6.3 (**a**) Normal_2_3. (**b**) Variation of $\mathscr{E}_K$ value with number of clusters. (**c**) Variation of D_K value with number of clusters. (**d**) Variation of *Sym*-index value with number of clusters. (**e**) Variation of the numerator of PS-index with number of clusters. (**f**) Variation of minimum separation (denominator of PS-index) between any two cluster centers with number of clusters. (**g**) Variation of PS-index with number of clusters

ation of the minimum separation between two cluster centers (d_{min}) with respect to the number of clusters is shown in Fig. 6.3(f). The variation of the *Sym*-index with the number of clusters (shown in Fig. 6.3(d)) reveals that it obtains its optimum value for $K = 3$.

6.4 Experimental Results

6.4.1 Data Sets

The data sets that are used for the experiments are as follows:

1. Group 1: This group consists of two artificial data sets: *Sym_3_2* and *AD_5_2*:
   a. *Sym_3_2*: This data set is described in Sect. 5.8.1.
   b. *AD_5_2*: The clusters present in this data set are internally symmetrical about their centers, and the clusters themselves are symmetrical with respect to some third cluster center. This data set is described in Sect. 5.8.1.
2. Group 2: This group consists of two real life data sets. These are *Iris* and *Cancer* data sets described in Sect. 5.8.1.

6.4.2 Comparative Results

The effectiveness of the *Sym*-index in determining the appropriate clustering algorithm as well as the number of clusters is established for the above-mentioned four data sets. Six clustering algorithms are used as the underlying partitioning techniques. These are described below:

- The newly developed point symmetry-based genetic clustering technique (GAPS) [27] described in Chap. 5.
- The GAK-means algorithm [188].
- The average linkage clustering algorithm [96] (source code obtained from http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/5/659).
- Self-organizing map (SOM) [163] (source obtained from http://www.cs.tau.ac.il/~rshamir/expander).
- EM algorithm assuming spherical-shaped clusters (EM-spherical).
- EM algorithm assuming ellipsoidal-shaped clusters (EM-elliptical) [44] (matlab source codes obtained from http://www.mathworks.com/matlabcentral/fileexchange/).

The number of clusters, K, was varied from 2 to $\sqrt{n}$, where n is the number of data points, and the variation of the *Sym*-index was noted. Its maximum value across different algorithms and different values of K indicates the appropriate algorithm and the appropriate $K = K^*$. Comparison is also made with four existing cluster validity indices, namely a point symmetry-based PS-index [58], I-index [189], a recently

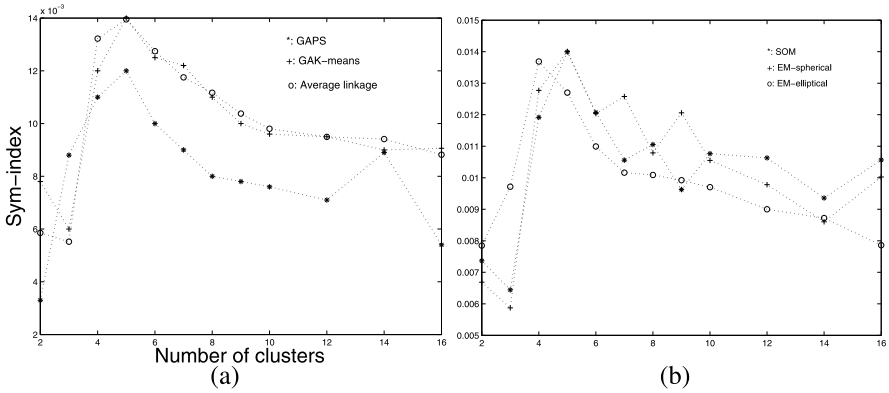


Fig. 6.4 Variation of *Sym*-index with number of clusters for *AD_5_2* using (**a**) GAPS, GAK-means, average linkage and (**b**) SOM, EM-spherical, EM-elliptical

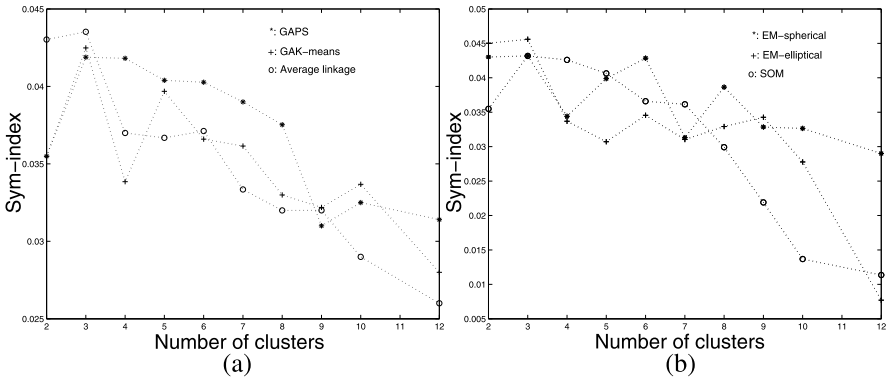


Fig. 6.5 Variation of *Sym*-index with number of clusters for *Iris* using (**a**) GAPS, GAK-means, average linkage and (**b**) SOM, EM-spherical, EM-elliptical

proposed CS-index [59], and the well-known Xie-Beni (XB)-index [295], in terms of their ability to provide the appropriate number of clusters and the partitioning. The parameters of the genetic clustering algorithms (GAPS and GAK-means) were as follows: population size is equal to 100 and maximum generations is equal to 50. For GAPS, the crossover and mutation probabilities are chosen adaptively as in [262]. For GAK-means, the crossover and mutation probabilities are chosen as 0.8 and 0.001 (as specified in [188]), respectively. As already mentioned, the codes for average linkage, EM-elliptical, EM-spherical, and SOM were obtained from different sources and were executed using default parameters. The results reported in the tables are the average values obtained over ten runs of the algorithms.

Figures 6.4 and 6.5 show the variations of *Sym*-index with the number of clusters for the *AD_5_2* and *Iris* data sets, respectively, for the purpose of illustration. Similar figures were obtained using the other data sets and other indices as well.

Table 6.1 Optimal values of the five indices for *Sym_3_2*, *AD_5_2*, *Iris*, and *Cancer* using the six algorithms (GAPS: A_1, GAK-means: A_2, average linkage: A_3, EM-spherical: A_4, EM-elliptical: A_5, SOM: A_6) where K is varied from 2 to $\sqrt{n}$. The number within brackets is the corresponding cluster number (best results for each data set are marked in bold)

Data set	Method	*Sym*-index value (K^*)	PS-index value (K^*)	*I*-index value (K^*)	CS-index value (K^*)	XB-index value (K^*)
Sym_3_2	A_1	0.057140(3)	0.015113(3)	7.200510(8)	0.883900(6)	0.250966(4)
	A_2	0.051(9)	0.022023(8)	**7.243650(6)**	0.821136(4)	0.160450(4)
	A_3	0.018659(4)	0.051505(4)	3.266627(5)	1.132787(3)	0.227456(2)
	A_4	0.047218(16)	0.020472(10)	5.190689(5)	**0.757086(4)**	**0.157517(4)**
	A_5	**0.062554(3)**	**0.009304(3)**	5.294871(10)	0.941965(6)	0.275887(2)
	A_6	0.044741(10)	0.020531(9)	6.894(8)	0.821136(4)	0.16045(4)
AD_5_2	A_1	0.012243(5)	0.019606(4)	1315.883622(6)	0.895751(4)	0.144343(4)
	A_2	0.013994(5)	0.01936(4)	1277.890304(5)	0.826734(5)	0.138853(4)
	A_3	0.013951(5)	0.018921(4)	1240.832405(4)	0.824517(4)	0.155164(4)
	A_4	**0.014009(5)**	**0.015858(5)**	**1544.990475(9)**	**0.776564(4)**	0.163565(4)
	A_5	0.013688(4)	0.019617(4)	1271.288857(5)	0.843807(4)	0.169602(4)
	A_6	0.013924(5)	0.019787(4)	1246.679(5)	0.821682(5)	**0.137765(5)**
Iris	A_1	0.041885(3)	0.027815(2)	**693.469990(3)**	0.715700(2)	0.065800(2)
	A_2	0.042486(3)	0.024805(2)	619.605970(3)	0.715700(2)	0.065800(2)
	A_3	0.043527(3)	0.024805(2)	653.958238(3)	**0.626869(2)**	**0.065409(2)**
	A_4	0.043180(3)	0.024992(2)	633.815597(3)	**0.626869(2)**	**0.065409(2)**
	A_5	**0.045598(3)**	**0.021853(2)**	447.609573(3)	**0.626869(2)**	**0.065409(2)**
	A_6	0.043180(3)	0.026918(2)	633.815597(3)	0.715700(2)	0.065800(2)
Cancer	A_1	**0.000522(2)**	0.125165(2)	27662.411940(2)	1.098512(2)	0.148525(2)
	A_2	0.0005(2)	0.131208(5)	**28055.566482(3)**	1.097088(2)	**0.148209(2)**
	A_3	0.000485(2)	**0.093310(3)**	27058.824291(3)	1.234792(2)	0.226168(3)
	A_4	0.000475(2)	0.099740(3)	35000.722399(6)	**1.027397(2)**	0.206329(2)
	A_5	0.000477(2)	0.118425(2)	17917.064259(2)	1.05373(2)	0.202678(2)
	A_6	**0.000522(2)**	0.103135(2)	27662.411940(2)	1.098512(2)	0.148525(2)

Based on the results in the figures, Table 6.1 presents the optimum values of the five validity indices, *Sym*-index, PS-index, *I*-index, CS-index, and XB-index, and the corresponding number of clusters obtained after application of the six clustering algorithms for the different data sets. Let S denote the set of clustering algorithms and $CV(A, l)$ denote the value of some cluster validity index CV for $K = l$ provided by a clustering algorithm $A \in S$. Then, the most appropriate algorithm and the corresponding $K = K^*$, denoted by the tuple (A^*, K^*), is given by $(A^*, K^*) = \text{argopt}_{\forall A \in S \text{ and } l=2,3,\ldots,\sqrt{n}}\{CV(A, l)\}$. Table 6.2 presents the overall (A^*, K^*) values obtained using the different indices for all the data sets.

Table 6.2 Most appropriate algorithms (A^*) and appropriate cluster number (K^*) identified by the five validity indices for the different data sets. Here the algorithms are represented as follows: GAPS: A_1, GAK-means: A_2, average linkage: A_3, EM-spherical: A_4, EM-elliptical: A_5, SOM: A_6, and AC means actual number of clusters

Data set	AC	Appropriate algo (known)	(K^*, A^*)				
			Sym	PS	*I*	CS	XB
Sym_3_2	3	A_1, A_5	$(3, A_5)$	$(3, A_5)$	$(6, A_2)$	$(4, A_4)$	$(4, A_4)$
AD_5_2	5	A_1, A_2, A_4, A_6	$(5, A_4)$	$(5, A_4)$	$(9, A_4)$	$(4, A_4)$	$(5, A_6)$
Iris	3	A_5	$(3, A_5)$	$(2, A_5)$	$(3, A_1)$	$(2, \{A_3, A_4, A_5\})$	$(2, \{A_3, A_4, A_5\})$
Cancer	2	A_1, A_2, A_6	$(2, \{A_1, A_6\})$	$(3, A_3)$	$(3, A_2)$	$(2, A_4)$	$(2, \{A_1, A_2\})$

6.4.3 Analysis of Results

This section analyzes the experimental results mentioned in the previous section.

1. *Sym_3_2*: EM-elliptical along with both symmetry distance-based indices, *Sym*-index and PS-index, is able to detect the proper cluster number (see Table 6.2). The corresponding partitioning is shown in Fig. 6.6(b), where the elliptical-shaped cluster is found to be extended and includes some points from the ring cluster. GAPS with both *Sym*-index and PS-index is also able to detect the proper cluster number (see Table 6.1, partitioning is shown in Fig. 6.6(a)). However, the optimal value of *Sym*-index corresponds to the partitioning obtained with EM-elliptical. In order to investigate the reason for this, we have noted the values of the components of *Sym*-index. For the partitioning obtained by EM-elliptical, the ellipsoidal cluster has $E_i = 2.0515$ where as that for the ring and spherical clusters are $E_i = 1.8249$ and $E_i = 4.2294$, respectively, resulting in $\mathscr{E}_3 = 8.1059$ and $\frac{1}{\mathscr{E}_3} = 0.12337$. The value of D_3 is 1.521174. For the partitioning obtained by GAPS, the E_i for ellipsoidal, ring, and spherical clusters are 1.8123, 2.9249, and 4.2294, respectively, resulting in $\mathscr{E}_3 = 8.9666$ and $\frac{1}{\mathscr{E}_3} = 0.11152$. In this case, $D_3 = 1.537$. Thus, it is observed that the E_i value for the ellipsoidal cluster (2.0515) is much smaller for EM-elliptical (due to the inclusion of some points on the ring) as compared with that for GAPS (2.9249). This results in a larger *Sym*-index value for EM-elliptical. *I*-index and XB-index are not able to find the proper clustering and the proper cluster number with any of the algorithms. CS-index is able to indicate the proper cluster number with average linkage (refer to Table 6.1), but its optimum value corresponds to $K^* = 4$ with EM-spherical (refer to Table 6.2).
2. *AD_5_2*: As clusters present in this data set are spherical in nature, so *Sym*-index is able to find the proper cluster number (=5) for all the clustering algorithms except EM-elliptical. Figures 6.7(a), 6.7(b), 6.7(c), 6.8(a), and 6.8(b) show the corresponding partitionings. It is evident from the figures that, although the algorithms indicate five clusters as optimal, the resultant partitionings, especially

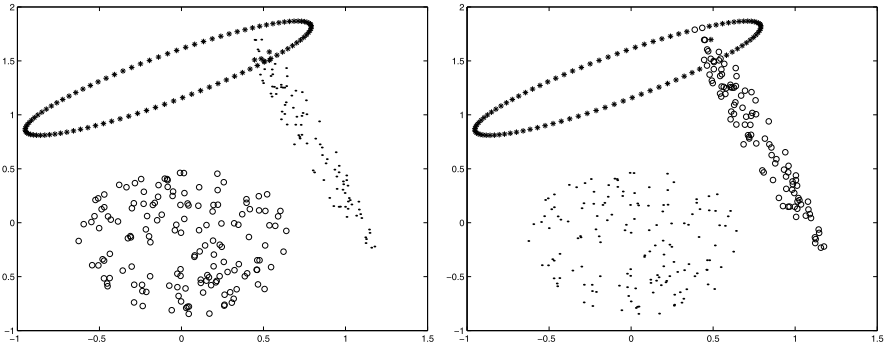


Fig. 6.6 Clustered *Sym_3_2* after application of (**a**) GAPS for $K = 3$, (**b**) EM-elliptical for the best value of *Sym*-index and PS-index providing $K^* = 3$

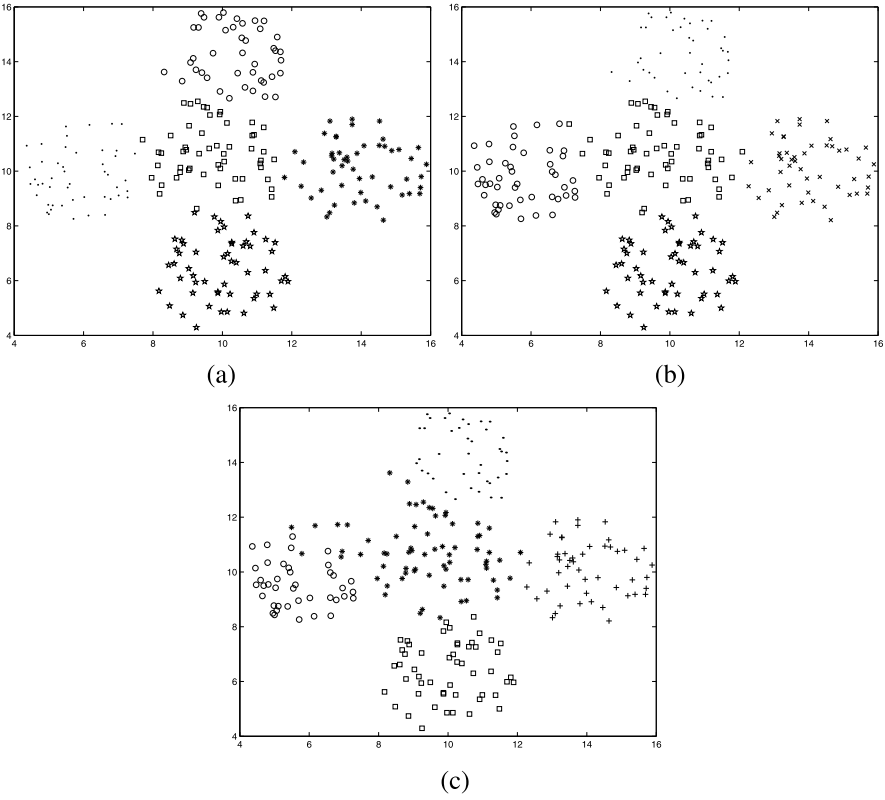


Fig. 6.7 Clustered *AD_5_2* after application of (**a**) GAK-means for $K = 5$, (**b**) EM-spherical for the best value of *Sym*-index and PS-index providing $K^* = 5$, and (**c**) GAPS for $K = 5$

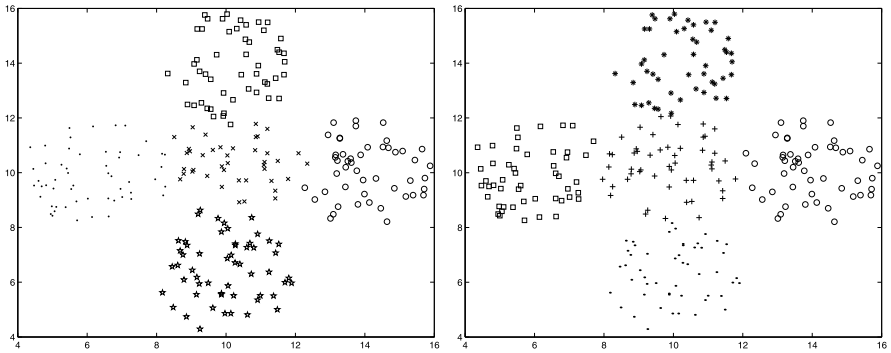


Fig. 6.8 Clustered *AD_5_2* after application of (**a**) average linkage for $K = 5$, and (**b**) SOM for the best value of XB-index providing $K^* = 5$

the cluster appearing in the middle, are different. The *Sym*-index value for EM-spherical is marginally superior to that of GAK-means (see Table 6.1). GAPS performs poorly for this data since it tends to spread out the middle cluster. Optimum values of PS-index and XB-index also indicate the correct number of clusters when used with EM-spherical and SOM, respectively (see Table 6.2). Although it can be seen that I-index finds the proper cluster number with GAK-means, EM-elliptical, and SOM, (see Table 6.1), its optimum value over all the algorithms is obtained with $K = 9$ for EM-elliptical (see Table 6.2). CS-index is able to find the proper number of clusters with GAK-means and SOM (refer to Table 6.1), but its optimum value over all the algorithms is obtained with $K = 4$ for EM-spherical (refer to Table 6.2).

3. *Iris*: *Sym*-index and I-index are able to indicate the proper cluster number with all the algorithms (refer to Table 6.1). *Sym*-index attains its optimum value for the partitioning obtained by EM-elliptical for $K = 3$ (refer to Table 6.2). I-index obtains its optimum value with GAPS for $K = 3$ (refer to Table 6.2). PS-index, CS-index, and XB-index are unable to indicate three clusters with any of the algorithms (refer to Tables 6.1 and 6.2). However, they mostly indicate two clusters, which is also often obtained by many other methods for *Iris*. Since visual display of higher-dimensional data is difficult, here the *Minkowski score* [146] (defined in Eq. 5.24) is reported. It is calculated after the application of each algorithm taking $K = 3$. This is a measure of the quality of a solution given the true clustering. For MS, the optimum score is 0, with lower scores being "better". Each of the above-mentioned six algorithms was executed ten times, the *Minkowski scores* were computed, and ANOVA [5] statistical analysis was performed. The results for the *Iris* data set are reported in Table 6.3. From Table 6.3 it can be seen that EM-elliptical finds the best clustering among the six algorithms. This table also reveals that, although the mean MS value of GAPS is slightly better than those of GAK-means, EM-spherical, and SOM, the differences of the values over ten runs are not statistically significant i.e., GAPS, GAK-means, EM-spherical, and SOM

Table 6.3 Estimated marginal means and pairwise comparisons of *Minkowski scores* (MS) for different algorithms obtained by ANOVA testing for *Iris* data (GAPS: A_1, GAK-means: A_2, average linkage: A_3, EM-spherical: A_4, EM-elliptical: A_5, Self organizing map: A_6)

Algo. name (I)	Mean MS	Comp. algo. (J)	Mean difference ($I-J$)	Significance value
A_1	0.59 ± 0.003	A_2	$-1.0 \times 10^{-2} \pm 0.004$	0.54
		A_3	$-1.1 \times 10^{-1} \pm 0.004$	<0.01
		A_4	$-1.0 \times 10^{-2} \pm 0.004$	0.59
		A_5	0.23 ± 0.004	<0.01
		A_6	$-1.0 \times 10^{-2} \pm 0.004$	0.59
A_2	0.60 ± 0.003	A_1	$1.0 \times 10^{-2} \pm 0.004$	0.54
		A_3	-0.10 ± 0.004	<0.01
		A_4	0.00 ± 0.004	1
		A_5	0.24 ± 0.004	<0.01
		A_6	0.00 ± 0.004	1
A_3	0.70 ± 0.003	A_1	$1.1 \times 10^{-1} \pm 0.004$	<0.01
		A_2	0.10 ± 0.004	<0.01
		A_4	0.10 ± 0.004	<0.01
		A_5	0.34 ± 0.004	<0.01
		A_6	0.10 ± 0.004	<0.01
A_4	0.60 ± 0.003	A_1	$1.0 \times 10^{-2} \pm 0.004$	0.59
		A_2	0.00 ± 0.004	1
		A_3	-0.10 ± 0.004	<0.01
		A_5	0.24 ± 0.004	<0.01
		A_6	0.00 ± 0.004	1
A_5	0.36 ± 0.003	A_1	-0.23 ± 0.004	<0.01
		A_2	-0.24 ± 0.004	<0.01
		A_3	-0.34 ± 0.004	<0.01
		A_4	-0.24 ± 0.004	<0.01
		A_6	-0.24 ± 0.004	<0.01
A_6	0.60 ± 0.003	A_1	$1.0 \times 10^{-2} \pm 0.004$	0.59
		A_2	0.00 ± 0.004	1
		A_3	$-0.10 \times 10^{-2} \pm 0.004$	<0.01
		A_4	0.00 ± 0.004	1
		A_5	0.24 ± 0.004	<0.01

clustering algorithms perform similarly for this data set. From Table 6.2 we find that only the *Sym*-index indicates that EM-elliptical is the most appropriate algorithm, and that three clusters are present in the data. Therefore, combining the results of Tables 6.2 and 6.3 reveals that the partitioning indicated by *Sym*-index is the best.

4. *Cancer*: For this data set, like the previous one the *Minkowski score* (MS) [146] of the partitionings obtained after application of all six algorithms with $K = 2$ were calculated, being 0.36, 0.36, 0.45, 0.43, 0.43, and 0.37 for GAPS, GAK-means, average linkage, EM-spherical, EM-elliptical, and SOM, respectively. From ANOVA analysis it is observed that the GAPS, GAK-means, and SOM algorithms perform similarly for this data (the difference between the mean MS for any pair of algorithms is not significant), indicating that the two clusters present in the *Cancer* data set are convex as well as highly symmetrical. They are able to find the best clustering among all the algorithms. As can be noted from Table 6.2, the *Sym*-index indicates GAPS and SOM as the most appropriate choices. The XB-index is also successful in correctly identifying the GAPS and GAK-means algorithms with $K^* = 2$ as the most appropriate clustering algorithms. Even though the I-index is able to detect the proper cluster number with GAPS, EM-elliptical, and SOM (see from Table 6.1), it attains its optimum value for GAK-means with $K = 3$ (refer to Table 6.2). The CS-index attains its optimum value corresponding to the partitioning obtained by EM-spherical for $K = 2$ (see Table 6.2). The PS-index is also able to find the proper cluster number with GAPS, EM-elliptical, and SOM (refer to Table 6.1), but its optimum value indicates average linkage with $K = 3$ as the proper choice (refer to Table 6.2).

Interestingly, it is observed that, for all the data sets, *Sym*-index is able to detect the proper number of clusters as well as a few of the suitable clustering algorithms. This is not the case for any of the other indices, which sometimes fail to identify either the proper algorithm or the appropriate number of clusters, or both; for example, for *Sym_3_2* which has symmetrical clusters, the *Sym*-index (as well as the PS-index) correctly identifies three clusters and GAPS/EM-elliptical as the appropriate clustering algorithms. However, the other three indices are able to identify neither the correct number of clusters nor the appropriate algorithm. Overall, the *Sym*-index is able to detect the proper number of clusters and the appropriate clustering algorithms for all the data sets.

6.5 Incorporating d_{ps} in Some Existing Cluster Validity Indices

As mentioned earlier, in order to validate the obtained partitionings and to determine the appropriate number of clusters from a data set, several cluster validity indices have been proposed.

Most of the existing cluster validity indices use the Euclidean distance in their computation. They are mostly therefore able to characterize only convex clusters. It has been shown in [27] that the symmetry-based distance is effective not only for convex clusters, but also in cases where the clusters are non-convex but satisfy the property of point symmetry. In this chapter we conjecture that incorporation of the symmetry measure in the above-mentioned validity indices will impart the property of characterizing non-convex, symmetric clusters to them. Thus, here we

incorporate the newly developed point symmetry-based distance, rather than the Euclidean distance, to develop symmetry-based versions of the Davies-Bouldin index (DB-index) [73], Dunn's index [87], generalized Dunn's index [38], PS-index [58], Xie-Beni index (XB index) [295], FS-index [107], K-index [172], and SV-index [154]. GAPS clustering is used as the underlying clustering algorithm. The number of clusters is varied from K_{min} to K_{max}. As a result, a total of $(K_{max} - K_{min} + 1)$ partitions will be generated, $U^*_{K_{min}}, U^*_{K_{min}+1}, \ldots, U^*_{K_{max}}$, with the corresponding validity index values computed as $V_{K_{min}}, V_{K_{min}+1}, \ldots, V_{K_{max}}$ with V representing one of the above-mentioned validity indices. Let $K^* = \text{argopt}_{i=K_{min},\ldots,K_{max}}[V_i]$. Therefore, according to the index V, K^* is the correct number of clusters present in the data. The corresponding U^*_K is the partitioning obtained by GAPS clustering with the number of clusters set to K^*. The tuple $\langle U^*_{K^*}, K^* \rangle$ is presented as the solution to the clustering problem.

The effectiveness of the point symmetry-based cluster validity indices, namely the *Sym-DB* index, *Sym-Dunn* index, *Sym-GDunn* index, *Sym-PS* index, *Sym* index, *Sym-XB* index, *Sym-FS* index, *Sym-K* index, and *Sym-SV* index, in identifying the proper number of clusters is demonstrated for two artificially generated and three real-life data sets of varying complexities. Results also reveal that the *Sym*-index performs the best compared with all the other eight indices. For the purpose of comparison, the cluster number indicated by the original versions of the existing eight cluster validity indices are also provided for all the artificial and real-life data sets. Experimental results show that incorporation of the point symmetry distance improves the capabilities of these indices to detect any type of cluster irrespective of their shapes, sizes, and convexity, as long as they possess the property of point symmetry.

6.6 Point Symmetry-Based Cluster Validity Indices

In this section, the eight point symmetry distance-based cluster validity indices are defined. Note that the definitions of these indices are inspired by those of eight well-known existing cluster validity indices.

6.6.1 Symmetry-Based Davies-Bouldin Index (Sym-DB Index)

This index is developed along the lines of the popular Davies-Bouldin (DB) index [73]. This is a function of the ratio of the sum of *within-cluster symmetry* to *between-cluster separation*. The scatter within the ith cluster, S_i, is computed as

$$S_i = \frac{\sum_{\overline{x} \in C_i} d^*_{ps}(\overline{x}, \overline{z}_i)}{|C_i|},$$

where $\overline{z}_i$ represents the center of cluster i and $d^*_{ps}(\overline{x}, \overline{z}_i)$ is computed using Eq. 5.10 with some constraint. Note that here the *knear* nearest neighbors of the reflected

point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_i$ and $\overline{x}$ should belong to the ith cluster; i.e., the first *knear* nearest neighbors of $\overline{x}^* = 2 \times \overline{z}_i - \overline{x}$ are searched among the points which are already in cluster i. The distance between cluster C_i and C_j, denoted by d_{ij}, is defined as $d_{ij} = d_e(\overline{z}_i, \overline{z}_j)$, where d_e stands for Euclidean distance computation. Then symmetry-based DB index, *Sym-DB* index, is defined as

$$Sym\text{-}DB = \frac{\sum_{i=1}^{K} R_i}{K},$$

where $R_i = \max_{j, j \neq i} \{\frac{S_i + S_j}{d_{ij}}\}$. Here K is the number of clusters. The objective is to minimize the *Sym-DB* index for achieving the proper clustering.

6.6.2 Symmetry-Based Dunn's Index (Sym-Dunn Index)

This index is developed along the lines of the popular Dunn's index [87]. Let S and T be two nonempty subsets of R^N. Then the radius $\triangle$ of S is defined as

$$\triangle(S) = \max_{x \in S} \left\{ d_{ps}^*(\overline{x}, \overline{z}) \right\},$$

where $\overline{z}$ represents the center of set S and $d_{ps}^*(\overline{x}, \overline{z})$ is computed using Eq. 5.10. Note that here also the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}$ and $\overline{x}$ should belong to the set S. The set distance δ between S and T is defined as

$$\delta(S, T) = \min_{\overline{x} \in S, \overline{y} \in T} \left\{ d_e(\overline{x}, \overline{y}) \right\}.$$

Here, $d_e(\overline{x}, \overline{y})$ indicates the Euclidean distance between points $\overline{x}$ and $\overline{y}$. For any partition, the *Sym-Dunn* index is defined as follows:

$$Sym\text{-}Dunn = \min_{1 \leq i \leq K} \left\{ \min_{1 \leq j \leq K, j \neq i} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \leq k \leq K} \triangle(C_k)} \right\} \right\}.$$

Larger values of the *Sym-Dunn* index correspond to good clustering, and the number of clusters that maximizes this index value is taken as the optimal number of clusters.

6.6.3 Symmetry-Based Generalized Dunn's Index (Sym-GDunn Index)

This index is developed along the lines of the generalized Dunn's index [38]. The generalized Dunn's index was developed after demonstrating the sensitivity of the original Dunn's index [87] to changes in cluster structure, since not all of the data

points were involved in the computation of the index. The symmetry-based GDunn cluster validity index, *Sym-GDunn* index, is defined as

$$Sym\text{-}GDunn = \min_{1\le s\le K}\left\{\min_{1\le t\le K, t\ne s}\left\{\frac{\delta(C_s, C_t)}{\max_{1\le k\le K}\triangle(C_k)}\right\}\right\}.$$

The two measures δ and $\triangle$ are defined as follows:

$$\triangle(S) = 2\times\frac{\sum_{x\in S} d^*_{ps}(\overline{x}, \overline{z}_S)}{|S|}$$

and

$$\delta(S,T) = \frac{1}{|S||T|}\sum_{\overline{x}\in S, \overline{y}\in T} d_e(\overline{x}, \overline{y}).$$

Here, $\overline{z}_S$ and $\overline{z}_T$ are the centers of the sets S and T, respectively. Here, $d^*_{ps}(\overline{x}, \overline{z}_S)$ is computed by Eq. 5.10 with some constraint. Note that the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_S$, and $\overline{x}$ should belong to the set S. Larger values of *Sym-GDunn* correspond to good clusters, and the number of clusters that maximizes *Sym-GDunn* is taken as the optimal number of clusters.

6.6.4 New Symmetry Distance-Based PS-Index (Sym-PS Index)

This index is developed along the lines of the PS-index [58]. The cluster validity index, *Sym-PS* index, is defined as

$$Sym\text{-}PS(K) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{n_i}\sum_{\overline{x}\in S_i}\frac{d^*_{ps}(\overline{x}, \overline{z}_i)}{\min_{m,n=1,\ldots,K,\ m\ne n} d_e(\overline{z}_m, \overline{z}_n)} \tag{6.20}$$

$$= \frac{1}{K}\sum_{i=1}^{K}\frac{1}{n_i}\sum_{\overline{x}\in S_i}\frac{d^*_{ps}(\overline{x}, \overline{z}_i)}{d_{min}} \tag{6.21}$$

where S_i is the set whose elements are the data points assigned to the ith cluster, n_i is the number of elements in S_i, or, $n_i = |S_i|$, d_{min} is the minimum Euclidean distance between any two cluster centers, and $d^*_{ps}(\overline{x}, \overline{z}_i)$ is computed by Eq. 5.10 with some constraint. Note that the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_i$ and $\overline{x}$ should belong to the ith cluster. The smallest $Sym\text{-}PS(K^*)$ indicates a valid optimal partition with the optimal cluster number K^*.

6.6.5 Symmetry-Based Xie-Beni Index (Sym-XB Index)

This index is developed along the lines of the popular XB-index [295]. It is defined as follows:

$$Sym\text{-}XB = \frac{\sum_{i=1}^{K}(\sum_{\overline{x}\in C_i} d_{ps}^{*2}(\overline{x},\overline{z}_i))}{n(\min_{i,k=1,\ldots K,i\neq k} d_e^2(\overline{z}_i,\overline{z}_k))}.$$

$d_{ps}^*(\overline{x},\overline{z}_i)$ is computed by Eq. 5.10. Note that here also the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_i$ and $\overline{x}$ should belong to the ith cluster. The most desirable partition (or an optimal value of K) is obtained by minimizing the *Sym-XB* index over $K = 2, 3, \ldots, K_{max}$.

6.6.6 Symmetry-Based FS-Index (Sym-FS Index)

This index is developed along the lines of the FS-index proposed in [107]. It is defined as follows:

$$Sym\text{-}FS = \sum_{i=1}^{K}\sum_{\overline{x}\in C_i} d_{ps}^{*2}(\overline{x},\overline{z}_i) - \sum_{i=1}^{K}\sum_{\overline{x}\in C_i} d_e^2(\overline{z}_i,\overline{z}),$$

where $\overline{z}$ is the center of the entire data set. $d_{ps}^*(\overline{x},\overline{z}_i)$ is computed by Eq. 5.10. Note that here also the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_i$ and $\overline{x}$ should belong to the ith cluster. The optimal partition is obtained by minimizing the *Sym-FS* index with respect to $K = 2, 3, \ldots, K_{max}$.

6.6.7 Symmetry-Based K-Index (Sym-K Index)

Kwon extended the index given by Xie and Beni [295] to eliminate its tendency to decrease monotonically when the number of clusters approaches the number of data points [172]. To achieve this, a punishing function was introduced into the numerator of Xie and Beni's original validity index. The resulting index is named the K-index. Here we have developed a new validity index along the lines of the K-index but using point symmetry-based distance. This is named the *Sym-K* index, being defined as follows:

$$Sym\text{-}K = \frac{\sum_{i=1}^{K}\sum_{\overline{x}\in C_i} d_{ps}^{*2}(\overline{x},\overline{z}_i) + \frac{1}{K}\sum_{i=1}^{K} d_e^2(\overline{z}_i,\overline{z})}{\min_{i\neq k} d_e^2(\overline{z}_i,\overline{z}_k)}.$$

In this equation again $\overline{z}$ represents the center of the entire data set. $d_{ps}^*(\overline{x},\overline{z}_i)$ is computed by Eq. 5.10. Note that here also the *knear* nearest neighbors of the reflected point $\overline{x}^*$ of the point $\overline{x}$ with respect to $\overline{z}_i$ and $\overline{x}$ should belong to the ith cluster. A minimum value of *Sym-K* index corresponds to the optimal cluster number.

6.6.8 Symmetry-Based SV-Index (Sym-SV Index)

Kim et al. attempted to determine the optimal cluster number by measuring the status of the given partition with both an under partition index and an over partition index [154]. Here, the newly developed *Sym-SV* index is defined along the lines of the SV-index proposed by Kim et al. [154].

$$\begin{aligned} Sym\text{-}SV &= v_{under}(Z:X) + v_{over}(Z) \\ &= \frac{1}{K}\sum_{i=1}^{K}\sum_{\overline{x}\in C_i}\frac{d_{ps}^*(\overline{x},\overline{z}_i)}{n_i} + \frac{K}{\min_{i\neq j} d_e(\overline{z}_i,\overline{z}_j)}. \end{aligned}$$

A minimum value of *Sym-SV* index indicates the optimal number of clusters.

6.7 Experimental Results

This section provides a description of the data sets and the partitionings indicated by different cluster validity indices after application of the GAPS clustering algorithm. Experiments were carried out with two artificial and three real-life data sets. The artificial data sets are *Sym_3_2* and *Bensaid_3_2* (described in Sect. 5.8.1).

The three real-life data sets are *Iris*, *Cancer*, and *LungCancer* (described in Sect. 5.8.1).

6.7.1 Discussion of Results

The parameters of GAPS clustering are set as detailed in Sect. 5.8.2. The results reported in the table are the average values obtained over ten runs of the algorithm. Here, K_{min} is set equal to 2 and K_{max} is set equal to $\sqrt{n}$, where n is the total number of data points present in the data set. Thus, for each data set, a total of $(\sqrt{n}-2+1) = (\sqrt{n}-1)$ partitions will be obtained with particular validity index (V) values $V_2, V_3, \ldots, V_{\sqrt{n}}$. Then, according to the index V, the optimal number of clusters will be denoted by $K^* = \text{argopt}_{i=2,\ldots,\sqrt{n}}[V_i]$.

Table 6.4 presents the optimum number of clusters identified by the nine point symmetry distance-based cluster validity indices, namely the *Sym-DB*, *Sym-Dunn*, *Sym-GDunn*, *Sym-PS*, *Sym*, *Sym-XB*, *Sym-FS*, *Sym-K* and *Sym-SV* indices, for all the data sets used here for the experiment. Figures 6.9(a) and 6.10(a) show, respectively, the partitionings obtained after application of GAPS clustering on the two artificial data sets, respectively, for the actual number of clusters present in the data sets. It can be seen that, for *Sym_3_2*, all the indices except *Sym-DB*, *Sym-FS*, and *Sym-SV* are able to find the proper partitioning and the proper number of partitions (the corresponding partitioning is shown in Fig. 6.9(a)). Optimum values of *Sym-DB*

Table 6.4 Optimal number of clusters identified by the symmetry version and the original version of eight cluster validity indices and *Sym*-index for five data sets, segmented using the GAPS clustering algorithm where K is varied from 2 to $\sqrt{n}$. Here, AC denotes the actual number of clusters present in the particular data set. Success rates (defined in Sect. 6.7.1) of two different versions of the eight cluster validity indices along with *Sym*-index in detecting the proper partitioning and the proper number of partitions are also provided

Validity index	Version	*Sym_3_2*	*Bensaid_3_2*	*Iris*	*Cancer*	*LungCancer*	Success rate
DB	Sym	6	3	2	2	3	0.6(3/5)
	Org	6	3	2	2	3	0.6(3/5)
Dunn	Sym	3	3	8	8	2	0.4(2/5)
	Org	3	6	7	7	3	0.4(2/5)
GDunn	Sym	3	3	2	2	4	0.6(3/5)
	Org	2	2	2	5	3	0.2(1/5)
PS	Sym	3	3	2	2	3	0.8(4/5)
	Org	3	3	2	2	3	0.8(4/5)
XB	Sym	3	3	2	2	3	0.8(4/5)
	Org	4	3	2	2	3	0.6(3/5)
FS	Sym	6	7	8	10	4	0(0/5)
	Org	10	7	8	10	4	0(0/5)
K	Sym	3	3	2	2	3	0.8(4/5)
	Org	4	3	2	2	3	0.6(3/5)
SV	Sym	2	6	2	2	3	0.4(2/5)
	Org	2	3	2	2	3	0.6(3/5)
Sym		3	3	3	2	3	1.00(5/5)

and *Sym-FS* indices indicate $K = 6$ as the proper number of clusters, whereas that of *Sym-SV* indicates $K = 2$ as the proper number of clusters. The corresponding partitionings are shown in Figs. 6.9(b) and 6.9(c), respectively. For the *Bensaid_3_2* data set, all the indices except *Sym-FS* and *Sym-SV* are able to detect the proper partitioning and the appropriate number of partitions after application of GAPS clustering (partitioning shown in Fig. 6.10(a)). *Sym-SV* wrongly indicates $K^* = 6$. The corresponding partitioning is shown in Fig. 6.10(b).

For the three higher-dimensional real-life data sets, *Iris*, *Cancer*, and *LungCancer*, the *Minkowski scores* (defined in Eq. 5.24) are calculated after application of the GAPS clustering algorithm. For, *Iris* data set, the MS value corresponding to the partitioning obtained by GAPS clustering for $K = 3$ is 0.59 ± 0.00. As can be seen from Table 6.4, only the *Sym*-index is able to detect the proper number of partitions for this data set. Optimum values of the *Sym-DB*, *Sym-GDunn*, *Sym-PS*, *Sym-XB*, *Sym-K*, and *Sym-SV* indices indicate two clusters, which is also often obtained for many other methods for *Iris*. The *Sym-Dunn* and *Sym-FS* indices perform poorly for this data set. For the *Cancer* dataset, the MS value corresponding to the

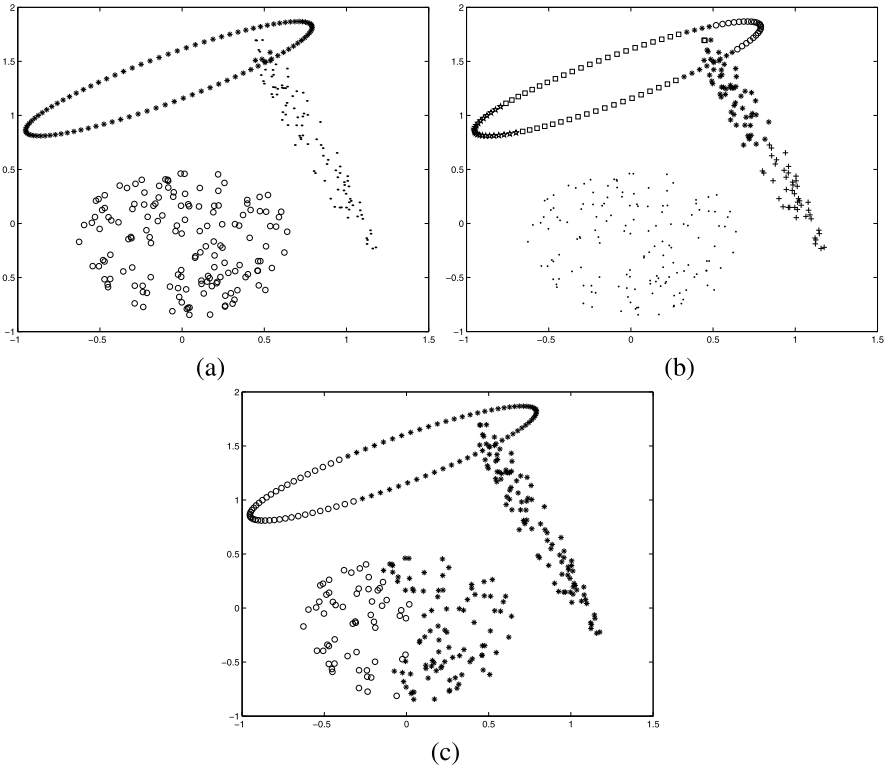


Fig. 6.9 Clustered *Sym_3_2* after application of GAPS for (**a**) $K = 3$, (**b**) $K = 6$, and (**c**) $K = 2$

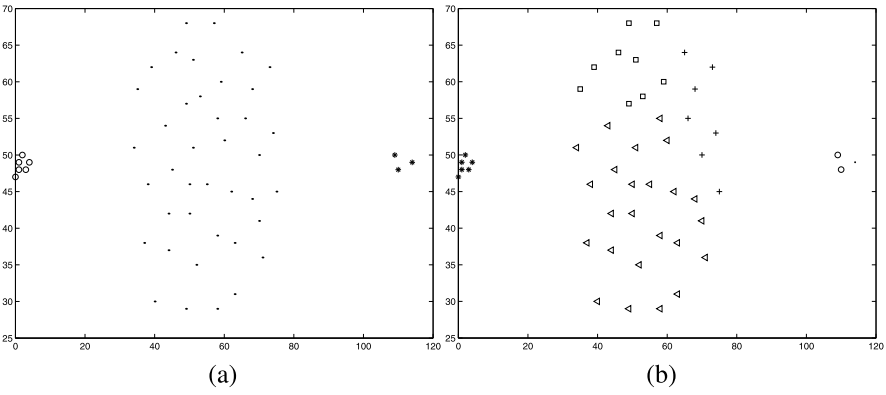


Fig. 6.10 Clustered *Bensaid_3_2* after application of GAPS for (**a**) $K = 3$ and (**b**) $K = 6$

partitioning obtained by GAPS clustering for $K = 2$ is 0.36 ± 0.00. The *Sym-DB*, *Sym-GDunn*, *Sym-PS*, *Sym*, *Sym-XB*, *Sym-K*, and *Sym-SV* indices are all able to indi-

cate this partitioning. But again for this data set, the performance of both *Sym-Dunn* and *Sym-FS* indices is poor. For the *LungCancer* data set, the *Sym-DB*, *Sym-PS*, *Sym*, *Sym-XB*, *Sym-SV*, and *Sym-K* indices are able to detect the proper number of clusters along with GAPS clustering (see Table 6.4). The corresponding MS value is 0.89 ± 0.01.

The above-mentioned results show that *Sym-DB* is able to detect the appropriate partitioning for three out of five data sets used here for the experiment. Similarly, the *Sym-Dunn*, *Sym-GDunn*, *Sym-PS*, *Sym*, *Sym-XB*, *Sym-FS*, *Sym-K*, and *Sym-SV* indices are able to detect the proper partitioning from two, three, four, five, four, zero, four, and two out of five data sets, respectively. Thus, it can be easily concluded that the *Sym*-index performs better than the other eight indices for detecting the proper number of clusters and the proper partitioning from data sets having symmetrical clusters.

Table 6.4 also provides the performance results of the original versions of the validity indices. The success rates of the two versions of the eight cluster validity indices (the original version and the symmetry version) in detecting the proper number of partitions and the proper partitioning are reported. Here, $SuccessRate(i) = \frac{A}{\text{total number of data sets}}$, where A is the number of data sets for which index i succeeds in determining the appropriate number of clusters. From the results provided in Table 6.4, it is easy to conclude that incorporation of the point symmetry-based distance into the definitions of the existing cluster validity indices makes them more effective in detecting any type of clusters from a data set irrespective of their shape and size as long as they possess the property of point symmetry. This is more evident from the results on the first artificial data set having clusters of different shapes possessing the point symmetry property. While the original versions of the eight cluster validity indices mostly fail in detecting the proper number of partitions from this data set, incorporation of the point symmetry distance imparts the property of characterizing these non-compact, symmetric clusters to them.

6.8 Application to Remote Sensing Imagery

An important task in remote sensing applications is classification of pixels in the images into homogeneous regions, each of which corresponds to some particular land cover type. This problem has often been modeled as a segmentation problem [190], and clustering methods have been used to solve it. However, since it is difficult to have a priori information about the number of clusters in satellite images, the clustering algorithms should be able to automatically determine this value. Moreover, in satellite images it is often the case that some regions occupy only a few pixels, while the neighboring regions are significantly large. Thus, automatically detecting regions or clusters of such widely varying sizes presents a challenge in designing segmentation algorithms.

The point symmetry (PS)-based cluster validity index, *Sym*-index, and GAPS have been used here for automatically determining the appropriate number of clus-

ters from different image data sets [243]. The number of clusters K is manually varied from K_{min} to K_{max}, and for each K, *Sym*-index is computed for the partitioning resulting from the application of GAPS clustering. The partitioning corresponding to the maximum value of *Sym*-index is presented as a solution to the segmentation problem.

The effectiveness of the new cluster validity index, *Sym*-index, in conjunction with GAPS clustering [27] for automatically detecting different types of regions is demonstrated on one simulated and one satellite image. Segmentation results are compared with those obtained by two other recently proposed cluster validity indices, namely the PS-index [58] and I-index [189], and another well-known index, the XB-index [295]. For each image, K is varied from 2 to 16.

6.8.1 Simulated Circle Image (SCI)

In order to show the effectiveness of *Sym*-index in identifying small clusters from much larger ones where there is significant overlap of the small clusters with the bigger one, we first generate artificial image of size 256×256 shown in Fig. 6.11(a). There are two small circles each of radius 20, centered at (113, 128) and (170, 128), respectively. The pixels of these two small circles take gray values randomly in the range [160–170] and [65–75], respectively. The background pixels take values randomly in the range [70–166]. Here also K is varied from 2 to 16. Figure 6.11(b) shows the segmented image using GAPS clustering with *Sym*-index, when three clusters were automatically found. We calculated the *Minkowski score* (MS) [146] (defined in Eq. 5.24) of the segmented image provided by the *Sym*-index. Smaller value of MS indicates better segmentation. The corresponding MS value is 0.177026. In contrast, the PS-index, I-index and XB-index attained their optimum values for $K^* = 9$, $K^* = 5$, and $K^* = 9$, respectively; i.e., they are not at all able to detect the proper number of clusters. K-means (with $K = 3$) is not able to find the proper clustering from this data set (shown in Fig. 6.11(c)). The MS value in this case is 0.806444. The EM algorithm is also not able to find the proper clustering from this overlapping dataset (Fig. 6.11(d)). The MS value in this case is 0.82.

6.8.2 SPOT Image of Kolkata

The French satellites SPOT (Systeme Probatoire d'Observation de la Terre) [232], launched in 1986 and 1990, carry two imaging devices that consist of a linear array of charge-coupled device (CCD) detectors. Two imaging modes are possible, the multispectral and panchromatic modes. The 512×512 SPOT image of a part of the City of Kolkata is available in three bands in the multispectral mode. These bands are:

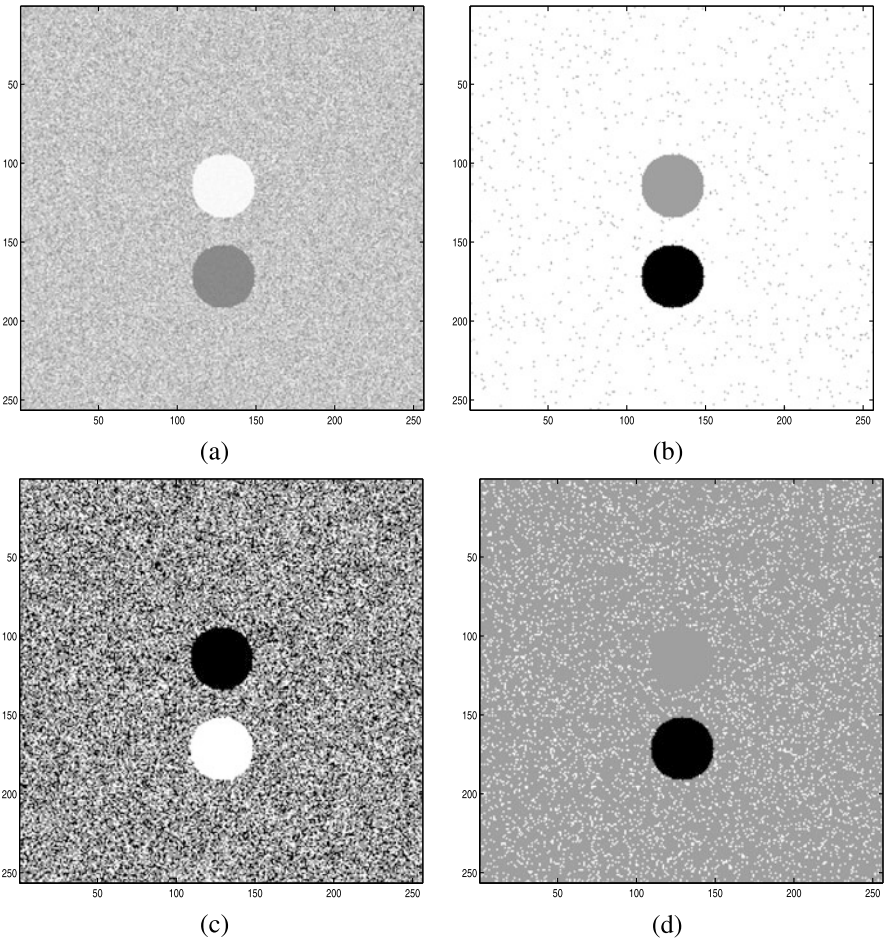


Fig. 6.11 (**a**) SCI. (**b**) Segmented SCI obtained by GAPS clustering with *Sym*-index (provides $K^* = 3$). (**c**) Segmented SCI obtained by K-means clustering for $K = 3$. (**d**) Segmented SCI obtained by EM-clustering for $K = 3$

Band 1: green band of wavelength 0.50–0.59 μm.
Band 2: red band of wavelength 0.61–0.68 μm.
Band 3: near infrared band of wavelength 0.79–0.89 μm.

Thus, here the feature vector of each image pixel is composed of three intensity values at different bands. The distribution of the pixels in the feature space of this image is shown in Fig. 6.13. It can be easily seen from Fig. 6.13 that the entire data can be partitioned into several hyperspherical clusters where symmetry does exist.

Some important landcovers of Kolkata are present in the image. Most of these can be identified, from knowledge about the area, more easily in the near infra-red band of the input image (Fig. 6.12(a)). These are the following: The prominent black

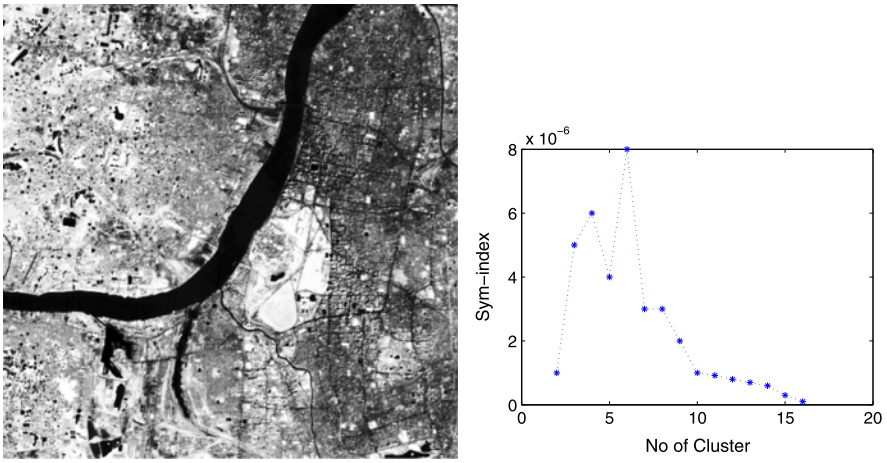
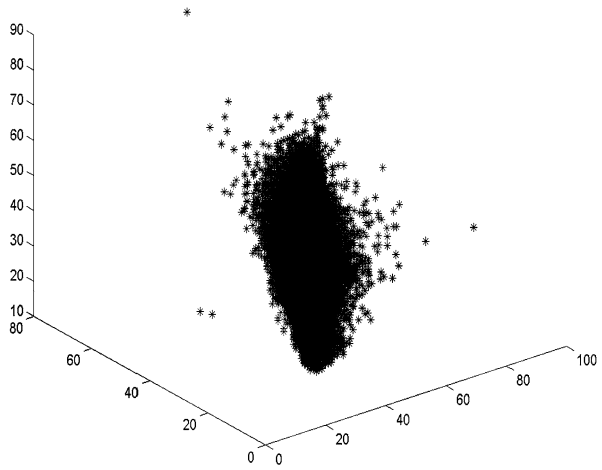


Fig. 6.12 (**a**) SPOT image of Kolkata in the NIR band with histogram equalization. (**b**) Variation of *Sym*-index with number of clusters for Kolkata image using GAPS

Fig. 6.13 Data distribution of SPOT image of Kolkata in the feature space



stretch across the figure is the River Hooghly. Portions of a bridge (referred to as the "second bridge"), which was under construction when the picture was taken, protrude into the Hooghly near its bend around the center of the image. There are two distinct black, elongated patches below the river, on the left side of the image. These are water bodies, the one to the left being Garden Reach Lake and the one to the right being Khidirpore Dockyard. Just to the right of these water bodies, there is a very thin line, starting from the right bank of the river, and going to the bottom edge of the picture. This is a canal called the Talis Nala. Above the Talis Nala, on the right side of the picture, there is a triangular patch, which is the

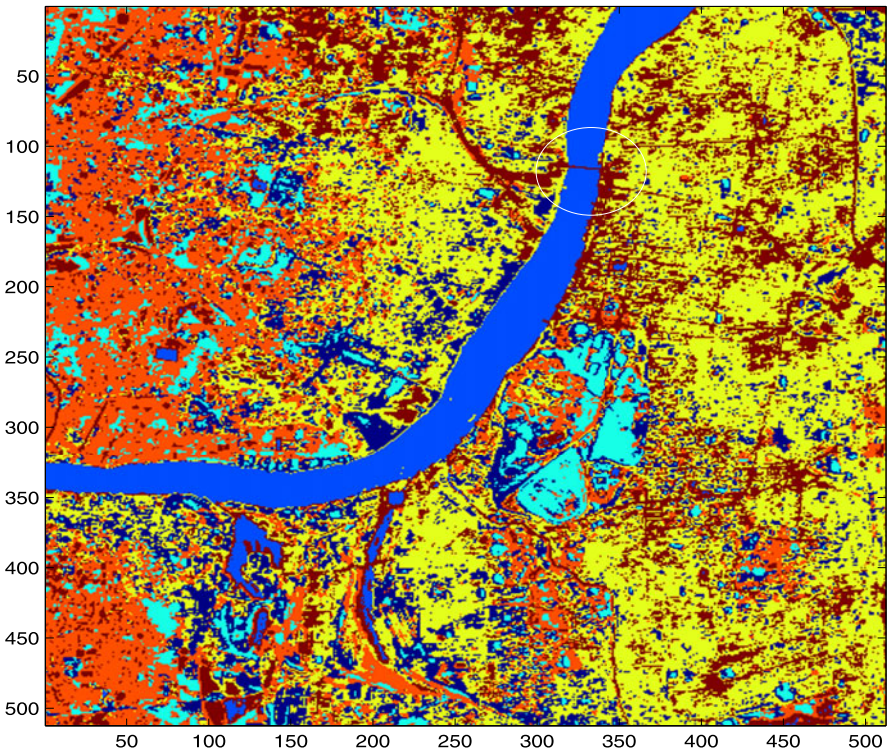


Fig. 6.14 Segmented Kolkata image obtained by GAPS clustering with *Sym*-index (provides $K^* = 6$)

race course. On the top, right hand side of the image, there is a thin line, stretching from the top edge, and ending on the middle, left edge. This is the Beleghata Canal with a road by its side. There are several roads on the right side of the image, near the middle and top portions. These are not very obvious from the images. A bridge cuts the river near the top of the image. This is referred to as the "first bridge".

GAPS clustering was applied on this image data set while varying the number of clusters K from 2 to 16. For each obtained partitioning, the values of four cluster validity indices (*Sym*-index, PS-index, I-index, and XB-index) were calculated. *Sym*-index obtained its optimal value for $K^* = 6$. The corresponding segmented image is shown in Fig. 6.14 (note that here different segments are shown using different colors). Similarly, I-index, PS-index, and XB-index obtained their optimum values for $K^* = 8$, $K^* = 3$, and $K^* = 2$, respectively, and the corresponding segmented images are shown in Figs. 6.15, 6.16, and 6.17, respectively. The segmentations corresponding to the optimum values of *Sym*-index and I-index are able to separate almost all the regions equally well (Figs. 6.14 and 6.15). Even the thin outline of the bridge on the river has been automatically identified (encircled in Fig. 6.14). This again illustrates the superiority of symmetry-based distance for detecting a small

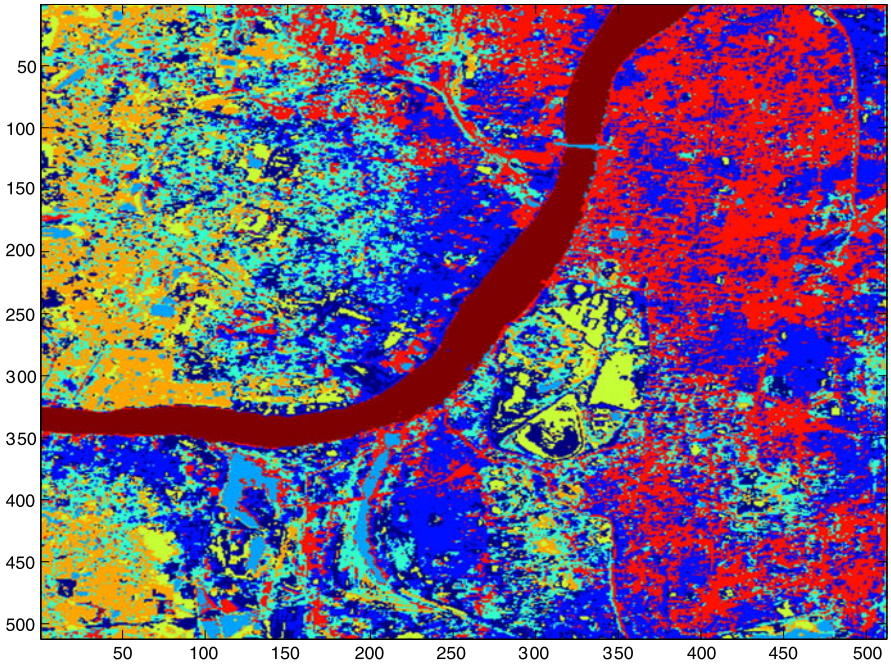


Fig. 6.15 Segmented Kolkata image obtained by GAPS clustering with I-index (provides $K^* = 8$)

cluster. To validate the results, 932 pixel positions were manually selected from 7 different land cover types which were labeled accordingly. For these points the *Minkowski score* (MS) [146] (defined in Eq. 5.24) is calculated after application of GAPS clustering for the optimal cluster number indicated by each of the indices. The MS scores corresponding to the *Sym*-index, I-index, PS-index, and XB-index are 0.865, 0.8799, 1.3692, and 1.4319, respectively, again demonstrating the superior result obtained with GAPS clustering in conjunction with *Sym*-index. PS-index and XB-index perform poorly for this image. For the segmented SPOT Kolkata image, the Davies Bouldin (DB) index [73] has been calculated corresponding to the optimal values of *Sym*-index, I-index, PS-index, and XB-index. The values are listed in Table 6.5. As smaller values of DB are preferable, this again signifies that segmentation corresponding to *Sym*-index is the best. Figure 6.12(b) shows the variations of the values of *Sym*-index with the number of clusters for this data set.

6.9 Discussion and Conclusions

Identifying the appropriate model and the model order are two crucial issues in unsupervised classification. A recently proposed symmetry-based cluster validity function, *Sym*-index, is defined in this chapter that exploits the property of point

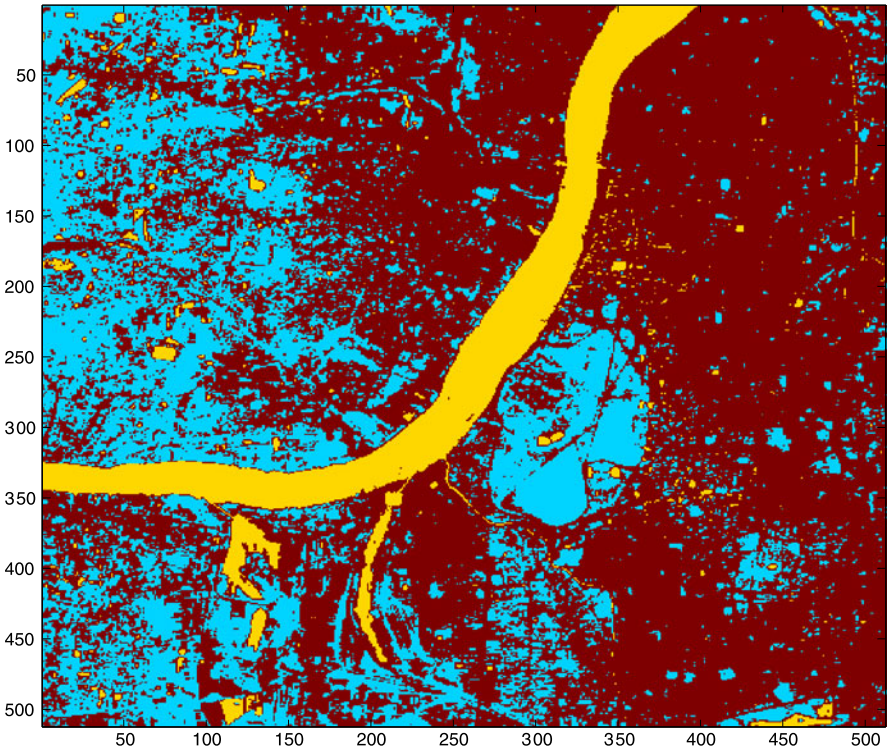


Fig. 6.16 Segmented Kolkata image obtained by GAPS clustering with PS-index (provides $K^* = 3$)

based symmetry to indicate both the appropriate number of clusters as well as the clustering algorithm. An elaborate description of the different components of *Sym*-index and an intuitive explanation of how they compete with each other to identify a proper clustering are provided. A mathematical justification of the new *Sym*-index is derived by establishing the relationship of the *Sym*-index with the well-known Dunn's index (however, note that *Sym*-index is not a generalization of the Dunn's index). The effectiveness of the *Sym*-index is demonstrated for two artificially generated and two real-life data sets. Six clustering algorithms, viz. GAPS, GAK-means, average linkage algorithm, two versions of the EM algorithm, and self organizing map, are used as the underlying partitioning methods. The experimental results establish the superiority of the new *Sym*-index as compared with four existing validity indices, namely PS index, *I*-index, CS-index, and XB-index, as long as the clusters present in it have a point-based symmetrical structure irrespective of their geometrical shape and convexity.

Thereafter, the point symmetry-based distance is incorporated into eight existing cluster validity indices. These indices exploit the property of point symmetry to indicate both the appropriate number of clusters as well as the appropriate partitioning. Results show that incorporation of the point symmetry distance into the definitions

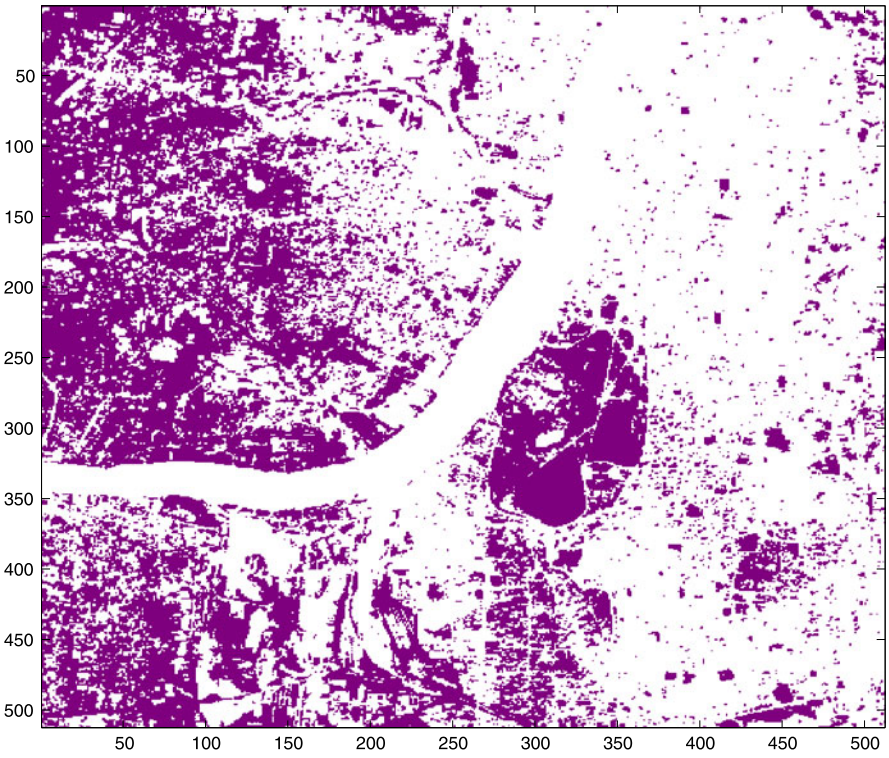


Fig. 6.17 Segmented Kolkata image obtained by GAPS clustering with XB-index (provides $K^* = 2$)

Table 6.5 DB-index values of the segmented Kolkata satellite image corresponding to the optimal values of four cluster validity indices

Validity index	SPOT image of Kolkata
Sym-index	0.669
I-index	0.775
PS-index	0.800
XB-index	0.724

of the existing eight cluster validity indices makes them more effective in determining the proper number of clusters and the appropriate partitioning from data sets having clusters of different shapes and sizes as long as they possess the property of point symmetry. In [244], results of the symmetry-based cluster validity indices have also been shown for data sets having clusters of different densities. Results show that, if the underlying partitioning technique is able to detect the appropriate partitioning in this case, some of the symmetry-based cluster validity indices, including *Sym*-index, are able to identify them.

Finally, an application of *Sym*-index in conjunction with the GAPS clustering technique is described for image segmentation. Its effectiveness, vis-à-vis other well-known validity indices, is first established for segmenting one artificially generated image. Thereafter, it is used for classifying different land cover types in a multispectral satellite image. The choice of the underlying clustering technique is important. Although the *Sym*-index has the capability of indicating the proper symmetric clusters, the underlying clustering technique should be able to first detect them; For example, both the well-known K-means and EM clustering algorithms are unable to find the proper clustering from data sets like synthetic images. In contrast, GAPS clustering [27] is able to tackle such situations, as is evident from its consistently good performance.

The present work determines the appropriate algorithm and the number of clusters in an iterated fashion. The next chapter deals with an approach of automating this process.

Chapter 7
Symmetry-Based Automatic Clustering

7.1 Introduction

Determining the appropriate number of clusters and the appropriate partitioning automatically from a given data set are important considerations in clustering. One of the approaches to this, as described in the previous chapter, is the following: (i) Execute a particular clustering algorithm for different values of the number of clusters (K); (ii) Determine the value of any cluster validity index for all these Ks; (iii) Consider that partitioning along with the corresponding K for which the validity index obtains its optimum value.

The above-mentioned method is an iterative one. Stochastic clustering algorithms using genetic algorithms (GAs) have also been used to optimize the cluster validity functions in order to automatically identify the appropriate number of clusters and the appropriate partitioning from a data set simultaneously [18, 20, 88, 254]. These approaches use cluster validity measures as the fitness function of the genetic algorithm, which guides the evolution to search for the appropriate solution. Several such algorithms are described later in this chapter.

In the genetic clustering techniques for automatic evolution of clusters, assignment of points to different clusters is done along the lines of the K-means clustering algorithm [28]. Consequently, all these approaches are only able to find compact hyperspherical, equisized, and convex clusters like those detected by the K-means algorithm [144]. If clusters of different geometric shapes are present in the same data set, the above methods will not be able to find all of them perfectly. This chapter describes an approach in this direction.

A variable string length GA (VGA)-based clustering method is described in this chapter. Here, assignment of points to different clusters is done based on the PS distance described in Chap. 5. The *Sym*-index, described in Chap. 6, is used as the optimizing criterion. The characteristic features of the new clustering technique, referred to as VGAPS clustering, are as follows: Use of variable string length GA allows the encoding of a variable number of clusters [18]. The *Sym*-index, used as the fitness function, provides the most approximate partitioning even when the number of clusters, K, is varied. Again use of GA enables the algorithm to come

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_7,

out of local optima, a typical problem associated with local search methods such as the K-means (note that optimizing *Sym*-index is not inherent to a GA framework. Any other optimization technique, such as simulated annealing [16], may be used). Finally, use of the PS-distance enables the evolution of clusters of any shape and size as long as they possess the symmetry property.

Using finite Markov chain theory, a convergence proof of VGAPS clustering to the globally optimal partition is also established. Thereafter, the new clustering technique is extended to handle the fuzzy partitioning of a data set. In order to determine the fitness function, a fuzzy symmetry-based cluster validity index, named the *FSym*-index [240], is used. The fuzzy algorithm is called fuzzy variable string length genetic point symmetry-based clustering technique (Fuzzy-VGAPS). As a real-life application, results are demonstrated on MR brain imaging data.

7.2 Some Existing Genetic Algorithm-Based Automatic Clustering Techniques

Evolutionary algorithms which automatically determine the number of clusters (K) present in a data set are described in the works by Cole [64], Cowgill et al. [67], Bandyopadhyay and Maulik [18, 20], Hruschka and Ebecken [135], Hruschka et al. [131–133], Ma et al. [183], and Alves et al. [256]. All these approaches use Euclidean distance for computing the similarity measures. Consequently, none of these algorithms are able to detect clusters having shapes other than hyperspheres.

In variable string length genetic clustering technique (GCUK clustering) [20], a variable string length genetic algorithm (VGA) [129] is applied with real parameter representation as the underlying search tool. The chromosome encodes the centers of a number of clusters, whose value may vary. Chromosomes are made up of real numbers (representing the coordinates of the centers of the clusters). If chromosome i encodes the centers of K_i clusters in N-dimensional space, $K_i \geq 2$, then its length l_i is $N \times K_i$. The initial K_i centers are randomly selected points from the data set. Modified versions of crossover and mutation operations are used. The Davies-Bouldin [73] cluster validity index is utilized for computing the fitness of the chromosomes. (Note that any other cluster validity indices can as well be used as the fitness function.) For each chromosome, the centers encoded in it are first extracted, then a partition is obtained by assigning points to different clusters based on the closest center criterion. The cluster centers encoded in the chromosome are then replaced by the centroids of the corresponding clusters. Given the above partition, the number of clusters and the validity index are computed. The fitness of a chromosome is then defined as a function of the corresponding cluster validity index. Results have been presented for four artificial and two real-life data sets [20]. In this chapter, results are presented for GCUK clustering for different data sets.

In the hybrid niching genetic algorithm (HNGA)-based clustering technique [254], a weighted sum validity function (WSVF), which is a weighted sum of several normalized cluster validity functions, is used for optimization to automatically

evolve the proper number of clusters and the appropriate partitioning of the data set. Within the HNGA, a niching method is developed to prevent premature convergence during the search. Additionally, in order to improve the computational efficiency, a hybridization between the niching method with the computationally attractive K-means is made. Here, WSVF is defined as $WSVF = \sum_{i=1}^{m} w_i f_i(x)$, where m is the number of component functions. In [254] $m = 6$ is used. The w_is are non-negative weighting coefficients representing the relative importance of the functions such that $\sum_{i=1}^{m} w_i = 1$, and $f_i(x)$ are component functions (as used in [254]) corresponding to the reciprocal of the DB-index [73], SIL-index [153], Dunn-index [87], generalized Dunn-index [38], CH-index [47], and I-index [189], respectively. The weighting coefficients are chosen as $w_1 = w_2 = \cdots = w_m = 1/m$. Center-based real encoding is used. Each individual i in the population is constructed by random assignment of real numbers to each of the attributes of the k_i cluster centers, where k_i is the number of cluster centers encoded in the individual. The initial values of the cluster centers are constrained to be in the range (determined from the data set) of the attribute to which they are assigned but are otherwise random. A single step of K-means algorithm is executed on all new offsprings during each generation to improve the computational efficiency. During this step each data point is assigned to the nearest cluster center encoded in a particular chromosome. After that, the cluster centers encoded in the individual's chromosome are replaced by the mean objects of the respective clusters. During crossover, the cluster centers are considered to be indivisible. Here, two-point crossover [112] is used. After crossover, a low probability of Gaussian mutation was applied to the offspring. Gaussian mutation adds a unit Gaussian-distributed random value to the selected feature value. The new attribute value is truncated if it falls outside the lower or upper bounds of the attribute. The stopping criterion adopted is that the fitness value of the best population individual does not change for some fixed number of generations. Experimental results are presented for three artificial and three real-life data sets. Comparisons are made with the clustering genetic algorithm [88] and GCUK clustering techniques [20].

In [176] an algorithm for evolutionary clustering with self-adaptive genetic operators (ECSAGO) is developed. This algorithm is based on the unsupervised niche clustering (UNC) and hybrid adaptive evolutionary (HAEA) algorithms [176]. The UNC is a genetic clustering algorithm which is robust to noise and can determine the appropriate number of clusters from data sets automatically. HAEA is a parameter adaptation technique that automatically learns the rates of its genetic operators at the same time that the individuals are evolved in an evolutionary algorithm (EA) [176]. In ECSAGO, real encoding and real genetic operators are used.

In [70] a differential evolution (DE) [264] based automatic clustering technique was proposed. DE is easy to implement and requires very little parameter tuning in order to attain considerably good search results. The conventional DE algorithm is modified here from its classical form to improve its convergence properties. In order to detect the optimal number of clusters, a new representation scheme for the search variables is also developed. The new algorithm is named as the automatic clustering DE (ACDE) algorithm. Two different sets of experiments with two different fitness functions, namely the CS measure [59] and DB measure [73], are conducted.

7.3 Description of VGAPS

In this section the variable string length genetic clustering technique with point symmetry-based distance (VGAPS) clustering algorithm [28] based on the *Sym*-index (described in Chap. 6) and using a genetic algorithm as the underlying optimization technique is described in detail. It includes determination of the number of clusters as well as the appropriate clustering of the data set.

7.3.1 Chromosome Representation and Population Initialization

In VGAPS clustering, the chromosomes are made up of real numbers which represent the coordinates of the centers of the partitions. If chromosome i encodes the centers of K_i clusters in d-dimensional space, then its length l_i is taken to be $d * K_i$. For example, in three-dimensional space, the chromosome $\langle 12.3\ 1.4\ 5.6\ \ 22.1\ 0.01\ 10.2\ \ 0.0\ 5.3\ 15.3\ \ 13.2\ 10.2\ 7.5\rangle$ encodes four cluster centers: (12.3, 1.4, 5.6), (22.1, 0.01, 10.2), (0.0, 5.3, 15.3), and (13.2, 10.2, 7.5). Each center is considered to be indivisible. Each string i in the population initially encodes the centers of a number, K_i, of clusters, such that $K_i = (\text{rand}() \bmod (K_{max} - 1)) + 2$. Here, rand() is a function returning an integer, and K_{max} is a soft estimate of the upper bound of the number of clusters. The number of clusters will therefore range from two to K_{max}. The K_i centers encoded in a chromosome are randomly selected distinct points from the data set.

7.3.2 Fitness Computation

Fitness computation is composed of two steps. Firstly, points are assigned to different clusters using the point symmetry-based distance, d_{ps}, as done in GAPS (described in Sect. 5.6.2). Next, the cluster validity index, *Sym*-index (defined in Sect. 6.3.1), is computed and used as a measure of the fitness of the chromosome. This fitness function is maximized using a genetic algorithm.

7.3.3 Genetic Operations and Terminating Criterion

The following genetic operations are performed on the population of strings for a number of generations.

7.3.3.1 Selection

The selection operator randomly selects a chromosome from the previous population according to the distribution

$$P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j),} \tag{7.1}$$

where $F(s_i)$ represents the fitness value (*Sym*-index) of the string s_i in the population and N denotes the population size. Here, a string receives a number of copies that is proportional to its fitness in the population.

7.3.3.2 Crossover

For the purpose of crossover, the cluster centers are considered to be indivisible; i.e., the crossover points can only lie in between two cluster centers. The crossover operation, applied stochastically, must ensure that information exchange takes place in such a way that both offspring encode the centers of at least two clusters. For this purpose, the operator is defined as follows [190]: Let parent chromosomes P_1 and P_2 encode cluster centers M_1 and M_2, respectively. The crossover point, τ_1, in P_1 is generated as $\tau_1 = \text{rand}() \bmod M_1$. Let τ_2 be the crossover point in P_2; it may vary in between [LB(τ_2), UB(τ_2)], where LB(τ_2) and UB(τ_2) indicate the lower and upper bounds of the range of τ_2, respectively. LB(τ_2) and UB(τ_2) are given by $\text{LB}(\tau_2) = \min[2, \max[0, 2 - (M_1 - \tau_1)]]$ and $\text{UB}(\tau_2) = [M_2 - \max[0, 2 - \tau_1]]$. Therefore, τ_2 is given by

$$\begin{aligned} \tau_2 &= \text{LB}(\tau_2) + \text{rand}() \bmod\big(\text{UB}(\tau_2) - \text{LB}(\tau_2)\big), \quad \text{if } \big(UB(\tau_2) \geq LB(\tau_2)\big), \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

It can be verified by some simple calculations that, if the crossover points τ_1 and τ_2 are chosen according to the above rules, then none of the offspring generated would have fewer than two clusters.

Example 7.1 Let $M_1 = 4$ and $M_2 = 5$; i.e., two parents P_1 and P_2 encode, respectively, four and five cluster centers. Let $\tau_1 = \text{rand}() \bmod M_1 = 2$. Then, the lower and upper bounds of τ_2 are calculated as

$$\begin{aligned} LB(\tau_2) &= \min\big[2, \max\big[0, 2 - (M_1 - \tau_1)\big]\big] \\ &= \min\big[2, \max[0, 4 - 2]\big] \\ &= \min[2, 2] \\ &= 2, \end{aligned} \tag{7.2}$$

$$\begin{aligned} UB(\tau_2) &= \big[M_2 - \max[0, 2 - \tau_1]\big] \\ &= \big[5 - \max[0, 2 - 2]\big] \\ &= [5 - 0] \\ &= 5. \end{aligned} \tag{7.3}$$

Thus, τ_2 should be a random number chosen in the range [2, 5].

The crossover probability, μ_c, is selected adaptively as in GAPS described in Sect. 5.6.4.

7.3.3.3 Mutation

Three types of mutations are considered here.

1. Mutation 1: Each cluster center encoded in a chromosome is replaced with a random variable drawn from a Laplacian distribution, $p(\varepsilon) \propto e^{-\frac{|\varepsilon-\mu|}{\delta}}$, where the scaling factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. The scaling factor δ is chosen equal to 1.0. The old value at the position is replaced with the newly generated value. Here, this type of mutation operator is applied for all dimensions independently.
2. Mutation 2: One randomly generated cluster center is removed from the chromosome; i.e., the total number of clusters encoded in the chromosome is decreased by 1.
3. Mutation 3: The total number of clusters encoded in the chromosome is increased by 1. One randomly chosen point from the data set is encoded as the new cluster center.

Example 7.2 Let a chromosome look like $\langle 3.5\ 1.5 \quad 2.1\ 4.9 \quad 1.6\ 1.2 \rangle$, representing three cluster centers in a 2-d plane $(3.5, 1.5)$, $(2.1, 4.9)$, and $(1.6, 1.2)$.

(i) If mutation type 1 is selected, then first one position from the chromosome has to be selected for perturbation. Let position 2 be selected randomly. Then, each dimension of $(2.1, 4.9)$ will be changed by some value generated using the Laplacian distribution.
(ii) If mutation type 2 is selected, a center will be removed from the string. Let center 3 be selected for deletion. Then, after deletion, the chromosome will look like $\langle 3.5\ 1.5 \quad 2.1\ 4.9 \rangle$.
(iii) If mutation type 3 is selected, a new center will be added to the chromosome. Let the randomly chosen point from the data set to be added to the chromosome be $(9.7, 2.5)$. After addition of this center, the chromosome will look like $\langle 3.5\ 1.5 \quad 2.1\ 4.9 \quad 1.6\ 1.2 \quad 9.7\ 2.5 \rangle$.

Any one of the above-mentioned types of mutation is applied to each chromosome of the population with some probability of mutation, p_m. The mutation probability is selected adaptively for each chromosome as in GAPS (described in Sect. 5.6.5).

7.3.3.4 Termination Criterion

The processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string having the largest

fitness (i.e., the largest *Sym*-index value) seen up to the last generation provides the solution to the clustering problem. Elitism is implemented at each generation by preserving the best string seen up to that generation in a location outside the population and also inside the population, replacing the string with the lowest fitness value. Thus, on termination, this outside location contains the centers of the final clusters.

7.4 On the Convergence Property of VGAPS

Using finite Markov chain theory it has been proved that the canonical genetic algorithms converge to the global optimum [236]. In [167] it is also proved along the lines of [236] that the genetic K-means algorithm also converges to the global optimum of square-error (SE) measure, depending on some conditions on its parameters. Here, the global convergence of VGAPS to the optimum value of *Sym*-index will be proved along similar lines by deriving some conditions on the parameters of VGAPS that ensure the global convergence. The proof is available in [28].

Consider the process $\{\mathscr{P}(t)\}, t \geq 0$, where $\mathscr{P}(t)$ represents the population maintained by VGAPS at generation t. The state space of this process is the space of all possible populations $\mathscr{S}$, and the states are numbered from 1 to $|\mathscr{S}|$. Here, the state space comprises the populations containing strings representing partitions with K clusters, where $K \in [K_{min}, K_{max}]$. From the definition of VGAPS, $\mathscr{P}(t+1)$ can be determined completely by $\mathscr{P}(t)$, i.e.,

$$\begin{aligned} &Pr\big\{\mathscr{P}(t) = p_t \big| \mathscr{P}(t-1) = p_{t-1}, \ldots, \mathscr{P}(0) = p_0\big\} \\ &= Pr\big\{\mathscr{P}(t) = p_t \big| \mathscr{P}(t-1) = p_{t-1}\big\}. \end{aligned}$$

Hence, $\{\mathscr{P}(t)\}, t \geq 0$, is a Markov chain. Also, the transition probabilities are independent of the time instant; i.e., if

$$p_{ij}(t) = Pr\big\{\mathscr{P}(t) = p_j \big| \mathscr{P}(t-1) = p_i\big\},$$

then $p_{ij}(s) = p_{ij}(t)$ for all $p_i, p_j \in \mathscr{S}$ and for all $s, t \geq 1$. Therefore, $\{\mathscr{P}(t)\}, t \geq 0$ is a time-homogeneous finite Markov chain. Let $\mathbf{P} = (p_{ij})$ be the *transition matrix* of the process $\{\mathscr{P}(t)\}, t \geq 0$. The entries of the matrix $\mathbf{P}$ satisfy $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|\mathscr{S}|} p_{ij} = 1, \forall i \in \mathscr{S}$. $\mathbf{P}$ is a *stochastic matrix*. Some definitions are given below for use in the rest of this section.

A square matrix $\mathbf{A}_{m \times m}$ is said to be positive if $a_{ij} > 0, \forall i, j \in \{1, 2, \ldots, m\}$, and is said to be *primitive* if there exists a positive integer k such that $\mathbf{A}^k$ is positive. A square matrix is said to be *column-allowable* if it has at least one positive entry in each column.

In the following theorem it is required that $\mathbf{P}$ be a primitive matrix. So, first we investigate the conditions on the operators which make the matrix $\mathbf{P}$ primitive. The probabilistic changes of the chromosome within the population caused by the

operators used in VGAPS are captured by the transition matrix **P**, which can be decomposed in a natural way into a product of stochastic matrices

$$\mathbf{P} = \mathbf{K} \times \mathbf{C} \times \mathbf{M} \times \mathbf{S}, \tag{7.4}$$

where **K**, **C**, **M**, and **S** describe the intermediate transitions caused by K-means, i.e., the update center, crossover, mutation, and selection operators, respectively. It is easy to consider that all these matrices are stochastic matrices.

Proposition 7.1 *Stochastic matrices form a group under matrix multiplication.*

Thus, for the two stochastic matrices **K** and **C**, by Proposition 7.1, $\mathbf{C}' = \mathbf{K} \times \mathbf{C}$ is also a stochastic matrix. Therefore, Eq. 7.4 can be written as

$$\mathbf{P} = \mathbf{C}' \times \mathbf{M} \times \mathbf{S}, \tag{7.5}$$

where $\mathbf{C}'$, **M**, and **S** are stochastic matrices.

Proposition 7.2 *Let* $\mathbf{C}'$, **M**, *and* **S** *be stochastic matrices, where* **M** *is positive and* **S** *is column-allowable. Then, the product* $\mathbf{C}' \times \mathbf{M} \times \mathbf{S}$ *is positive.*

Since every positive matrix is primitive, it is therefore enough to find the conditions which make **M** positive and **S** column-allowable.

7.4.1 To Check Whether the Mutation Matrix Is Positive

The matrix **M** is positive if any string $s \in \mathscr{S}$ can be obtained from any other string on application of the corresponding mutation operator. The mutation operator defined in Sect. 7.3.3.3 ensures the above condition. The mutation operator is of three types. The first type is for obtaining a valid position from any other valid position. By generating a random variable using a Laplacian distribution, there is a non-zero probability of generating any valid position from any other valid position, while the probability of generating a value near the old value is greater. The second type is for decreasing the value of K; i.e., from a chromosome consisting of K_1 number of centers, another chromosome having K_2 number of clusters, where $K_1 > K_2$, is generated by this type of mutation operation. The third type of mutation operator is for increasing the value of K in a particular chromosome; i.e., if a chromosome encodes K_1 clusters, where $K_1 < K_{max}$, then by the third type of mutation operation some new cluster centers can be included in it, increasing the number of clusters.

The above discussion implies that the mutation operation can change any string to any other string in the search space with non-zero probability. Hence, the transition matrix, **M**, corresponding to the above mutation operator is positive.

7.4.2 Conditions on Selection

The probability of survival of a string in the current population depends on the fitness value of the string; so is the transition matrix due to selection, **S**. Very little can be said about **S** if the fitness function is defined as only the *Sym*-index value of that particular partition. The following modification to the fitness function will ensure the column-allowability of **S**: Let

$$F(s) = c_s \times Sym_{max} + Sym(s), \tag{7.6}$$

where Sym_{max} is the maximum *Sym*-index value that has been encountered till the present generation and $c_s \geq 1$. $Sym(s)$ is the *Sym*-index value of the sth string. Then, the fitness value of each chromosome in the population is strictly positive. Therefore, the probability that selection does not alter the present state, s_{ii}, can be bounded as follows:

$$\begin{aligned} s_{ii} &\geq \frac{F(s_1)}{\sum_{l=1}^{N} F(s_l)} \times \frac{F(s_2)}{\sum_{l=1}^{N} F(s_l)} \times \cdots \times \frac{F(s_N)}{\sum_{l=1}^{N} F(s_l)} \\ &= \frac{\prod_{l=1}^{N} F(s_l)}{(\sum_{l=1}^{N} F(s_l))^N} > 0 \quad \forall i \in \mathscr{S}, \end{aligned}$$

where s_l is the lth string of the current population. Even though this bound changes with the generation, it is always strictly positive; hence, the selection matrix **S** is *column-allowable*.

Theorem 7.1 *Let $X(t) = Sym(s^*(t))$, where $s^*(t)$ is the string with maximum Sym-index value encountered during the evolution of VGAPS till the time instant t. Let the mutation operator be the same as defined in Sect. 7.3.3.3, and the fitness function be as defined in Eq. 7.6. Then*

$$\lim_{t \to \infty} Pr\{X(t) = Sym^*\} = 1, \tag{7.7}$$

where $Sym^* = \max\{Sym(i) | i \in \mathscr{T}\}$, $\mathscr{T}$ is the set of all legal strings.

Proof It is proved (Ref. [236, Theorem 6]) that a canonical GA whose transition matrix is primitive and which maintains the best solution found over time converges to the global optimum in the sense given in Eq. 7.7. It is proved in Proposition 7.2 that the transition matrix of VGAPS clustering with the same mutation operator as defined in Sect. 7.3.3.3 and the fitness function as defined in Eq. 7.6 is positive. Since every positive matrix is primitive, thus the transition matrix of VGAPS is also primitive. Moreover, VGAPS uses an elitist GA; i.e., it preserves the best solution found till the current time instant. Thus, the above theorem follows from Ref. [236, Theorem 6].

The above theorem implies that $X(t)$, the maximum *Sym*-index value of the strings encountered by VGAPS till the instant t, converges to the global optimum Sym^*, with probability 1 when it goes to infinity. □

7.5 Data Sets Used and Implementation Results

This section provides a description of the data sets and the implementation results of the VGAPS algorithm. Three artificial and two real-life data sets are used for the experiments.

7.5.1 Data Sets Used

1. Artificial data sets used: *Sym_3_2*, *AD_5_2* and *Bensaid_3_2*.
2. Real-life data sets: *Iris* and *Cancer*.

7.5.2 Results and Discussions

In VGAPS clustering, the population size is taken to be equal to 100. K_{min} (the minimum number of clusters) and K_{max} (the maximum number of clusters) are set equal to 2 and $\sqrt{n}$, respectively, where n is the total number of data points in the data set. VGAPS is executed for a total of 50 generations.

In order to evaluate the VGAPS method, two types of experiments are performed. First we show that VGAPS optimizing the *Sym*-index performs better than VGAPS optimizing the two other indices, viz. PS-index [58] and I-index [189]. After that, the properties of VGAPS optimizing the *Sym*-index are explored and its performance is compared with other genetic clustering methods, which do not need knowledge about the number of clusters a priori.

7.5.2.1 Exploring *Sym*-Index as a Fitness Function

In the first experiment, we establish the effectiveness of using the *Sym*-index with VGAPS clustering vis-à-vis another point symmetry-based validity index, i.e., the PS-index [58], and a Euclidean distance-based cluster validity index, i.e., the I-index [189]. The numbers of clusters obtained after applying VGAPS optimizing these three validity indices separately for all the data sets are shown in Table 7.1. It can be seen from the table that VGAPS clustering with the *Sym*-index is able to find the proper cluster number from data sets having symmetrical-shaped clusters. VGAPS clustering with the I-index is, in general, able to find the proper cluster number from data sets with spherically symmetrical structure, but it is not able to detect other shaped clusters. This is because the I-index essentially prefers hyperspherical clusters, which is not the case for *Sym_3_2*. VGAPS clustering with the PS-index is able to detect the proper clusters from those data sets where the clusters have strong point symmetry. However, as discussed in Sect. 5.2, the definition of point symmetry distance in the PS-index precludes the detection of symmetrical interclusters. Thus, it fails for *AD_5_2*, which has clearly symmetrical interclusters.

Table 7.1 Comparing the number of clusters found in the experimental data sets by VGAPS clustering using *Sym*-index, PS-index, and *I*-index as the cluster objective function for computing fitness, GCUK clustering and HNGA clustering. Here, AC denotes the actual number of clusters present in the data and OC denotes the obtained number of clusters

Data set	AC	OC by VGAPS using			OC by different methods		
		Sym	*I*	PS	VGAPS	GCUK	HNGA
Sym_3_2	3	3	8	3	3	3	16
AD_5_2	5	5	6	4	5	5	5
Bensaid_3_2	3	3	3	3	3	2	3
Iris	3	3	3	2	3	2	2
Cancer	2	2	2	2	2	2	2

7.5.2.2 Exploring the VGAPS-Clustering

In this section, we compare the performance of VGAPS clustering (in conjunction with the *Sym*-index) with that of GCUK clustering [20] and the recently developed HNGA clustering [254]. GCUK clustering utilizes the Davies-Bouldin [73] cluster validity index for computing the fitness of the chromosomes. In HNGA [254], a weighted sum validity function (WSVF), which is a weighted sum of several normalized cluster validity functions, is used for optimization.

Table 7.1 presents the number of clusters identified by the three clustering algorithms for all the data sets. As is evident from the table, VGAPS is able to find the appropriate number of clusters and the proper partitioning for all the data sets. Figures 7.1(a), 7.2(a), and 7.3(a) show the final partitionings obtained after application of VGAPS on *Sym_3_2*, *AD_5_2*, and *Bensaid_3_2*, respectively. Although for *AD_5_2* VGAPS is able to detect the clusters reasonably well, it is found to somewhat over approximate the central cluster (which extends to the left).

Final clustering results obtained after the application of the GCUK algorithm to the five artificial data sets are also shown in Figs. 7.1(b), 7.2(b), and 7.3(b), respectively. Results presented in Table 7.1 reveal that GCUK clustering is able to determine the proper cluster number only for the *Sym_3_2*, *AD_5_2*, and *Cancer* data sets. However, for *Sym_3_2*, even though GCUK clustering is able to detect the proper number of clusters, the final partitioning identified by it (shown in Fig. 7.1(b)) is not proper. Figures 7.1(c), 7.2(c), and 7.3(a) show, respectively, the clustering results obtained after application of HNGA clustering to the three artificial data sets. Again, results presented in Table 7.1 reveal that HNGA clustering is able to determine the proper cluster number only for the *AD_5_2*, *Bensaid_3_2*, and *Cancer* data sets. Thus, it is easy to conclude that HNGA clustering is only able to find hyperspherical clusters from a data set but not any other shaped clusters. The main reason behind such performance is that it optimizes a convex combination of some cluster validity indices all of which are only able to detect hyperspherical-shaped clusters.

The *Minkowski score* (MS) [146] (defined in Eq. 5.24) of the resultant partitionings was calculated after application of all three algorithms to both the artificial and

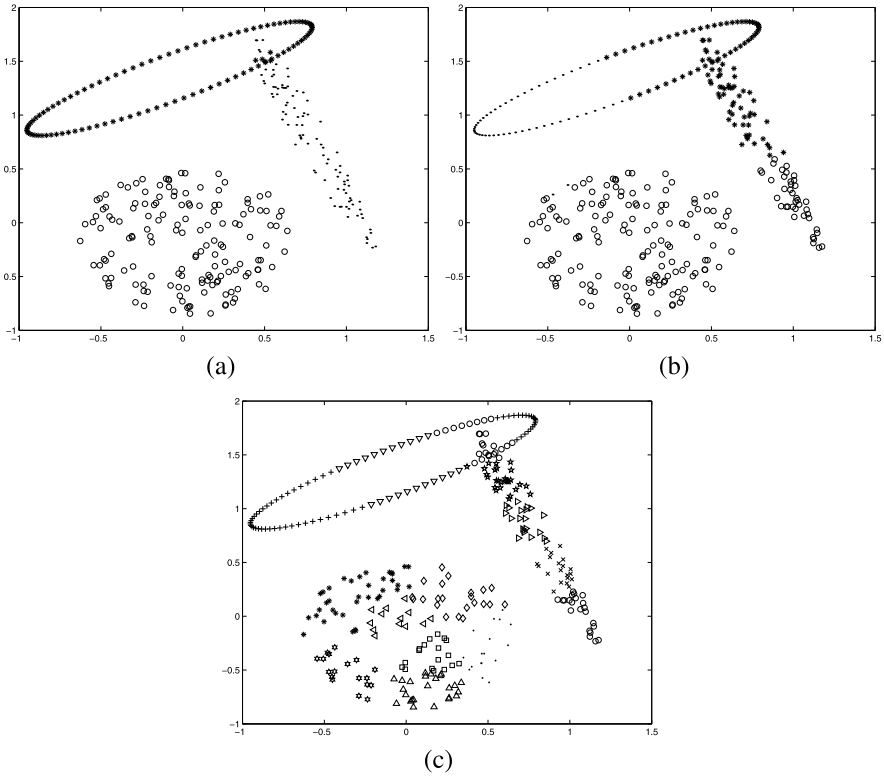


Fig. 7.1 Clustered *Sym_3_2* after application of (**a**) VGAPS clustering, where three clusters are detected, (**b**) GCUK clustering where, three clusters are detected, (**c**) HNGA clustering, where 16 clusters are detected

real-life data sets used here for the experiments. Each of the three algorithms was executed ten times on each data set. The average MS scores and their standard deviations for all the experimental data sets after application of the three algorithms are given in Table 7.2. Except for *AD_5_2*, VGAPS clustering was found to provide the lowest MS values for the other data sets, which indicates that the partitionings corresponding to VGAPS clustering are the best among the three clustering algorithms. ANOVA [5] statistical analysis was performed on the combined results of the three algorithms. The one-way ANOVA procedure produces a one-way analysis of variance for a quantitative dependent variable (here, the MS value) by a single independent variable (here, the algorithm). Analysis of variance is used to test the hypothesis that several means are equal. From the ANOVA statistical test, it is found that the differences in the mean MS values obtained by VGAPS clustering, compared with those obtained by the GCUK clustering and HNGA clustering algorithms, are statistically significant at the level of 0.05 for all data sets. This indicates better performance of VGAPS as compared with GCUK clustering and HNGA clustering. For *AD_5_2*, HNGA clustering performs the best in terms of MS score, and

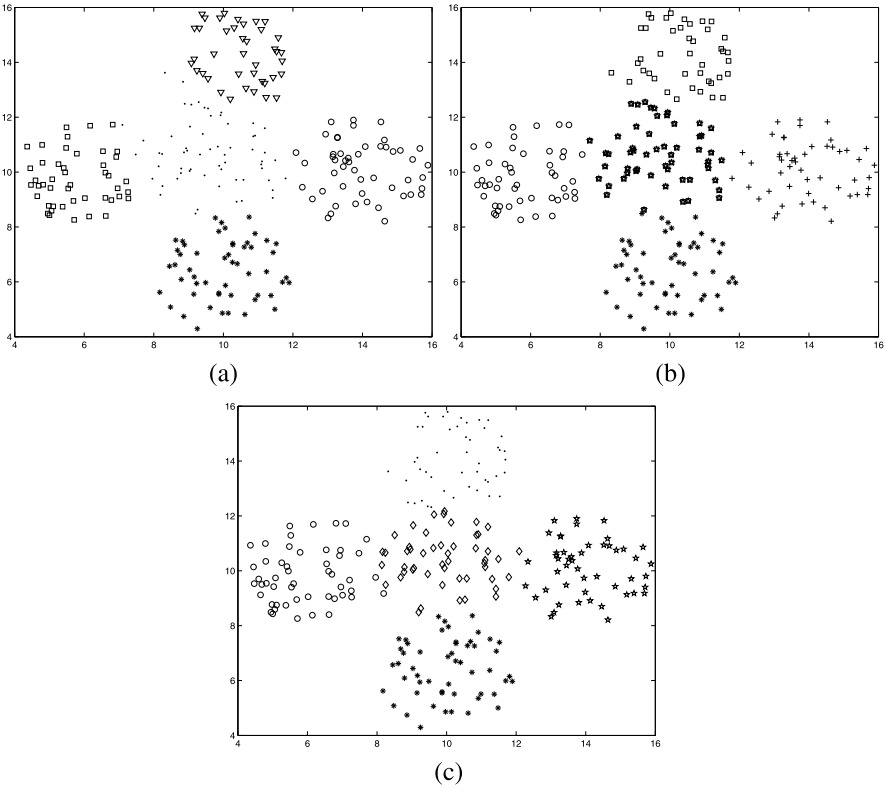


Fig. 7.2 Clustered *AD_5_2* using (**a**) VGAPS clustering, where five clusters are detected, (**b**) GCUK clustering, where five clusters are detected, (**c**) HNGA clustering, where five clusters are detected

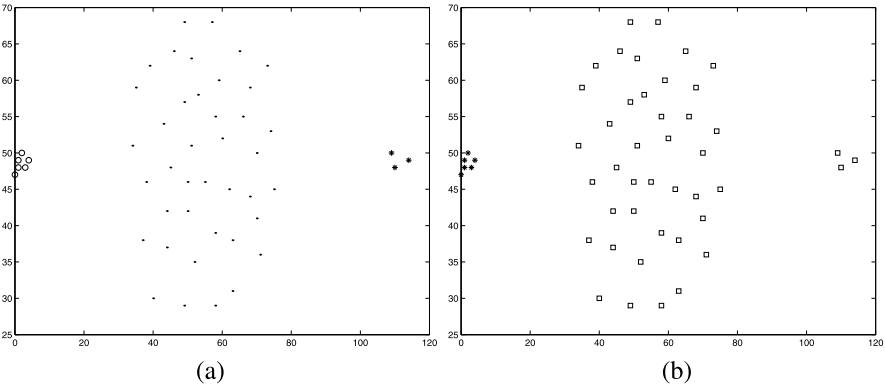


Fig. 7.3 (**a**) Clustered *Bensaid_3_2* by VGAPS clustering and HNGA clustering, where three clusters are detected, (**b**) Clustered *Bensaid_3_2* by GCUK clustering, where two clusters are detected

Table 7.2 *Minkowski scores* obtained by three algorithms for all data sets used here for the experiment. Smaller values of MS indicate better partitioning

Data set	VGAPS clustering	GCUK clustering	HNGA clustering
Sym_3_2	0.12 ± 0.00	1.05 ± 0.02	0.85 ± 0.02
AD_5_2	0.42 ± 0.02	0.14 ± 0.001	0.10 ± 0.002
Bensaid_3_2	0 ± 0.00	0.62 ± 0.02	0 ± 0.00
Iris	0.62 ± 0.02	0.85 ± 0.01	0.85 ± 0.025
Cancer	0.36 ± 0.001	0.38 ± 0.02	0.38 ± 0.023

the difference in the mean MS values obtained by the HNGA and VGAPS clustering techniques is statistically significant with significance value 2.4603×10^{-8}.

7.6 Extending VGAPS to Fuzzy Clustering

In order to extend VGAPS to fuzzy clustering, first the *Sym*-index is extended to define a fuzzy symmetry-based cluster validity index: *FSym*-index.

7.6.1 Fuzzy Symmetry-Based Cluster Validity Index

The fuzzy symmetry-based cluster validity index *FSym*-index [240], which measures the goodness of a partitioning in terms of "symmetry" and the separation present in the clusters, is presented below. Let K cluster centers be denoted by $\overline{c}_i$ where $1 \leq i \leq K$ and $U(X) = [u_{ij}]_{K \times n}$ be a partition matrix for the data. Then, the *FSym*-index is defined as follows:

$$FSym(K) = \left(\frac{1}{K} \times \frac{1}{\mathscr{E}_K} \times D_K \right), \tag{7.8}$$

where K is the number of clusters. Here,

$$\mathscr{E}_K = \sum_{i=1}^{K} E_i \tag{7.9}$$

such that

$$E_i = \sum_{j=1}^{n} \left(u_{ij}^m \times d_{ps}(\overline{x}_j, \overline{c}_i) \right) \tag{7.10}$$

and

$$D_K = \max_{i,j=1}^{K} \| \overline{c}_i - \overline{c}_j \|. \tag{7.11}$$

D_K is the maximum Euclidean distance between two cluster centers among all centers. $d_{ps}(\overline{x}_j, \overline{c}_i)$ is the point symmetry distance [27] between the point $\overline{x}_j$ and the cluster center $\overline{c}_i$. This index is inspired by the PBMF-index developed in [212]. The objective is to maximize this index in order to obtain the actual number of clusters.

Framework of the Formulation In order to obtain the actual number of clusters and to achieve the proper clustering from a data set, the *FSym*-index value has to be maximized. As formulated in Eq. 7.8, *FSym* is a composition of three factors: $1/K$, $1/\mathscr{E}_K$, and D_K. The first factor increases as K decreases; as *FSym* needs to be maximized for optimal clustering, so it will prefer to decrease the value of K. The second factor is the within-cluster total symmetrical measure. For clusters which have good symmetrical structure, E_i is small. This, in turn, indicates that formation of more clusters, which are symmetrical in shape, would be encouraged. Finally, the third factor, D_K, measuring the maximum separation between a pair of clusters, increases with the value of K. As these three factors are complementary in nature, so they are expected to compete and balance each other critically for determining the proper partitioning.

7.6.2 Fuzzy-VGAPS Clustering

In this section, a variable string length genetic point symmetry-based clustering technique (Fuzzy-VGAPS clustering) is described. Here, the best partition is considered to be the one that corresponds to the maximum value of the *FSym*-index. In Fuzzy-VGAPS, both the number of clusters as well as the appropriate fuzzy clustering of the data are evolved simultaneously using the search capability of genetic algorithms. The basic steps of the clustering technique are provided in Fig. 7.4. The technique is described below in detail.

7.6.2.1 Chromosome Representation and Population Initialization

In Fuzzy-VGAPS clustering, center-based encoding as in GAPS/VGAPS (described in Sect. 7.3.1) is used.

7.6.2.2 Fitness Computation

The fitness computation is performed for each chromosome. This is composed of two steps. Firstly, membership values of n points to different clusters are computed. Next, the *FSym*-index (defined in Sect. 7.6.1) is computed and used as a measure of the fitness of the chromosome.

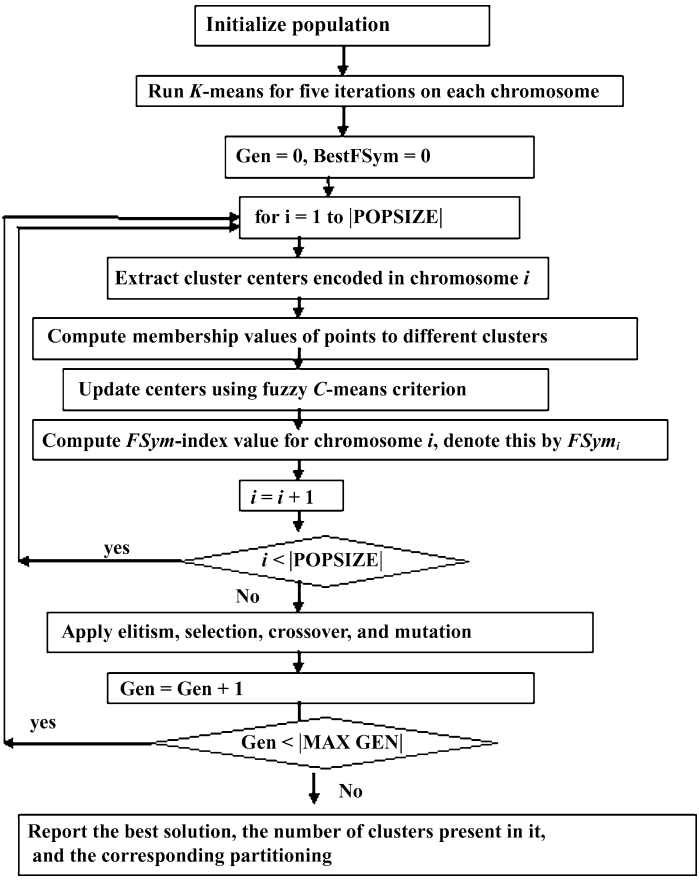


Fig. 7.4 Flowchart of Fuzzy-VGAPS clustering technique

Computing the Membership Values In Fuzzy-VGAPS, a particular point is assigned to a particular cluster with membership value 1 if it is truly symmetrical with respect to that cluster center. For points whose symmetrical measures to all the cluster centers are greater than some threshold (i.e., points which are not symmetrical with respect to any cluster center), membership values are calculated using the Euclidean distance. For each point $\overline{x}_j$, $j = 1, 2, \ldots, n$, the membership values to K different clusters are calculated in the following way: Find the cluster center nearest to $\overline{x}_j$ in the symmetrical sense; That is, we find the cluster center k that is nearest to the input pattern $\overline{x}_j$ using the minimum-symmetric-distance criterion: $k = \text{argmin}_{i=1,\ldots,K}\, d_{ps}(\overline{x}_j, \overline{c}_i)$, where the point symmetry-based distance $d_{ps}(\overline{x}_j, \overline{c}_i)$ is computed by Eq. 5.10. Here, $\overline{c}_i$ denotes the center of the ith cluster. If the corresponding $d_{sym}(\overline{x}_j, \overline{c}_k)$ (as defined in Eq. 5.9) is smaller than a prespecified parameter θ, then the membership u_{ij} is updated using the following criterion:

$$u_{ij} = 1, \quad \text{if } i = k,$$
$$u_{ij} = 0, \quad \text{if } i \neq k.$$

Otherwise, the membership u_{ij} is updated by using the following rule, which corresponds to the normal fuzzy C-means [37] algorithm:

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} (\frac{d_{ij}}{d_{kj}})^{\frac{2}{m-1}}}, \tag{7.12}$$

where $m \in [1, \infty)$ is a weighting exponent called the fuzzifier. Here, we set $m = 2$. d_{ij} represents the Euclidean distance between a pattern $\overline{x}_j$ and the cluster center $\overline{c}_i$. Note that, in Eq. 7.12, the Euclidean distance has been used instead of the symmetry-based one. This is because Eq. 7.12 is derived in [37] under the assumption that the distance measure used should be a norm. As mentioned earlier, the d_{ps} measure is not a norm. Thus, the point symmetry-based distance cannot be used in Eq. 7.12.

The value of θ is kept equal to d_{NN}^{max}, the maximum nearest neighbor distance in the data set, as in Sect. 5.6.2.

Updating the Centers The centers encoded in a chromosome are updated using the following equation as in FCM:

$$\overline{c}_i = \frac{\sum_{j=1}^{n} (u_{ij})^m \overline{x}_j}{\sum_{j=1}^{n} (u_{ij})^m}, \quad 1 \leq i \leq K. \tag{7.13}$$

Centers are updated as above in order to incorporate a limited amount of local search (provided by FCM-like center update) to speed up the convergence of Fuzzy-VGAPS.

Fitness Calculation The fitness of a chromosome indicates the degree of goodness of the solution it represents. It is computed using the newly developed *FSym*-index. The objective of the GA is to maximize this fitness function.

7.6.2.3 Selection, Crossover, and Mutation

Conventional proportional selection is applied on the population of strings. Here, a string receives a number of copies that is proportional to its fitness in the population. We use the roulette wheel strategy for implementing the proportional selection scheme. We use a crossover operation similar to that used in the VGAPS clustering technique (described in Sect. 7.3.3.2).

The mutation operation is also very similar to that used in the VGAPS clustering technique (described in Sect. 7.3.3.3).

7.6.2.4 Termination

Fuzzy-VGAPS is executed for a fixed number of generations. Moreover, the elitist model of GAs is used, where the best string seen so far is stored in a location within

the population. The best string of the last generation provides the solution to the clustering problem.

7.7 Implementation Results and Comparative Study

For the experimental results, five data sets are considered. These are categorized into two groups. The first group consists of two artificial data sets, and the second group consists of three real-life data sets obtained from [2]. The two artificial data sets are *Sym_3_2*, described in Sect. 5.8.1, and *AD_10_ 2*, described below. These artificial data sets contain overlapping clusters. The real-life data sets are *Iris*, *Cancer*, and *LungCancer*, described in Sect. 5.8.1.

1. *AD_10_2*: This data set, used in Ref. [24], consists of 500 two-dimensional data points distributed over 10 different clusters. Some clusters are overlapping in nature. Each cluster consists of 50 data points. This data set is shown in Fig. 7.5.

In the first part of the experimental results, the superiority of the *FSym*-index as compared with the PBMF-index [212], a new fuzzy cluster validity index-based on relative degree of sharing (fuzzy-RDS) proposed in [157], and a modified version of the XB-index, XB* [156], is demonstrated. In all these cases, Fuzzy-VGAPS is used as the underlying partitioning method. The parameters of the algorithm are as follows: The population size, P, is set equal to 100. Fuzzy-VGAPS is executed for a total of 50 generations. The mutation and crossover probabilities of the Fuzzy-VGAPS algorithm are selected adaptively as discussed earlier. The number of clusters automatically determined by Fuzzy-VGAPS clustering optimizing the four indices are presented in Table 7.3. This table shows that Fuzzy-VGAPS optimizing the *FSym*-index is able to determine the appropriate number of clusters for all the data sets. The other indices are able to do so for only two data sets. Thus, it can

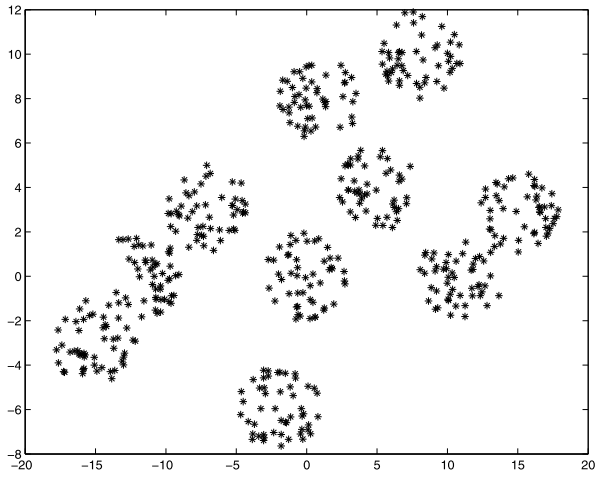


Fig. 7.5 *AD_10_2*

Table 7.3 Results obtained with the different data sets using Fuzzy-VGAPS clustering algorithm optimizing the *FSym*-index, PBMF-index, fuzzy cluster validity index, and XB*-index. Here, AC denotes the actual number of clusters present in the data set

Name	No. of points	Dimension	AC	Obtained number of clusters using			
				FSym	PBMF	fuzzy-RDS	XB*
Sym_3_2	350	2	3	3	8	16	4
AD_10_2	500	2	10	10	11	19	2
Iris	150	4	3	3	3	9	2
Cancer	683	9	2	2	4	17	2
LungCancer	32	56	3	3	3	3	3

be concluded that Fuzzy-VGAPS optimizing the *FSym*-index definitely has an edge over the others in identifying the appropriate number of clusters and the appropriate partitioning, even for mixed type of clusters.

In the second part of the experimental results, the Fuzzy-VGAPS clustering technique optimizing the *FSym*-index is compared with seven other clustering techniques:

- Fuzzy-VGA [190], optimizing the well-known XB-index [295].
- A crisp algorithm: The well-known K-means clustering technique with the newly developed point symmetry-based distance (PSKM). This algorithm follows the basic steps of SBKM proposed in [266] but uses the newly developed point symmetry-based distance, d_{ps} [27]. It a priori assumes the number of clusters present in a data set.
- An adaptation of the fuzzy C-means algorithm using the point symmetry distance (PSFCM) to evaluate the advantage provided by the genetic approach. It also assumes the number of clusters present in a data set.
- A recently developed fuzzy genetic clustering technique: The multistage random sampling genetic algorithm-based fuzzy C-means clustering algorithm (GM-FCM) developed in Ref. [83]. It also assumes a priori the number of clusters present in a data set.
- A recently developed fuzzy variant of an evolutionary algorithm for clustering (EAC-FCM) [48], which uses a fuzzy cluster validity index, fuzzy silhouette [49], and a fuzzy local search algorithm.
- Hybrid centroid-medoid heuristics (Hybrid) [117]: This is an enhancement of the K-means clustering technique where local search is used to accelerate convergence without greatly increasing the number of iterations. It also assumes a priori the number of clusters present in a data set.
- A shape-based clustering technique, SPARCL [53]. It also assumes a priori the number of clusters present in a data set.

The parameters of the Fuzzy-VGA clustering technique are as follows: population size = 100, total number of generations = 20, probability of mutation = 0.01 and probability of crossover = 0.8. Both the K-means algorithm with point

symmetry-based distance (PSKM) and fuzzy C-means with point symmetry-based distance (PSFCM) clustering techniques are executed until they converge. Here, the number of clusters (K) is set equal to the number of clusters identified by *FSym*-index for a particular data set. For GMFCM and EAC-FCM, the population size is kept equal to 100. In GMFCM, the other parameters are set as specified in Ref. [83]. For GMFCM, again the number of clusters is kept equal to the number of clusters identified by the *FSym*-index. In Fuzzy-VGAPS, Fuzzy-VGA, and EAC-FCM while generating the initial population the maximum value of the number of clusters is kept equal to $\sqrt{n}$, where n is the total number of points present in the data set, and the fuzzifier $m = 2.0$. For SPARCL, the results were obtained from the respective authors, thus being available for only one run for all data sets except *LungCancer*. Authors did not execute their code on *LungCancer* due to its higher dimensional nature. For all the data sets, the actual number of clusters present in the data set and those obtained by the three algorithms, Fuzzy-VGA, EAC-FCM, and Fuzzy-VGAPS, which can determine K automatically, are presented in Table 7.4. The results of the other algorithms, i.e., which assume a fixed K, are shown visually.

7.7.1 Discussion of Results

- *Sym_3_2*: This data set is utilized to show that Fuzzy-VGAPS is able to detect automatically different shaped clusters present in a data set as long as they are point symmetric. Fuzzy-VGAPS clustering is able to automatically detect the proper number of partitions as shown in Table 7.3 and the proper partitioning from this data set as clusters present here have "symmetrical" structure. The partitioning is shown in Fig. 7.6(a). As the data set contains non-convex clusters, Fuzzy-VGA is not able to detect the proper partitioning. The corresponding partitioning is shown in Fig. 7.6(b). The partitioning obtained by PSKM, PSFCM, and GMFCM for $K = 3$ (the number of clusters identified by Fuzzy-VGAPS) for this data set are shown in Figs. 7.6(c), 7.6(d), and 7.6(e), respectively. EAC-FCM automatically indicates $K = 11$ as the proper number of clusters (the corresponding partitioning is shown in Fig. 7.6(f)), as clusters present in this data set are not hyperspherical in nature. The partitionings indicated by the hybrid clustering technique [117] and SPARCL clustering technique [53] for this data set are shown in Figs. 7.6(g) and 7.6(h), respectively. The hybrid clustering technique, in general, fails to detect non-hyperspherical-shaped clusters. SPARCL is not able to detect the overlapping clusters correctly.
- *AD_10_2*: This data set is used to test the performance of the Fuzzy-VGAPS clustering technique for some overlapping clusters. Fuzzy-VGAPS is able to automatically detect the proper partitioning and the proper number of partitions from this data set. The corresponding partitioning is shown in Fig. 7.7(a). This result shows that the fuzzy characteristics of Fuzzy-VGAPS are used to make the partition more flexible, and thus it can handle overlapping clusters. Fuzzy-VGA automatically indicates $K = 16$ as the proper number of clusters, and the corresponding partitioning is shown in Fig. 7.7(b). The partitionings obtained by

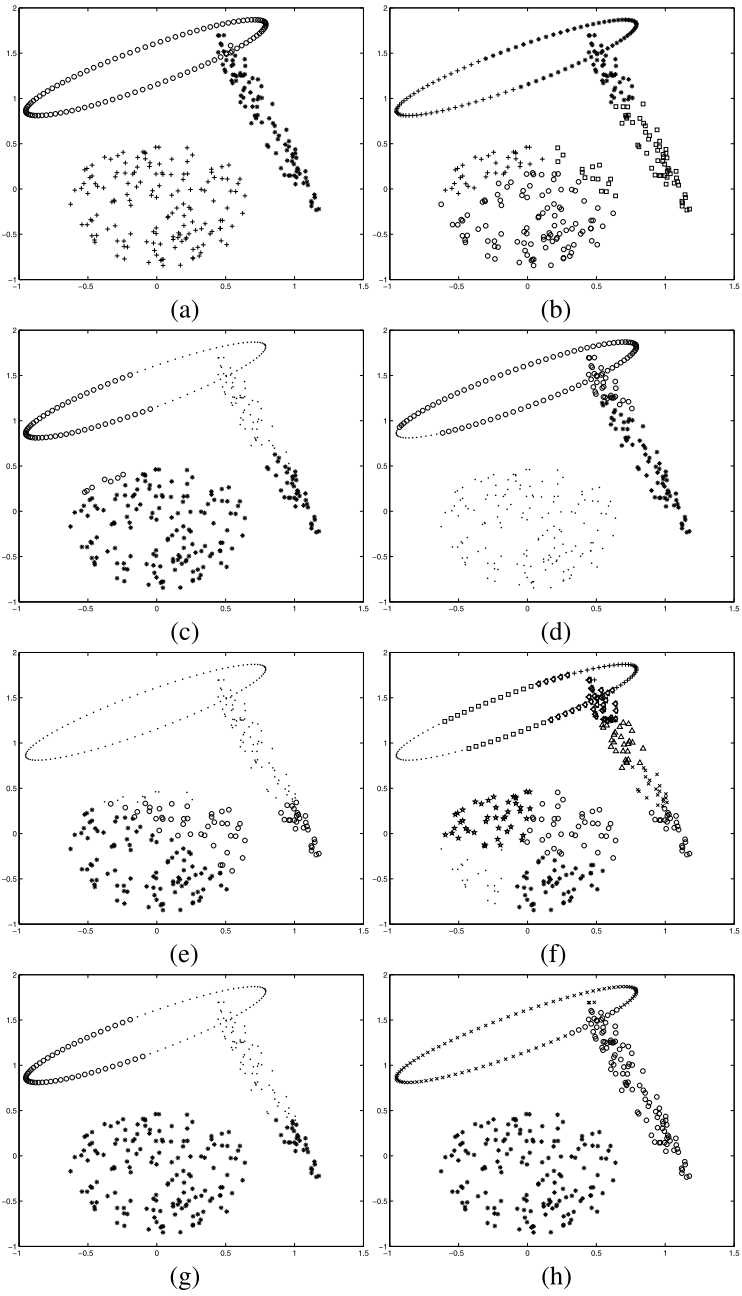


Fig. 7.6 Clustering results on *Sym_3_2* by (**a**) Fuzzy-VGAPS providing $K = 3$, (**b**) Fuzzy-VGA providing $K = 4$, (**c**) PSKM for $K = 3$, (**d**) PSFCM for $K = 3$, (**e**) GMFCM for $K = 3$, (**f**) EAC-FCM providing $K = 11$, (**g**) hybrid algorithm for $K = 3$, and (**h**) SPARCL algorithm for $K = 3$

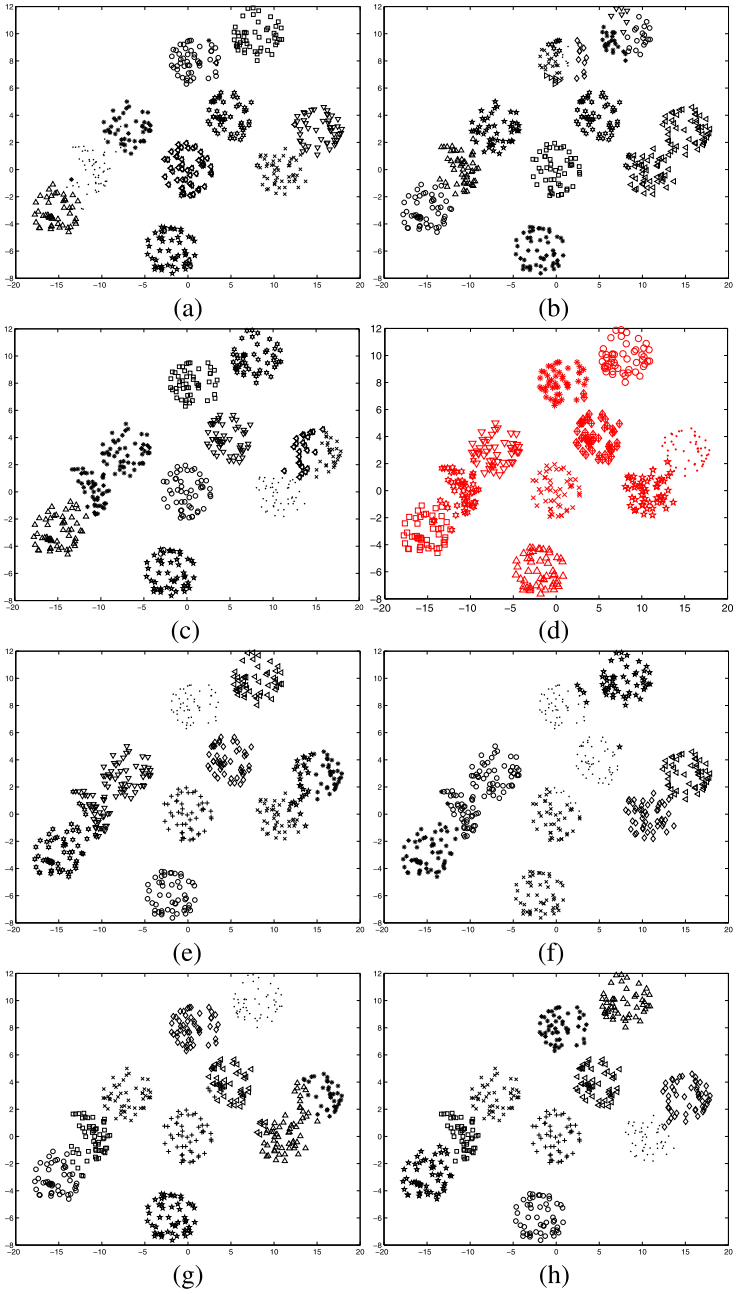


Fig. 7.7 Clustering results on *AD_10_2* by (**a**) Fuzzy-VGAPS providing $K = 10$, (**b**) Fuzzy-VGA providing $K = 16$, (**c**) PSKM for $K = 10$, (**d**) PSFCM for $K = 10$, (**e**) GMFCM for $K = 10$, (**f**) EAC-FCM providing $K = 7$, (**g**) Hybrid algorithm for $K = 10$, and (**h**) SPARCL algorithm for $K = 10$

Table 7.4 Results obtained with the different data sets using the Fuzzy-VGAPS, Fuzzy-VGA, and EAC-FCM clustering algorithms

Name	Actual number of clusters	Obtained number of clusters		
		Fuzzy-VGAPS	Fuzzy-VGA	EAC-FCM
Sym_3_2	3	3	4	11
AD_10_2	10	10	16	7
Iris	3	3	2	3
Cancer	2	2	2	2
LungCancer	2	2	5	3

Table 7.5 Estimated marginal means and variances of the *Minkowski score* of eight algorithms obtained for three real-life data sets. Smaller values of MS indicate better partitioning

Data set	Minkowski score							
	Fuzzy-VGAPS	Fuzzy-VGA	PSKM	PSFCM	GMFCM	EAC-FCM	Hybrid	SPARCL
Iris	0.62 ±0.001	0.85 ±0.002	0.65 ±0.02	0.62 ±0.024	0.52 ±0.024	0.77 ±0.033	0.82 ±0.003	0.57
Cancer	0.32 ±0.0004	0.39 ±0.0001	0.37 ±0.021	0.33 ±0.0021	0.31 ±0.0021	0.39 ±0.00234	0.35 ±0.00345	0.33
LungCancer	0.82 ±0.001	1.13 ±0.0022	0.84 ±0.0132	0.88 ±0.0012	0.90 ±0.003	1.15 ±0.0021	1.39 ±0.0043	– –

the PSKM, PSFCM, GMFCM, hybrid, and SPARCL clustering techniques for $K = 10$ (the number of clusters identified by Fuzzy-VGAPS) for this data set are shown in Figs. 7.7(c), 7.7(d), 7.7(e), 7.7(g), and 7.7(h), respectively. EAC-FCM indicates $K = 7$ as the proper number of clusters. The corresponding partitioning is shown in Fig. 7.7(f).

- *Iris*: For the real-life data sets, the *Minkowski scores* (MS) [146] were computed for each algorithm. The MS scores and their variances provided by the eight clustering algorithms for these three data sets are reported in Table 7.5. Statistical analysis of variance (ANOVA) [5] was performed for the real-life data sets on the combined MS values of all the algorithms when each is executed ten times. For SPARCL, the results were obtained from the respective authors, thus being available for only one run for all data sets except *LungCancer*. As mentioned earlier, the authors did not execute their code on *LungCancer* due to its higher dimensional nature.

As is evident from Table 7.4, Fuzzy-VGAPS is able to automatically determine the proper number of clusters for this data set. Fuzzy-VGA automatically determines two clusters for this data set, which is also often reported for many other methods for *Iris*. The MS score corresponding to the partitioning provided by Fuzzy-VGAPS is

smaller compared with that of Fuzzy-VGA (refer to Table 7.5). ANOVA tests show that the difference in mean MS score between Fuzzy-VGAPS and Fuzzy-VGA clustering is significant, again revealing that the partitioning provided by Fuzzy-VGAPS is better than that of Fuzzy-VGA. EAC-FCM also automatically determines three clusters from this data set, but the corresponding MS score (shown in Table 7.5) is worse than that of Fuzzy-VGAPS. The MS scores of the partitionings provided by PSKM, PSFCM, GMFCM, hybrid, and SPARCL for $K = 3$ (the number of partitions identified by Fuzzy-VGAPS) are also provided in Table 7.5. ANOVA analysis reveals that the performance of Fuzzy-VGAPS and PSFCM is the same for this data set. Table 7.5 shows that GMFCM performs the best for this data set in terms of MS score. The hybrid clustering technique attains a higher MS value for this data set (refer to Table 7.5), but SPARCL attains a smaller MS score for this particular data set.

- *Cancer*: For this data set, all three algorithms, Fuzzy-VGAPS, Fuzzy-VGA, and EAC-FCM, are able to determine the proper number of clusters (refer to Table 7.4), but the MS score corresponding to Fuzzy-VGAPS is the minimum (refer to Table 7.5) and ANOVA analysis shows that this improvement is statistically significant. This again shows that the partitioning obtained by Fuzzy-VGAPS clustering is better than that of Fuzzy-VGA and EAC-FCM. The PSKM, PSFCM, GMFCM, hybrid, and SPARCL algorithms were also executed on this data set with $K = 2$ (the number of partitions identified by Fuzzy-VGAPS), and the corresponding MS scores are also reported in Table 7.5. The table shows that Fuzzy-VGAPS and PSFCM again perform similarly for this data set, and this is corroborated by the ANOVA analysis. It is also revealed from Table 7.5 that GMFCM again performs the best for this data set in terms of MS score. The hybrid and SPARCL clustering techniques attain slightly higher MS values for this data set.
- *LungCancer*: It can be seen from Table 7.4 that only Fuzzy-VGAPS is able to find the proper number of clusters from this data set. The MS scores, reported in Table 7.5, again demonstrate the superior performance of Fuzzy-VGAPS over Fuzzy-VGA and EAC-FCM for automatically detecting the proper partitioning from a data set. The PSKM, PSFCM, GMFCM, hybrid, and SPARCL algorithms were also executed on this data set with $K = 2$ (the number of partitions identified by Fuzzy-VGAPS), and the corresponding MS scores are also reported in Table 7.5. The table shows that Fuzzy-VGAPS performs the best for this data set in terms of MS score. ANOVA statistical analysis was also done here. The analysis (results shown in Table 7.5) shows that the mean MS differences of all the algorithms are statistically significant.

Summary of Results It can be observed from the above results that Fuzzy-VGAPS is able to determine automatically the proper number of partitions and the proper partitioning for a wide variety of data sets having any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristic of symmetry. Note that it will fail for data sets where clusters do not have any point symmetry property (where the other algorithms will fail as well).

The PSKM and PSFCM clustering algorithms are based on local search, and hence they may often get stuck at local optima depending on the choice of the initial cluster centers. The use of a genetic algorithm and the cluster validity index, *FSym*-index, in Fuzzy-VGAPS enables it to detect automatically the number of clusters present in a data set. The presented results clearly illustrate the properties of Fuzzy-VGAPS. Results on *AD_10_2* show that the fuzzy characteristics of Fuzzy-VGAPS make the partitions more flexible and thus enable it to handle more overlapping clusters. It may be noted that, when the clusters are compact and well separated, a crisp clustering technique is expected to be able to work well. In such cases, application of fuzzy clustering may not be able to provide any advantage.

However, when the data are overlapping, fuzzy clustering methods are expected to handle them more efficiently vis-à-vis crisp methods. This is evident for *AD_10_2*. Fuzzy-VGA and EAC-FCM, two recently developed fuzzy clustering techniques for automatically determining the number of clusters, only succeed for data sets having hyperspherical clusters. They cannot detect the proper number of clusters from data sets having some symmetrical clusters other than hyperspheres. However, Fuzzy-VGAPS succeeds in such situations. The hybrid clustering technique is also able to detect only hyperspherical-shaped clusters but it fails for non-hyperspherical-shaped clusters. The SPARCL clustering technique can detect any shaped clusters only when the clusters are well separated and also when the dimensionality is low. Thus, it fails for overlapping clusters. Moreover, neither the hybrid nor the SPARCL clustering technique can determine the number of clusters automatically from a data set. Based on these observations, and the fact that the property of symmetry is widely evident in real-life situations, application of Fuzzy-GAPS in most clustering tasks seems justified and is therefore recommended.

7.7.2 Application to MR Brain Image Segmentation

The process of partitioning an image space into some nonoverlapping meaningful homogeneous regions is termed segmentation [114]. In general, these regions will have a strong correlation with the objects in the image. The quality of segmentation determines the success of an image analysis system. Segmentation is a necessary preprocessing task in the analysis of medical images for computer-aided diagnosis and therapy. Due to the intrinsically imprecise nature of the images, medical image segmentation is a hard and challenging task.

Automatically classifying brain tissues from magnetic resonance imaging (MRI) is an important task for research and clinical study of neurological pathology. Accurate partitioning of MR images into different tissue classes like gray matter (GM), white matter (WM) and cerebrospinal fluid (CRF) is a pressing problem. Additionally, regional volume calculations may provide even more useful diagnostic information. Among them, quantization of gray and white matter volumes may be helpful to detect neurodegenerative disorders such as Alzheimer disease, movement disorders such as Parkinson or Parkinson-related syndrome, white matter metabolic or

inflammatory disease, congenital brain malformations or perinatal brain damage, or posttraumatic syndrome.

Automatic segmentation of brain MR images is a complex problem. This is due to the presence of overlap between the MR intensities of different tissue classes and the presence of spatially smoothly varying intensity inhomogeneity. Statistical approaches are often used for MR image segmentation. These methods assign classes to pixels according to probability values, which are determined based on the intensity distribution of the image. The hidden Markov random field (HMRF) model has also been used for MR image segmentation combined with the expectation maximization (EM) algorithm [44] to estimate the involved model parameters [303, 304]. Clustering approaches have been widely used for segmentation of MR brain images. Unsupervised clustering methods are reliable and have high reproducibility because the results are primarily based on information in the image data, requiring little or no assumption in the model, and the distribution of the image data. Use of neural networks, evolutionary computation, and/or fuzzy clustering techniques for MR image segmentation has been investigated in [40, 269]. In [240] the problem of automatic partitioning of MR brain images is posed as clustering in intensity space. The Fuzzy-VGAPS clustering technique is applied to automatically segment MR normal brain images and MR brain images with multiple sclerosis lesions.

7.7.3 Experimental Results

The MR image of the brain chosen for the experiment is available in three bands: T_1-weighted, proton density (p_d)-weighted, and T_2-weighted. The normal brain images were obtained from the Brainweb database [1]. The images correspond to 1 mm slice thickness, 3 % noise (calculated relative to the brightest tissue), and 20 % intensity nonuniformity. Images of size 217×181 are available in 181 different z planes. Fuzzy-VGAPS was executed on seven of these z planes. The algorithm was applied to a particular z plane at a time to get the clusters and the cluster centers. The parameters of the Fuzzy-VGAPS algorithm were as follows: population size $= 20$, total number of generations $= 15$. The mutation and crossover probabilities were calculated adaptively. The number of clusters, K, was varied from 2 to 20. For the normal MR brain image, the ground truth information is available. There are a total of 10 classes present in the images, but the number of classes varies along the different z planes. The ten classes are background, CSF, grey matter, white matter, fat, muscle/skin, skin, skull, glial matter, and connective. Table 7.6 presents the actual number of clusters and the number of clusters automatically determined by the Fuzzy-VGAPS clustering technique (after application to the above-mentioned brain images projected on different z-planes). In order to measure the segmentation solution quantitatively, we also calculated the *Minkowski score* (MS) [146]. The MS scores obtained by Fuzzy-VGAPS clustering corresponding to the seven brain images are also reported in Table 7.6. For the purpose of comparison, the fuzzy C-means [36] and expectation maximization (EM)

Table 7.6 *Minkowski scores* (MS) obtained by the FCM, EM, and Fuzzy-VGAPS clustering algorithms on simulated MRI volumes for normal brain projected on different z planes. Here, #AC and #OC denote, respectively, the actual number of clusters and the automatically obtained number of clusters (after application of Fuzzy-VGAPS). Fuzzy-VGAPS is denoted by 'FVGAPS'. Smaller values of MS indicate better partitioning

z plane	#AC	MS for AC		#OC	MS for OC		
		FCM	EM		FCM	EM	FVGAPS
1	6	1.08	1.05	9	0.70	1.019	0.69
2	6	0.76	0.78	9	0.65	0.83	0.62
3	6	0.57	0.76	8	0.62	0.64	0.59
36	9	0.89	0.98	8	0.88	1.12	0.84
72	10	0.75	0.74	8	0.72	0.70	0.59
108	9	0.79	0.58	9	0.79	0.58	0.52
144	9	0.82	0.72	6	0.34	0.76	0.33

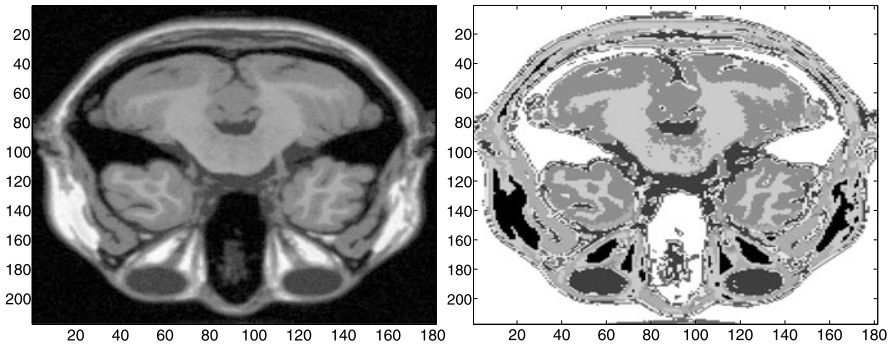


Fig. 7.8 (**a**) Original T_1-weighted MR image of the normal brain in z36 plane. (**b**) Segmentation obtained by Fuzzy-VGAPS clustering

[145] algorithms were also executed on the above-mentioned brain data sets with two different K values. In the first case, K was equal to the actual cluster number present in that particular plane. Next, it was set equal to that automatically determined by the Fuzzy-VGAPS algorithm. The corresponding MS scores of all the above runs using both algorithms are also reported in Table 7.6 for all the seven images. Results show that the MS score corresponding to the partitioning provided by Fuzzy-VGAPS clustering, in general, is the minimum among all the partitions. This indicates the superior performance of Fuzzy-VGAPS in automatically detecting the proper partitioning from the normal brain MR images. Figures 7.8(a) and 7.9(a) show the original normal brain images in the T_1 band in the z36 and z108 planes, respectively. Figures 7.8(b) and 7.9(b) show, respectively, the corresponding segmented images obtained after application of the Fuzzy-VGAPS clustering algorithm.

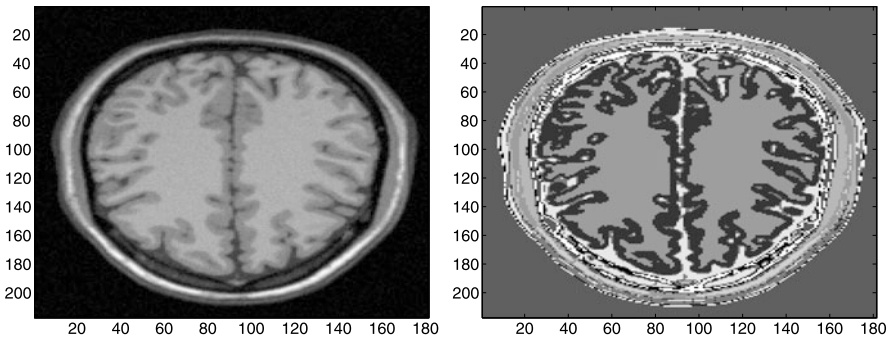


Fig. 7.9 (**a**) Original T_1-weighted MR image of the normal brain in z108 plane. (**b**) Segmentation obtained by Fuzzy-VGAPS clustering

Table 7.7 *Minkowski scores* (MS) obtained by the FCM, EM and Fuzzy-VGAPS clustering algorithms on Simulated MRI volumes for brain with multiple sclerosis lesions projected on different z planes. Here #AC and #OC denote, respectively, the actual number of clusters and the automatically obtained number of clusters (after application of Fuzzy-VGAPS). Fuzzy-VGAPS is denoted by 'FVGAPS'. Smaller values of MS indicate better partitioning

z plane	AC	MS for AC		OC	MS for OC		
		FCM	EM		FCM	EM	FVGAPS
1	6	0.59	0.59	10	0.74	0.66	0.58
2	6	0.75	0.76	10	0.75	0.67	0.58
5	6	0.74	0.76	8	0.69	0.77	0.62
36	9	0.99	1.01	9	0.99	1.01	0.81
72	11	0.72	0.63	11	0.72	0.63	0.62
108	10	0.78	0.58	9	0.79	0.70	0.57
144	9	0.31	0.76	10	0.80	0.81	0.31

Next, Fuzzy-VGAPS was executed on some simulated MR volumes for brain with multiple sclerosis lesions obtained from [1]. These images are again available in three modalities: T_1-weighted, proton density (p_d)-weighted, and T_2-weighted. These images also correspond to 1 mm slice thickness, 3 % noise (calculated relative to the brightest tissue), and 20 % intensity nonuniformity. Now, the images contain a total of 11 classes. These are background, CSF, grey matter, white matter, fat, muscle/skin, skin, skull, glial matter, connective, and MS lesion. However, the number of classes varies along the z planes. The image is available in 181 different z planes. The Fuzzy-VGAPS clustering algorithm was executed on the images projected on seven different z-planes. The parameters of the algorithm were the same as earlier. The ground truth information is available. The MS score [146] was calculated after application of the Fuzzy VGAPS-clustering technique in order to measure the 'goodness' of the solutions. Table 7.7 presents the actual number of clusters present in the image, the ob-

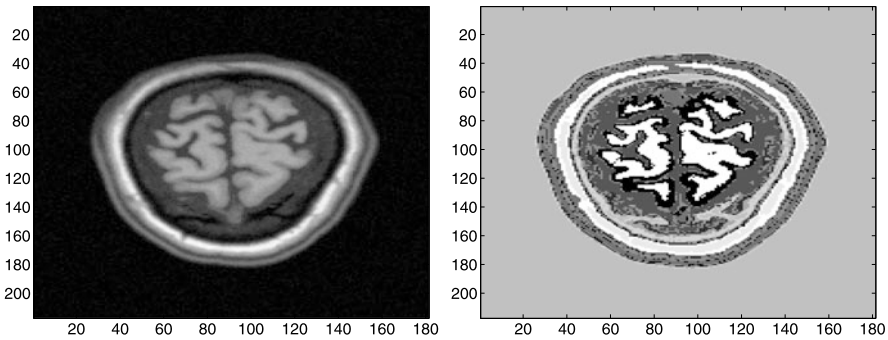


Fig. 7.10 (**a**) Original T_1-weighted MR image of the brain with multiple sclerosis lesions in the z144 plane: (**b**) Segmentation obtained by Fuzzy-VGAPS

tained number of clusters, and the 'goodness' of the corresponding partitioning in terms of the MS score after application of the Fuzzy-VGAPS clustering algorithm on these seven different MS lesion brain images. For the purpose of comparison, the fuzzy C-means algorithm [36] and EM algorithm [145] were again executed on these images, firstly with the number of clusters automatically determined by Fuzzy-VGAPS and then with the actual number of clusters present in the images. The MS scores of the corresponding partitionings are also provided in Table 7.7. The results again show that the MS score corresponding to the partitioning provided by the Fuzzy-VGAPS clustering is, in general, the minimum. This again reveals the effectiveness of the Fuzzy-VGAPS clustering in automatically segmenting the MR brain image with multiple sclerosis lesions. Figure 7.10(a) shows the original MS lesion brain image in the T_1 band projected on the z144 plane. Figure 7.10(b) shows the corresponding automatically segmented image obtained after application of the Fuzzy-VGAPS clustering algorithm. The calculated MS scores show that Fuzzy-VGAPS clustering performs well in segmenting both normal brain images and brain images with multiple sclerosis lesions.

Further experiments were carried out in order to establish the effectiveness of the symmetry-based distance in Fuzzy-VGAPS clustering. Fuzzy-VGAPS without adaptive crossover and mutation operators (crossover probability $= 0.9$, mutation probability $= 0.01$) was now executed on the first ten z planes of the MR brain images with multiple sclerosis lesions. The corresponding MS scores are provided in Table 7.8. The results were then compared with those obtained by Fuzzy-VGA [190] (fuzzy variable string length genetic algorithm), optimizing the popular Euclidean distance-based Xie-Beni (XB) index [295], where Euclidean distance is used to compute the membership values of different points to different clusters. The number of clusters and the MS scores obtained by Fuzzy-VGA after applying it on the first ten z planes of the MR brain images with multiple sclerosis lesions are also provided in Table 7.8. The results show that Fuzzy-VGAPS is more effective than Fuzzy-VGA. This establishes the fact that symmetry-based distance is more effective in segmenting the MR brain image than the existing Euclidean distance.

Table 7.8 The automatically obtained cluster (OC) number and the corresponding *Minkowski scores* (MS) after application of the Fuzzy-VGA and Fuzzy-VGAPS clustering algorithms on simulated MRI volumes for brain with multiple sclerosis lesions projected on the first ten z planes. Smaller values of MS indicate better partitioning

z plane No.	AC	Fuzzy-VGA		Fuzzy-VGAPS	
		OC	MS	OC	MS
1	6	2	1.21	10	0.58
2	6	2	1.20	10	0.58
3	6	2	1.19	7	0.71
4	6	5	0.69	5	0.67
5	6	2	1.184	8	0.62
6	6	2	1.18	3	0.71
7	6	2	1.17	8	0.70
8	6	2	1.16	9	0.71
9	6	2	1.16	9	0.68
10	9	2	1.17	9	0.65

7.8 Summary

Most clustering methods make prior assumptions about the structure of the clusters; For example, GCUK clustering, which is a genetic K-means technique for automatic determination of clusters, can only detect equisized hyperspherical clusters from a data set. In this chapter, a new point symmetry-based distance is utilized to develop a variable string length genetic clustering technique (VGAPS clustering) which automatically evolves the number of clusters present in a data set. The new cluster validity index, *Sym*-index, which is capable of detecting both the proper partitioning and the proper number of clusters present in a data set, is used as the fitness of the chromosomes. In VGAPS clustering, the assignment of points to different clusters is done based on the point symmetry distance rather than the Euclidean distance when the point is indeed symmetric with respect to a center. Moreover, the use of adaptive mutation and crossover probabilities helps VGAPS clustering to converge faster. Kd-tree-based nearest neighbor search is utilized to reduce the computational complexity of computing the point symmetry-based distance. The global convergence property of VGAPS-clustering is also established. The effectiveness of VGAPS clustering, as compared with two recently proposed automatic clustering techniques, namely GCUK clustering and HNGA clustering, is demonstrated on three artificially generated and two real-life data sets of different characteristics. Results on the five data sets establish the fact that VGAPS clustering is wellsuited to detect the number of clusters and the proper partitioning from data sets having clusters with widely varying characteristics, irrespective of their convexity, or overlap or size, as long as they possess the property of symmetry. VGAPS seeks clusters which are point symmetric with respect to their centers. Thus, VGAPS will fail if the clusters do not have this property. Based on these observations, and the fact that the property of symmetry is widely evident in real-life situations, application of VGAPS clustering to automatically determine the proper number of clusters and the proper partitioning from different data sets seems justified and is therefore recommended.

Thereafter in this chapter, a variable string length genetic point symmetry-based fuzzy clustering technique, Fuzzy-VGAPS, is described. It utilizes a new symmetry-based fuzzy cluster validity index, the *FSym*-index. The characteristic features of the Fuzzy-VGAPS clustering technique which distinguishes it from the state-of-the-art approaches are as follows: Use of variable string length GA allows encoding of a variable number of clusters. The *FSym*-index, used as the fitness function, provides the most appropriate partitioning even when the number of clusters, K, is varied. Moreover, clusters of any shape (e.g., hyperspherical, linear, ellipsoidal, ring shaped, etc.) and size (mixture of small and large clusters, i.e., clusters of unequal sizes) can be detected as long as they satisfy the property of point symmetry. Again, use of GA enables the algorithm to come out of local optima, a typical problem associated with local search methods such as the K-means and FCM. The effectiveness of the Fuzzy-VGAPS clustering technique as compared with seven different clustering techniques, namely the Fuzzy-VGA clustering algorithm, EAC-FCM clustering algorithm, K-means clustering technique with point symmetry-based distance (PSKM), fuzzy C-means clustering technique with point symmetry-based distance (PSFCM), genetic algorithm-based fuzzy C-means clustering technique, GMFCM, a hybrid centroid-medoid clustering technique, and a shape-based clustering technique, SPARCL, is shown for five data sets. Note that Fuzzy-VGAPS is not applicable for data sets not containing point-symmetric clusters (where the other algorithms will also fail). In a part of the experiment, a real-life application of Fuzzy-VGAPS to automatically segment magnetic resonance brain images with multiple sclerosis lesions is demonstrated.

Line symmetry and point symmetry are two important concepts of symmetry. Many objects around us contain some form of line symmetry, such as the human face, leaves of plants, etc. So far, we have discussed different clustering algorithms based on the point symmetry property. In the next chapter we discuss some recently developed line symmetry-based distances and the clustering techniques based on them.

Chapter 8
Some Line Symmetry Distance-Based Clustering Techniques

8.1 Introduction

In the last few chapters we have discussed some point symmetry-based clustering techniques. Both point symmetry and line symmetry are important aspects of geometrical symmetry [12]. Many objects around us contain the line symmetry property such as leaves of plants, the human face, human body, etc. Thus, it is necessary to develop some line symmetry-based distances to detect line-symmetrical clusters. Inspired by this, some line symmetry-based distances are developed in [61, 238, 246]. In Sect. 5.2, a line symmetry-based clustering technique proposed by Chung and Lin is discussed. In this chapter we describe in detail some newly developed line symmetry-based distances and the clustering algorithms based on them.

In order to determine the line symmetry-based distance, first the line of symmetry of a cluster has to be determined. Two approaches are discussed in this chapter to detect the line of symmetry of clusters. Thereafter, the definition of line symmetry-based distance is provided. The first approach to determining the line of symmetry is based on moment calculation. It is restricted to two-dimensional data sets. On the other hand, the second approach is based on principal component analysis (PCA). Thus, it is applicable for data sets of any number of dimensions.

The chapter is organized as follows: In the first part of the chapter, the moment-based approach for determining the line of symmetry is described, followed by a definition of the line symmetry-based distance. Thereafter, a genetic clustering technique based on this distance is described and some experimental results are provided. An application in the domain of human face detection is demonstrated. In the second part of this chapter the approach based on principal component analysis (PCA) is described in detail.

8.2 The Line Symmetry-Based Distance

In this section a newly developed line symmetry-based distance [238] is described.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_8, © Springer-Verlag Berlin Heidelberg 2013

8.2.1 Definition

Given a particular data set, the symmetrical line of each cluster is determined by using the central moment technique [61, 114]. Let the points in a particular cluster be denoted by $X = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Then, the (p, q)th order moment is calculated as [61]

$$m_{pq} = \sum_{\forall (x_i, y_i) \in X} x_i^p y_i^q. \tag{8.1}$$

Using Eq. 8.1, the centroid of the given cluster is calculated as $(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$. The central moment is defined as

$$u_{pq} = \sum_{\forall (x_i, y_i) \in X} (x_i - \overline{x})^p (y_i - \overline{y})^q, \tag{8.2}$$

where $\overline{x} = \frac{m_{10}}{m_{00}}$ and $\overline{y} = \frac{m_{01}}{m_{00}}$. Then the major axis of each cluster can be determined after considering the following two items [61]:

1. Centroid of a cluster should be on the major axis.
2. The angle between the major axis and the x axis is equal to $0.5 \times \tan^{-1}(\frac{2 \times u_{11}}{u_{20} - u_{02}})$.

Consequently, for a cluster, its major axis is expressed by $((\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}), 0.5 \times \tan^{-1}(\frac{2 \times u_{11}}{u_{20} - u_{02}}))$. The major axis can be considered as the symmetrical line of the relevant cluster. This symmetrical line can be used to measure the amount of line symmetry of a particular point in that cluster.

In order to measure the amount of line symmetry of a point $(\overline{x})$ with respect to a particular line i, $d_{ls}(\overline{x}, i)$, the following steps are followed:

1. For a particular data point $\overline{x}$, calculate the projected point $\overline{p}_i$ on the relevant symmetrical line i.
2. Find $d_{sym}(\overline{x}, \overline{p}_i)$ as

$$d_{sym}(\overline{x}, \overline{p}_i) = \frac{\sum_{i=1}^{knear} d_i}{knear}, \tag{8.3}$$

where *knear* unique nearest neighbors of $\overline{x}^* = 2 \times \overline{p}_i - \overline{x}$ are at Euclidean distances d_i, $i = 1, 2, \ldots, knear$. The ANN library [202] utilizing Kd-tree-based nearest neighbor search is used to reduce the complexity of computing these d_i (as described in Sect. 5.5). The amount of line symmetry of a point $\overline{x}$ with respect to the symmetrical line of cluster i is calculated as

$$d_{ls}(\overline{x}, i) = d_{sym}(\overline{x}, \overline{p}_i) \times d_e(\overline{x}, \overline{c}), \tag{8.4}$$

where $\overline{c}$ is the centroid of cluster i, i.e., $\overline{c} = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$.

The above line symmetry-based distance is realized more clearly from Fig. 8.1. Here, let $\overline{x}$ be a particular data point, and $\overline{c}$ the center of a particular cluster. The symmetrical line of the given cluster is denoted by *Line l*. The projected point of $\overline{x}$

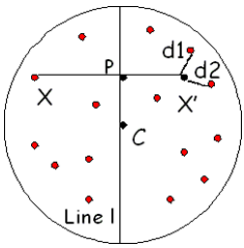


Fig. 8.1 Example of line symmetry-based distance

on this symmetrical line is denoted by $\overline{p}$. Let the reflected point of $\overline{x}$ with respect to $\overline{p}$ be denoted by $\overline{x}'$. The first two nearest neighbors (here *knear* is chosen equal to 2) of $\overline{x}'$ are at Euclidean distances of d_1 and d_2, respectively. Then, the total amount of symmetry of $\overline{x}$ with respect to the projected point $\overline{p}$ on *Line* l is calculated as follows: $d_{sym}(\overline{x}, \overline{p}) = \frac{d_1+d_2}{2}$. Therefore, the total line symmetry-based distance of point $\overline{x}$ with respect to the symmetrical line of cluster l is calculated as $d_{ls}(\overline{x}, l) = d_{sym}(\overline{x}, \overline{p}) \times d_e(\overline{x}, \overline{c})$, where $d_e(\overline{x}, \overline{c})$ is the Euclidean distance between the point $\overline{x}$ and the cluster center $\overline{c}$.

8.3 GALSD: The Genetic Clustering Scheme with Line Symmetry-Based Distance

In this section, a genetic clustering scheme along the lines of the genetic clustering technique with point symmetry-based distance (GAPS) [27] is described. Unlike GAPS, here the above-described line symmetry based distance [238] is used for cluster assignment and for calculation of the fitness function. A brief overview of the basic steps of GALSD are enumerated below. Given a particular value of the number of clusters, K, GALSD partitions the data into K line-symmetrical clusters.

8.3.1 String Representation and Population Initialization

Here, center-based encoding of the chromosome as done in GAPS (refer to Chap. 5) is used.

8.3.2 Fitness Computation

In order to compute the fitness of the chromosomes, the following steps are executed:

1. Find the symmetrical line for each cluster. As described in the first paragraph of Sect. 8.2.1, for each cluster, the moment-based approach is used to find the relevant symmetrical line.
2. For each data point $\overline{x}_i$, $i = 1, \ldots, N$, where N is the total number of points present in the data set, compute $d_{ls}(\overline{x}_i, k)$ (the line symmetric distance between the point $\overline{x}_i$ and the symmetrical line of cluster k) by Eq. 8.4, $k = 1, \ldots, K$, where K is the total number of clusters.
3. The point $\overline{x}_i$ is assigned to cluster k if and only if $d_{ls}(\overline{x}_i, k) \leq d_{ls}(\overline{x}_i, j)$, $j = 1, \ldots, K$, $j \neq k$ and $(d_{ls}(\overline{x}_i, k)/d_e(\overline{x}_i, \overline{c}_k)) \leq \theta$. For $(d_{ls}(\overline{x}_i, k)/d_e(\overline{x}_i, \overline{c}_k)) > \theta$, point $\overline{x}_i$ is assigned to some cluster m iff $d_e(\overline{x}_i, \overline{c}_m) \leq d_e(\overline{x}_i, \overline{c}_j)$, $j = 1, 2 \ldots, K$, $j \neq m$. In other words, point $\overline{x}_i$ is assigned to that cluster with respect to whose symmetrical line its LS distance is minimum, provided the total "symmetricity" with respect to it is less than some threshold θ. Otherwise, assignment is done based on the minimum Euclidean distance criterion as normally used in [188] or the K-means algorithm.

Here, the threshold θ is kept equal to d_{NN}^{max} (as described in Sect. 5.6.2).

After the assignments are done, the cluster centers encoded in the chromosome are replaced by the mean points of the respective clusters. Subsequently, for each chromosome the *clustering_metric*, M, is calculated as

$$M = \sum_{i=1}^{K} \sum_{j=1}^{N_i} d_{ls}\left(\overline{x}_i^j, i\right),$$

where N_i denotes the number of points in the ith cluster, and $\overline{x}_i^j$ denotes the jth point of the ith cluster. Then, the fitness function of that chromosome, *fit*, is defined as the inverse of M, i.e., $fit = \frac{1}{M}$. This fitness function, *fit*, will be maximized by using a genetic algorithm. (Note that there could be other ways of defining the fitness function.)

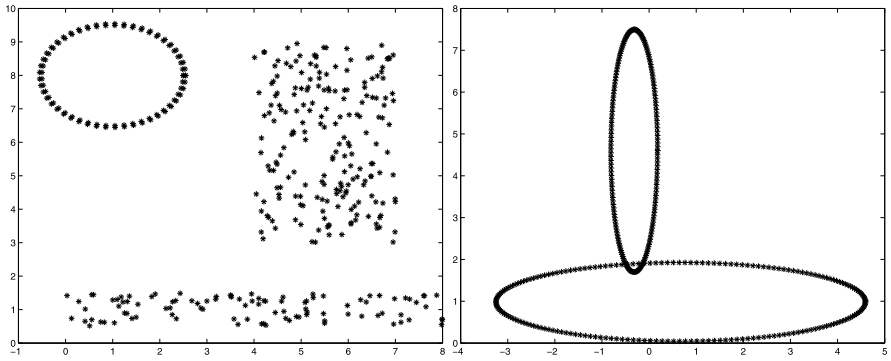
8.3.3 Genetic Operators

Here, genetic operators similar to those used in GAPS are used.

8.4 Experimental Results

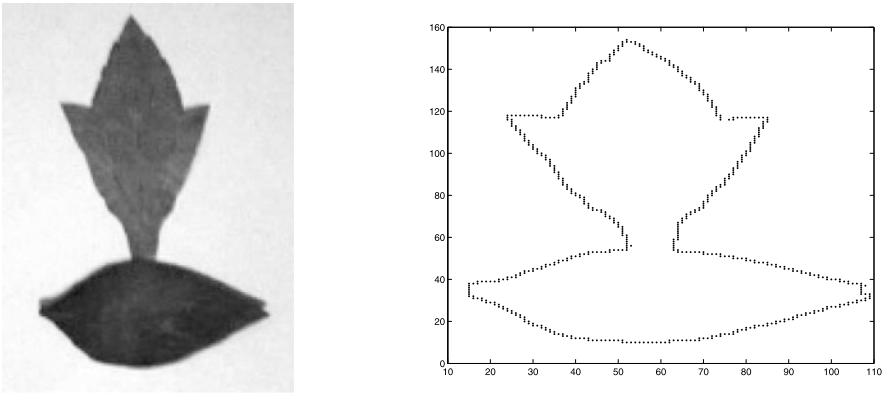
8.4.1 Data Sets Used

1. *Sym_3_2*: This data set is described in Sect. 5.8.1.
2. *Rect_3_2*: This data set, containing 400 points, is a combination of a ring-shaped cluster, a rectangular cluster and a linear cluster, as shown in Fig. 8.2(a).



(a) (b)

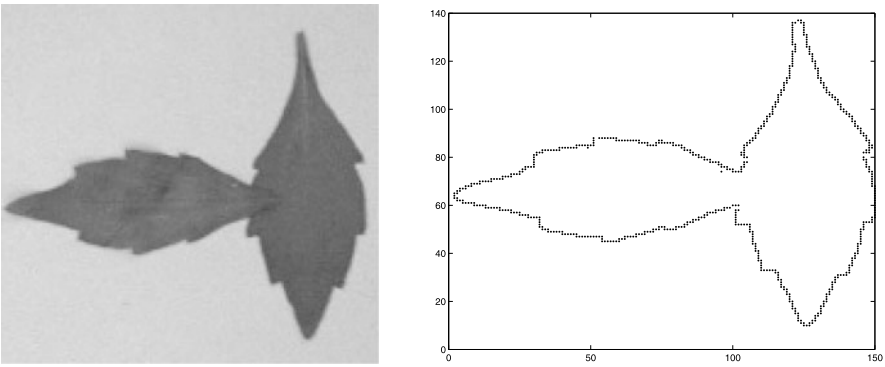
Fig. 8.2 (**a**) *Rect_3_2*. (**b**) *Ellip_2_2*



(a) (b)

Fig. 8.3 (**a**) *Two_leaves1* data. (**b**) Edge pixels of leaves as input data points

3. *Ellip_2_2*: This data set, used in Ref. [28], contains 400 points distributed on two crossed ellipsoidal shells as shown in Fig. 8.2(b).
4. *Two_leaves1*: The line-symmetry property is evident in many natural scenes. Leaves of plants are important examples of this. Figure 8.3(a) shows two real leaves of *Ficus microcapa* overlapping slightly with each other. First, the Sobel edge detector [114] is used to obtain the edge pixels in the input data points, as shown in Fig. 8.3(b).
5. *Two_leaves2*: Figure 8.4(a) shows two real leaves of *Erechitites valerianifolia*. The edge map obtained after application of the Sobel edge detector is shown in Fig. 8.4(b).



(a) (b)

Fig. 8.4 (**a**) *Two_leaves2* data. (**b**) Edge pixels of leaves as input data points

8.4.2 Discussion of Results

The experimental results comparing the performance of the GALSD, GAPS, and K-means clustering algorithms are provided for three artificial and two real-life data sets. For the GALSD clustering technique, value of θ is determined from the data set as discussed in Sect. 8.3.2. For both the GALSD and GAPS [27] clustering algorithms, the crossover probability, μ_c, and the mutation probability, μ_m, are determined adaptively as described in Sects. 5.6.4 and 5.6.5, respectively. The population size, P, is set equal to 100. The total number of generations is kept equal to 30; Executing them further did not improve performance.

1. *Sym_3_2*: The final clustering results obtained after application of K-means, GAPS, and GALSD for this data set are shown in Figs. 8.5(a), 8.5(b), and 8.5(c), respectively, where K-means is found to fail in providing the proper partitioning. The performance of both the GAPS and GALSD clustering algorithms is similar for this data set.
2. *Rect_3_2*: The final clustering results corresponding to the K-means, GAPS, and GALSD clustering algorithms for this data set are shown in Figs. 8.6(a), 8.6(b), and 8.6(c), respectively. As is evident from Fig. 8.6(a), K-means fails to correctly detect the linear cluster; it includes points from the rectangular cluster in the linear cluster. As expected, the ring is properly detected. The GALSD clustering algorithm is able to detect the proper partitioning, similarly to GAPS.
3. *Ellip_2_2*: This is a nonconvex symmetrical data set. The final clustering results corresponding to the K-means, GAPS, and GALSD clustering algorithms for this data set are shown in Figs. 8.7(a), 8.7(b), and 8.7(c), respectively. As expected, K-means is not able to detect the proper partitioning, but GALSD and GAPS are able to do so.
4. *Two_leaves1*: After running the K-means algorithm, the obtained partitioning is shown in Fig. 8.8(a). The clustering results obtained after execution of GAPS and GALSD are shown in Figs. 8.8(b) and 8.8(c), respectively. Both K-means

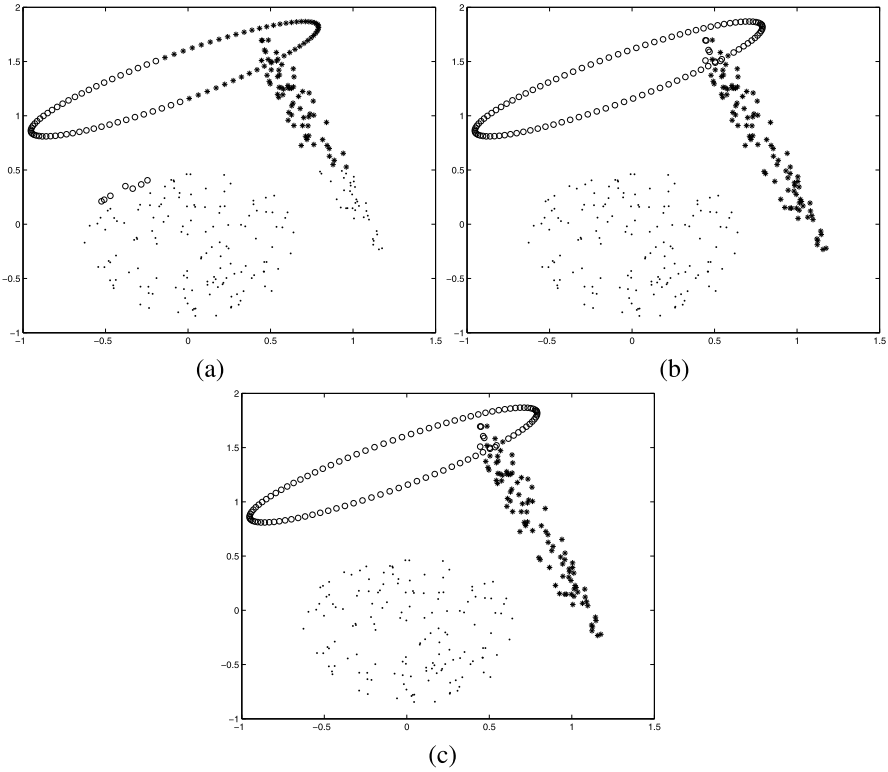


Fig. 8.5 Clustered *Sym_3_2* for $K = 3$ after application of the (**a**) K-means clustering algorithm, (**b**) GAPS clustering technique, and (**c**) GALSD clustering algorithm

and GAPS fail to detect the proper partitioning from this data set, but GALSD demonstrates a satisfactory clustering result. This shows the superiority of the line symmetry-based clustering technique over the existing GAPS.

5. *Two_leaves2*: Figures 8.9(a), 8.9(b), and 8.9(c) show the final clustering results obtained after application of the K-means, GAPS, and GALSD clustering algorithms, respectively. Again for this data set, both K-means and GAPS do not perform well, but the GALSD clustering technique illustrates a satisfactory clustering result.

8.4.3 Computation Time

The execution time taken by the three clustering techniques, K-means, GAPS, and GALSD, for all the data sets were provided in Table 8.1. All the algorithms were executed on an HP xw8400 Workstation with dual-core 3.0-GHz Intel Xeon processors, 4 MB cache memory, and 2 GB primary memory. Results show that K-means is the fastest, followed by GAPS and GALSD.

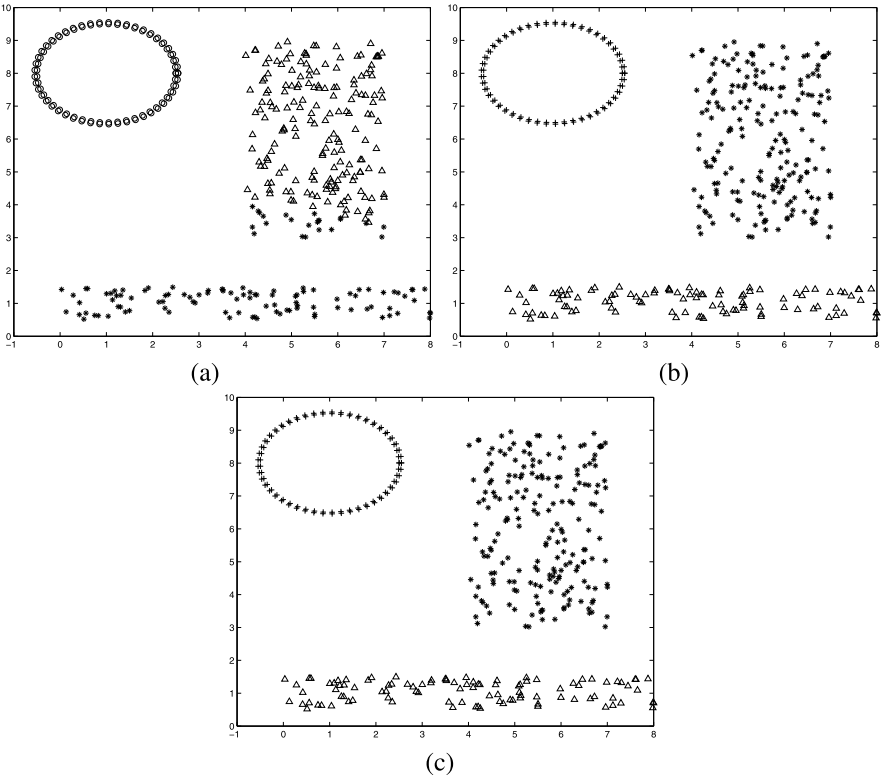


Fig. 8.6 Clustered *Rect_3_2* for $K = 3$ after application of the (**a**) K-means clustering algorithm, (**b**) GAPS clustering technique, and (**c**) GALSD clustering algorithm

Table 8.1 Time taken (in seconds) by three clustering techniques for all data sets

Index	Time taken (seconds)		
	K-means	GAPS	GALSD
Sym_3_2	15	28	50
Rect_3_2	17	27	50
Ellip_2_2	13	26	42
Two_leaves1	16	42	51
Two_leaves2	15	40	51

8.5 Application to Face Recognition

The problem of human face recognition has become very popular in recent years [124]. A good overview of algorithms developed to solve this problem can be found in [306]. There are wide applications of face recognition systems, including secure access control and financial transactions. Human face detection is the first important

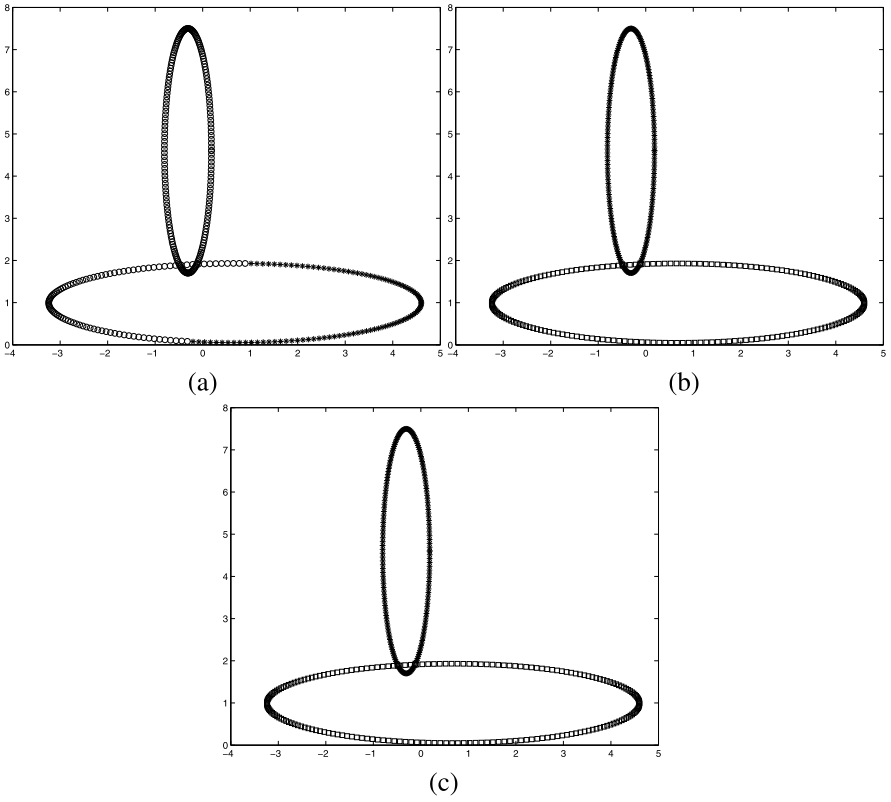


Fig. 8.7 Clustered *Ellip_2_2* for $K = 2$ after application of the (**a**) K-means clustering algorithm, (**b**) GAPS clustering technique, and (**c**) GALSD clustering algorithm

step of fully automatic human face recognition. Face detection is the technique to automatically determine the locations and sizes of faces in an input image. In this section an efficient method to locate human faces in a complex background is described. Here, the symmetry property of human faces is utilized to quickly locate the candidate faces. The line symmetry-based distance [238] is used for this purpose.

8.5.1 Human Face Detection Algorithm

This human face detection algorithm uses the line symmetry property of human faces to quickly detect a candidate face from some background. Here, it is assumed that the human face can be approximated by an ellipse. Thus, the goal of human face detection technique is to quickly locate elliptical-shaped objects in an image. This is not an easy task, as the sizes and orientations of the objects of interest may vary a lot. In order to effectively solve the problem, the following five-step procedure is used (the following procedure is very much similar to the method developed in [265]):

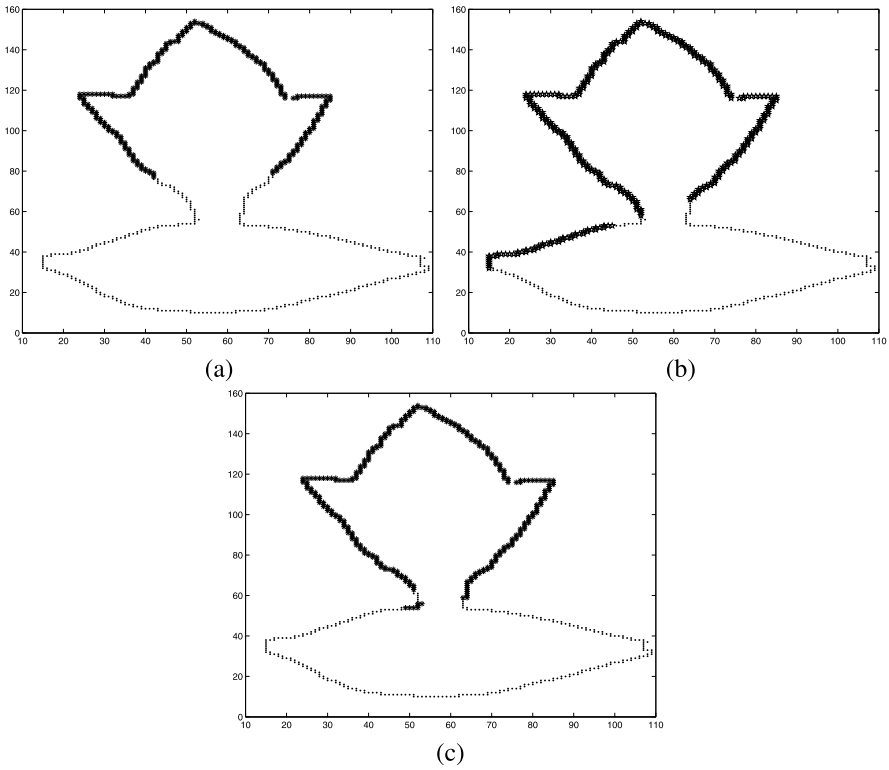


Fig. 8.8 Clustered *Two_leaves1* data for $K = 2$ after application of the (**a**) K-means clustering algorithm, (**b**) GAPS clustering algorithm, and (c) GALSD clustering algorithm

1. First the Sobel edge detection operator [114] is used to find the edge image.
2. Short segments that contain fewer than five pixels determined by the Sobel edge detection operator are deleted. This step is executed by using connected component analysis. The algorithm for finding 8-connected components of an image is found in [114]. Here, first the 8-connected components of the edge image are found and then the connected components having fewer than five pixels are removed from the edge image.
3. A $w_1 \times w_2$ window is used to scan the processed edge image from top to bottom and from left to right. For each window, first compute its major axis. Then the above-mentioned d_{ls} distance is used to measure the degree of symmetry of an edge pixel within the window relative to the major axis of the window. Here, corresponding to each window one counter, $count_w$, is maintained. For a particular window w, where there are a total of N_w number of edge pixels, for each edge pixel $\overline{x}_i$, $1 \leq i \leq N_w$ of that particular window w, calculate $d_{ls}(\overline{x}_i, w)$ (by Eq. 8.4). If $d_{ls}(\overline{x}_i, w) < \theta$, then $count_w$ is increased by 1. Therefore, the final value of $count_w$ represents the number of symmetrical points in that window.

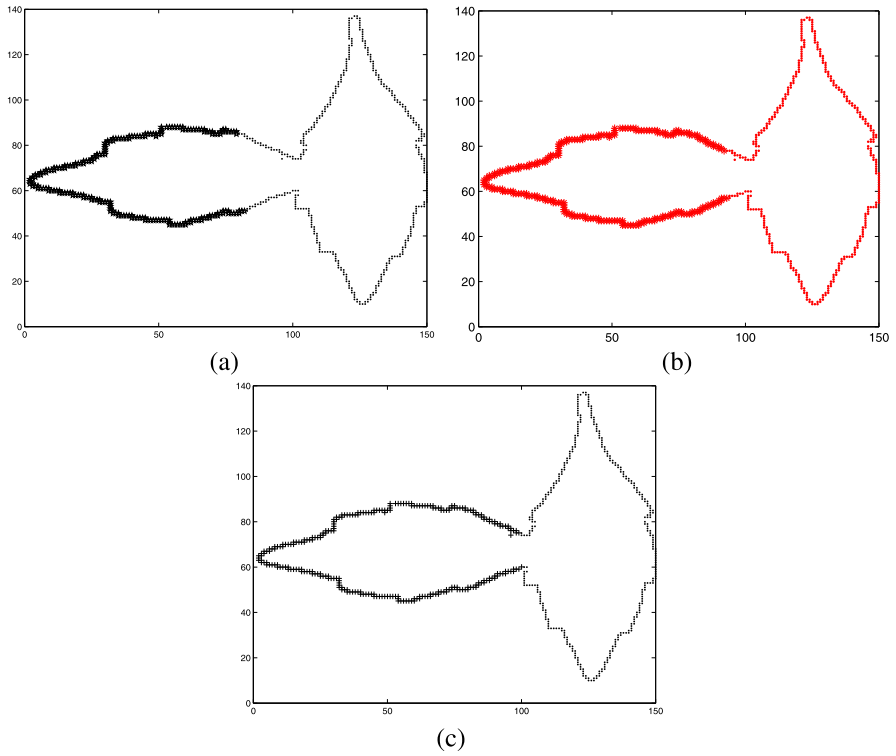


Fig. 8.9 Clustered *Two_leaves2* data for $K = 2$ after application of the (**a**) K-means clustering algorithm, (**b**) GAPS clustering algorithm, and (**c**) GALSD clustering algorithm

The threshold value θ is kept equal to the maximum nearest neighbor distance in the data set (as illustrated in Sect. 8.3.2).

4. The values of $count_w$ are sorted in decreasing order. Then, the window with the largest $count_w$ locates the region most likely to contain a face.

8.5.2 Experimental Results

The face detection algorithm was tested on a face database collected from CMU (http://www.ius.cs.cmu.edu/IUS/usrp0/har/FaceDemo/gallery-inline.html). Some examples of how the face detection algorithm works to detect faces from some images are shown in Figs. 8.10–8.14. Figures 8.10(a), 8.11(a), 8.12(a), 8.13(a), and 8.14(a) show, respectively, the five original images to which the algorithm is applied. Figures 8.10(b), 8.11(b), 8.12(b), 8.13(b), and 8.14(b) show, respectively, the five edge images after application of the Sobel edge detection operator. Figures 8.10(c), 8.11(c), 8.12(c), 8.13(c), and 8.14(c) show, respectively, the human faces detected from the corresponding images after application of this method. Note

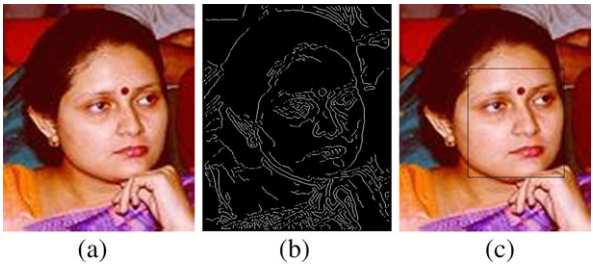


Fig. 8.10 (**a**) Original image. (**b**) The result after applying the Sobel filter. (**c**) The detected face region

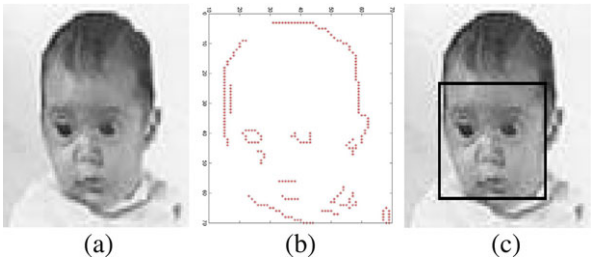


Fig. 8.11 (**a**) Original image of a baby's face. (**b**) The result after applying the Sobel filter. (**c**) The detected face region

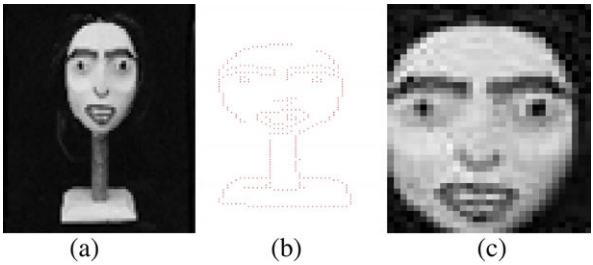


Fig. 8.12 (**a**) Original image of the face of a puppet. (**b**) The result after applying the Sobel filter. (**c**) The detected face region

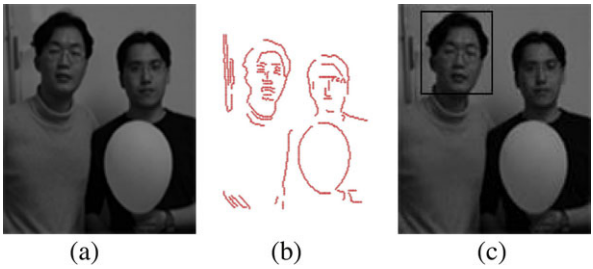


Fig. 8.13 (**a**) Original image of two human faces and a balloon. (**b**) The result after applying the Sobel filter. (**c**) The window having largest symmetrical points, e.g., the detected face region

that the present work can only detect one face from a particular image. For that reason, from Fig. 8.13(a), where there are two human faces in the image, the present method is only able to detect one single human face. However, the present method can be easily extended to detect multiple faces from an image by keeping track of the windows having the second highest number of symmetrical pairs, etc. The size of the window is determined adaptively depending on the considered image. Note that, in Fig. 8.13(a), a balloon is present, which is of elliptical shaped. However, this

(a) (b) (c)

Fig. 8.14 (**a**) Original image of a human face. (**b**) The result after applying the Sobel filter. (**c**) The detected face region

algorithm is still able to detect the window containing the face because the number of symmetrical points present in the window containing the human face is much larger than that present in the window containing the balloon.

8.6 A Generalized Line Symmetry-Based Distance and Its Application to Data Clustering

The above-discussed line symmetry-based distance has several drawbacks. The major shortcoming is that its application is limited to two-dimensional data sets only. Thus, some line symmetry-based distance needs to be defined which can detect clusters having the line symmetry property in data sets having any number of dimensions. The line symmetry-based distance discussed next [246] calculates the amount of symmetry of a point with respect to the first principal axis of the data points in a cluster. Principal component analysis (PCA) [148] is used for this purpose. The data set has a maximum amount of variation along the first principal axis. In this clustering technique a particular point is assigned to that cluster with respect to whose principal axis its line symmetry-based distance is minimum. Thus, it can detect clusters of any shape which are symmetric with respect to the principal axis. A genetic algorithm has been used as the underlying search technique. The GA with line symmetry distance-based clustering technique (GALS) [246] is able to detect both convex and nonconvex clusters of any shape and sizes as long as the clusters have some line symmetry property. A Kd-tree-based nearest neighbor search is utilized to reduce the computational complexity of computing the line symmetry-based distance. The effectiveness of the algorithm is demonstrated in identifying line-symmetric clusters from two artificial and four real-life datasets of varying characteristics. The clustering results are compared with those obtained by the well-known GAK-means clustering algorithm [188] and average linkage clustering technique.

A Generalized Line Symmetry-Based Distance Given a particular data set, the first principal axis of this data set is found using principal component analysis [148]. Let the eigenvector of the covariance matrix of the data set with the highest eigenvalue be $[eg_1\ eg_2\ eg_3\ eg_4\ \ldots\ eg_d]$, where d is the dimension of the original data. Then, the first principal axis of the data set is given by

$$\frac{(x_1 - c_1)}{eg_1} = \frac{(x_2 - c_2)}{eg_2} = \cdots = \frac{(x_d - c_d)}{eg_d},$$

where the center of the data set is $\overline{c} = \{c_1, c_2, \ldots, c_d\}$.

The obtained principal axis is treated as the symmetrical line of the relevant cluster; i.e., if the data set is indeed symmetrical, then it should also be symmetric with respect to the first principal axis of the dataset identified by principal component analysis (PCA) . This symmetrical line is used to measure the amount of line symmetry of a particular point in that cluster [246]. In order to measure the amount of line symmetry of a point $(\overline{x})$ with respect to a particular line i, $d_{ls}(\overline{x}, i)$, the steps described in Sect. 8.2.1 are followed. A genetic clustering technique, GALS, is thereafter developed which mimics the steps of GALSD but uses principal component analysis to detect the symmetrical line of each cluster.

8.7 Implementation Results

The experimental results comparing the performance of the GALS, GAK-means, and average linkage clustering techniques are provided for two artificial and four real-life data sets. For average linkage, source code was obtained from http://bioinformatics.oxfordjournals.org/cgi/content/abstract. For both genetic clustering techniques, GAK-means and GALS, the following parameter values were kept: population size = 100, number of generations = 30. For GAK-means, the probability of crossover was kept equal to 0.8, whereas the probability of mutation was kept equal to 0.1. For GALS, the crossover and mutation probabilities were calculated adaptively. In order to measure the goodness of the partitioning obtained by all the three clustering techniques for all artificial and real-life data sets, the *Minkowski score* (MS) [146] was calculated. These are reported in Table 8.2.

Table 8.2 *Minkowski scores* (MS) obtained by the GAK-means, GALS, and average linkage (AL) clustering techniques for all the data sets. Here 'n', 'd', and 'K' denote, respectively, the number of data points, dimension of the data, the number of clusters present in the data set

Data set	n	d	K	*Minkowski score*		
				GAK-means	GALS	AL
Line_2_2	400	2	2	0.83	**0.39**	0.85
AD_4_3	400	3	4	**0.00**	**0.00**	**0.00**
Two_leaves1	580	2	2	0.65	**0.00**	0.54
Two_leaves2	657	2	2	0.69	**0.19**	0.28
Iris	150	4	3	0.62	**0.60**	0.70
Cancer	683	9	2	0.36	**0.35**	0.45

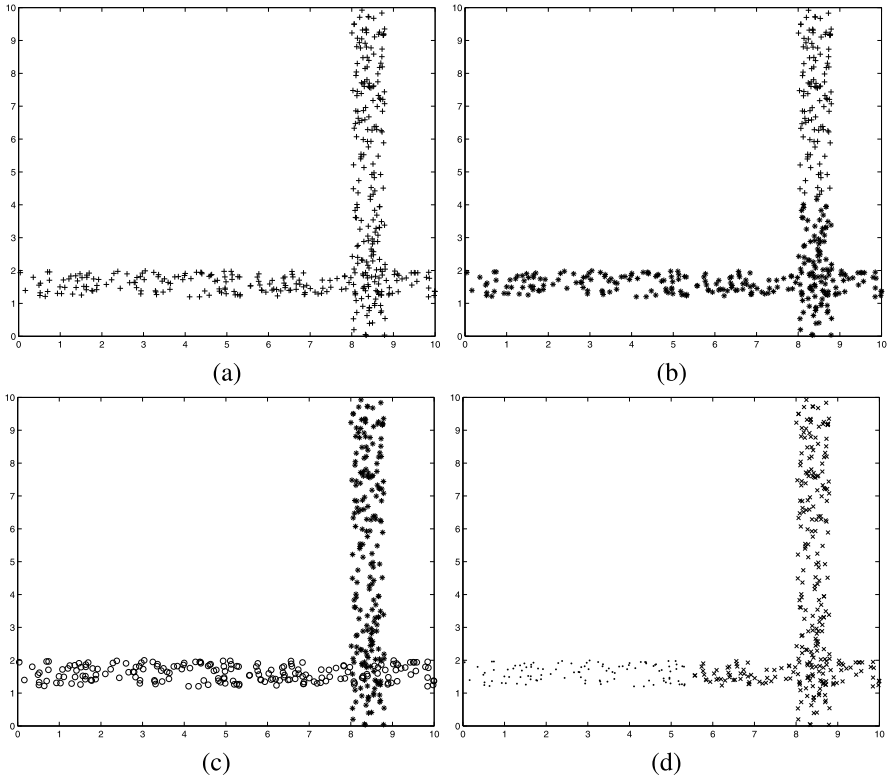


Fig. 8.15 (**a**) *Line_2_2*. (**b**) Partitioning obtained by the GAK-means algorithm for $K = 2$. (**c**) Partitioning obtained by the GALS clustering algorithm for $K = 2$. (**d**) Partitioning obtained by the average linkage clustering algorithm for $K = 2$

8.7.1 Data Sets Used

1. *Line_2_2*: This data set, used in [27], consists of two bands as shown in Fig. 8.15(a), where each band consists of 200 data points.
2. *AD_4_3*: This data set consists of 400 data points in three dimensional space distributed over four hyperspherical disjoint clusters where each cluster contains 100 data points. This data set is shown in Fig. 8.16(a).
3. *Two_leaves1*: This data set is described in Sect. 8.4.1.
4. *Two_leaves2*: This data set is described in Sect. 8.4.1.
5. *Iris*: This data set is described in Sect. 5.8.1.
6. *Cancer*: This data set is described in Sect. 5.8.1.

8.7.2 Discussion of Results

1. *Line_2_2*: The final clustering results obtained by the GAK-means, GALS, and average linkage clustering techniques are provided in Figs. 8.15(b), 8.15(c),

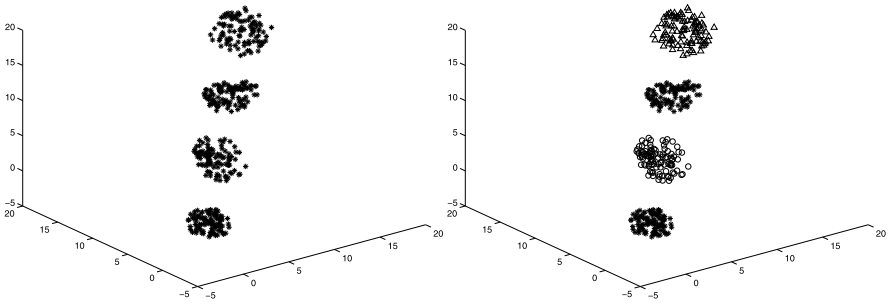


Fig. 8.16 (**a**) *AD_4_3*. (**b**) Partitioning obtained by the GAK-means/GALS/average linkage algorithm for $K = 4$

and 8.15(d), respectively. As expected, both GAK-means and average linkage perform poorly for this data. GAK-means fails since the clusters are not hyperspherical in nature. Average linkage fails because there are overlaps among the clusters. GALS is able to detect the proper partitioning from this data set as the clusters possess the line symmetry property. Table 8.2 reveals that the *Minkowski score* corresponding to the partitioning obtained by GALS is the minimum. This again establishes the superior performance of GALS for this data set.

2. *AD_4_3*: The final clustering results corresponding to the GAK-means, GALS, and average linkage clustering techniques are shown in Fig. 8.16(b). All three clustering techniques are able to detect the proper partitioning from this data set. The corresponding MS value is also the optimum (refer to Table 8.2).
3. *Two_leaves1*: The obtained partitionings after execution of the GAK-means, GALS, and average linkage clustering techniques are shown in Figs. 8.17(c), 8.17(d), and 8.17(e), respectively. GALS again demonstrates a satisfactory clustering result. This is again supported by the minimum MS score attained by the GALS clustering technique (refer to Table 8.2).
4. *Two_leaves2*: The obtained partitionings after execution of the GAK-means, GALS, and average linkage clustering techniques are shown in Figs. 8.18(c), 8.18(d), and 8.18(e), respectively. GALS again demonstrates a satisfactory clustering result.
5. *Iris*: As this is a higher dimensional data set, no visualization is possible. The GALS clustering technique obtains a slightly smaller MS score for this data set (refer to Table 8.2).
6. *Cancer*: Again for this data set, the performance of GALS clustering is slightly better than that of the GAK-means and average linkage clustering techniques in terms of the obtained MS scores (refer to Table 8.2).

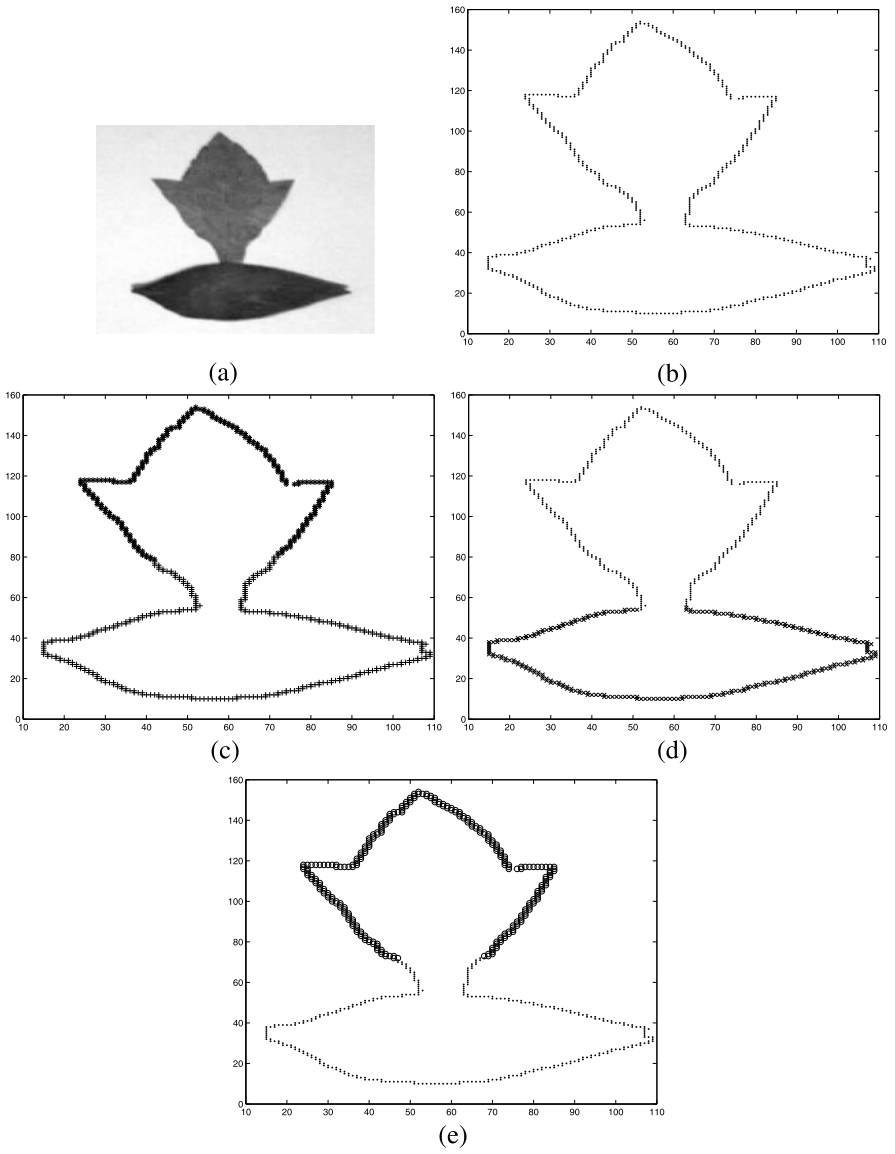


Fig. 8.17 (**a**) *Two_leaves1* data. (**b**) Edge pixels of leaves as input data points. (**c**) Partitioning obtained by GAK-means for $K = 2$. (**d**) Partitioning obtained by GALS for $K = 2$. (**e**) Partitioning obtained by the average linkage clustering technique for $K = 2$

Summary of Results The results on two artificial and four real-life data sets having clusters with different characteristics establish that GALS can detect clusters having any size, shape or convexity as long as they possess the line symmetry prop-

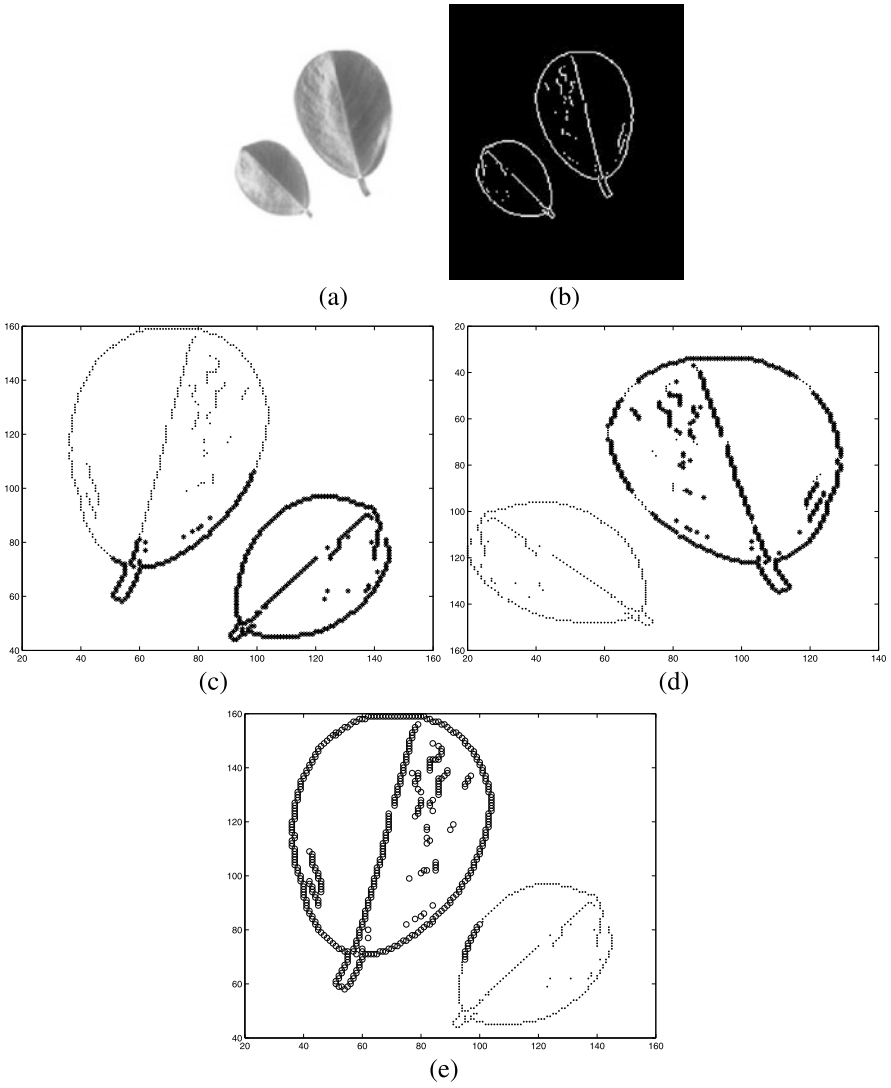


Fig. 8.18 (**a**) *Two_leaves2* data. (**b**) Edge pixels of leaves as input data points. (**c**) Partitioning obtained by the GAK-means clustering algorithm for $K = 2$. (**d**) Partitioning obtained by the GALS clustering algorithm for $K = 2$. (**e**) Partitioning obtained by the average linkage clustering technique for $K = 2$

erty. Results show that, while GAK-means is able to detect only hyperspherical-shaped clusters, average linkage is able to detect clusters having well-separated structures. As the property of line symmetry is evident in many real-life situations, use of GALS for data clustering is highly recommended.

8.8 Discussion and Conclusions

In this chapter some line symmetry-based distance functions are described based on the existing point symmetry-based distance developed in Ref. [27]. The first distance uses a moment-based approach and is applicable only for two-dimensional data sets. Kd-tree-based nearest neighbor search is used to reduce the complexity of computing the line symmetry-based distances. Thereafter, a genetic clustering technique along the lines of GAPS clustering is described, which incorporates the line symmetry-based distance for performing cluster assignment of the points and in the fitness computation. The GALSD clustering technique [238] can cluster data sets with the property of line symmetry successfully. The effectiveness of this algorithm is demonstrated in detecting clusters having the line symmetry property from three artificial and two real-life data sets. As a real-life application, GALSD is also used for locating human faces in an image.

Thereafter, in this chapter a second line symmetry-based distance is described which measures the total amount of symmetry of a point with respect to the first principal axis of a cluster. It is applicable for data sets irrespective of the number of dimensions. A genetic clustering technique (GALS) using this line symmetry based distance is also described. The effectiveness of this algorithm is demonstrated in detecting clusters having the line symmetry property from two artificial and four real-life data sets. Results are compared with those obtained by the GAK-means algorithm and average linkage clustering technique.

So far we have discussed different clustering techniques based on either point symmetry or line symmetry. All these techniques are able to optimize only one objective function. However, in real-life a particular data set may consist of several different types of clusters; thus, clustering techniques optimizing a single cluster objective function are not able to determine the appropriate partitioning. Thus, it was required to design some clustering techniques which can optimize more than one objective function each capturing different data properties, simultaneously so that these can detect clusters of different shapes from a given data set. The next chapter deals with such multiobjective clustering techniques.

Chapter 9
Use of Multiobjective Optimization for Data Clustering

9.1 Introduction

Clustering is considered to be a difficult task as no unambiguous partitioning of the data exists for many data sets. Most of the existing clustering techniques are based on only one criterion which reflects a single measure of goodness of a partitioning. However, a single cluster quality measure is seldom equally applicable for different kinds of data sets with different characteristics. Hence, it may become necessary to simultaneously optimize several cluster quality measures that can capture different data characteristics. In order to achieve this, the problem of clustering a data set has been posed as one of multiobjective optimization in literature.

In [121], a multiobjective clustering technique named multiobjective clustering for automatic K-determination (MOCK) is developed which uses a multiobjective evolutionary algorithm (MOEA) to perform the optimization. PESA-II [65], a well-known multiobjective evolutionary algorithm, is used as the underlying optimization technique. For the encoding, the locus-based adjacency representation proposed in [218] is used. In this graph-based representation, each individual g consists of N genes $g_1, \ldots, g_N$, where N is the size of the clustered data set, and each gene g_i can take allele values j in the range $\{1, \ldots, N\}$. Thus, a value of j assigned to the ith gene is then interpreted as a link between data items i and j; in the resultant clustering solution they will be in the same cluster. The decoding requires the identification of all subgraphs. All data items assigned to the same subcluster are assigned the same cluster number. MOCK uses two complementary objective functions to be optimized simultaneously. The first objective function is the *overall deviation* of a partitioning. This is simply the overall summed distances between data items and their corresponding cluster centers

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \overline{c}_k), \tag{9.1}$$

where C is the set of all clusters, $\overline{c}_k$ is the center of cluster C_k, and $\delta(i, \overline{c}_k)$ is the chosen distance function. This overall deviation should be minimized. This is strongly biased towards spherically shaped clusters.

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2_9,

The second objective function measures the degree of connectivity between the clusters. It calculates the degree to which neighboring data points have been placed in the same cluster. It is computed as

$$Conn(C) = \sum_{i=1}^{N}\left(\sum_{j=1}^{L} x_{i,nn_i(j)}\right), \quad \text{where } x_{r,s} = \frac{1}{j} \text{ if } \nexists C_k : r, s \in C_k \tag{9.2}$$

$$= 0, \quad \text{otherwise.} \tag{9.3}$$

$nn_i(j)$ is the jth nearest neighbor of data i, and L is a parameter determining the number of neighbors that contribute to the connectivity measure. Connectivity should be minimized for achieving the proper clustering. Uniform crossover [112] is used in favor of one- or two-point crossover. A restricted *nearest neighbor mutation* where each data item can only be linked to its L nearest neighbors is used here. Here, $g_i \in \{nn_{i1}, \ldots, nn_{iL}\}$, where nn_{il} denotes the lth nearest neighbor of data item i. To the end, an approach inspired from Tibshirani's Gap statistic [280] is used to select a single solution from the final Pareto-optimal front.

An EA for MOO clustering is proposed in [165]. Here, two objectives are minimized simultaneously. These are the total intracluster variation (computed over all the clusters) and the number of clusters. These two objectives are conflicting with each other. Using MOO, the EA provides a set of nondominated solutions. Here, for each different number of clusters, the algorithm manages to provide the smallest possible total intracluster variance. Users can then make a more informed choice about the solution to be used in practice.

A multiobjective evolutionary algorithm for fuzzy clustering has been proposed in [21]. Here again, two objectives are simultaneously optimized. The first one is the objective function optimized in fuzzy C-means algorithm [37], and the other is the well-known Xie-Beni index [295]. This is defined as a ratio between a global measure of intracluster variation and a local measure of cluster separation. The separation between clusters is measured using the distance between the two closest clusters. Although the numerator of the second objective is similar to the first objective, the denominator measures a qualitative aspect of the clustering, which is eventually not captured by the first objective. The minimum value of the second objective corresponds to the partitioning where all clusters have an intracluster variation as small as possible and the two closest clusters are as far away from each other as possible.

Another multiobjective evolutionary clustering algorithm with two objectives is proposed in [233]. The first objective function is again a kind of measure of the average intracluster variation computed over all clusters. Rather than using the total summation across clusters, its average value across clusters is used. This is done in order to produce a normalized value of the measure taking into account the number of clusters, which varies across different individuals in the evolutionary algorithm's population. The second objective measures the intercluster distance, which is the average distance between a pair of clusters computed over all pairs of clusters.

9.2 MOPS: Multiobjective Clustering Using Point Symmetry Distance

The point symmetry-based clustering technique, GAPS, optimizes only the total symmetrical compactness for clustering. However, as stated earlier, a single cluster quality measure is seldom equally applicable for different kinds of data sets with different characteristics. Hence, it is necessary to simultaneously optimize several cluster quality measures that can capture different data characteristics. In order to achieve this, the problem of clustering a data set is posed as one of multiobjective optimization (MOO) [77], where search is performed over a number of, often conflicting, objective functions. The newly developed simulated annealing-based multiobjective optimization technique AMOSA [29], described in detail in Sect. 2.4 is used to determine the K cluster centers and the corresponding partitioning. The resulting clustering technique is called multiobjective clustering with point symmetry-based distance (MOPS).

The encoding of a fixed number of cluster centers, and the assignment of the points to the different clusters, is done as explained in detail for GAPS in Sect. 5.6.2. Two objectives are computed for each solution. The first objective function measures the total symmetry present in a partitioning of the data, and the second objective function measures the degree of goodness in terms of the total compactness of the partitioning. These are explained below:

Let K cluster centers be denoted by $\overline{c}_i$ where $1 \leq i \leq K$ and $U(X) = [u_{ij}]_{K \times n}$ be a partition matrix for the data. Then, the first objective function for the AMOSA-based multiobjective clustering technique is defined as follows:

$$totalSym(K) = \sum_{i=1}^{K} E_i, \tag{9.4}$$

where K is the number of clusters. Here,

$$E_i = \sum_{j=1}^{n_i} d_{ps}\left(\overline{x}_j^i, \overline{c}_i\right), \tag{9.5}$$

where n_i denotes the total number of points present in the ith cluster and $\overline{x}_j^i$ denotes the jth point of the ith cluster. $d_{ps}(\overline{x}_j^i, \overline{c}_i)$ is computed by Eq. 5.10. Note that $totalSym(K)$ measures the within-cluster total symmetrical distance. For clusters which have good symmetrical structure, the E_i value will be less. This, in turn, indicates that formation of symmetrical-shaped clusters would be encouraged. Thus, $totalSym(K)$ needs to be minimized for achieving better partitioning.

The second objective function is the total variation σ. Here, σ is written as

$$\sigma(K) = \sum_{i=1}^{K} \sum_{k=1}^{n_i} d_e\left(\overline{c}_i, \overline{x}_k^i\right),$$

where $d_e(\overline{c}_i, \overline{x}_k^i)$ is the Euclidean distance between the kth point of the ith cluster, $\overline{x}_k^i$, and the cluster center $\overline{c}_i$. Note that, when the partitioning is compact and good, the total deviation (σ) should be low. Thus, $\sigma(K)$ needs to be minimized for achieving better clustering. MOPS provides a set of nondominated solutions [77] in the archive. Each of these solutions provides a way of clustering the given data set. All the solutions are equally important from the algorithmic point of view, but sometimes the user may want only a single solution. In the following section a method of selecting a single solution from the set of solution is described.

9.2.1 Selection of a Solution from the Archive

This method is a semisupervised one, where we assume that the class labels of some of the points (denoted *test patterns*) are known to us. The MOPS algorithm is executed on the unlabeled data sets for which no class information is known beforehand. A set of Pareto-optimal solutions is generated. For each clustering associated with a solution from the final Pareto-optimal set, the *test patterns* are also assigned cluster labels based on the nearest center criterion, and the amount of misclassification is calculated by computing the *Minkowski score* values (see Eq. 5.24). The solution with the minimum *Minkowski score* calculated over the *test patterns* is selected as the best solution. Note that this is only one possible way of selecting an appropriate solution from the archive. There may be other ways of doing the same.

9.2.2 Experimental Results

The parameters of AMOSA with the symmetry-based multiobjective clustering algorithm (MOPS) are as follows: $Tmax = 100$, $Tmin = 0.00001$, $\alpha = 0.8$, $SL = 200$, and $HL = 100$. Here, K is set equal to the actual number of clusters present in the data set. MOPS produces a number of nondominated solutions on the final nondominated front. The best solution is identified by the method defined earlier. The performance of MOPS is compared with that of GAPS for the four artificial and four real-life data sets (described in Sect. 5.8.1). In order to compare the performance of the algorithms quantitatively, the *Minkowski scores* (MS) [146] corresponding to their final partitionings are also computed. These are reported in Table 9.1.

As is evident from the table, except for *AD_5_2* and *LungCancer*, the performance of MOPS is similar to that of GAPS (though the former, in general, performs slightly better). For *AD_5_2* and *LungCancer*, MOPS outperforms GAPS by a large margin. As explained for *AD_5_2*, GAPS overestimates the central cluster. Consideration of the two objectives in MOPS reduces this problem. The clustering results are shown in Figs. 9.1(a) and 9.1(b), for MOPS and GAPS, respectively. The results on *Bensaid_3_2* show that MOPS is also able to detect symmetric clusters irrespective of their densities/sizes. In summary, the improved performance of MOPS

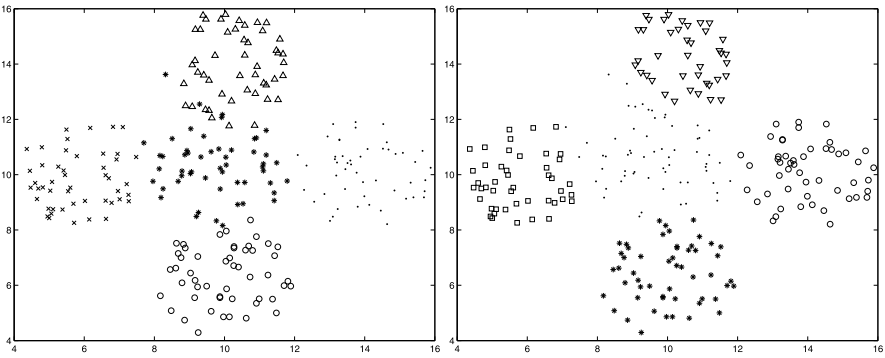


Fig. 9.1 Clustered *AD_5_2* for $K = 5$ after application of the (**a**) MOPS clustering technique and (**b**) GAPS clustering technique

Table 9.1 Best *Minkowski scores* (MS) obtained by the AMOSA with symmetry-based multiobjective clustering technique (MOPS) and GAPS for all the data sets used here for experiment. Smaller values of MS indicate better partitioning

Data set	*Minkowski score*	
	MOPS	GAPS
Sym_3_2	0.21	0.21
AD_5_2	0.37	0.51
Bensaid_3_2	0	0
Iris	0.55	0.59
Cancer	0.31	0.33
LungCancer	0.78	1.16

demonstrates the effectiveness of using MOO for optimizing both the Euclidean compactness and symmetrical compactness of a partitioning simultaneously.

9.3 VAMOSA: Symmetry-Based Multiobjective Clustering Technique for Automatic Evolution of Clusters

The point symmetry-based genetic clustering technique named VGAPS clustering for automatic determination of the number of clusters and the appropriate partitioning from data sets having point-symmetric clusters optimizes a single cluster validity measure, the point symmetry-based cluster validity index *Sym*-index, as the fitness function to reflect the goodness of an encoded partitioning. However, a single cluster validity measure such as *Sym*-index is seldom equally applicable for different kinds of data sets with different characteristics. Hence, it is necessary to simultaneously optimize several validity measures that can capture different data characteristics. Thus, a new multiobjective clustering technique, VAMOSA, based on symmetry is

developed in [241] for automatic evolution of the number of clusters. AMOSA [29] is used as the underlying optimization technique.

In VAMOSA, encoding of a variable number of cluster centers and the assignment of the points to the different clusters are done as in VGAPS and GAPS, respectively (discussed in Sects. 7.3.1 and 5.6.2, respectively). Two cluster validity measures are optimized simultaneously: the well-known Euclidean distance-based XB-index [295], and another recently developed point symmetry distance-based index, the *Sym*-index (defined in Sect. 6.3.1). Note that any other and any number of objective functions could be used instead of the above-mentioned two.

For computing these two validity indices, the centers encoded in a string are first extracted. Let there be K number of cluster centers encoded in a particular string. Let these be denoted as $\mathbf{C} = \overline{c}_1, \overline{c}_2, \ldots, \overline{c}_K$.

The XB-index is defined as a function of the ratio of the total variation σ to the minimum separation sep of the clusters. Here, σ and sep are written as $\sigma(\mathbf{C}; \mathbf{X}) = \sum_{i=1}^{K} \sum_{k=1}^{n_i} d_e^2(\overline{c}_i, \overline{x}_k^i)$ and $sep(\mathbf{C}) = \min_{i \neq j}\{\|\overline{c}_i - \overline{c}_j\|^2\}$, where $\|.\|$ is the Euclidean norm, $d_e(\overline{c}_i, \overline{x}_k^i)$ is the Euclidean distance between the kth point of the ith cluster, $\overline{x}_k^i$, and the cluster center $\overline{c}_i$, and n_i denotes the number of points present in the ith cluster. $\mathbf{C}$ and $\mathbf{X}$ represent the set of cluster centers and the data set, respectively. The XB-index is then written as

$$XB = \frac{\sigma(\mathbf{C}; \mathbf{X})}{sep(\mathbf{C})} = \frac{\sum_{i=1}^{K}(\sum_{k=1}^{n_i} d_e^2(\overline{c}_i, \overline{x}_k^i))}{n(\min_{i \neq j}(\|\overline{c}_i - \overline{c}_j\|^2)}.$$

Note that, when the partitioning is compact and good, the total deviation (σ) should be low while the minimal separation (sep) between any two cluster centers should be high. Thus, the objective is to minimize the XB-index to achieve the proper clustering.

The second objective function is the newly defined point symmetry distance based *Sym*-index (defined in Sect. 6.3.1).

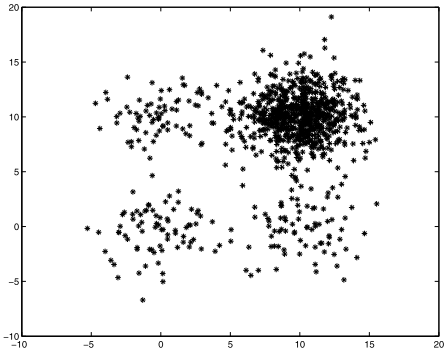
Thus, the objective functions corresponding to the ith string of VAMOSA are $f_1 = \frac{1}{XB_i}$ and $f_2 = Sym_i$, where XB_i and Sym_i, respectively, the XB-index and *Sym*-index values corresponding to the ith string. These two objective functions are maximized simultaneously in VAMOSA using the search capability of AMOSA.

Here, a mutation operation similar to VGAPS (described in detail in Sect. 7.3.3.3) is used. Finally, the best solution is selected from the final archive following the procedure mentioned in Sect. 9.2.1.

9.3.1 Data Sets Used for Experiment

Six data sets are used for the experiment: three of them are artificial data (*Sym_3_2*, *AD_5_2*, and *Sizes5*) and three are real-life data sets (*Iris*, *Cancer*, and *LungCancer*). Real-life data sets are obtained from [2]. *Sym_3_2*, *AD_5_2*, *Iris*, *Cancer*, and *LungCancer* are described in Sect. 5.8.1. *Sizes5* is described below:

Fig. 9.2 *Sizes5*



1. *Sizes5*: This data set, used in Ref. [121], consists of 1,000 data points distributed over four clusters. The densities of these clusters are not uniform. This is shown in Fig. 9.2.

9.3.2 Experimental Results

The parameters of the VAMOSA clustering technique were as follows: $Tmax = 100$, $Tmin = 0.00001$, $\alpha = 0.8$, $SL = 200$, and $HL = 100$. Here, K^{max} is set equal to $\sqrt{n}$, where n is the size of the data set. For the purpose of comparison, another MO clustering technique, MOCK [121] is also executed on the above-mentioned data sets with default parameter settings. The source code for MOCK was obtained from http://dbkgroup.org/handl/mock/. In MOCK, the best solution from the final Pareto-optimal front is selected by GAP statistics [280]. Note that, for every data set used here for the experiment, class labels of all the data points are available. Thus, in order to quantify the obtained partitionings by different algorithms, their corresponding *Minkowski score*s [146] (defined in Eq. 5.24) were computed. The number of clusters identified by the best solution of the VAMOSA clustering technique and MOCK clustering technique, and the *Minkowski score* (MS) values of the corresponding partitionings for all the data sets used here for the experiment are reported in Table 9.2.

In order to show the efficacy of the MO clustering technique over existing single-objective clustering techniques, two recently developed genetic algorithm-based automatic clustering techniques, genetic clustering for unknown K (GCUK clustering) [20] and VGAPS clustering (described in Sect. 7.3), were also executed on the above-mentioned six data sets. These single-objective automatic clustering techniques provide a single solution after their execution. The GCUK clustering technique optimizes a Euclidean distance-based cluster validity index, i.e., the XB-index [295], by using the search capability of genetic algorithms to automatically determine the appropriate partitioning from data sets. The parameters of the XB-GCUK clustering technique are as follows: population size = 100, number of generations = 40, probability of mutation = 0.2 and probability of crossover = 0.8 (as

Table 9.2 Number of clusters and the *Minkowski score* (MS) values obtained by the VAMOSA clustering technique, another automatic MO clustering technique, MOCK, two single-objective clustering techniques, the VGAPS clustering optimizing the *Sym*-index, and GCUK clustering optimizing the XB-index, for all the data sets used here for experiment. Smaller values of MS indicate better partitioning

Data set	AC	VAMOSA		MOCK		VGAPS		XB-GCUK	
		OC	MS	OC	MS	OC	MS	OC	MS
Sym_3_2	3	**3**	**0.12**	2	0.69	**3**	**0.12**	4	0.74
AD_5_2	5	**5**	**0.25**	6	0.39	5	0.42	5	0.39
Sizes5	4	**4**	**0.14**	2	0.64	5	0.22	4	0.25
Iris	3	2	0.80	2	0.82	**3**	**0.62**	2	0.84
Cancer	2	**2**	**0.32**	2	0.39	2	0.37	2	0.38
LungCancer	3	**3**	**0.85**	7	0.97	2	0.97	6	0.94

specified in [20]). The parameters of the VGAPS clustering technique are set as detailed in Sect. 7.5.2. The numbers of clusters automatically determined by these clustering techniques for the six data sets are also reported in Table 9.2. The MS values were also calculated for the partitionings obtained by these two single-objective clustering techniques for these six data sets. These are also reported in Table 9.2.

1. *Sym_3_2*: As seen from Table 9.2, both symmetry-based clustering techniques, VAMOSA and VGAPS, are able to detect the proper number of clusters and the proper partitioning from this data set. The corresponding partitioning is shown in Fig. 9.3(a). MOCK merges the two overlapping clusters into one cluster and provides $K = 2$ as the optimal number of clusters. The corresponding partitioning is shown in Fig. 9.3(b). The XB-GCUK clustering technique identifies a total $K = 4$ number of clusters from this data set. The corresponding partitioning is shown in Fig. 9.3(c). The MS scores reported in Table 9.2 also show the poorer performance of both the MOCK and XB-GCUK clustering techniques for this data set.
2. *AD_5_2*: As can be seen from Table 9.2, for this data set, the VAMOSA clustering technique performs much better than the VGAPS clustering technique. The corresponding partitionings are shown in Figs. 9.4(a) and 9.4(b), respectively. The best solution provided by MOCK is not able to determine the appropriate number of clusters from this data set. The corresponding partitioning is shown in Fig. 9.4(c). GCUK clustering optimizing the XB-index is able to detect the appropriate number of clusters from this data set, and the corresponding partitioning is very near to the actual partitioning of the data set (refer to Table 9.2). The corresponding partitioning is shown in Fig. 9.4(d).
3. *Sizes5*: Here, only the VAMOSA clustering technique and the XB-GCUK clustering technique are able to detect the appropriate number of clusters. The corresponding partitionings are shown in Figs. 9.5(a) and 9.6(b), respectively. However, the MS value corresponding to the partitioning obtained by VAMOSA is much lesser than that of XB-GCUK (refer to Table 9.2). The best solution of

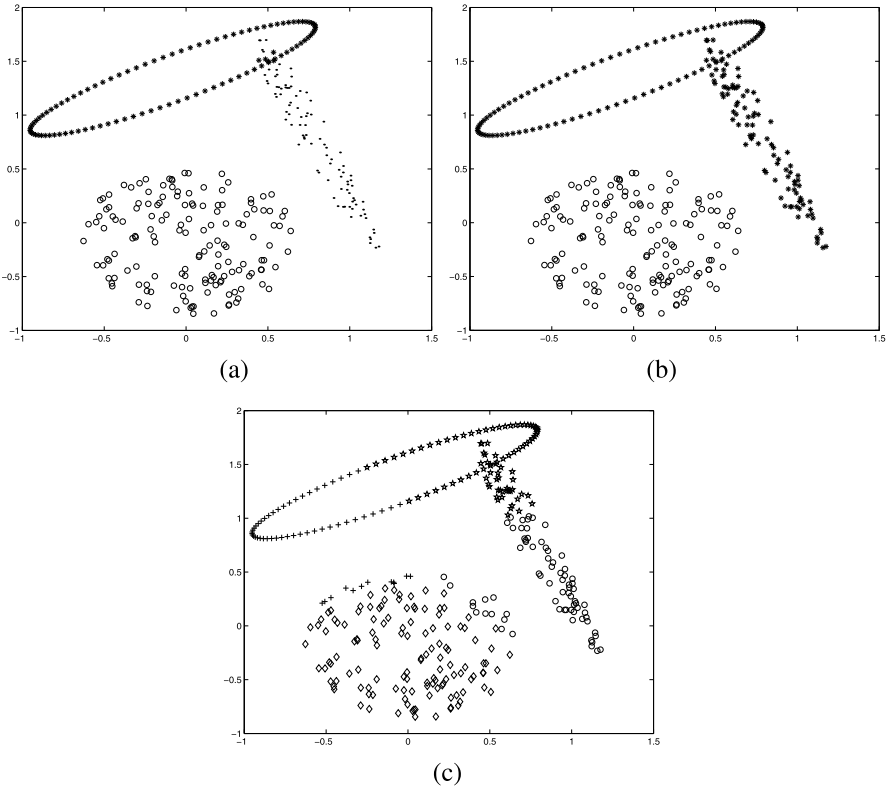


Fig. 9.3 Automatically clustered *Sym_3_2* after application of the (**a**) VAMOSA/VGAPS clustering technique for $K = 3$, (**b**) MOCK clustering technique for $K = 2$, (**c**) XB-GCUK clustering technique for $K = 4$

MOCK provides $K = 2$ as the optimal number of clusters. The corresponding partitioning is shown in Fig. 9.5(b). VGAPS overestimates the number of clusters from this data set. It breaks the maximum densed cluster into two clusters. The corresponding partitioning is shown in Fig. 9.6(a).

4. *Iris*: For this real-life data set, only the VGAPS clustering technique is able to determine the appropriate number of clusters. The corresponding MS score is also the minimum (refer to Table 9.2). As this is a higher dimensional data set, no visualization is possible. The other three clustering algorithms, VAMOSA, MOCK, and XB-GCUK, provide $K = 2$ as the optimal number of clusters, which is also often obtained for many other methods for the *Iris* data set. However, the MS score corresponding to VAMOSA for $K = 2$ is the minimum among these three clustering techniques (refer to Table 9.2).
5. *Cancer*: For this data set all four clustering techniques are able to detect the appropriate number of clusters ($K = 2$ for this case), but the MS value obtained by the VAMOSA clustering technique is the minimum (refer to Table 9.2).

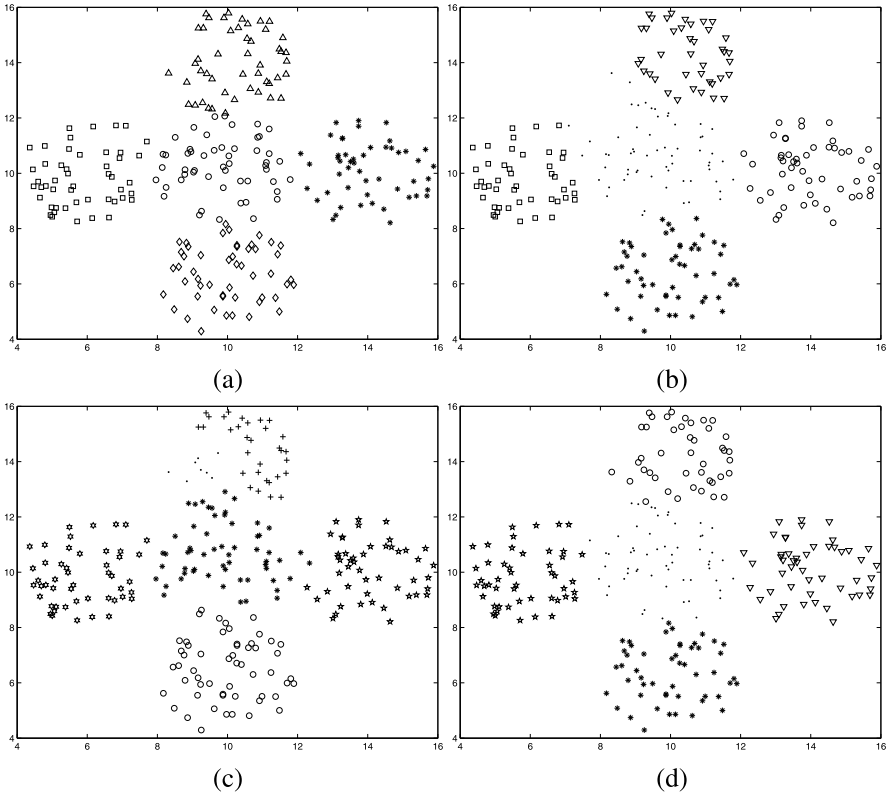


Fig. 9.4 Automatically clustered *AD_5_2* after application of the (**a**) VAMOSA clustering technique for $K = 5$, (**b**) VGAPS clustering technique for $K = 5$, (**c**) MOCK clustering technique for $K = 6$, and (**d**) XB-GCUK clustering technique for $K = 5$

6. *LungCancer*: For this high-dimensional data set, only the VAMOSA clustering technique is able to detect the appropriate number of clusters. None of the other algorithms are able to detect the correct number of clusters. The MS value obtained by VAMOSA is again the minimum (refer to Table 9.2).

Summary of Results It can be seen from the above results that the VAMOSA clustering technique is able to detect the appropriate partitioning and the appropriate number of clusters for most of the data sets used here for the experiment. It outperforms another MO clustering technique, MOCK, and two single-objective genetic algorithm-based clustering techniques. The superiority of VAMOSA is also established on three real-life data sets. These real-life data sets have different characteristics with the number of dimensions varying from 4 to 56. Results on six artificial and real-life data sets establish the fact that VAMOSA is wellsuited to detect the number of clusters automatically from data sets having clusters with widely varying characteristics, as long as they possess the property of point symmetry.

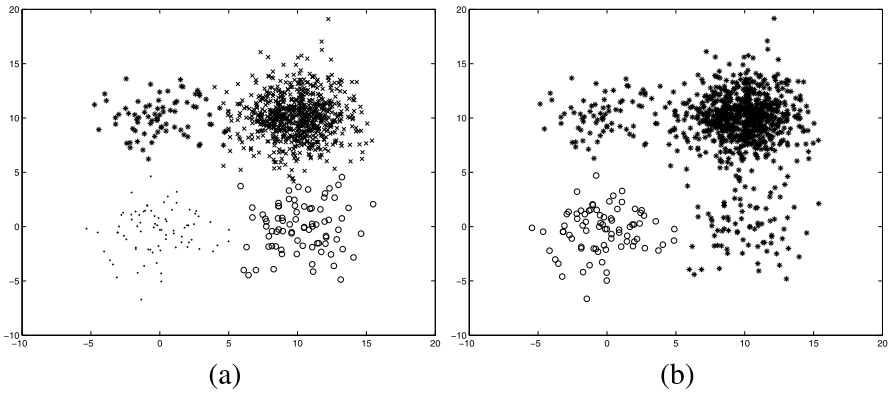


Fig. 9.5 Automatically clustered *Sizes5* after application of the (**a**) VAMOSA clustering technique for $K = 4$, and (**b**) MOCK clustering technique for $K = 2$

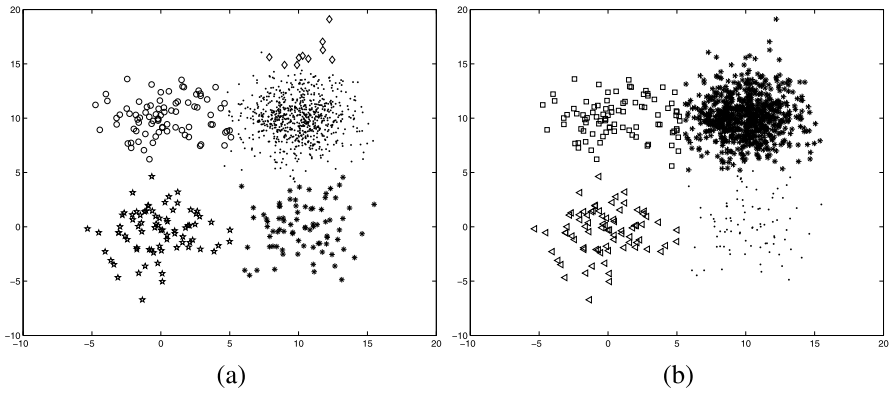


Fig. 9.6 Automatically clustered *Sizes5* after application of the (**a**) VGAPS clustering technique for $K = 5$, and (**b**) XB-GCUK clustering technique for $K = 4$

9.4 A Generalized Automatic Clustering Algorithm in a Multiobjective Framework

In this section a new multiobjective clustering technique with encoding of cluster centers as in [188], which can detect the appropriate number of clusters and the appropriate partitioning from data sets with many different types of cluster structures, is discussed. AMOSA is again used as the underlying optimization strategy. The concept of "multiple centers" corresponding to each cluster is used here. Each cluster is divided into several nonoverlapping small hyperspherical subclusters, and the centers of these subclusters are encoded in a string to represent a particular cluster. Three cluster validity indices are optimized simultaneously using the search capability of AMOSA. One of these cluster validity indices reflects the total compactness of a particular partitioning, another represents the total symmetry present

in a particular partitioning, and the last one measures, in a novel way, the degree of "connectedness" of a particular partitioning. The superiority of *GenClustMOO* in comparison with the MOCK-clustering technique, VAMOSA clustering technique, and a single-objective genetic clustering technique, i.e., VGAPS clustering [28], is shown for many artificial data sets (including most of the data sets used in [121]) and many real-life data sets of varying complexities. In a part of the experiment, the effectiveness of AMOSA as the underlying optimization technique in *GenClustMOO* is also demonstrated in comparison to another evolutionary MO algorithm, PESA-II.

9.4.1 GenClustMOO: Multiobjective Clustering Technique

This section describes the new multiobjective clustering technique, *GenClustMOO*, in detail.

9.4.1.1 String Representation and Population Initialization

In *GenClustMOO*, a state of AMOSA comprises a set of real numbers which represents the coordinates of the centers of the partitions. AMOSA attempts to evolve an appropriate set of cluster centers and hence the associated partitioning of the data. Here, each cluster is divided into several small nonoverlapping hyperspherical subclusters. Then, each cluster is represented by the centers of these individual subclusters. Suppose a particular string encodes the centers of K number of clusters and each cluster is divided into C number of subclusters. If the data set is of dimension d, then the length of the string will be $C \times K \times d$. This concept of representing one cluster using multicenters is shown in Fig. 9.7. Suppose a particular string contains $K = 2$ number of clusters. Each cluster is divided into 10 smaller subclusters, i.e., here $C = 10$. Let the dimension (d) of the data set be 2. Suppose the center of the jth subcluster of the ith cluster is denoted by $\overline{c}_j^i = (cx_j^i, cy_j^i)$. Then, this string will look like: $\langle cx_1^1, cy_1^1, cx_2^1, cy_2^1, \ldots, cx_{10}^1, cy_{10}^1, cx_1^2, cy_1^2, \ldots, cx_{10}^2, cy_{10}^2 \rangle$. Each string i in the archive initially contains K_i number of clusters, such that $K_i = (\text{rand}() \bmod (K^{max} - 1)) + 2$. Here, rand() is a function returning an integer, and K^{max} is a soft estimate of the upper bound of the number of clusters. The number of initial clusters will therefore lie between 2 and K^{max}. Our initialization procedure is motivated by that of Ref. [121]. Here, the initialization procedure is partly random and partly based on two different single-objective algorithms in order to obtain a good initial spread of solutions. One-third of the solutions in the archive is initialized after running the single linkage clustering algorithm for different values of K. These solutions perform well when clusters present in the data set are wellseparated. Another one-third of the solutions in the archive are generated using the K-means algorithm. These solutions perform well under overall deviations. The last one-third of the solutions are generated randomly; i.e., for these strings the K_i

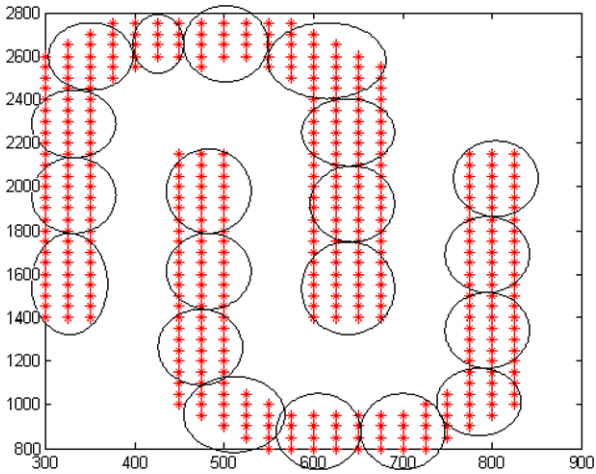


Fig. 9.7 Example of representing a single cluster using multiple cluster centers

centers encoded in a string are randomly selected distinct points from the data set. The initial partitioning is obtained using a minimum center distance-based criterion. For all the initial encoded solutions, C number of distinct points are selected from each cluster randomly. These $C \times K$ number of points are encoded in that particular string.

9.4.1.2 Assignment of Points

For the purpose of assignment, each subcluster is considered as a separate cluster. Here, assignment is done based on the minimum Euclidean distance criterion. A data point $\overline{x}_j$ is assigned to the kth subcluster, where

$$k = \text{argmin}_{i=1}^{K \times C} d_e(\overline{c}_i, \overline{x}_j).$$

Thereafter, the partition matrix is formed in the following way: $u(kj) = 1$ and $u(ij) = 0, \forall i = 1, \ldots, K \times C, k \neq i$.

9.4.1.3 Objective Functions Used

For the purpose of optimization, three different cluster validity indices are considered. These three objective functions reflect three different aspects of good clustering solutions. The first quantifies the amount of symmetry present in a particular partitioning. The second quantifies the connectedness of the clusters, and the third measures the compactness of the partitionings in terms of the Euclidean distance. These indices are described below. The first objective function is the newly developed point symmetry-based cluster validity index, *Sym*-index (defined in Sect. 6.3.1).

9.4.1.4 Connectivity-Based Cluster Validity Index: *Con*-index

A new cluster validity index based on the concept of connectedness of the clusters is developed [245]. This index is capable of detecting the appropriate partitioning from data sets having clusters of any shape, size or convexity as long as they are wellseparated. The concept of a relative neighborhood graph (RNG) [282] has been successfully applied for solving several pattern recognition problems. An unsupervised clustering technique based on the concepts of RNG is developed in Ref. [15]. In this chapter, RNG is used to develop a new cluster validity index, *Con*-index, that quantifies the degree of connectivity of well-separated clusters.

9.4.1.5 Relative Neighborhood Graph [282]

Suppose D is an integer and $\mathbf{x}, \mathbf{y}$ *are two points in D-dimensional Euclidean space. Then the lune of* $\mathbf{x}$ *and* $\mathbf{y}$ *(denoted by lun*$(\mathbf{x}, \mathbf{y})$ *or lun*$(\mathbf{xy})$*) is the set of points*

$$\left\{z \in R^D : d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{y}) \text{ and } d(\mathbf{y}, \mathbf{z}) < d(\mathbf{x}, \mathbf{y})\right\},$$

where d denotes the Euclidean distance. Alternatively, lun$(\mathbf{x}, \mathbf{y})$ *denotes the interior of the region formed by the intersection of two D-dimensional hyperspheres of radius $d(\mathbf{x}, \mathbf{y})$ where one hypersphere is centered at* $\mathbf{x}$ *and the other at* $\mathbf{y}$*. If V is a set of n points in D-space, then the relative neighborhood graph of V (denoted $RNG(V)$ or simply RNG when V is understood) is an undirected graph with vertices V such that, for each pair* $\mathbf{x}, \mathbf{y} \in V$, $\mathbf{xy}$ *is an edge of $RNG(V)$ if and only if lun*$(\mathbf{x}, \mathbf{y}) \cap V = \emptyset$*. Here, the edge weight of a particular edge* $(\mathbf{xy})$ *is kept equal to $d(\mathbf{x}, \mathbf{y})$, the Euclidean distance between the points* $\mathbf{x}$ *and* $\mathbf{y}$*.*

Figure 9.8(a) shows a set V of points in the plane; Fig. 9.8(b) shows the RNG of this set of points V. The RNG problem is: Given a set V, find $RNG(V)$.

9.4.1.6 Measuring the Connectivity Among a Set of Points

A novel way of measuring the connectivity among a set of points using the above-discussed RNG is proposed in [245]. The distance between a pair of points is measured in the following way:

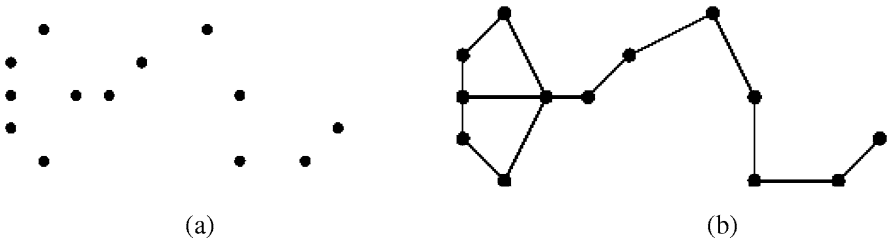


Fig. 9.8 (**a**) A set of points in the plane. (**b**) RNG of the points in (**a**)

- Construct the relative neighborhood graph of the whole data set.
- The distance between any two points, $\mathbf{x}$ and $\mathbf{y}$, denoted as $d_{short}(\mathbf{x}, \mathbf{y})$, is measured along the relative neighborhood graph. Find all possible paths between these two points along the RNG. Suppose there are total p paths between $\mathbf{x}$ and $\mathbf{y}$, and the number of edges along the ith path is n_i, for $i = 1, \ldots, p$. If the edges along the ith path are denoted as $ed_1^i, \ldots, ed_{n_i}^i$ and the corresponding edge weights are $w(ed_1^i), \ldots, w(ed_{n_i}^i)$, then the shortest distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as follows:

$$d_{short}(\mathbf{p}, \mathbf{q}) = \min_{i=1}^{p} \max_{j=1}^{n_i} w\left(ed_j^i\right). \tag{9.6}$$

In order to improve the efficiency of computing d_{short}, the following pruning strategy is adopted. The maximum value of $w(ed_j^i)$ corresponding to the first path is stored in a temporary variable *max*. If, in any of the next path being traced, a weight value greater than *max* is obtained, that path is pruned. However, if a smaller value of the maximum weight is found in any of the subsequent paths, then *max* is updated to this smaller value and the process repeats.

9.4.1.7 Definition of the Connectivity-Based Cluster Validity Index

The cluster validity index is defined as follows: Suppose the clusters formed are denoted by C_k, for $k = 1, \ldots, K$, where K is the number of clusters. Then, the medoid of the kth cluster, denoted by $\overline{m}_k$, is the point of that cluster which has the minimum average distance to all the other points in that cluster. Suppose the point which has the minimum average distance to all the points in the kth cluster is denoted by $\overline{x}^k_{\text{minindex}}$. Then,

$$\text{minindex} = \text{argmin}_{i=1}^{n_k} \frac{\sum_{j=1}^{n_k} d_e(\overline{x}_i^k, \overline{x}_j^k)}{n_k},$$

where n_k is the total number of points in the kth cluster and $\overline{x}_i^k$ denotes the ith point of the kth cluster. Then,

$$\overline{m}_k = \overline{x}^k_{\text{minindex}}.$$

The newly developed *Con*-index is defined as follows:

$$Con = \frac{\sum_{i=1}^{K} \sum_{j=1}^{n_k} d_{short}(\overline{m}_i, \overline{x}_j^i)}{n \times \min_{i,j=1 \wedge i \neq j}^{K} d_{short}(\overline{m}_i, \overline{m}_j)},$$

where $d_{short}(\overline{m}_i, \overline{x}_j^i)$ is the shortest distance along the relative neighborhood graph between the two points $\overline{m}_i$ and $\overline{x}_j^i$, the jth point of the ith cluster. It is calculated using the procedure mentioned in Sect. 9.4.1.6. n denotes the total number of points present in the data set. Intuitively, smaller values of the *Con*-index correspond to

good partitioning. In order to achieve the proper partitioning, the value of *Con*-index has to be minimized.

Con-index has two components. Its denominator measures the minimum shortest distance between any two medoids among a total of K clusters. Thus, when the clusters are wellseparated, this distance is maximum, which in turn minimizes the *Con*-index value. The numerator of the *Con*-index measures the total connectedness of a particular partitioning. If the clusters are wellconnected then the shortest distance between the medoid and any point of that particular cluster is small and thus the numerator of the *Con*-index also takes a very small value. Thus the *Con*-index obtains its minimum value when clusters are connected as well as separated.

The third objective function used here is a Euclidean distance-based cluster validity index, i.e., the *I*-index [189]. It has already been described in detail in Sect. 6.2.8.

9.4.2 Subcluster Merging for Objective Function Calculation

Before computing the above-mentioned three objective functions for each string, first the total $C \times K$ number of subclusters encoded in a particular string are merged to form a total of K clusters. The merging operation is done in the following way: First, the shortest distance between each pair of $C \times K$ cluster medoids along the relative neighborhood graph is computed. This provides a distance matrix denoted as $distance_{short}$, i.e.,

$$distance_{short} = \left[d_{short}(c_i, c_j)\right]_{i,j=1,\ldots,C\times K}.$$

Thereafter, the single linkage clustering technique [95] is executed on these cluster centers K times with this modified distance measure, $distance_{short}$, each time merging C number of clusters to form a single cluster.

After the merging operation is done, the three cluster validity indices are computed for each string. Thus, the objective functions for a particular string are

$$obj = \{sym(K), 1/Con(K), I(K)\},$$

where $sym(K)$, $Con(K)$, and $I(K)$ are, respectively, the calculated *Sym*-index value, *Con*-index value, and *I*-index value for that particular string. Here, K denotes the number of clusters present in that particular string. These three objective functions are simultaneously optimized by using the simulated annealing-based MOO algorithm AMOSA.

A new string is generated following the procedure of VGAPS (defined in Sect. 7.3.3.3). Finally, a solution is selected from the final archive using the procedure described in Sect. 9.2.1.

Table 9.3 Results on different data sets by the *GenClustMOO*, MOCK, VGAPS, *GenClustPESA2*, and VAMOSA clustering algorithms. Smaller values of MS indicate better partitioning (best values are marked in bold)

Data set	N	d	K	*GenClust-MOO*		MOCK		VGAPS		*GenClust-PESA2*		VAMOSA	
				OC	MS	OC	MS	OC	MS	OC	MS	OC	MS
AD_10_2	500	2	10	**10**	**0.13**	6	1.01	7	0.84	11	0.32	10	0.43
Pat1	557	2	3	**3**	**0.00**	10	0.89	4	0.93	**3**	**0.00**	2	0.83
Long1	1000	2	2	**2**	**0.00**	2	0.00	3	1.00	**2**	**0.00**	8	0.73
Sizes5	1000	2	4	**4**	**0.14**	2	0.64	5	0.76	3	0.69	**4**	**0.14**
Spiral	1000	2	2	**2**	**0.00**	3	0.39	6	1.00	**2**	**0.00**	4	0.97
Square4	1000	2	4	**4**	**0.49**	4	0.60	5	0.52	**4**	**0.49**	4	0.51
Twenty	1000	2	20	**20**	**0.00**	**20**	**0.00**	20	1.35	24	0.31	20	0.77
Iris	150	4	3	**3**	**0.54**	2	0.82	3	0.62	3	0.55	2	0.80
Cancer	683	9	2	**2**	**0.32**	2	0.39	2	0.36	2	0.35	**2**	**0.32**
Lung-Cancer	33	56	3	2	**0.77**	7	0.97	2	0.97	4	0.83	3	0.85
Glass	214	9	6	**6**	**0.49**	5	0.53	5	0.53	5	0.53	5	2.75

9.5 Experimental Results

9.5.1 Data Sets Used

Seven artificial data sets and four real-life data sets were used for the experiment. The artificial data sets are *AD_10_2*, *Pat1*, *Long1*, *Spiral*, *Square4*, *Sizes5*, and *Twenty*. The real-life data sets were obtained from [2]. These are *Iris*, *Cancer*, *LungCancer*, and *Glass*. The description of *AD_10_2* is provided in Sect. 7.7. The descriptions of *Iris*, *Cancer*, and *LungCancer* are provided in Sect. 5.8.1. The remaining data sets are described below. A description of the data sets in terms of the number of points present, dimension of the data set, and number of clusters is presented in Table 9.3.

1. *Pat1*: This dataset, used in Ref. [217], consists of 880 patterns. There are three non-convex clusters present in this data set. This is shown in Fig. 9.9(a).
2. *Long1*: This data set, used in Ref. [121], consists of 1,000 data points distributed over two long clusters. This is shown in Fig. 9.9(b).
3. *Spiral*: This data set, used in Ref. [121], consists of 1,000 data points distributed over two spiral clusters. This is shown in Fig. 9.9(c).
4. *Square4*: This data set, used in Ref. [121], consists of 1,000 data points distributed over four squared clusters. This is shown in Fig. 9.9(d).
5. *Sizes5*: This data set, used in Ref. [121], consists of 1,000 data points distributed over four clusters. The densities of these clusters are not uniform. This is shown in Fig. 9.9(e).

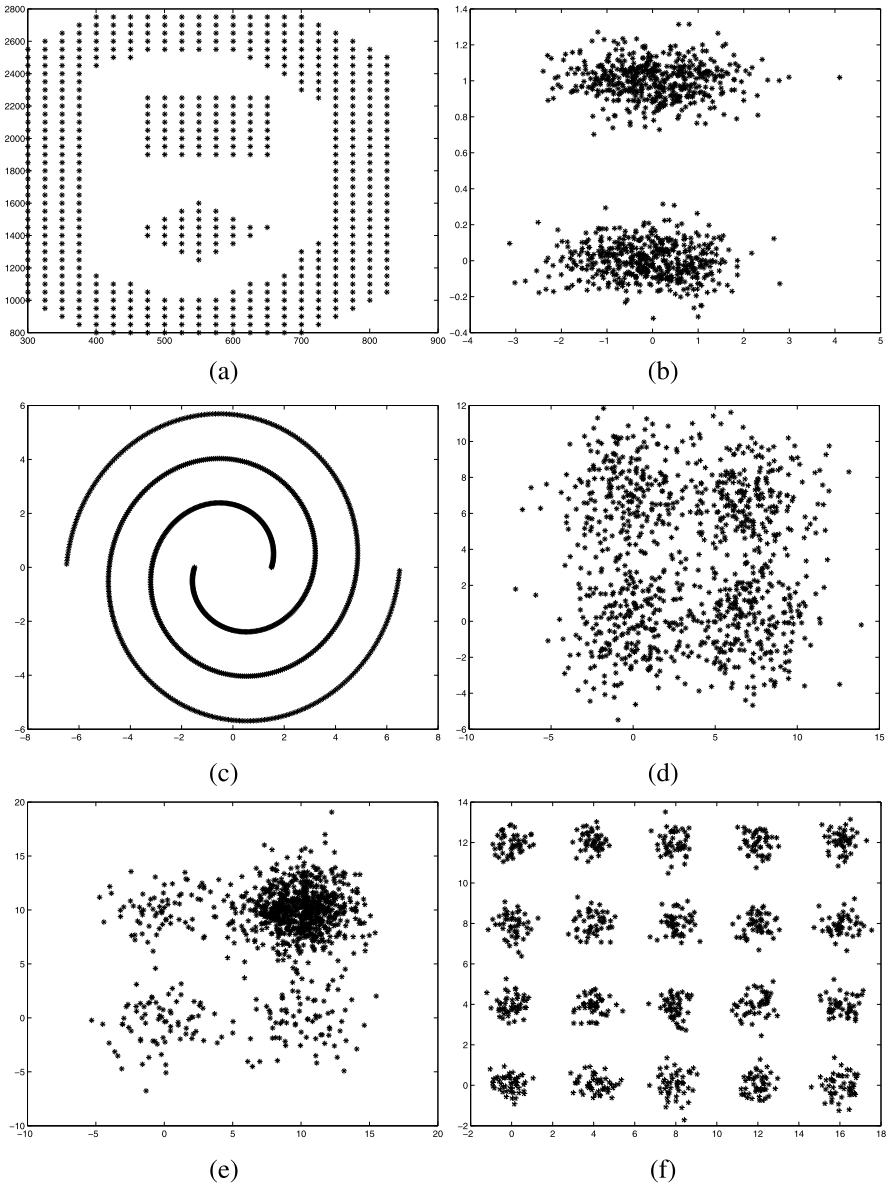


Fig. 9.9 (**a**) *Pat1*. (**b**) *Long1*. (**c**) *Spiral*. (**d**) *Square4*. (**e**) *Sizes5*. (**f**) *Twenty*

6. *Twenty*: This data set, used in Ref. [121], consists of 1,000 data points distributed over 20 small clusters. This is shown in Fig. 9.9(f).
7. *Glass*: This is a glass identification data set consisting of 214 instances having nine features (an Id# feature has been removed). Criminological investigation inspires the study of the classification of the types of glass. At the scene of a

crime, the glass left can be used as evidence, if it is correctly identified. There are six categories present in this data set.

9.5.2 Discussion of Results

In *GenClustMOO*, the newly developed simulated annealing-based MOO technique, AMOSA is used as the underlying optimization strategy. The parameters of the *GenClustMOO* clustering technique were as follows: $SL = 100$, $HL = 50$, $iter = 50$, $Tmax = 100$, $Tmin = 0.00001$, and cooling rate, $\alpha = 0.9$. *GenClustMOO* was executed on all the data sets used in the previous chapters. For all the data sets having symmetrical-shaped clusters (e.g., *Ellip_2_2*, *Sym_3_2*, described in earlier chapters) it performs similarly to VAMOSA and VGAPS. For the *AD_ 5_2* data set it performs similarly to the VAMOSA clustering technique. Thus, results are reported here for only those data sets for which it outperforms the previously defined clustering techniques, VAMOSA and VGAPS. For the purpose of comparison, another automatic MOO clustering technique, MOCK [121], was also executed on the above-mentioned data sets. In MOCK, the final best solution is selected by GAP statistics [280]. The number of clusters automatically determined by the *GenClustMOO* and MOCK clustering techniques for all the above-mentioned data sets are presented in Table 9.3. This table also contains the *Minkowski score* [146] (defined in Eq. 5.24) values of the final partitionings identified by these two algorithms. Here, the best *Minkowski score* (MS) values obtained by the algorithms over five runs for all data sets are reported. In [121] it has already been established that the performance of the MO clustering technique, MOCK, is much better than K-means, average linkage, single linkage, and Strehl's ensemble method. Thus, comparisons of *GenClustMOO* with these well-known clustering techniques are omitted from this chapter. In order to show the effectiveness of AMOSA as the underlying optimization technique in *GenClustMOO*, the results are also shown for all the data sets obtained by *GenClustPESA2* which uses exactly the same approach as *GenClusMOO* but with the underlying MOO strategy replaced by PESA-II [65]. The number of clusters and the corresponding *Minkowski score* values are reported in Table 9.3. The performance of *GenClustMOO* is also compared with that of the VAMOSA clustering technique. The number of clusters obtained by this technique along with the corresponding *Minkowski score* values are also reported in Table 9.3.

In order to show that the multiobjective clustering technique (*GenClustMOO*) performs better than a single-objective version, VGAPS clustering [28] was also executed on the above-mentioned data sets used here for the experiment. The parameter values of the VGAPS clustering technique were set as detailed in Sect. 7.5.2.

The clusters present in *AD_10_2* are hyperspherical in shape. *GenClustMOO* is able to identify automatically the appropriate number of clusters and the appropriate partitioning from this data set. The partitioning obtained by *GenClustMOO* for this data set is shown in Fig. 9.10(a). MOCK is not able to detect the appropriate number of clusters from this data set. The partitioning obtained by MOCK for this data set

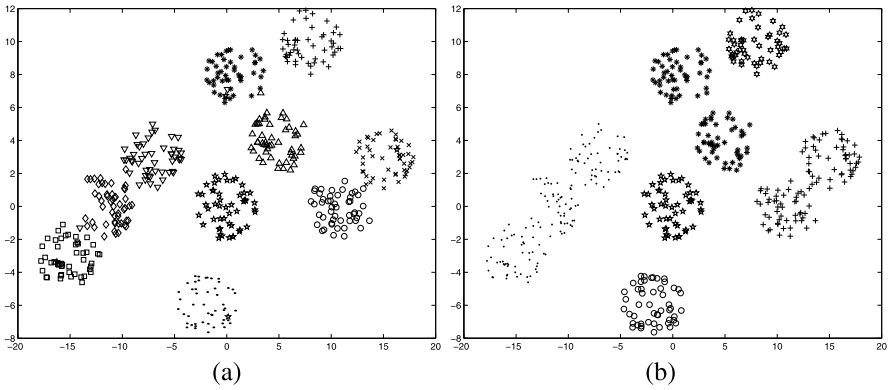


Fig. 9.10 Automatically clustered *AD_10_2* after application of the (**a**) *GenClustMOO* clustering technique for $K = 10$ and (**b**) MOCK clustering technique for $K = 6$

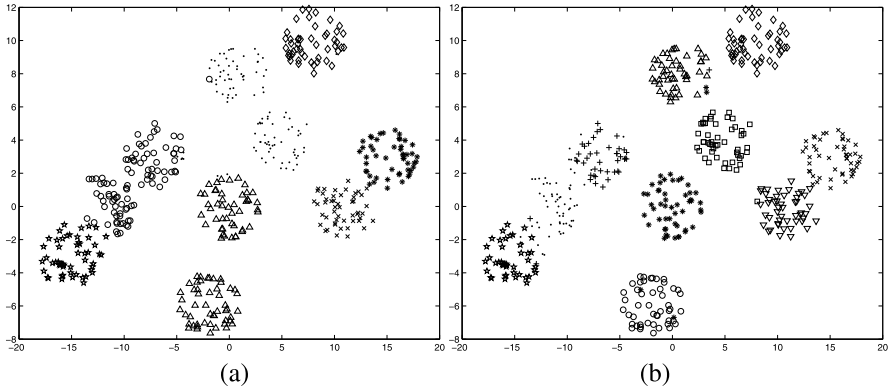


Fig. 9.11 Automatically clustered *AD_10_2* after application of the (**a**) VGAPS clustering technique for $K = 7$ and (**b**) *VAMOSA* clustering technique for $K = 10$

is shown in Fig. 9.10(b). The MS value obtained by *GenClustMOO* is also less than that obtained by MOCK (refer to Table 9.3). VGAPS is not able to detect the proper partitioning for the *AD_10_2* data set (the corresponding partitioning is shown in Fig. 9.11(a)). The *GenClustPESA2* clustering technique fails to detect the proper number of partitions from the *AD_10_2* data set. The partitioning obtained by VAMOSA for *AD_10_2* is shown in Fig. 9.11(b).

The clusters present in *Pat1*, *Long1*, *Spiral*, *Sizes5*, *Square4*, and *Twenty* are wellseparated, having any shape, size or convexity. These data sets are used to show the performance of the algorithms for detecting some well-separated clusters. *GenClustMOO* is able to detect the appropriate number of partitions and the appropriate partitioning from all six data sets. The partitionings identified by *GenClustMOO* for all these six data sets are shown in Figs. 9.12(a), 9.14(a), 9.16(a), 9.18(a), 9.20(a),

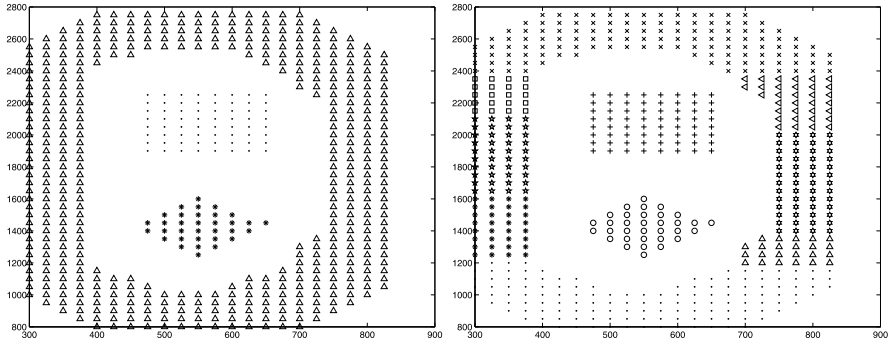


Fig. 9.12 Automatically clustered *Pat1* after application of the (**a**) *GenClustMOO* clustering technique for $K = 3$ and (**b**) MOCK clustering technique for $K = 10$

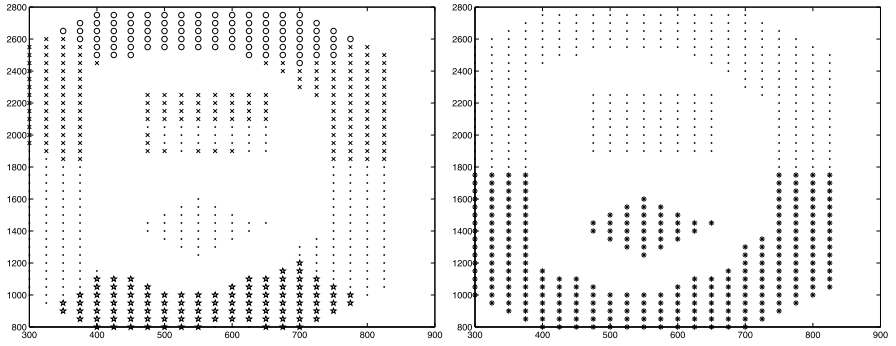


Fig. 9.13 Automatically clustered *Pat1* after application of the (**a**) VGAPS clustering technique for $K = 4$ and (**b**) VAMOSA clustering technique for $K = 2$

and 9.22(a), respectively. MOCK is able to detect the appropriate number of partitions from three out of six data sets. The partitionings are shown in Figs. 9.12(b), 9.14(b), 9.16(b), 9.18(b), 9.20(b), and 9.22(b), respectively. VGAPS clustering is able to detect the proper partitioning and the proper number of clusters from only one out of six data sets. The partitionings are shown in Figs. 9.13(a), 9.15(a), 9.17(a), 9.19(a), 9.21(a), and 9.23(a), respectively. The *GenClustPESA2* clustering technique performs poorly for the *Sizes5* and *Twenty* data sets (refer to Table 9.3). For other data sets of this group, the *GenClustPESA2* and *GenClustMOO* clustering techniques perform similarly. The partitionings identified by VAMOSA for all these six data sets are shown in Figs. 9.13(b), 9.15(b), 9.17(b), 9.19(b), 9.21(b), and 9.23(b), respectively. Table 9.3 shows that, for most of the data sets of this group VAMOSA is not able to detect the proper partitioning and the proper number of partitions. This is because most of these data sets contain clusters having nonsymmetrical shapes but wellseparated structures.

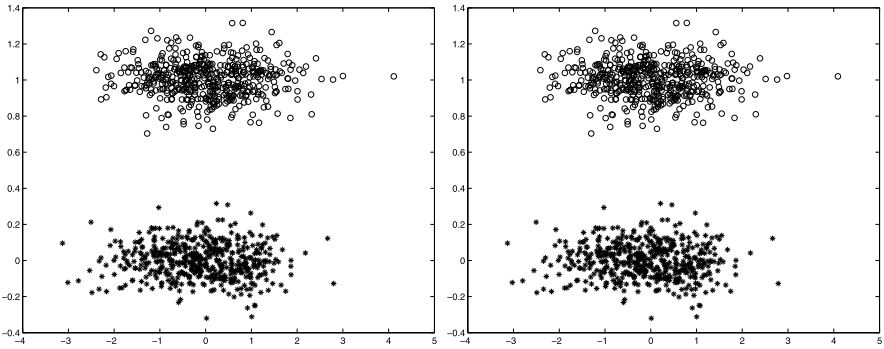


Fig. 9.14 Automatically clustered *Long1* after application of the (**a**) *GenClustMOO* clustering technique for $K = 2$ and (**b**) MOCK clustering technique for $K = 2$

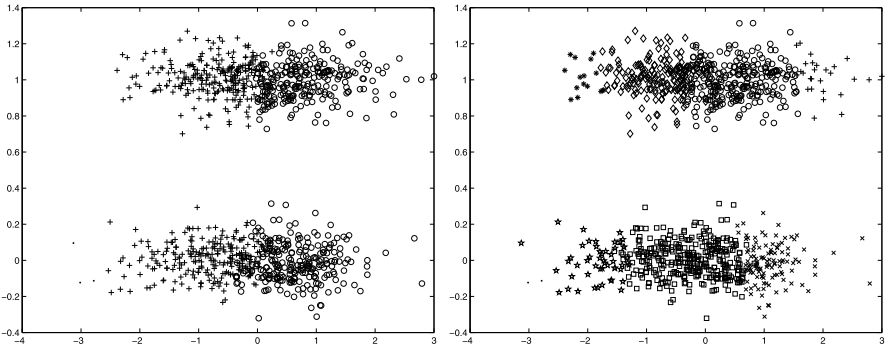


Fig. 9.15 Automatically clustered *Long1* after application of the (**a**) VGAPS clustering technique for $K = 3$ and (**b**) VAMOSA clustering technique for $K = 8$

For the real-life data sets no visualization is possible as these are higher dimensional data sets. For the *Iris* data set, both the *GenClustMOO* and VGAPS clustering techniques are able to detect the appropriate number of partitions, but the *Minkowski score* value attained by the *GenClustMOO* clustering technique is slightly smaller than that obtained by VGAPS (refer to Table 9.3). MOCK and VAMOSA automatically identify $K = 2$ number of clusters for this data set, which is also often obtained for many other methods on *Iris* [188]. For the *Cancer* data set, all five algorithms are able to detect the appropriate number of clusters ($K = 2$), but the MS value obtained by *GenClustMOO* is smaller than those corresponding to MOCK, *GenClustPESA2*, and VGAPS. This in turn indicates that *GenClustMOO* provides better partitioning for this data set than MOCK, *GenClustPESA2* or VGAPS. For the *LungCancer* data set, only VAMOSA is able to detect the appropriate number of partitions, but the MS value attained by *GenClustMOO* is the smallest among

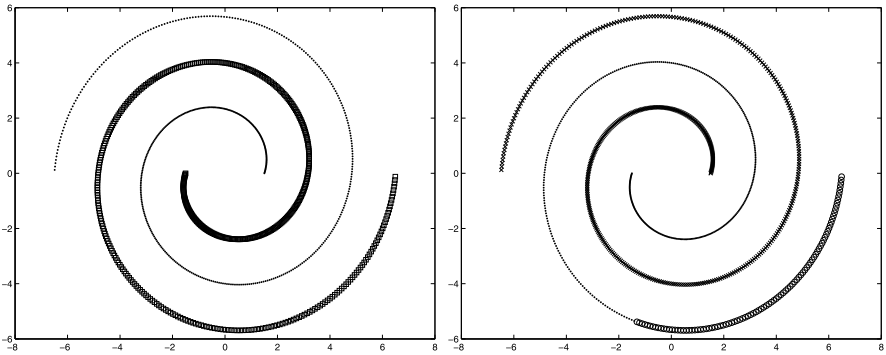


Fig. 9.16 Automatically clustered *Spiral* after application of the (**a**) *GenClustMOO* clustering technique for $K = 2$ and (**b**) MOCK clustering technique for $K = 3$

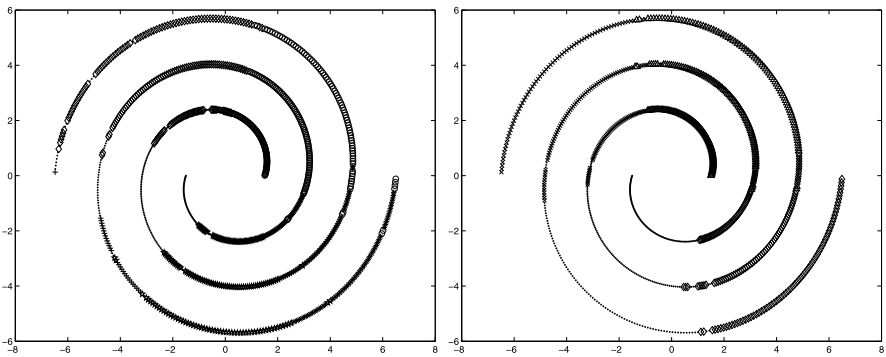


Fig. 9.17 Automatically clustered *Spiral* after application of the (**a**) VGAPS clustering technique for $K = 6$ and (**b**) VAMOSA clustering technique for $K = 4$

algorithms. For the *Glass* data set, only *GenClustMOO* is able to detect the appropriate number of clusters, and the corresponding MS value is also optimum (refer to Table 9.3).

Summary of Results Results on a wide variety of data sets show that *GenClustMOO* is able to detect the appropriate number of partitions and the appropriate partitioning from data sets having many different types of clusters. Results on artificial data sets show that *GenClustMOO* is able to identify various symmetrical-shaped clusters (hyperspheres, linear, elliptical, ring shaped, etc.) having overlaps, as well as some well-separated clusters having any shape. Results on real-life data sets also show that *GenClustMOO* is able to detect partitioning from real-life data sets with varying characteristics. The results on seven artificial and four real-life data sets establish the fact that *GenClustMOO* is wellsuited to detect clusters with

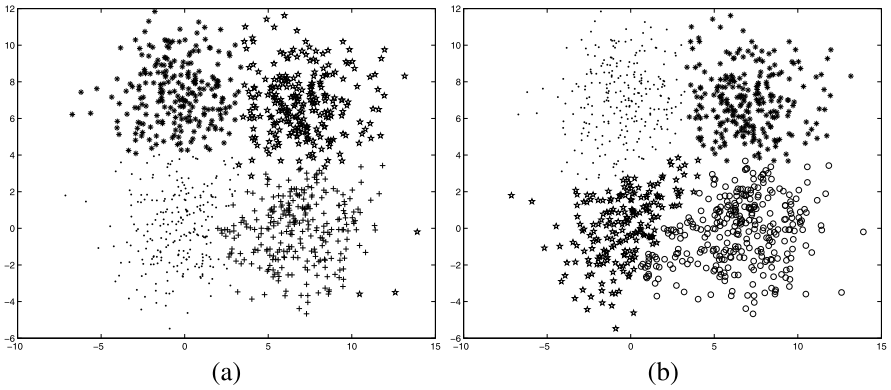


Fig. 9.18 Automatically clustered *Square4* after application of the (**a**) *GenClustMOO* clustering technique for $K = 4$ and (**b**) MOCK clustering technique for $K = 4$

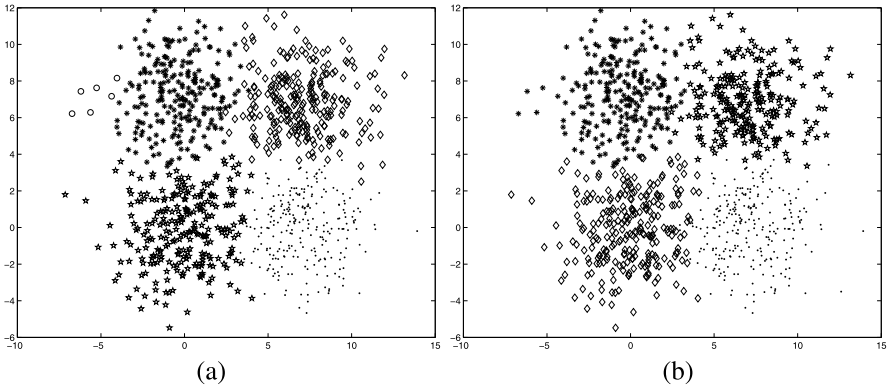


Fig. 9.19 Automatically clustered *Square4* after application of the (**a**) VGAPS clustering technique for $K = 5$ and (**b**) VAMOSA clustering technique for $K = 4$

widely varying characteristics. Results show that, while MOCK is only able to detect well-separated or hyperspherical-shaped clusters well, VGAPS is capable of doing so for symmetrical-shaped clusters, either overlapping or nonoverlapping. The *GenClustMOO* clustering technique is able to find the proper clustering automatically where MOCK succeeds while VGAPS fails, as well as where VGAPS succeeds while MOCK fails. Experimental results also show that the VAMOSA clustering technique is only able to detect the appropriate partitioning automatically from data sets having symmetrical-shaped clusters. The VGAPS and VAMOSA clustering techniques fail when clusters are non-symmetrical in shape, but *GenClustMOO* succeeds for symmetrical as well as well-separated clusters having any shape. In a part of the experiment, we also compared the effectiveness of the underlying multiobjective optimization techniques, AMOSA and PESA-II,

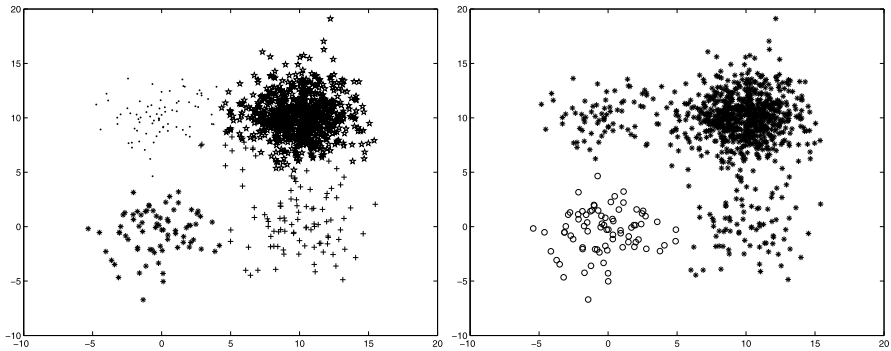


Fig. 9.20 Automatically clustered *Sizes5* after application of the (**a**) *GenClustMOO* clustering technique for $K = 4$ and (**b**) MOCK clustering technique for $K = 2$

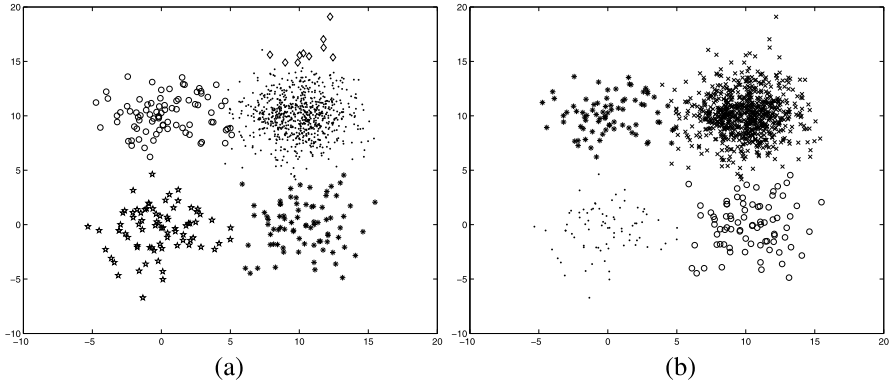


Fig. 9.21 Automatically clustered *Sizes5* after application of the (**a**) VGAPS clustering technique for $K = 5$ and (**b**) VAMOSA clustering technique for $K = 4$

in the clustering algorithm, *GenClustMOO*. The *GenClustPESA2* clustering technique, utilizing PESA-II [65] as the underlying optimization technique in the *GenClustMOO* framework, performs similarly to the *GenClustMOO* clustering technique using AMOSA for data sets with a small number of equisized, equidensity clusters.

The improved performance of *GenClustMOO* can be attributed to the following facts: The use of multicenter approach for each cluster enables it to detect any shaped clusters. The symmetry-based cluster validity index captures the total symmetry present in the obtained partitioning. The use of a relative neighborhood graph to compute the *Con*-index enables it to detect any shaped clusters as long as they are wellseparated. AMOSA, the underlying optimization technique, makes it capable of optimizing three cluster validity indices efficiently.

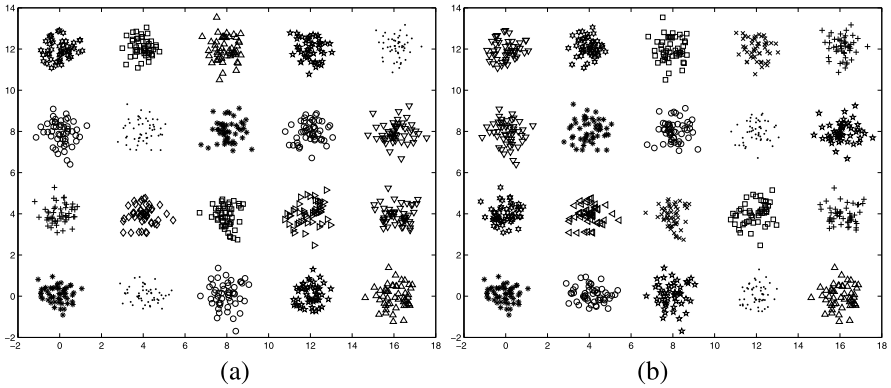


Fig. 9.22 Automatically clustered *Twenty* after application of the (**a**) *GenClustMOO* clustering technique for $K = 20$ and (**b**) MOCK clustering technique for $K = 20$

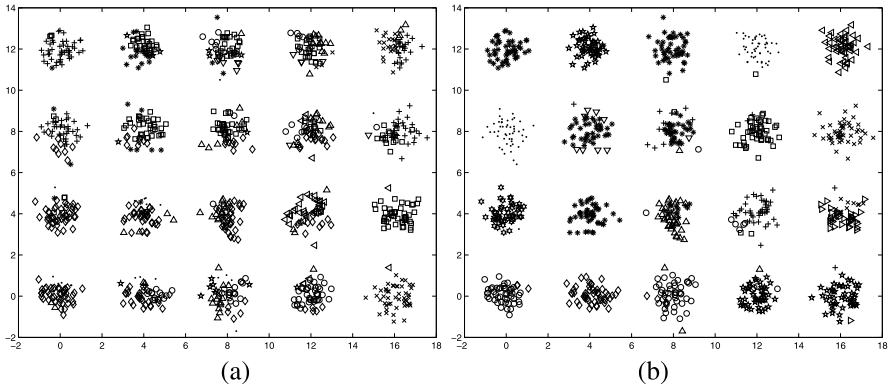


Fig. 9.23 Automatically clustered *Twenty* after application of the (**a**) VGAPS clustering technique for $K = 20$ and (**b**) VAMOSA clustering technique for $K = 20$

9.6 Discussion and Conclusions

In this chapter, some multiobjective clustering techniques are described in detail. These use a newly developed simulated annealing-based multiobjective optimization technique, AMOSA, as the underlying optimization strategy. First, a new multiobjective clustering technique, named MOPS, which can detect the appropriate partitioning from data sets with a specified value of number of clusters, is described. This is the multiobjective extension of the GAPS clustering technique. Results show that MOPS performs better than GAPS for most of the data sets, as long as the clusters in the data set possess the property of symmetry.

Thereafter in this chapter, an automatic multiobjective clustering technique based on symmetry, named VAMOSA, is described. This is the multiobjective version of

the VGAPS clustering technique. Results show that VMAOSA performs much better than VGAPS in determining the number of clusters automatically from data sets with point-symmetric clusters. Finally in this chapter, a generalized clustering technique, named *GenClustMOO*, is described, which is wellsuited to detect the appropriate partitioning from data sets having either point-symmetric or well-separated clusters. Here, multiple cluster centers are used to encode a particular cluster. Three cluster validity indices, namely a Euclidean distance-based cluster validity index, a point symmetry distance-based cluster validity index, and a connectivity-based cluster validity index, are optimized simultaneously. A relative neighborhood graph [282] is utilized to compute the connectivity index. The performance of *GenClustMOO* is compared with the existing multiobjective clustering techniques, MOCK and VAMOSA, and one single-objective clustering technique, VGAPS, for several data sets having different characteristics. In a part of the experiment, the effectiveness of AMOSA as the underlying optimization technique in *GenClustMOO* is also demonstrated in comparison with another evolutionary MO algorithm, PESA-II.

References

1. BrainWeb: Simulated brain database. http://www.bic.mni.mcgill.ca/brainweb
2. UC Irvine Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html
3. Multiobjective simulated annealing. http://www.dcs.ex.ac.uk/people/kismith/mosa/results/tec/
4. Anderberg, M.R.: Cluster Analysis for Application. Academic Press, New York (1973)
5. Anderson, T.W., Sclove, S.L.: Introduction to the Statistical Analysis of Data. Houghton Mifflin, Boston (1978)
6. Andrews, H.C.: Mathematical Techniques in Pattern Recognition. Wiley-Interscience, New York (1972)
7. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), pp. 49–60. ACM, Philadelphia (1999)
8. Asharaf, S., Murty, M.N.: An adaptive rough fuzzy single pass algorithm for clustering large data sets. Pattern Recogn. **36**(12), 3015–3018 (2003)
9. Asharaf, S., Murty, M.N.: Scalable non-linear support vector machine using hierarchical clustering. In: ICPR (1), pp. 908–911 (2006)
10. Asharaf, S., Murty, M.N., Shevade, S.K.: Cluster based core vector machine. In: ICDM, pp. 1038–1042 (2006)
11. Asharaf, S., Shevade, S.K., Murty, M.N.: Rough support vector clustering. Pattern Recogn. **38**(10), 1779–1783 (2005)
12. Attneave, F.: Symmetry information and memory for pattern. Am. J. Psychol. **68**, 209–222 (1995)
13. Babu, G.P., Murty, M.N.: A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm. Pattern Recognit. Lett. **14**(10), 763–769 (1993)
14. Backhaus, K., Erichson, B., Plinke, W., Weiber, R.H.: Multivariate Analysis Methods. An Application-Oriented Introduction (in German). Springer, Berlin (2000)
15. Bandyopadhyay, S.: An automatic shape independent clustering technique. Pattern Recogn. **37**(1), 33–45 (2004)
16. Bandyopadhyay, S.: Simulated annealing using reversible jump Markov chain Monte Carlo algorithm for fuzzy clustering. IEEE Trans. Knowl. Data Eng. **17**(4), 479–490 (2005)
17. Bandyopadhyay, S.: Genetic algorithms for clustering and fuzzy clustering. WIREs Data Min. Knowl. Discov. **1**(6), 524–531 (2011)
18. Bandyopadhyay, S., Maulik, U.: Non-parametric genetic clustering: Comparison of validity indices. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. **31**(1), 120–125 (2001)
19. Bandyopadhyay, S., Maulik, U.: An evolutionary technique based on K-means algorithm for optimal clustering in R^N. Inf. Sci. **146**(1–4), 221–237 (2002)

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2, © Springer-Verlag Berlin Heidelberg 2013

20. Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit. **35**(6), 1197–1208 (2002)
21. Bandyopadhyay, S., Maulik, U., Mukhopadhyay, A.: Multiobjective genetic clustering for pixel classification in remote sensing imagery. IEEE Trans. Geosci. Remote Sens. **45**(5), 1506–1511 (2007)
22. Bandyopadhyay, S., Maulik, U., Pakhira, M.K.: Clustering using simulated annealing with probabilistic redistribution. Int. J. Pattern Recognit. Artif. Intell. **15**(2), 269–285 (2001)
23. Bandyopadhyay, S., Mukhopadhyay, A., Maulik, U.: An improved algorithm for clustering gene expression data. Bioinformatics **23**(21), 2859–2865 (2007)
24. Bandyopadhyay, S., Pal, S.K.: Classification and Learning Using Genetic Algorithms Applications in Bioinformatics and Web Intelligence. Springer, Heidelberg (2007)
25. Bandyopadhyay, S., Pal, S.K., Aruna, B.: Multi-objective GAs, quantitative indices and pattern classification. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **34**(5), 2088–2099 (2004)
26. Bandyopadhyay, S., Pal, S.K., Murthy, C.A.: Simulated annealing based pattern classification. Inf. Sci. **109**(1–4), 165–184 (1998)
27. Bandyopadhyay, S., Saha, S.: GAPS: A clustering method using a new point symmetry based distance measure. Pattern Recognit. **40**(12), 3430–3451 (2007)
28. Bandyopadhyay, S., Saha, S.: A point symmetry based clustering technique for automatic evolution of clusters. IEEE Trans. Knowl. Data Eng. **20**(11), 1–17 (2008)
29. Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K.: A simulated annealing based multiobjective optimization algorithm: AMOSA. IEEE Trans. Evol. Comput. **12**(3), 269–283 (2008)
30. Bargiela, A., Pedrycz, W., Hirota, K.: Granular prototyping in fuzzy clustering. IEEE Trans. Fuzzy Syst. **12**(5), 697–709 (2004)
31. Battiti, R.: Using mutual information for selecting features in supervised neural net learning. IEEE Trans. Neural Netw. **5**, 537–550 (1994)
32. Beliakov, G., King, M.: Density based fuzzy c-means clustering of non-convex patterns. Eur. J. Oper. Res. **173**, 717–728 (2006)
33. Bensaid, A.M., Hall, L.O., Bezdek, J.C., Clarke, L.P., Silbiger, M.L., Arrington, J.A., Murtagh, R.F.: Validity-guided (re)clustering with applications to image segmentation. IEEE Trans. Fuzzy Syst. **4**(2), 112–123 (1996)
34. Bentley, J.L., Weide, B.W., Yao, A.C.: Optimal expected-time algorithms for closest point problems. ACM Trans. Math. Softw. **6**(4), 563–580 (1980)
35. Berg, M.D., Kreveld, M.V., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (2008)
36. Bezdek, J.C.: Fuzzy mathematics in pattern classification. Ph.D. thesis, Cornell University, Ithaca, NY (1973)
37. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York (1981)
38. Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Trans. Syst. Man Cybern. **28**(3), 301–315 (1998)
39. Bezdek, J.C., Pal, S.K. (eds.): Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data. IEEE Press, New York (1992)
40. Bhandarkar, S.M., Zhang, H.: Image segmentation using evolutionary computation. IEEE Trans. Evol. Comput. **3**(1), 1–21 (1999)
41. Bhuyan, J.N., Raghavan, V.V., Elayavalli, V.K.: Genetic algorithm for clustering with an ordered representation. In: Proc. Int. Conf. on Genetic Algorithm '91, pp. 408–415. Morgan Kaufmann, San Mateo (1991)
42. Bouchachia, A., Pedrycz, W.: Data clustering with partial supervision. Data Min. Knowl. Discov. **12**(1), 47–78 (2006)
43. Bouchachia, A., Pedrycz, W.: Enhancement of fuzzy clustering by mechanisms of partial supervision. Fuzzy Sets Syst. **157**(13), 1733–1759 (2006)
44. Bradley, P.S., Fayyad, U.M., Reina, C.: Scaling EM (expectation maximization) clustering to large databases. Tech. rep., Microsoft Research Center (1998)

45. Bradley, P.S., Fayyad, U.M., Reina, C.: Scaling clustering algorithms to large databases. In: Proc. Fourth International Conference on Knowledge Discovery and Data Mining, pp. 9–15 (1998)
46. Bray, J.R., Curtis, J.T.: An ordination of the upland forest communities of southern Wisconsin. Ecol. Monogr. **27**, 325–349 (1957)
47. Calinski, R.B., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat., Theory Methods **3**(1), 1–27 (1974)
48. Campello, R.J., Hruschka, E.R., Alves, V.S.: On the efficiency of evolutionary fuzzy clustering. J. Heuristics **15**(1), 43–75 (2009)
49. Campello, R.J.G.B., Hruschka, E.R.: A fuzzy extension of the silhouette width criterion for cluster analysis. Fuzzy Sets Syst. **157**, 2858–2875 (2007)
50. Carpenter, G., Grossberg, S.: A massively parallel architecture for a self-organizing neural pattern recognition machine. Comput. Vis. Graph. Image Process. **37**(3), 54–115 (1987)
51. Carpenter, G., Grossberg, S.: ART2: Self-organization of stable category recognition codes for analog input patterns. Appl. Opt. **26**(23), 4919–4930 (1987)
52. Caves, R., Quegan, S., White, R.: Quantitative comparison of the performance of SAR segmentation algorithms. IEEE Trans. Image Process. **7**(11), 1534–1546 (1998)
53. Chaoji, V., Hasan, M.A., Salem, S., Zaki, M.J.: SPARCL: Efficient and effective shape-based clustering. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pp. 93–102. IEEE Comput. Soc., Washington (2008). http://dl.acm.org/citation.cfm?id=1510528.1511311
54. Charalampidis, D.: A modified K-means algorithm for circular invariant clustering. IEEE Trans. Pattern Anal. Mach. Intell. **27**(12), 1856–1865 (2005)
55. Chen, Y.L., Hu, H.L.: An overlapping cluster algorithm to provide non-exhaustive clustering. Eur. J. Oper. Res. **173**, 762–780 (2006)
56. Chipperfield, A., Whidborne, J., Fleming, P.: Evolutionary algorithms and simulated annealing for MCDM. In: Multicriteria Decision Making – Advances in MCDM Models, Algorithms, Theory and Applications, pp. 16.1–16.32. Kluwer Academic, Boston (1999)
57. Choi, J.N., Oh, S.K., Pedrycz, W.: Structural and parametric design of fuzzy inference systems using hierarchical fair competition-based parallel genetic algorithms and information granulation. Int. J. Approx. Reason. **49**(3), 631–648 (2008)
58. Chou, C.H., Su, M.C., Lai, E.: Symmetry as a new measure for cluster validity. In: 2nd WSEAS Int. Conf. on Scientific Computation and Soft Computing, Crete, Greece, pp. 209–213 (2002)
59. Chou, C.H., Su, M.C., Lai, E.: A new cluster validity measure and its application to image compression. Pattern Anal. Appl. **7**(2), 205–220 (2004)
60. Chung, K.L., Lin, J.S.: Faster and more robust point symmetry-based K-means algorithm. Pattern Recognit. **40**(2), 410–422 (2007)
61. Chung, K.L., Lin, K.S.: An efficient line symmetry-based K-means algorithm. Pattern Recognit. Lett. **27**(7), 765–772 (2006)
62. Coello Coello, C.A.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inf. Syst. **1**(3), 129–156 (1999)
63. Coello Coello, C.A., Veldhuizen, D.V., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic, Boston (2002)
64. Cole, R.M.: Clustering with genetic algorithms. Master's thesis, Department of Computer Science, University of Western Australia, Australia (1998)
65. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: Region-based selection in evolutionary multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 283–290. Morgan Kaufmann, San Francisco (2001). citeseer.ist.psu.edu/corne01pesaii.html
66. Corne, D.W., Knowles, J.D., Oates, M.J.: The Pareto-envelope based selection algorithm for multiobjective optimisation. In: Proceedings of the Parallel Problem Solving from Nature – PPSN VI, Springer Lecture Notes in Computer Science, pp. 869–878 (2000)

67. Cowgill, M.C., Harvey, R.J., Watson, L.T.: A genetic algorithm approach to cluster analysis. Comput. Math. Appl. **37**(7), 99–108 (1999)
68. Czyzak, P., Jaszkiewicz, A.: Pareto simulated annealing – A metaheuristic technique for multiple-objective combinatorial optimization. J. Multi-Criteria Decis. Anal. **7**(1), 34–47 (1998)
69. Das, I., Dennis, J.: A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Struct. Optim. **14**(1), 63–69 (1997)
70. Das, S., Abraham, A., Konar, A.: Automatic clustering using an improved differential evolution algorithm. IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum. **38**(1), 218–237 (2008)
71. Dave, R.N.: Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. Intell. Robots Comput. Vis. VIII **1192**, 600–611 (1989)
72. Dave, R.N., Bhaswan, K.: Adaptive fuzzy c-shells clustering and detection of ellipses. IEEE Trans. Neural Netw. **3**(5), 643–662 (1992)
73. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(4), 224–227 (1979)
74. Davis, L. (ed.): Genetic Algorithms and Simulated Annealing. Morgan Kaufmann, Los Altos (1987)
75. Davis, L. (ed.): Handbook of Genetic Algorithms. Van Nostrand-Reinhold, New York (1991)
76. Deb, K.: Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evol. Comput. **7**(3), 205–230 (1999)
77. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley, England (2001)
78. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
79. DeJong, K.: Learning with genetic algorithms: An overview. Mach. Learn. **3**(2-3), 121–138 (1988)
80. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc., Ser. B **39**(1), 1–38 (1977)
81. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice Hall, London (1982)
82. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer, Berlin (2009)
83. Dong, Y.Y., Zhang, Y.J., Chang, C.L.: Multistage random sampling genetic-algorithm-based fuzzy c-means clustering algorithm. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2069–2073 (2004)
84. Dubes, R.C., Jain, A.K.: Clustering techniques: The user's dilemma. Pattern Recognit. **8**(4), 247–260 (1976)
85. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
86. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
87. Dunn, J.C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J. Cybern. **3**(3), 32–57 (1973)
88. Eduardo, R.H., Nelson, F.F.E.: A genetic algorithm for cluster analysis. Intell. Data Anal. **7**, 15–25 (2003)
89. Emmanouilidis, C., Hunter, A., MacIntyre, J.: A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In: Proceedings of the 2000 Congress on Evolutionary Computation CEC00, pp. 309–316. IEEE Press, La Jolla (2000). citeseer.nj.nec.com/emmanouilidis00multiobjective.html
90. Engrand, P.: A multi-objective approach based on simulated annealing and its application to nuclear fuel management. In: 5th International Conference on Nuclear Engineering, Nice, France, pp. 416–423 (1997)
91. Erickson, M., Mayer, A., Horn, J.: Multi-objective optimal design of groundwater remediation systems: Application of the niched Pareto genetic algorithm (NPGA). Adv. Water Resour. **25**(1), 51–65 (2002)

92. Ester, M., Kriegal, H.P., Sander, J.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the Second International Conference on Knowledge Discovery and Data-mining, pp. 226–231. AAAI Press, Menlo Park (1996)
93. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: Density-based algorithm for discovering clusters in large spatial databases. In: Proceedings of the Second International Conference on Data Mining (KDD'96), Portland, OR, pp. 226–231 (1996)
94. Estivill-Castro, V., Murray, A.T.: Spatial clustering for data mining with genetic algorithms. In: Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, pp. 317–323 (1997)
95. Everitt, B.S.: Cluster Analysis, 3rd edn. Halsted, New York (1993)
96. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis. Arnold, London (2001)
97. Everitt, B.: Graphical Techniques for Multivariate Data. Heinemann Educational, London (1978)
98. Falkenauer, E.: Genetic Algorithms and Grouping Problems. Wiley, New York (1998)
99. Fieldsend, J., Everson, R., Singh, S.: Using unconstrained elite archives for multi-objective optimisation. IEEE Trans. Evol. Comput. **7**(3), 305–323 (2003)
100. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annu. Eugen. **3**, 179–188 (1936)
101. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. **3**(1), 1–16 (1995)
102. Fränti, P., Kivijärvi, J., Kaukoranta, T., Nevalainen, O.: Genetic algorithms for large scale clustering problems. Comput. J. **40**, 547–554 (1997)
103. Friedman, J.H., Bently, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. **3**(3), 209–226 (1977)
104. Friedman, M., Kandel, A.: Introduction to Pattern Recognition, Statistical, Structural, Neural and Fuzzy Logic Approaches. World Scientific, Singapore (1999)
105. Fu, K.S.: Syntactic Pattern Recognition and Applications. Academic Press, London (1982)
106. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, New York (1990)
107. Fukuyama, Y., Sugeno, M.: A new method of choosing the number of clusters for the fuzzy c-means method. In: Proc. of the Fifth Fuzzy Systems Symposium, pp. 247–250 (1989)
108. Gan, G., Ma, C., Wu, J.: Data Clustering – Theory, Algorithms, and Applications. SIAM, Philadelphia (2007)
109. Gath, I., Geva, A.B.: Unsupervised optimal fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **11**(7), 773–781 (1989)
110. Gelsema, E.S., Kanal, L. (eds.): Pattern Recognition in Practice II. North-Holland, Amsterdam (1986)
111. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**(6), 721–741 (1984)
112. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, New York (1989)
113. Gonzalez, R.C., Thomason, M.G.: Syntactic Pattern Recognition: An Introduction. Addison-Wesley, Reading (1978)
114. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley, Massachusetts (1992)
115. Grabusts, P., Borisov, A.: Using grid-clustering methods in data classification. In: 2002 International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002), 22–25 September 2002, Warsaw, Poland, pp. 425–426. IEEE Comput. Soc., Los Alamitos (2002)
116. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. **16**, 122–128 (1986)
117. Grira, N., Houle, M.E.: Best of both: A hybridized centroid-medoid clustering heuristic. In: ICML '07: Proceedings of the 24th International Conference on Machine Learning, pp. 313–320. ACM, New York (2007)

118. Grubbs, F.E.: Procedures for detecting outlying observations in samples. Technometrics **11**, 1–21 (1969)
119. Gustafson, D.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. Proc. IEEE Conf. Decision Contr. **17**, 761–766 (1979)
120. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
121. Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE Trans. Evol. Comput. **11**(1), 56–76 (2007)
122. Hansen, P., Mladenovic, N.: J-means: A new local search heuristic for minimum sum of squares clustering. Pattern Recognit. **34**(2), 405–413 (2001)
123. Hapke, M., Jaszkiewicz, A., Slowinski, R.: Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. J. Heuristics **6**(3), 329–345 (2000)
124. Harmon, L.D., Hunt, W.F.: Automatic recognition of human face profiles. Comput. Graph. Image Process. **6**(2), 135–156 (1977)
125. Hartigan, J.A.: Clustering Algorithms. Wiley, New York (1975)
126. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Inf. Process. Lett. **76**, 175–181 (2000). http://dl.acm.org/citation.cfm?id=364456.364469
127. Hartwig, F., Dearing, B.: Exploratory Data Analysis. Sage, Thousand Oaks (1979)
128. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley, Reading (1991)
129. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
130. Höppner, F.: Fuzzy shell clustering algorithms in image processing: Fuzzy c-rectangular and 2-rectangular shells. IEEE Trans. Fuzzy Syst. **5**(4), 599–613 (1997)
131. Hruschka, E.R., Campello, R.J.G.B., de Castro, L.N.: Evolutionary algorithms for clustering gene-expression data. In: Proc. 4th IEEE Int. Conference on Data Mining, pp. 403–406 (2004)
132. Hruschka, E.R., Campello, R.J.G.B., de Castro, L.N.: Improving the efficiency of a clustering genetic algorithm. In: Proc. 9th Ibero-American Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol. 3315, pp. 861–870 (2004)
133. Hruschka, E.R., Campello, R.J.G.B., de Castro, L.N.: Evolving clusters in gene-expression data. Inf. Sci. **176**(13), 1898–1927 (2006)
134. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., de Carvalho, A.C.P.L.F.: A survey of evolutionary algorithms for clustering. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. **39**(2), 133–155 (2009)
135. Hruschka, E.R., Ebecken, N.F.F.: A genetic algorithm for cluster analysis. Intell. Data Anal. **7**(1), 15–25 (2003)
136. Hu, M.K.: Visual pattern recognition by moment invariants. IEEE Trans. Inf. Theory **8**(2), 179–187 (1962)
137. Hughes, E.J.: Evolutionary many-objective optimization: Many once or one many. In: Proceedings of 2005 Congress on Evolutionary Computation, Edinburgh, Scotland, UK, September 2–5, 2005, pp. 222–227 (2005)
138. Ingber, L.: Very fast simulated re-annealing. Math. Comput. Model. **12**(8), 967–973 (1989)
139. Ishibuchi, H., Doi, T., Nojima, Y.: Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. In: Parallel Problem Solving from Nature IX (PPSN-IX), vol. 4193, pp. 493–502 (2006)
140. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. **28**(3), 392–403 (1998)
141. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans. Evol. Comput. **6**(6), 721–741 (1984)
142. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull. Soc. Vaud. Sci. Nat. **37**, 547–579 (1901)

143. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
144. Jain, A.K., Duin, P., Jianchang, M.: Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 4–37 (2000)
145. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. **31**(3), 264–323 (1999)
146. Jardine, N., Sibson, R.: Mathematical Taxonomy. Wiley, New York (1971)
147. Jaszkiewicz, A.: Comparison of local search-based metaheuristics on the multiple objective knapsack problem. Found. Comput. Dec. Sci. **26**(1), 99–120 (2001)
148. Jolliffe, I.: Principal Component Analysis. Springer Series in Statistics. Springer, England (1986)
149. Kandel, A.: Fuzzy Techniques in Pattern Recognition. Wiley-Interscience, New York (1982)
150. Kandel, A.: Fuzzy Mathematical Techniques with Applications. Addison-Wesley, New York (1986)
151. Kankanala, L., Murty, M.N.: Hybrid approaches for clustering. In: PReMI, pp. 25–32 (2007)
152. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient K-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002)
153. Kaufman, L., Rosseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990)
154. Kim, D.J., Park, Y.W., Park, D.J.: A novel validity index for determination of the optimal number of clusters. IEICE Trans. Inf. Syst. **D-E84**(2), 281–285 (2001)
155. Kim, D.W., Lee, K.H., Lee, D.: Fuzzy cluster validation index based on inter-cluster proximity. Pattern Recognit. Lett. **24**(15), 2561–2574 (2003)
156. Kim, T.H., Barrera, L.O., Zheng, M., Qu, C., Singer, M.A., Richmond, T.A., Wu, Y., Green, R.D., Ren, B.: A high-resolution map of active promoters in the human genome. Nature **436**, 876–880 (2005)
157. Kim, Y.I., Kim, D.W., Lee, D., Lee, K.H.: A cluster validation index for GK cluster analysis based on relative degree of sharing. Inf. Sci. **168**(1–4), 225–242 (2004)
158. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative studies. J. Stat. Phys. **34**(5/6), 975–986 (1984)
159. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
160. Kirpatrick, S., Vecchi, M.P.: Global wiring by simulated annealing. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **CAD-2**(4), 215–222 (1983)
161. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evol. Comput. **8**(2), 149–172 (2000)
162. Kohonen, T.: The 'neural' phonetic typewriter. IEEE Comput. **27**(3), 11–12 (1988)
163. Kohonen, T.: Self-Organization and Associative Memory, 3rd edn. Springer, New York (1989)
164. Konak, A., Coit, D., Smith, A.: Multi-objective optimization using genetic algorithms: A tutorial. Reliab. Eng. Syst. Saf. **91**(9), 992–1007 (2006). http://linkinghub.elsevier.com/retrieve/pii/S0951832005002012
165. Korkmaz, E.E., Du, J., Alhajj, R., Barker, K.: Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. Intell. Data Anal. **10**(2), 163–182 (2006)
166. Kövesi, B., Boucher, J.M., Saoodi, S.: Stochastic K-means algorithm for vector quantization. Pattern Recognit. Lett. **22**(6–7), 603–610 (2001)
167. Krishna, K., Murty, M.N.: Genetic K-means algorithm. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **29**(3), 433–439 (1999)
168. Krishnapuram, R., Nasraoui, O., Frigui, H.: The fuzzy c-spherical shells algorithm: A new approach. IEEE Trans. Neural Netw. **3**(5), 663–671 (1992)
169. Krovi, R.: Genetic algorithms for clustering: A preliminary investigation. In: Proceedings of the 25th Hawaii Int. Conference on System Sciences, vol. 4, pp. 540–544 (1992)

170. Kuncheva, L.I., Bezdek, J.C.: Selection of cluster prototypes from data by a genetic algorithm. In: Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing, pp. 1683–1688 (1997)
171. Kwanghoon, S., Jung, K.H., Alexander, W.E.: A mean field annealing approach to robust corner detection. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **28**(1), 82–90 (1998)
172. Kwon, S.H.: Cluster validity index for fuzzy clustering. Electron. Lett. **34**(22), 2176–2177 (1998)
173. Lance, G., Williams, W.: Mixed-data classificatory programs. I. Agglomerative systems. Aust. Comput. J. **1**, 15–20 (1967)
174. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Comput. **16**(6), 1299–1323 (2004)
175. Laszlo, M., Mukherjee, S.: A genetic algorithm using hyper-quadtrees for low-dimensional K-means clustering. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 533–543 (2006)
176. Leon, E., Nasraoui, O., Gomez, J.: ECSAGO: Evolutionary clustering with self adaptive genetic operators. In: Proc. IEEE Congress on Evolutionary Computation, July 16–21, 2006, pp. 1768–1775 (2006)
177. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recognit. **36**, 451–461 (2003)
178. Lin, J.Y., Peng, H., Xie, J.M., Zheng, Q.L.: Novel clustering algorithm based on central symmetry. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 26–29 August 2004, vol. 3, pp. 1329–1334 (2004)
179. Loia, V., Pedrycz, W., Senatore, S.: Semantic web content analysis: A study in proximity-based collaborative clustering. IEEE Trans. Fuzzy Syst. **15**(6), 1294–1312 (2007)
180. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.J.: FGKA: A fast genetic K-means clustering algorithm. In: SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 622–623. ACM, New York (2004)
181. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.J.: Incremental genetic K-means algorithm and its application in gene expression data analysis. BMC Bioinform. **5**, 172 (2004)
182. Lucasius, C.B., Dane, A.D., Kateman, G.: On K-medoid clustering of large data sets with the aid of a genetic algorithm: Background, feasibility and comparison. Anal. Chim. Acta **282**, 647–669 (1993)
183. Ma, P.C.H., Chan, K.C.C., Yao, X., Chiu, D.K.Y.: An evolutionary clustering algorithm for gene expression microarray data analysis. IEEE Trans. Evol. Comput. **10**(3), 296–314 (2006)
184. Man, Y., Gath, I.: Detection and separation of ring-shaped clusters using fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **16**(8), 855–861 (1994)
185. Mao, J., Jain, A.K.: A self-organizing network for hyperellipsoidal clustering. IEEE Trans. Neural Netw. **7**(1), 16–29 (1996)
186. Marden, J.I.: Analyzing and Modeling Rank Data. Chapman & Hall, London (1995)
187. Matake, N., Hiroyasu, T., Miki, M., Senda, T.: Multiobjective clustering with automatic k-determination for large-scale data. In: GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 861–868. ACM, New York (2007)
188. Maulik, U., Bandyopadhyay, S.: Genetic algorithm based clustering technique. Pattern Recognit. **33**(9), 1455–1465 (2000)
189. Maulik, U., Bandyopadhyay, S.: Performance evaluation of some clustering algorithms and validity indices. IEEE Trans. Pattern Anal. Mach. Intell. **24**(12), 1650–1654 (2002)
190. Maulik, U., Bandyopadhyay, S.: Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. IEEE Trans. Geosci. Remote Sens. **41**(5), 1075–1081 (2003)
191. Maulik, U., Bandyopadhyay, S., Trinder, J.: SAFE: An efficient feature extraction technique. J. Knowl. Inf. Syst. **3**(3), 374–387 (2001)
192. Maulik, U., Mukhopadhyay, A., Bandyopadhyay, S.: Combining Pareto-optimal clusters using supervised learning for identifying co-expressed genes. BMC Bioinform. **27**, 1197–1208 (2009)

193. Maulik, U., Bandyopadhyay, S., Mukhopadhyay, A.: Multiobjective Genetic Algorithms for Clustering – Applications in Data Mining and Bioinformatics. Springer, Heidelberg (2011)
194. Merz, P., Zell, A.: Clustering gene expression profiles with memetic algorithms. In: PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, pp. 811–820. Springer, London (2002)
195. Metropolis, N., Rosenbluth, A.W., Rosenbloth, M.N., Teller, A.H., Teller, E.: Equation of state calculation by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
196. Mezzich, J.E.: Evaluating clustering methods for psychiatric-diagnosis. Biol. Psychiatry **13**, 265–281 (1978)
197. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1992)
198. Mikheev, A., Vincent, L., Faber, V.: High-quality polygonal contour approximation based on relaxation. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, p. 361. IEEE Comput. Soc., Washington (2001). http://dl.acm.org/citation.cfm?id=876867.877738
199. Milligan, G.: An algorithm for generating artificial test clusters. Psychometrika **50**(1), 123–127 (1981)
200. Milligan, G.W., Cooper, C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika **50**(2), 159–179 (1985)
201. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
202. Mount, D.M., Arya, S.: ANN: A library for approximate nearest neighbor searching (2005). http://www.cs.umd.edu/~mount/ANN
203. Mukhopadhyay, A., Maulik, U.: Unsupervised pixel classification in satellite imagery using multiobjective fuzzy clustering combined with SVM classifier. IEEE Trans. Geosci. Remote Sens. **47**(4), 1132–1138 (2009)
204. Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S.: Multi-objective genetic algorithm based fuzzy clustering of categorical attributes. IEEE Trans. Evol. Comput. **13**(5), 991–1005 (2009)
205. Murthy, C.A., Chowdhury, N.: In search of optimal clusters using genetic algorithms. Pattern Recognit. Lett. **17**(8), 825–832 (1996)
206. Nam, D., Park, C.H.: Pareto-based cost simulated annealing for multiobjective optimization. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02), vol. 2, pp. 522–526. Nanyang Technical University, Orchid Country Club, Singapore (2002)
207. Nam, D.K., Park, C.H.: Multiobjective simulated annealing: A comparative study to evolutionary algorithms. Int. J. Fuzzy Syst. **2**(2), 87–97 (2000)
208. Ng, R., Han, J.: Efficient and effective clustering method for spatial data mining. In: Proceedings of the 1994 International Conference on Very Large Data Bases, Santiago, Chile, pp. 144–155 (1994)
209. Ng, R., Han, J.: Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In: Proceedings of the 4th International Symposium on Large Spatial Databases (SSD'95), Portland, ME, pp. 67–82 (1995)
210. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, Heidelberg (2007)
211. Nock, R., Nielsen, F.: On weighting clustering. IEEE Trans. Pattern Anal. Mach. Intell. **28**(8), 1223–1235 (2006)
212. Pakhira, M.K., Maulik, U., Bandyopadhyay, S.: Validity index for crisp and fuzzy clusters. Pattern Recognit. **37**(3), 487–501 (2004)
213. Pal, P., Chanda, B.: A symmetry based clustering technique for multi-spectral satellite imagery. In: ICVGIP (2002)
214. Pal, S.K.: Fuzzy set theoretic measures for automatic feature evaluation – II. Inf. Sci. **64**, 165–179 (1992)
215. Pal, S.K., Majumder, D.D.: Fuzzy Mathematical Approach to Pattern Recognition. Wiley, New York (1986)

216. Pal, S.K., Mandal, D.P.: Linguistic recognition system based on approximate reasoning. Inf. Sci. **61**, 135–161 (1992)
217. Pal, S.K., Mitra, S.: Fuzzy versions of Kohonen's net and MLP-based classification: Performance evaluation for certain nonconvex decision regions. Inf. Sci. **76**, 297–337 (1994)
218. Park, Y.J., Song, M.S.: A genetic algorithm for clustering problems. In: Proc. 3rd Annual Conference on Genetic Programming, Paris, France, pp. 568–575 (1998)
219. Pavlidis, T.: Structural Pattern Recognition. Springer, Berlin (1977)
220. Pedrycz, W.: A fuzzy cognitive structure for pattern recognition. Pattern Recognit. Lett. **9**(5), 305–313 (1989)
221. Pedrycz, W.: Fuzzy sets in pattern recognition: Methodology and methods. Pattern Recognit. **23**, 121–146 (1990)
222. Pedrycz, W.: Fuzzy clustering with a knowledge-based guidance. Pattern Recognit. Lett. **25**(4), 469–480 (2004)
223. Pedrycz, W.: Knowledge-based clustering in computational intelligence. In: Challenges for Computational Intelligence, pp. 317–341. Springer, Heidelberg (2007)
224. Pedrycz, W.: A dynamic data granulation through adjustable fuzzy clustering. Pattern Recognit. Lett. **29**(16), 2059–2066 (2008)
225. Pedrycz, W., Amato, A., Lecce, V.D., Piuri, V.: Fuzzy clustering with partial supervision in organization and classification of digital images. IEEE Trans. Fuzzy Syst. **16**(4), 1008–1026 (2008)
226. Pedrycz, W., Hirota, K.: A consensus-driven fuzzy clustering. Pattern Recognit. Lett. **29**(9), 1333–1343 (2008)
227. Pedrycz, W., Loia, V., Senatore, S.: P-FCM: A proximity-based fuzzy clustering. Fuzzy Sets Syst. **148**(1), 21–41 (2004)
228. Pedrycz, W., Rai, P.: Collaborative clustering with the use of fuzzy c-means and its quantification. Fuzzy Sets Syst. **159**(18), 2399–2427 (2008)
229. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
230. Raftery, A.: A note on Bayes factors for log-linear contingency table models with vague prior information. J. R. Stat. Soc. **48**(2), 249–250 (1986)
231. Rechenberg, I.: Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
232. Richards, J.A.: Remote Sensing Digital Image Analysis: An Introduction. Springer, New York (1993)
233. Ripon, K.S.N., Tsang, C.H., Kwong, S., Ip, M.K.: Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In: ICPR'06: Proceedings of the 18th International Conference on Pattern Recognition, pp. 1200–1203. IEEE Comput. Soc., Washington (2006)
234. Rousseeuw, P.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
235. Ruck, D.W., Rogers, S.K., Kabrisky, M.: Feature selection using a multilayer perceptron. Network **2**(2), 1–14 (1990). http://portal.acm.org.offcampus.lib.washington.edu/citation.cfm?id=1497653.1498412
236. Rudolph, G.: Convergence analysis of canonical genetic algorithms. IEEE Trans. Neural Netw. **5**(1), 96–101 (1994)
237. Runyon, R., Haber, A.: Fundamentals of Behavioral Statistics. Addison-Wesley, Reading (1976)
238. Saha, S., Bandyopadhyay, S.: A new line symmetry distance and its application to data clustering. J. Comput. Sci. Technol. **24**(3), 544–556 (2009)
239. Saha, S., Bandyopadhyay, S.: A new multiobjective simulated annealing based clustering technique using symmetry. Pattern Recognit. Lett. **30**(15), 1392–1403 (2009)
240. Saha, S., Bandyopadhyay, S.: A new point symmetry based fuzzy genetic clustering technique for automatic evolution of clusters. Inf. Sci. **179**(19), 3230–3246 (2009)
241. Saha, S., Bandyopadhyay, S.: A new symmetry based multiobjective clustering technique for automatic evolution of clusters. Pattern Recognit. **43**(3), 738–751 (2010)

242. Saha, S., Bandyopadhyay, S.: A generalized automatic clustering algorithm in a multiobjective framework. Appl. Soft Comput. **13**(1), 89–108 (2013)
243. Saha, S., Bandyopadhyay, S.: Application of a new symmetry based cluster validity index for satellite image segmentation. IEEE Geosci. Remote Sens. Lett. **5**(2), 166–170 (2008)
244. Saha, S., Bandyopadhyay, S.: Performance evaluation of some symmetry based cluster validity indices. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. **39**(4), 420–425 (2009)
245. Saha, S., Bandyopadhyay, S.: A validity index based on connectivity. In: ICAPR, pp. 91–94. IEEE Comput. Soc., Los Alamitos (2009)
246. Saha, S., Bandyopadhyay, S.: On principle axis based line symmetry clustering techniques. Memetic Comput. **3**(2), 129–144 (2011)
247. Saha, S., Maulik, U.: Use of symmetry and stability for data clustering. Evol. Intell. **3**(3-4), 103–122 (2010)
248. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. Data Min. Knowl. Discov. **2**(2), 169–194 (1998)
249. Schaffer, J.: Multiple objective optimization with vector evaluated genetic algorithms. In: Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100 (1985)
250. Schott, J.R.: Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Boston, MA (1995)
251. Selim, S.Z., Ismail, M.A.: K-means type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 81–87 (1984)
252. Serafini, P.: Simulated annealing for multiple objective optimization problems. In: Proceedings of the Tenth International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application, vol. 1, pp. 283–292. Springer, Berlin (1994)
253. Sheng, W., Liu, X.: A hybrid algorithm for k-medoid clustering of large data sets. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 77–82 (2004)
254. Sheng, W., Swift, S., Zhang, L., Liu, X.: A weighted sum validity function for clustering with a hybrid niching genetic algorithm. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **35**(6), 56–67 (2005)
255. Siedlecki, W., Sklansky, J.: A note on genetic algorithms for large-scale feature selection. Pattern Recognit. Lett. **10**, 335–347 (1989). http://dl.acm.org/citation.cfm?id=78354.78362
256. Alves, V.S., Campello, R.J.G.B., Hruschka, E.R.: Towards a fast evolutionary algorithm for clustering. In: Proc. IEEE Congress on Evolutionary Computation, pp. 6240–6247 (2006)
257. Smith, K.I., Everson, R.M., Fieldsend, J.E.: Dominance measures for multi-objective simulated annealing. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'04), pp. 23–30 (2004)
258. Smith, K.I., Everson, R.M., Fieldsend, J.E., Murphy, C., Misra, R.: Dominance-based multiobjective simulated annealing. IEEE Trans. Evol. Comput. **12**(3), 323–342 (2008)
259. Sontag, E., Sussman, H.: Image restoration and segmentation using the annealing algorithm. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **CAD-2**(4), 215–222 (1983)
260. Spath, H.: Cluster Analysis Algorithms. Ellis Horwood, Chichester (1989)
261. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**(3), 221–248 (1994)
262. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994)
263. Staiano, A., Tagliaferri, R., Pedrycz, W.: Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering. Neurocomputing **69**(13-15), 1570–1581 (2006)
264. Storn, R., Price, K.: Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
265. Su, M.C., Chou, C.H.: Application of associative memory in human face detection. In: 1999 International Joint Conference on Neural Networks, pp. 3194–3197 (1999)

266. Su, M.C., Chou, C.H.: A modified version of the K-means algorithm with a distance based on cluster symmetry. IEEE Trans. Pattern Anal. Mach. Intell. **23**(6), 674–680 (2001)
267. Su, M.C., DeClaris, N., Kang Liu, T.: Application of neural networks in cluster analysis. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1997, 'Computational Cybernetics and Simulation', Orlando, FL, vol. 1, pp. 1–6 (1997)
268. Su, M.C., Liu, Y.C.: A new approach to clustering data with arbitrary shapes. Pattern Recognit. **38**, 1887–1901 (2005)
269. Suckling, J., Sigmundsson, T., Greenwood, K., Bullmore, E.: A modified fuzzy clustering algorithm for operator independent brain tissue classification of dual echo MR images. J. Magn. Reson. Imaging **17**(7), 1065–1076 (1999)
270. Suman, B.: Study of self-stopping PDMOSA and performance measure in multiobjective optimization. Comput. Chem. Eng. **29**(5), 1131–1147 (2005)
271. Suman, B.: Multiobjective simulated annealing – A metaheuristic technique for multiobjective optimization of a constrained problem. Found. Comput. Dec. Sci. **27**(3), 171–191 (2002)
272. Suman, B.: Simulated annealing based multiobjective algorithm and their application for system reliability. Eng. Optim. **35**(4), 391–416 (2003)
273. Suman, B.: Study of simulated annealing based multiobjective algorithm for multiobjective optimization of a constrained problem. Comput. Chem. Eng. **28**(9), 1849–1871 (2004)
274. Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multiobjective optimization. J. Oper. Res. Soc. **57**(10), 1143–1160 (2006)
275. Suppapitnarm, A., Seffen, K., Parks, G., Clarkson, P.: A simulated annealing algorithm for multiobjective optimization. Eng. Optim. **33**(1), 59–85 (2000)
276. Szu, H.H., Hartley, R.L.: Fast simulated annealing. Phys. Lett. A **122**(3–4), 157–162 (1987)
277. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Springer, Berlin (2005)
278. Teknomo, K.: Similarity measurement. http://people.revoledu.com/kardi/tutorial/Similarity/
279. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 3rd edn. Academic Press, Orlando (2006)
280. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters via the gap statistics. J. R. Stat. Soc. **63**, 411–423 (2001)
281. Tou, J.T., Gonzalez, R.C.: Pattern Recognition Principles. Addison-Wesley, Reading (1974)
282. Toussaint, G.T.: The relative neighborhood graph of a finite planar set. Pattern Recognit. **12**, 261–268 (1980)
283. Toussaint, G.T.: Pattern recognition and geometrical complexity. In: Proc. Fifth International Conf. on Pattern Recognition, Miami Beach, December 1980, pp. 1324–1347 (1980)
284. Tseng, L., Yang, S.: Genetic algorithms for clustering, feature selection, and classification. In: Proceedings of the IEEE International Conference on Neural Networks, Houston, pp. 1612–1616 (1997)
285. Tuyttens, D., Teghem, J., El-Sherbeny, N.: A particular multiobjective vehicle routing problem solved by simulated annealing. In: Metaheuristics for Multiobjective Optimization, vol. 535, 133–152 (2003)
286. Ulungu, E.L., Teghaem, J., Fortemps, P., Tuyttens, D.: MOSA method: A tool for solving multiobjective combinatorial decision problems. J. Multi-Criteria Decis. Anal. **8**(4), 221–236 (1999)
287. Varma, S., Asharaf, S., Murty, M.N.: Rough core vector clustering. In: PReMI, pp. 304–310 (2007)
288. Vijaya, P.A., Murty, M.N., Subramanian, D.K.: Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognit. Lett. **25**(4), 505–513 (2004)
289. Vijaya, P.A., Murty, M.N., Subramanian, D.K.: An efficient hybrid hierarchical agglomerative clustering (HHAC) technique for partitioning large data sets. In: PReMI, pp. 583–588 (2005)
290. Wang, W., Zhang, Y.: On fuzzy cluster validity indices. Fuzzy Sets Syst. **158**(19), 2095–2117 (2007)
291. Wang, W., Yang, J., Muntz, R.R.: STING: A statistical information grid approach to spatial data mining. In: Proceedings of the 23rd International Conference on Very Large Data

Bases, VLDB'97, pp. 186–195. Morgan Kaufmann, San Francisco (1997). http://dl.acm.org/citation.cfm?id=645923.758369
292. Wang, W., Yang, J., Muntz, R.R.: STING+: An approach to active spatial data mining. In: Proceedings of the 15th IEEE International Conference on Data Engineering, Sydney, Australia, March 1999, pp. 116–125 (1999)
293. Webb, A.: Statistical Pattern Recognition. Wiley, Chichester (2002)
294. Wong, C., Chen, C., Su, M.: A novel algorithm for data clustering. Pattern Recognit. **34**, 425–442 (2001)
295. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **13**(8), 841–847 (1991)
296. Xu, R.: II, D.W.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
297. Yan, H.: Fuzzy curve-tracing algorithm. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **31**(5), 768–780 (2001)
298. Yao, X.: A new simulated annealing algorithm. Int. J. Comput. Math. **56**, 161–168 (1995)
299. Yip, A.M., Ding, C., Chan, T.F.: Dynamic cluster formation using level set methods. IEEE Trans. Pattern Anal. Mach. Intell. **28**(6), 877–889 (2006)
300. Zabrodsky, H., Peleg, S., Avnir, D.: Symmetry as a continuous feature. IEEE Trans. Pattern Anal. Mach. Intell. **17**(12), 1154–1166 (1995)
301. Zadeh, L.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
302. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. In: Proc. of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 103–114 (1996)
303. Zhang, Y., Brady, M., Smith, S.: A hidden Markov random field model for segmentation of brain MR images. In: Proceedings of SPIE Medical Imaging 2000, vol. 3979, pp. 1126–1137 (2000)
304. Zhang, Y., Brady, M., Smith, S.: Segmentation of brain MR image through a hidden Markov random field model and the expectation maximization algorithm. IEEE Trans. Med. Imaging **20**(1), 45–57 (2001)
305. Zhang, Y., Wang, W., Zhang, X., Li, Y.: A cluster validity index for fuzzy clustering. Inf. Sci. **178**(4), 1205–1218 (2008)
306. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Comput. Surv. **35**(4), 399–458 (2003)
307. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Comput. **8**(2), 173–195 (2000)
308. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Tech. Rep. 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001)
309. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)

Index

A
ACDE algorithm, 167
Adaptive crossover probability, 107
Adaptive mutation probability, 108
AMOSA, 36
  Tmax, 38
  Tmin, 38
  amount of domination, 40
  archive, 37
  archive initialization, 38
  binary-coded, 48
  clustering of archive solutions, 38
  complexity analysis, 45
  current-pt, 41
  HL, 38
  main process, 41
  new-pt, 41
  real-coded, 51
  SL, 38
ANN, 104
ANOVA, 118, 145

B
Binary variable, 3

C
Cancer data, 114
Categorical variable, 3
Central moment technique, 198
Classification, 7
  supervised, *see* Supervised classification
  unsupervised, *see* Unsupervised classification, *see also* Clustering
Cluster validity indices, 11
Clustering, 8, 76
  cluster types, 10
    concentric shaped, 10
    hyperspherical shaped, 10
    linear shaped, 10
    ring shaped, 10
  cluster validity indices, 11
  density, 83
    DBSCAN, 83
    GDBSCAN, 84
    OPTICS, 84
  evolutionary approaches, 87
    for fixed number of clusters, 88
    for variable number of clusters, 89
  grid based, 85
    STING, 85
  hierarchical, 82
    agglomerative, 82
    average linkage, 83
    complete linkage, 83
    divisive, 82
    linkage, 82
  model order selection, 10
  model selection, 10
  MOO-based approaches, 90
    MOCK, 90
  partitional, 77
    K-means, 77
    K-medoid, 79
    EM, 80
    FCM, 79
  recent techniques, 85
    BIRTH, 86
    CLARA, 86
    CLARANS, 86
    FCS, 86
    PAM, 86
    SMCK-means, 86
    SOFM, 87

S. Bandyopadhyay, S. Saha, *Unsupervised Classification*,
DOI 10.1007/978-3-642-32451-2, © Springer-Verlag Berlin Heidelberg 2013

Clustering (*cont.*)
techniques
partitional, 9
single linkage, 82
Column-allowable matrix, 111
Compactness, 126
Connectivity among a set of points, 230

D
Data acquisition, 6
Decision boundary, 8
Decision function, 8
Discriminant function, *see* Decision function
Dissimilarity, 60
Distance, 9
binary variables, 61
Hamming, 63
Jaccard, 62
simple matching, 62
categorical variables, 63
normalization, 72
ordinal variables, 65
footrule distance, 67
normalized rank transformation, 66
spearman distance, 67
quantitative variables, 68
angular separation, 70
Bray-Curtis, 70
Canberra, 70
Chebyshev, 69
city block, 69
correlation coefficient, 71
Euclidean distance, 9, 68
Mahalanobis, 10, 72
Minkowski, 68

E
EAC-FCM clustering, 183
ECSAGO clustering, 167
EDR property, 102
Enumerative techniques, 18
Erechitites valerianifolia, 201

F
Face recognition, 204
human face detection, 205
Feature selection, 7
Ficus microcapa, 201
Finite Markov chain theory, 166
Fuzzy-VGAPS, 179
encoding, 179
fitness, 179

G
GALSD, 199
fitness, 199
Gap statistics, 218
GAPS, 104
chromosome, 104
complexity analysis, 109
convergence analysis, 110
fitness, 106
GenClustMOO
assignment of points, 227, 229
encoding, 228
initialization, 228
objective functions, 229
subcluster merging, 232
subclusters, 228
Genetic algorithms, 19
crossover, 22
probability of crossover, 23
elitism, 22
encoding, 21
features, 19
mutation, 23
probability of mutation, 23
objective function, 22
population, 21
selection, 22
Glass data, 234
GMFCM clustering, 183
GUCK clustering, 166

I
Iris data, 114

H
HNGA clustering, 166

K
K-means, 9
Kd-tree, 103

L
Laplacian distribution, 51, 109, 170
Learning, 4
Line symmetry-based distance, 210
LSK, 99
LungCancer data, 114

M
Machine learning, 4
Major axis, 198
Markov chain, 110
Metric, 60
Minkowski score, 114, 145
Missing value, 12

MOCK, 90, 217
Mod-SBKM, 98
MOEA, 29
  aggregating approaches, 29
    ε-constraint, 29
    goal attainment, 29
    goal programming, 29
    weighted sum, 29
  pareto-based elitist approaches, 30
    NSGA-II, 32
    PAES, 31
    PESA, 31
    PESA-II, 32
    SPEA, 30
    SPEA2, 31
  pareto-based non-elitist approaches
    MOGA, 30
    NPGA, 30
    NSGA, 30
  pareto-based nonelitist approaches, 30
  population-based non-Pareto approaches, 29
    game theory, 30
    lexicographic, 29
    MOGA, 29
    VEGA, 29
MOPS, 219
  selection of a solution from the archive, 220
  total variation, 219
MPSK, 99
MR brain image, 189
  cerebrospinal fluid, 189
  gray matter, 189
  white matter, 189
MSSL, 100
Multiobjective clustering, 217
Multiobjective optimization, 25
  dominance relation, 27
  pareto optimality, 27
  performance measures, 28, 47
    Convergence, 47
    Displacement, 47
    Minimal Spacing, 47
    Purity, 47
    Spacing, 47
  SA-based MOO
    AMOSA, *see* AMOSA

N
Numerical methods, 18

O
Object data, 6
Optimization, 17
  multiobjective, *see* Multiobjective optimization, 25
  single objective, 18
Ordinal variable, 3
Outliers, 11

P
Pattern recognition, 75
  applications, 13
  fuzzy set-theoretic, 12
  supervised, 7
  unsupervised, 7
PESA-II, 217
Positive square matrix, 111
Primitive matrix, 111
Principal component analysis, 197, 209, 210
PSFCM clustering, 183
PSKM clustering, 183

Q
Quantitative variable, 3

R
Relational data, 6
Relative neighborhood graph, 230
Remote sensing imagery, 155
  SPOT image of Kolkata, 156

S
SA based MOO
  MOSA, 35
  PCSA, 35
  PDMOSA, 35
  PSA, 34
SA-based MOO, 33
SBKM, 96
  θ, 98
Separability, 126
Similarity, 59
Simulated annealing, 23
  annealing schedule, 24, 56
Simulated circle image, 156
Sobel edge detection, 206
Sobel edge detector, 201
SPARCL clustering, 183
SPOT, 156
SSL operator, 99
Supervised classification, 7
Sym-index, 130
  mathematical justification, 134
  separation, 130
  symmetrical deviation, 130
  total compactness, 130
Symmetrical interclusters, 99, 103

Symmetrical line, 198
Symmetry, 10, 93
central symmetry, 99
circular symmetry, 94
line symmetry, 197
d_{ls}, 197
mirror symmetry, 94
point symmetry, 94, 96
d_{ps}, 100
radial symmetry, 94
rotational symmetry, 94
symmetry distance, 95
symmetry transform, 95
Systeme Probatoire d'Observation de la Terre, 156

T
Tissue classes, 189
Transition matrix, 110

U
Unsupervised classification, 8

V
Validity index, 125
external, 126
internal, 126
FSym-index, 178
Sym, *see also Sym*-index
Sym-DB, 148
Sym-Dunn, 149
Sym-FS, 151
Sym-GDunn, 149
Sym-K, 151
Sym-PS, 150
Sym-SV, 152
Sym-XB, 151
BIC, 126
CH, 127
Con, 230, 231
CS, 129
DB, 127
Dunn, 128
I, 129
PS, 129
Silhouette, 127
XB, 128, 222
VAMOSA, 221
Variable, 3
Variable string length GA, 165
VGAPS, 168
convergence, 171
encoding, 168
fitness, 168
genetic operators, 168

W
WSVF, 167