SINGLE AND MULTIOBJECTIVE APPROACHES TO CLUSTERING WITH POINT SYMMETRY

Sriparna Saha
Machine Intelligence Unit
Indian Statistical Institute
Kolkata 700108, India

A thesis submitted to the *Indian Statistical Institute*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2009

ACKNOWLEDGEMENTS

This thesis owes so much to so many people who helped me learn and who made the experience all the more enjoyable. Above all, I am grateful to Prof. (Dr.) Sanghamitra Bandyopadhyay for her precision and insightful supervision, untiring encouragement, and not least, her eye for detail. I would like to thank her for assisting me in getting started with this work and for her efforts, comments, ideas and advice that have assisted me throughout this adventurous period. Words seem insufficient to express my gratitude to her who not only supervised my thesis work, but also acted as a friend and philosopher in my time of need.

Special note of thanks to Prof. S. K. Pal, Prof. C. A. Murthy, Prof. M. K. Kundu, Prof. S. Sur-Kolay, Prof. P. Dasgupta, Dr. R. K. De, Prof. S. Mitra, Prof. A. Ghosh, Dr. D. P. Mandal, Dr. S. N. Biswas, Dr. P. Maji, Dr. B. Uma Shankar, Dr. K. Ghosh and my all research colleagues specially Anirban Mukherjee, Malay Bhattacharyya, Ramkrishna Mitra, Debarka Sengupta, Indranil Aich and Tapas Bhadra. I also acknowledge the office staff of Machine Intelligence Unit for providing a friendly environment. Heartfelt thanks are due to Prof. Ujjwal Maulik who acted as a friend, philosopher and guide in various phases of my student life.

I whole heartedly thank all my friends who were always there for me during my times of distress and were a constant source of inspiration. I am thankful to my ex-roommate Krithika who has made my hostel life cheerful. My friend and junior Soumi Sengupta deserves special mention for reinforcing my spirits at some crucial junctures.

I am grateful to all those people who contributed ideas to this thesis, directly or indirectly, to the development and culmination of this research.

I express my special thanks to my husband Dr. Asif Ekbal, who is my inspiration. His persistent dedication, love and confidence in me are highly appreciated. I am thankful to him for his perpetual understanding and tolerance during all the phases of this work.

Lastly, but most important of all, I would like to thank my family for being there. I express my sincere gratitude and appreciation to my parents and my sister for their unwavering moral support, especially in this endeavor. My parents have always given top priority to education, and raised me to set high goals for myself. Their gentle love, care and belief in me encouraged me to do my best in all matters of life and I owe a lot to them.

Finally, I express my sincere thanks to the authorities of ISI for the facilities extended to carry out my research work and for providing me fellowship to support my research.

And many others are not mentioned, but none is forgotten.

All errors are - of course - mine.

ISI, Kolkata
August, 2009 *Sriparna Saha*

CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In our every day life, we make decisions consciously or unconsciously. This decision can be very simple such as selecting the color of 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 decision is easy to take, while the latter one might take several years due to the level of complexity involved in it. 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 galore in real life. Development of optimization algorithms has therefore been of 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 (MOOP). In this regard, a multitude of metaheuristic single objective optimization techniques like genetic algorithms, simulated annealing, differential evolution, and their multiobjective versions have been developed.

Computational pattern recognition can be viewed as a two fold 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 [20]. The latter classification task can be either supervised or unsupervised depending on the availability of labelled patterns. Clustering is an important unsupervised classification technique where a number of patterns, usually vectors in a multi-dimensional 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 a variety of ways of measuring the similarity and dissimilarity concepts, which do not have any universal definition. Therefore, seeking for an appropriate cluster is experiment-oriented with the assumption that clustering algorithms capable of performing as per the demand are yet to be investigated. A good review of clustering can be found in [97].

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 basic features of shapes and objects is symmetry which is considered to be important for enhancing their recognition [10]. As symmetry is commonly found in the natural world, it may be interesting to exploit this property while clustering a data set [24][25][26][43][169][187].

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 of defining multiple objectives that are to be simultaneously optimized. This makes the process not only computationally intensive, but also leads to a possibility of losing 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.

The first part of the present thesis deals with development of a complex multiobjective optimization (MOO) algorithm based on simulated annealing (SA), a well known search and optimization technique. It is called archived multiobjective simulated annealing based technique or AMOSA. Thereafter a symmetry based similarity measurement is developed. Subsequently some single objective clustering techniques (using genetic algorithm and related methods as the underlying optimization tools) and multiobjective clustering techniques (using AMOSA and related methods as the underlying optimization tools) using the symmetry based distance are proposed. Finally, a multi-seed based clustering technique is proposed, exploiting both the multiobjective approach as well as symmetry property. Before we describe the scope of the thesis, we provide an overview of optimization techniques, both single and multiobjective, as well as of the different clustering approaches.

Section 1.2 presents a description of the basic concepts, features and techniques of both single and multiobjective optimization. A short description of the use of simulated annealing technique for solving multiobjective optimization problems is also provided in Section 1.2. In Section 1.3, we have provided a detailed description of the clustering problem along with various methods of solving it. This section also discusses about the use of symmetry as a measure of similarity. Thereafter the clustering problem is formulated as one of multiobjective optimization. Finally Section 1.4 discusses the scope of the present thesis.

1.2 Overview of Optimization Techniques

In decision science, optimization is a quite an obvious and important tool [144]. In the optimization process, development of an appropriate model, which involves identifying objectives, variables and constraints for a given problem, is the first and the most important step. If a model is too simple, it may not provide useful insights into the optimization process. On the other hand, a complex model will make the problem difficult to solve.

An optimization algorithm can be used to search for the solution after the model is fixed. In general, it is difficult to design an universally accepted optimization technique. Each of the existing algorithms is applicable to a particular type of optimization problem. The choice of the appropriate algorithm for a particular application depends upon the user. Depending on the number of objectives, the optimization technique can be single or multiobjective. Evolutionary algorithms and simulated annealing, from the family of metaheuristic search and optimization techniques, have shown promise in solving complex single as well as multiobjective optimization problems in a wide variety of domains [14][18][19][39][131][134][147].

1.2.1 Single Objective Optimization Problem

Optimization is the process of minimizing or maximizing a function subject to several constraints on its variables [144]. Generally the following notations are used:

$\overline{x}$ is the vector of *variables*, also called *unknowns* or *parameters*.

f is the *objective function*, a function of $\overline{x}$ that has to be minimized or maximized.

$\overline{c}$ is the vector of *constraints* that the *variables* must satisfy. This is a vector function of $\overline{x}$.

The single objective optimization problem can be written as

$$\min_{\overline{x}\in R^*} f(\overline{x}) \quad \text{subject to} \quad \begin{cases} c_i(\overline{x}) = 0 & \text{if} \quad i \in \xi \\ c_i(\overline{x}) \geq 0 & \text{if} \quad i \in \mathcal{I} \end{cases} \tag{1.1}$$

Here f and c_i are scalar valued functions of the variable $\overline{x}$, and ξ and $\mathcal{I}$ are sets of indices.

Some popular single objective meta heuristic optimization techniques include genetic algorithm [78], simulated annealing [108], evolutionary strategies [156] etc. Genetic algorithms (GAs) [78] [83] 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 like selection, crossover and mutation. They perform multimodal search in complex landscapes and provide near optimal solutions for objective or fitness function of an optimization problem. They are efficient, adaptive and robust search processes with a large amount of implicit parallelism [78]. Genetic algorithms have diversified applications in solving problems requiring efficient and effective search, in business, scientific and engineering circles [16][21]. Genetic algorithms find plenty of applications in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields. A detailed discussion on genetic algorithm is found in Chapter 2 of the present thesis.

Simulated annealing (SA) [108] belongs to a class of local search algorithm. It utilizes the principles of statistical mechanics, regarding the behaviour of a large number of atoms at low temperature, for finding minimal cost functions to large optimization problems by minimizing the associated energy. SA has been applied in diverse areas [18][39][134] by optimizing a single criterion. A detailed discussion on simulated annealing is found in Chapter 2 of the present thesis.

1.2.2 Multiobjective Optimization Problem

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). Some definitions and basic concepts related to MOOPs are given below.

Formal Definition of MOOP

The multiobjective optimization (MOO) can be formally stated as [48]: Find the vector $\overline{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ of decision variables which will satisfy the m inequality constraints :

$$g_i(\overline{x}) \geq 0, \quad i = 1, 2, \ldots, m, \tag{1.2}$$

the p equality constraints

$$h_i(\overline{x}) = 0, \quad i = 1, 2, \ldots, p, \tag{1.3}$$

and simultaneously optimize the M objective values

$$f_1(\overline{x}), f_2(\overline{x}), \ldots, f_M(\overline{x}). \tag{1.4}$$

The constraints given in Eqns. (1.2) and (1.3) define the feasible region $\mathcal{F}$ 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 $\mathcal{F}$. 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.

Dominance Relation and Pareto Optimality

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

A solution $\overline{x_i}$ is said to dominate $\overline{x_j}$ if [60]

$\forall k \in 1, 2, \ldots, M, f_k(\overline{x_i}) \geq f_k(\overline{x_j})$ and $\exists k \in 1, 2, \ldots, M$ such that $f_k(\overline{x_i}) > f_k(\overline{x_j})$.

This concept can be explained using a two-objective optimization problem. It has five different solutions, as shown in the figure below. 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 in Figure 1.1. Evidently solution 1 dominates solution 2 since the former is better than the latter on both the 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.

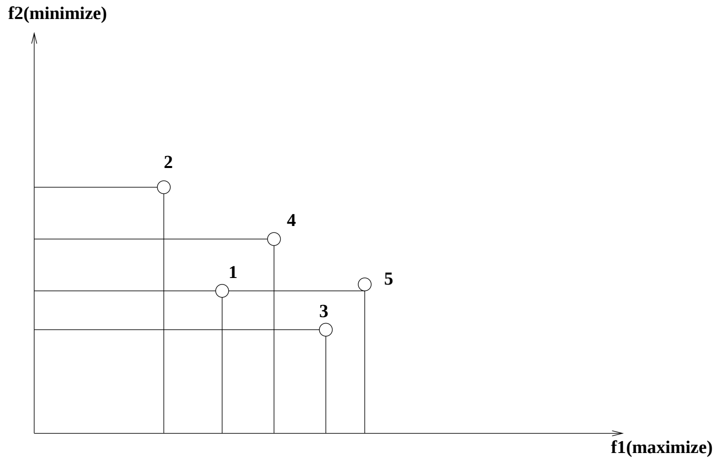


Figure 1.1: Example of dominance and Pareto optimality

Assume a set of solutions P. The solutions of P that are not dominated by any other solution in P comprise the non-dominated set [60]. The rank of a solution $\overline{x}$ in P is defined as the number of solutions in P that dominate $\overline{x}$ [60]. In Figure 1.1, solutions 3 and 5 are in the non-dominated set, and their

ranks are 0.
The non-dominated set of the entire search space S is the globally Pareto optimal set [60].

Performance Measures

In order to evaluate the performance of different MOO algorithms, several measures have been proposed in the literature. Some such measures are convergence measure γ [61], *Purity* [22], *Spacing* [176] and *Minimal Spacing* [22]. The convergence measure γ indicates how close an obtained non-dominated front is from the true Pareto optimal front. *Purity* of an MOO algorithm measures the fraction of the non-dominated solutions that remain non-dominated with respect to the solutions obtained using several other MOO techniques. *Spacing* and *Minimal Spacing* measure the diversity of the solutions in the final non-dominated set. These measures have been used in this thesis for comparing the performance of different MOO algorithms. There are many other measures available in the literature. Details can be found in [47][60].

1.2.3 Various Methods to Solve MOOP

A large number of approaches exist in the literature to solve multiobjective optimization problems [48][60]. These are aggregating, population based non-Pareto and Pareto based techniques. In case of aggregating techniques, the different objectives are generally combined into one using weighting or goal based method. One of the techniques in the population based non-Pareto approach is 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), niched Pareto GA. Note that all these techniques were essentially non-elitist in nature. Some recent elitist techniques are NSGA-II [60], SPEA [212] and SPEA2 [211].

Simulated annealing (SA) performs reasonably well in solving single-objective optimization problems. But 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 MOOP. 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 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. [92][195][208], that take care of this crucial issue very effectively [142]. In this thesis a new multiobjective version of the standard simulated annealing algorithm is proposed. In this context the next section reviews some existing multiobjective simulated annealing based techniques in detail.

1.2.4 MOOP and SA

As already mentioned, one of the major obstacles of using SA for MOOPs is that it produces only a single solution at a time. Since solving 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 Chapter 9 of [47] and in [192]. A part of the following discussion is based on [47] and [192].

In most of the earlier attempts, a single objective function is constructed by combining the different objectives into one [52][65][82][143][177][194][200][201]. 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 w_i's are the corresponding weights. This composite objective is then used as the energy to be minimized in a scalar SA optimization method. The problem here is how to choose the weights in advance. Some alternative approaches have also been used in this regard. For example, in [65], sum of $log f_i(\overline{x})$ has been taken.

Multiobjective simulated annealing with a composite energy clearly converges to the true Pareto front if the objectives have ratios given by w_i^{-1}, if such points, in general, exist. In [53], it has been proved that part of the front will be inaccessible with fixed weights. In [99] several different schemes were explored for adapting the w_is during the annealing process to encourage exploration along the front. However, a proper choice of the w_is remains a challenging task.

An integration of the weighted sum approach and the concept of Pareto optimality was introduced in [52]. 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 quad-trees are used to store and retrieve them efficiently. Another new approach taken into consideration in PSA is that when a new solution $f(\overline{x'})$ is generated in the neighborhood of $f(\overline{x})$ ($f(\overline{x'})$ denotes the closest neighborhood solution of $f(\overline{x})$), 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 the fact 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 [41]. Here the problem is to minimize the maximum deviations of each objective with respect to a set of user defined goals.

Suppapitnarm et al. [193] 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 add in the external file, a new solution has to be non-dominated with respect to all the solutions of the external file. This external population takes 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 $p = \prod_{i=1}^{k} \exp\left\{-\frac{f_i(\overline{x})-f_i(\overline{x'})}{T_i}\right\}$ 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" strategy 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 to MOGA but its only advantage is its simplicity.

In [143] and [201] different non-linear and stochastic composite energy functions have been investigated. In [143] six different criteria for energy difference calculation for MOSA are suggested and evaluated. These are i) minimum cost criterion, ii) maximum cost criteria, iii) random cost criteria, iv) self cost criteria, v) average cost criteria, and vi) fixed cost criteria. Since each run of the SA provides just a single solution, the algorithm attempts to evolve the set of PO solution by using multiple SA runs. As a result of the independent runs, the diversity of the set of solutions suffered. After performing some comparative study, the authors conclude that the criteria that work best are the random, average and fixed criterion [47]. For comparison with respect to some existing multiobjective evolutionary algorithms (MOEAs), the average criterion is taken into consideration. Authors have compared their approach with respect to NPGA [85]. The proposed approach presents a competitive performance but has some diversity problems. The use of niches is suggested to deal with this problem.

A modified multiobjective simulated annealing named Pareto Cost Simulated Annealing (PCSA) is proposed in [142]. 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 is compared with the existing MOEA techniques, MOGA [72], NPGA [85] and NSGA [185] for 18 test problems. These problems are basically two-objective problems, having 2-4 number of variables. Authors have shown that in 67% of the cases PCSA performs better than MOEAs.

A multiobjective version of simulated annealing based on Pareto dominance is proposed by Suman [188][189][190][191]. An external archive and a scheme to handle constraints within the expression used to determine the probability of moving to a different state are also proposed in [188][189][190][191]. 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, SMOSA [189], UMOSA [190], PSA [191], WMOSA [188] and PDMOSA are compared for several constrained multiobjective optimization problems. The selection criterion adopted by SPEA [212] 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 [191] provides the best qualitative results where as PDMOSA provides best results with respect to diversity. Some more recent Pareto dominance based multiobjective SA methods are proposed in [181][182][188]. Since the technique in [181][182] has been used in this thesis for the purpose of comparison, it is described in detail in Chapter 2.

In the Pareto domination based multiobjective SAs developed relatively recently [181][182][188], 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. In this thesis, a new multiobjective SA is proposed, hereafter referred to as AMOSA (Archived Multiobjective Simulated Annealing), 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. It is described in detail in Section 2.4 of Chapter 2.

1.3 Clustering

1.3.1 Definition of Clustering

Clustering [96][197], 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 multi-dimensional 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

$$\begin{array}{ll} \sum_{j=1}^{n} u_{kj} \geq 1 & for \quad k = 1, \ldots, K, \\ \sum_{k=1}^{K} u_{kj} = 1 & for \quad j = 1, \ldots, n \text{ and }, \\ \sum_{k=1}^{K} \sum_{j=1}^{n} u_{kj} = n. & \end{array}$$

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 $u_{kj} = 0$.

1.3.2 Some Clustering Techniques

There exists a large number of clustering techniques in the literature [97]. Traditional clustering algorithms are classified into three classes [96]: *hierarchical*, *partitional* and *density-based*. Some examples of *hierarchical* clustering methods are *Single Linkage Clustering Algorithm*, *Average Linkage Clustering Algorithm* and *Complete Linkage Clustering Algorithm*. K-means clustering algorithm is an example of the *partitional* method [197]. *DB scan* clustering algorithm is an example of the density-based clustering technique. In this section two well-known clustering techniques, K-means and single linkage clustering technique are described in detail since these two have been used in the present thesis.

K-means Clustering Technique

The K-means algorithm [68][96] is an iterative clustering technique that evolves K crisp, compact and hyperspherical clusters in a data set such that the measure:

$$J = \sum_{j=1}^{n} \sum_{k=1}^{K} u_{kj} \times \|\overline{x}_j - \overline{z}_k\|^2 \tag{1.5}$$

is minimized. Here u_{kj} is equal to 1 if the jth point belongs to cluster k, and 0 otherwise; $\overline{z}_k$ denotes the center of the cluster k and $\overline{x}_j$ denotes the jth point of the data. In K-means, K 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 with the means of the respective clusters. The process of partitioning followed by updating centers are 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 (c) maximum number of iterations have been exhausted. The different steps of K-means algorithm are enumerated in Figure 1.2. In general, if the process does not terminate in step 4 normally, then it is executed for a maximum fixed number of iterations.

```
Step1: Choose $K$ cluster centers $\overline{z}_1, \overline{z}_2, \ldots \overline{z}_K$ randomly from the
$n$ points $\overline{x}_1, \overline{x}_2, \ldots. \overline{x}_n$.
Step2: Assign point $\overline{x}_i$, $i = 1, 2, \ldots. n$ to cluster $C_j$, $j \in 1, 2, \ldots. K$ iff
    $\|\overline{x}_i - \overline{z}_j\| < \|\overline{x}_i - \overline{z}_p\|, \quad p = 1, 2, \ldots K, \quad$ and $\quad j \neq p$
  Ties are resolved arbitrarily.
Step3: Compute new cluster centers $\overline{z}_1^*, \overline{z}_2^*, \ldots.., \overline{z}_K^*$ as follows:
    $\overline{z}_i^* = \frac{\sum_{\overline{x}_j \in C_i} \overline{x}_j}{n_i}, \quad i = 1, 2, \ldots. K.$
    where $n_i$ is the number of elements belonging to cluster $C_i$.
Step4: If $\overline{z}_i^* = \overline{z}_i, i = 1, 2, \ldots. K$ than terminate.
Otherwise continue from step2.
```

Figure 1.2: The K-means algorithm

In order to improve the performance of the K-means algorithm, several improved versions have been developed in the past several years [40][101][113].

Single-linkage Clustering Algorithms

The single-linkage clustering technique is a non-iterative method based on a local connectivity criterion [97]. Instead of using one single data point $\overline{x}$ in a data set, single linkage 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

$$\delta_{SL}(C_i, C_j) = \min_{\overline{x} \in C_i, \overline{y} \in C_j} \{d(\overline{x}, \overline{y})\}, \tag{1.6}$$

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 distance 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 2) it works with both numeric or categorical attributes. The disadvantages of this approach are its computational complexity and inability to handle overlapping clusters. ♣

Several clustering algorithms with different distance measures have been developed for clustering data sets with different geometric shapes [54][55][76][80][84][128]. These algorithms were used to detect compact clusters [76], straight lines [54][76], ring shaped clusters [128], or contours with polygonal boundaries [55][84]. However the performance of these algorithms were poor when the data had clusters of other shapes. In [12] 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 [28][35][36][42][123] [150][151][152][154][155][186]. Some other recent clustering techniques can be found in [6][7][8][9] [100][149] [153][202][203][204].

1.3.3 Distance Measures in Clustering

The main goal of clustering is to maximize both the homogeneity within each cluster and the heterogeneity among different clusters [67][89] irrespective of the type of clustering algorithm (partitional, hierarchical or overlapping). Alternatively, the 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 amount of similarity between two objects. Several measures have been employed in the literature for clustering [96][207]. 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 [96] 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 [129]. Here $\overline{x}$ represents an input pattern, the matrix $\sum$ is the covariance matrix of a pattern population constituting a particular cluster, $\overline{m}$ is the mean vector of the vectors which are in the same cluster. A disadvantage of 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 [187]. But 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 areas such as bioinformatics, text clustering or document categorization.

Symmetry is considered as an important feature in recognition and reconstruction of shapes and objects [10][187]. Almost every interesting area around us consists of 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 in [43][45][46][122][187]. It is also one of the main focusses of this thesis. A detailed discussion is provided in Chapter 3.

1.3.4 Some Evolutionary Approaches to Clustering

Clustering can be treated as a particular kind of NP-hard grouping problem [69] from an optimization perspective. Evolutionary algorithms are meta-heuristics that are generally used for solving NP-hard problems. The ability of these algorithms to provide near-optimal solutions in a reasonable time stimulates their use in solving clustering problems [89].

Algorithms for a Fixed Value of the Number of Clusters

Several papers use evolutionary algorithms to solve clustering problems with fixed number of clusters (K), such as Bandyopadhyay and Maulik [15]; Castro and Murray [66]; Fränti et al. [73], Krishna and Murty [114]; Krovi [115]; Kuncheva and Bezdek [116]; Lu et al. [124][125], Lucasius et al. [126]; Maulik and Bandyopadhyay [131]; Merz and Zell [136]; Murthy and Chowdhury [141], Sheng and Liu [178]. A good review of these clustering techniques is available in [89].

Genetic algorithms have been used for finding globally optimal solutions to clustering problems [34]. But it has been pointed out in [114] that these GA-based methods are computationally inefficient as they either use complex crossover operators or computationally hard fitness functions. To avoid these problems, Krishna and Murty [114] have proposed a new GA based algorithm (GKA). This algorithm uses 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 $n = 50$, 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 combinatorially and the problem of finding the global solution becomes very difficult. 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 [11]. 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 concatenation 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 [131] have used center based encoding in chromosome while performing clustering using GA. GA is used to evolve the appropriate set of cluster centers. The crossover operators [131] exchange randomly selected sub strings. 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 [16].

Laszlo and Mukherjee [120] have proposed a new genetic algorithm based K-means clustering technique. Here GA is used to evolve centers in the K-means algorithm that simultaneously identifies good partitions for a range of values around a specified K. The set of centers is represented using a hyper-quadtree constructed on the data. This representation is used in the proposed genetic clustering technique to generate an initial population of good centers and to support a novel crossover operation that selectively passes good subsets of neighboring centers from parents to offspring by swapping subtrees. Experimental results show that the proposed technique is well-suited for large data sets as well as small ones.

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 [49], Cowgill et al. [51], Bandyopadhyay and Maulik [14][16], Hruschka and Ebecken [90], Hruschka et al. [86][87][88], Ma et al. [127], Alves et al. [180]. 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 GCUK-clustering [16], variable string length Genetic Algorithm (VGA) [83] 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. Davies-Bouldin cluster validity index [56] is utilized for computing the fitness of the chromosomes.

In Hybrid Niching Genetic Algorithm (HNGA) [179], 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, specifically $m = 6$ is used here. w_is are the 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 [179]) corresponding to 1/(DB-index [56]), SIL-index [102], Dunn-index [64], Generalized Dunn-index [32], CH-index [38] and I-index [132], respectively. Here weighting coefficients are chosen as $w_1 = w_2 = \ldots = w_m = 1/m$.

In [121] 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 (HNEA) 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. HNEA 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). In ECSAGO, real-encoding and real genetic operators are used.

1.3.5 Some Cluster Validity Indices

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 may be the clustering technique, one has to determine the number of clusters and also the validity of the clusters formed [63]. The measure of validity of clusters should be such that it will be able to impose an ordering of the clusters in terms of its goodness. In other words, if $U_1, U_2, \ldots, U_m$ be the m partitions of X, and the corresponding values of a validity measure be $V_1, V_2, \ldots V_m$, then $V_{k1} \geq V_{k2} \geq \ldots V_{km}, \forall ki \in 1, 2, \ldots, m,\ i = 1, 2, \ldots, m$ will indicate that $U_{k1} \uparrow \ldots \uparrow U_{km}$. 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., Davies-Bouldin (DB) index [56], Dunn's index [64], Xie-Beni (XB) index [206], *I*-index [132], CS-index [44], XB* index [105], index proposed in [104][106], fuzzy cluster validity indices proposed in [205][209] etc., to name just a few. A good review of the cluster validity indices and their categorization can be found in [105]. 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 [139] 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 [132] evaluated the performance of four validity indices, namely, the Davies-Bouldin index [56], Dunn's index [64], Calinski-Harabasz index [132], and an index I, in conjunction with three different algorithms viz., the well-known K-means [68], single-linkage algorithm [68] and an SA-based clustering method [132]. Several researchers have used a cluster validity index as the optimizing criterion in evolutionary approaches to clustering [14][16][179]. However, since a single validity index is seldom able to capture different characteristics of the partitioning, using a set of indices along with MOO is attracting the attention of researchers in recent times.

1.3.6 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 the 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 Ref. [81], a multiobjective clustering technique called 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 other measuring the total "connectedness" of the obtained partitioning are optimized simultaneously. Although the objectives of [81] are very useful, it can only handle clusters either having hyperspherical shape or "connected" but well-separated structures. It fails for datasets having overlapping clusters which do not contain any hyperspherical shape, e.g., the data sets in Figures 3.6(a) and 3.6(b) of Chapter 3. Moreover MOCK uses locus-based adjacency representation as proposed in Ref. [148]. As a result when the number of data points is too large the string length becomes high too and convergence becomes slow. Here "Gap-statistics" [119] is used to select a single solution from the set of final Pareto optimal solutions obtained by MOCK. In [130], 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 scheme reduces the Pareto-size and the appropriate number of clusters can usually be determined.

An EA for MOO clustering is proposed in [112]. Here two objectives are minimized simultaneously. These are total intra cluster 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 [17]. Here again, two objectives are simultaneously optimized. The first one is the objective function optimized in Fuzzy C-means algorithm [31] and the other is the well-known Xie-Beni index [206]. 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.

Another multiobjective evolutionary clustering algorithm with two objectives is proposed in [158]. 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, that 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.

1.4 Scope of the Thesis

The present thesis deals with the development of some single and multiobjective clustering techniques based on a newly proposed definition of point symmetry. A new MOO algorithm based on simulated annealing called AMOSA is also proposed. In order to solve the single objective clustering problem genetic algorithm [78] is used, while AMOSA is used for the multiobjective version. A symmetry based cluster validity index is also developed. The proposed symmetry distance is incorporated in some existing cluster validity indices to develop the symmetry versions of these indices. In order to automatically determine the appropriate number of clusters from the data sets, the concept of variable length representations is incorporated. Finally a multicenter based multiobjective clustering technique is developed which can detect any type of clusters from data sets having either symmetrical shapes (convex or non-convex, overlapping or non-overlapping) or asymmetrical but well-separated structures. The results of investigation are summarized below on the basis of chapter headings.

1.4.1 Some Single and Multi Objective Optimization Techniques

In Chapter 2, some existing single and multiobjective optimization techniques are first discussed in detail. Then this chapter introduces a new simulated annealing based algorithm that solves the MOOP. As already mentioned, although SA has a strong theoretical background, it has seldom been used in MOOPs primarily because of its search from a point nature.

The newly proposed algorithm incorporates the concept of amount of domination, and obtains the Pareto optimal solutions in an archive. This algorithm is hereafter referred to as AMOSA (Archived Multiobjective Simulated Annealing). A new concept of the amount of dominance is introduced that is utilized in AMOSA in order to determine the acceptance of a new solution. The results of binary-coded AMOSA are compared with those of two existing well-known multiobjective optimization algorithms - NSGA-II (binary-coded) [61] and PAES [110] for a suite of seven 2-objective test problems having different complexity levels. In a part of the investigation, comparison of the real-coded version of the proposed algorithm is conducted with a very recent multiobjective simulated annealing algorithm MOSA [181] and real-coded NSGA-II for six 3-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 like *Convergence*, *Purity*, *MinimalSpacing*, and *Spacing*, and the time taken are used. In this regard, a measure called *displacement* has also been used that 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 has also been performed.

It has been observed from the given results that the performance of the proposed 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 like 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 [61] do not work well on many-objective optimization problems as pointed out in some recent studies [91][93]. An interesting feature of AMOSA, as in other versions of multiobjective SA algorithms, is that it has a non-zero probability of allowing a dominated solution to be chosen as the current solution in favour 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 in other MOO algorithms in order to improve their performance.

1.4.2 A Point Symmetry Based Distance Measure and its Application to Single and Multi Objective Clustering

In Chapter 3 a symmetry based similarity measurement [24] has been defined. Kd-tree [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 GAPS (genetic algorithm with point symmetry based clustering technique) [24] which is able to detect any type of cluster possessing the property of point symmetry. The global convergence property of the proposed GAPS clustering is established.

Experimental results of GAPS are demonstrated for four artificial data sets and four real-life data sets. Results demonstrate the superiority of GAPS as compared to SBKM [187], Mod-SBKM [43], K-means algorithm, genetic algorithm based K-means clustering technique (GAK-means), average linkage clustering technique (AL) and Expectation Maximization clustering technique (EM).

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 Chapter 3 the problem of clustering a data set into a fixed number of clusters is posed as one of the multiobjective optimization (MOO), where search is performed over a number of objective functions [160]. The newly proposed simulated annealing based multiobjective optimization technique, AMOSA, has been used as the underlying optimization technique. Two cluster quality measures, the total Euclidean compactness and the total symmetrical compactness are optimized simultaneously exploiting the search capability of AMOSA. This enables the algorithm to detect clusters that are well characterized by Euclidean compactness as also those which are not compact in the conventional sense, but are symmetric about a point. In the proposed multiobjective clustering technique with point symmetry based distance, MOPS, assignment of points to different clusters is done based on the point symmetry based distance rather than the Euclidean distance. The performance of MOPS is compared with GAPS in order to establish its effectiveness.

1.4.3 Validity Index Based on Symmetry

In Chapter 4 the newly proposed symmetry based distance is first used to develop a cluster validity index called *Sym*-index [162][168]. It is shown experimentally that the *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. An elaborate description of the different components of *Sym*-index along with an intuitive explanation of how they compete with each other to identify a proper clustering are provided. A mathematical justification of the newly proposed *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 *Sym*-index is demonstrated for four artificially generated and three real life data sets. 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 newly proposed *Sym*-index as compared to four existing validity indices, namely, PS index, *I*-index, CS-index and XB-index as long as the clusters present in it have point-based symmetrical structures irrespective of their geometrical shapes and convexities.

The concept of point symmetry is thereafter incorporated into several well-known cluster validity indices [172]. Point symmetry versions of eight cluster validity indices are developed which 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 make 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 [169]. Its effectiveness, vis-a-vis, other well-known validity indices is first established for segmenting one artificially generated image. Thereafter, it is used for classifying the different land covers in two multispectral satellite images.

1.4.4 Symmetry Based Automatic Clustering

In Chapter 5, a variable string length genetic algorithm based clustering technique (VGAPS clustering) is proposed that can automatically determine the appropriate number of clusters and the appropriate partitioning from a data set having symmetrical shaped clusters [26]. The newly proposed 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. The global convergence property of the proposed VGAPS-clustering is also established. The effectiveness of the VGAPS-clustering, as compared to two recently proposed automatic clustering techniques, namely, GCUK-clustering and HNGA-clustering, is demonstrated on five artificially generated and three real-life data sets of different characteristics. Results on the eight data sets establish the fact that VGAPS-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, or overlap or size, as long as they possess the property of symmetry.

The corresponding MO version of VGAPS clustering technique utilizing the

AMOSA algorithm is also developed [171] which simultaneously optimizes *Sym*-index and an Euclidean distance based cluster validity index, XB-index [206]. Thus it can automatically detect the proper partitioning and the proper number of partitions of a data set having clusters of either symmetrical or hyperspherical shape. The effectiveness of the proposed clustering technique (VAMOSA) in detecting the proper number of partitions and the proper partitioning is shown for four artificial and four real-life data sets and the results are compared with those obtained by another MO clustering technique, MOCK [81], two single objective automatic genetic clustering techniques, GCUK clustering optimizing XB-index [16] and VGAPS clustering [26] optimizing *Sym*-index [169].

1.4.5 A Generalized Automatic Clustering Algorithm in a Multiobjective Framework

Chapter 6 deals with the development of a MO clustering technique [25] using AMOSA which can automatically determine any type of clusters having either symmetrical (may be convex or non-convex, overlapping or non-overlapping) or asymmetric but well-separated structures from a data set. Here each cluster is divided into several nonoverlapping small hyperspherical sub-clusters and the centers of these sub-clusters are encoded in the states of AMOSA. For the assignment of points, all these sub-clusters are considered individually. However, for objective function calculation, the sub-clusters corresponding to each cluster are identified based on a proximity measure and these are merged together. Three cluster validity indices, an Euclidean distance based cluster validity index, a point symmetry distance based cluster validity index, and a connectivity based cluster validity index are optimized simultaneously. Relative neighborhood graph [198] is utilized to compute the connectivity index. The performance of the proposed algorithm called *GenClustMOO* is compared with the existing multiobjective clustering technique, MOCK, another multiobjective clustering technique developed in this thesis, VAMOSA and a single objective clustering technique, VGAPS, for several data sets having different characteristics. Results show that the proposed technique is well-suited to detect the appropriate partitioning from data sets having either the point symmetric clusters or well-separated clusters.

1.4.6 Conclusions and Scope of Future Research

The conclusions and possible directions of future research in this field have been explored in Chapter 7.

1.5 Contributions of the Thesis

This thesis contributes to the current understanding of the benefits of single and multiobjective optimizations for unsupervised classification while exploiting the property of symmetry within the clusters. The contributions are enlisted below

1. The development of a simulated annealing based multiobjective optimization technique (AMOSA) (in Chapter 2) which is used later to develop some multiobjective clustering techniques (Chapters 3, 5, 6).

2. Development of a symmetry based distance measure. This new distance is then used to develop some single and multiobjective clustering techniques based on symmetry where the number of clusters is assumed to be known *apriori* (Chapter 3).

3. Development of a symmetry based cluster validity index. Application of this index in automatic image segmentation. Thereafter the new point symmetry based distance is incorporated in several existing cluster validity indices to develop the symmetry versions of these indices (Chapter 4).

4. Development of some point symmetry based automatic clustering techniques using both single and multiobjective optimization methods. These can automatically detect the appropriate number of clusters and the appropriate partitioning from the data sets (Chapter 5).

5. Development of a generalized multiobjective multicenter based clustering technique which can determine any type of clusters from data sets having either symmetrical shapes (convex or non-convex, overlapping or non-overlapping) or asymmetrical but well-separated structures (Chapter 6).

Chapter 2

Some Single and Multi Objective Optimization Techniques

2.1 Introduction

Optimization deals with the study of those kinds of problems in which one has to minimize or maximize a real function. This is executed in a systematic way by choosing the proper values of real or integer variables within an allowed set. It is represented by a scalar real valued objective function. Given a defined domain, the main goal of optimization is to study the means of obtaining the best values of some objective function. Here, the elaboration depends on the types of functions, conditions and nature of the objects present in the problem domain.

An optimization problem [144] can be represented in the following way:
Given: a function $f : A \rightarrow R$ from some set A to the real numbers.
Sought: an element $\overline{x_0}$ in A such that $f(\overline{x_0}) \leq f(\overline{x})$ $\forall \overline{x} \in A$ ("minimization") or such that $f(\overline{x_0}) \geq f(\overline{x})$ $\forall \overline{x} \in A$ ("maximization").

Here, A denotes a subset of the Euclidean space R^n that is denoted in terms of a set of entities like constraints, equalities or inequalities. The members of A should satisfy these entities. A, the domain of f is called the search space and the elements of A 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 [60] (multi-criteria or multi-attribute optimization) can be regarded as the process of simultaneously optimizing two or more conflicting objectives with respect to a set of certain constraints. Often we come across these types of problems in several fields including product and process design, finance, aircraft design, oil and gas industry, automobile design etc.

In this chapter at first we discuss some existing single and multiobjective optimization techniques. Thereafter a new simulated annealing based multiobjective optimization technique named AMOSA (archived multiobjective simulated annealing based optimization technique) is proposed in the present paper.

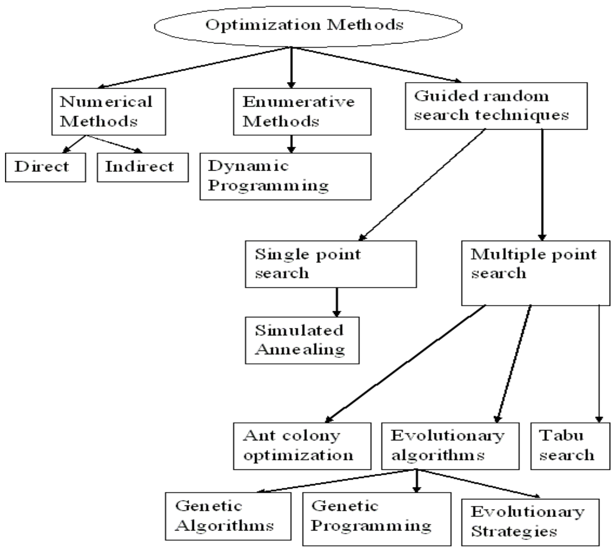


Figure 2.1: The different search and optimization techniques

2.2 Single Objective Optimization Techniques

Different optimization techniques that are found in the literature can be broadly classified into three categories (Figure 2.1) [78]:

- 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. They can be further subdivided into two categories, viz., direct and indirect. Direct search methods perform a hill climbing on 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 in 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. Dynamic programming is a well-known example of enumerative search. It is obvious that enumerative techniques will break down even on problems of 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. 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 single-point search technique that uses thermodynamic evolution to search for the minimum energy states. Evolutionary algorithms like genetic algorithms are the 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, multi modal 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 thesis, 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 for directing the search in 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 like *selection, crossover* and *mutation* are applied on the chromosomes to yield potentially better solutions.

Note that the classical gradient search techniques perform efficiently when the problems under consideration satisfy tight constraints. But when the search space is discontinuous, noisy, high dimensional and multimodal, then GAs have been found to consistently outperform both the gradient descent method and various forms of random search [79].

Genetic Algorithms: Basic Principles and Features

Genetic algorithms (GAs) [58][78][138] are adaptive computational procedures modeled on the mechanics of natural genetic systems. They express their ability by efficiently exploiting the historical information to speculate on new offspring with expected improved performance [78].

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

- GAs work with the coding of the parameter set, not with the parameter themselves.
- GAs work simultaneously with multiple points, and not a single point.
- GAs search via sampling (a blind search) using only the payoff information.

- GAs search using stochastic operators, not deterministic rules.

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, like derivative 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 generation (or iteration) when *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 reproduction* may be used, where one or more individuals are replaced at a time, instead of the whole population [58]. 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).
- Objective function and associated fitness evaluation techniques.
- Selection/reproduction procedure.
- Genetic operators (*crossover* and *mutation*).
- Probabilities to perform genetic operations.

These components are now briefly described.

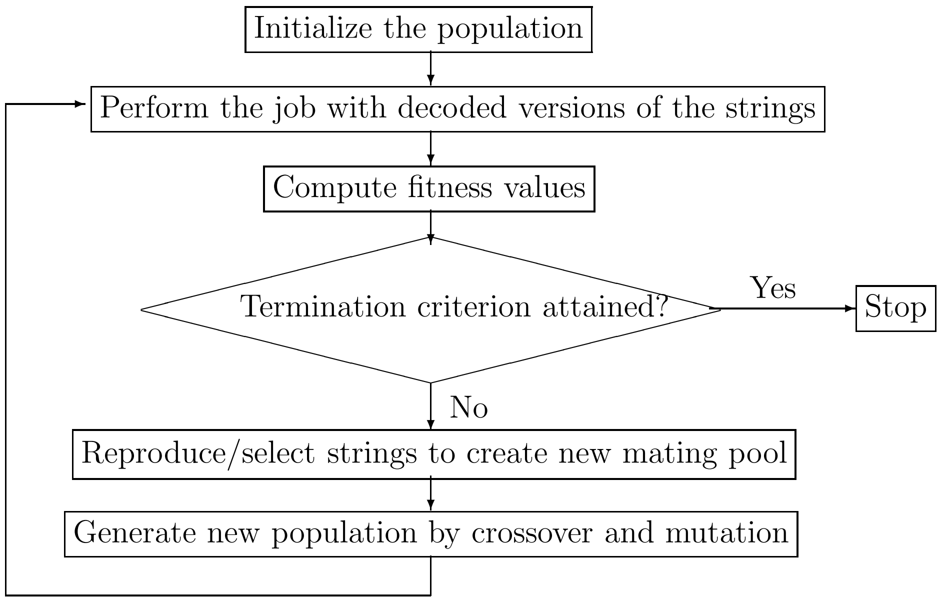

Initialize the population
Perform the job with decoded versions of the strings
Compute fitness values
Termination criterion attained?
Yes
Stop
No
Reproduce/select strings to create new mating pool
Generate new population by crossover and mutation

Figure 2.2: Basic steps of a genetic algorithm

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 0's and 1's. 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, …, a_p be 10110, 00100, …, 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* [78]. 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 [78]. Hence, binary encoding is one of the commonly used strategies, although other techniques like floating point coding [62][138] 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 mating pool) for genetic operations. Number of copies that an individual receives for 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* [78] and *linear selection* [58] are two of the most frequently used selection procedures.

As proved in [159], 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 upto a given generation will be retained in future generations. To overcome this, a commonly used strategy known as the *elitist selection* [79] is adopted, thereby providing an *elitist GA* (EGA), where the best chromosome of the current generation in retained in the next generation.

Genetic operators are applied on 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 the multiple point crossover, shuffle-exchange crossover, uniform crossover [58].

Mutation: The main aim of mutation is to introduce genetic diversity into the population. Sometimes, it helps to regain the 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 in between 0.6 to 0.9 [58][78].

Since mutation occurs occasionally, it is clear that the probability of performing mutation operation will be very low. Typically the value lies between $1/l$ and 0.1 [58][78].

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: Basic Principles

Introduction

Simulated Annealing (SA) [108] 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 [18][39][134] by optimizing a single criterion.

Basic Principles of Simulated Annealing

Application of techniques having physical or natural correspondence for solving difficult optimization problems has been receiving widespread attention for the last two decades. It has been found that these techniques consistently outperform classical methods like gradient descent search when the search space is large, complex and multimodal. Simulated annealing (SA) is one such paradigm having its foundation in statistical mechanics, which studies the behavior of a very large system of interacting components in thermal equilibrium.

In statistical mechanics, if the 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 the Boltzmann's constant and $E(s)$ is the energy of the system in state s.

Metropolis [137] developed a technique to simulate the behaviour of the 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 $T \rightarrow \infty$, the probability that the system is in state s is given by $\pi_T(s)$ irrespective of the starting configuration [77].

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.

In this thesis the annealing process of the Boltzmann machine is used, which is a variant of the Metropolis algorithm. Here, at a given temperature T, the new state is chosen with a probability

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

The parameters of the search space are usually encoded in the form of strings of fixed length. The objective value associated with the string is computed and mapped to its energy. The string with the minimum energy value provides the solution to the problem. The initial string (say q) of 0s and 1s is generated randomly and its energy value is computed. Keeping the initial temperature high (say $T = T_{max}$), a neighbor of the string (say s) is generated by randomly flipping one bit. The energy of the new string is computed and it is accepted in favour of q with a 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 Figure 2.3.

```
Begin
  generate the initial state $q$
  $T = T_{max}$
  Let $E(q,T)$ be the associated energy
  while ($T \geq T_{min}$)
    for $i = 1$ to $k$
      Perturb $q$ to yield $s$
      Let $E(s,T)$ be the associated energy
      Set $q \leftarrow s$ with probability $\frac{1}{1+e^{-(E(q,T)-E(s,T))/T}}$
    end for
    $T = rT$
  end while
Decode $q$ to provide the solution of the problem.
End
```

Figure 2.3: Steps of Simulated Annealing

Simulated Annealing has been successfully applied in various domains [57]. The domains include computer design [108][109], image restoration and segmentation [183], contour detection [33][39][134], edge detection [117], combinatorial problems such as traveling salesman problem [107] and artificial intelligence [18][23]. 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 Some Multiobjective Optimization Techniques

The multiobjective optimization (MOO) problem has a rather different perspective compared to one having a single objective. In single-objective optimization there is only one global optimum, but in multiobjective optimization there is a set of solutions, called the Pareto optimal (PO) set, which are considered to be equally important; all of them constitute global optimum solutions. Over the decade, a number of multiobjective Evolutionary Algorithms (MOEAs) have been suggested (see, [47][60] for some reviews). The main reason for the popularity of Evolutionary algorithms (EAs) for solving multiobjective optimization is their population based nature and ability of finding multiple optima simultaneously.

Simulated Annealing (SA) [108] is another popular search algorithm. Though simulated annealing has been in the literature for a longer time than evolutionary algorithms, it has been rarely used in the study of MOOP. The performance of the earlier SA based MOOP solving algorithms are not comparable with those of the existing multiobjective evolutionary algorithms [61][110]. In this thesis an attempt in this direction has been made by proposing an algorithm using simulated annealing as its underlying tool.

2.3.1 Recent MOEA Algorithms

During 1993-2003, a number of different evolutionary algorithms were suggested to solve multiobjective optimization problems. Among these, two well-known ones namely, PAES [110] and NSGA-II [61], are described first since these are used in this chapter for the purpose of comparison.

Knowles and Corne [110] suggested a simple MOEA using a single parent, single child evolutionary algorithm which is similar to (1+1) evolutionary strategy. Here binary representation and bit-wise mutation are used while creating offspring. At 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 yes, 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 with 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 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.

NSGA-II proposed by Deb *et al.* [61] is the other popular algorithm for multi-objective optimization. Here, initially a random parent population P_0 of size N is created. Then the population is sorted based on the non-domination relation. Each solution of the population is assigned a fitness which is equal to its non-domination level. A child population Q_0 is created from the parent population P_0 by using binary tournament selection, recombination, and mutation operators. In general, initially a combined population $R_t = P_t + Q_t$ is formed of size R_t, which is $2N$. Now all the solutions of R_t are sorted based on their non-domination status. If the total number of solutions belonging to the best non-dominated set F_1 is smaller than N, F_1 is completely included into $P_{(t+1)}$. The remaining members of the population $P_{(t+1)}$ are chosen from subsequent non-dominated fronts in the order of their ranking. To choose exactly N solutions, the solutions of the last included front are sorted using the crowded comparison operator and the best among them (i.e., those with larger values of the crowding distance) are selected to fill in the available slots in $P_{(t+1)}$. The new population $P_{(t+1)}$ is now used for selection, crossover and mutation to create a new population $Q_{(t+1)}$ of size N, and the process continues. The crowding distance operator is also used in the parent selection phase in order to break a tie in the binary tournament selection. This operator is basically responsible for maintaining diversity in the Pareto front.

2.3.2 Recent MOSA Algorithm

One of the recently developed MOSA algorithm is by Smith et al. [181][182]. 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 dominates $\overline{x}$. 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 non-dominating 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 dominates $\overline{x}$. 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 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 attainment surface sampling method is adopted to increase the number of solutions in the final Pareto front. Authors have perturbed a decision variable with a random number generated from the Laplacian distribution. Two different sets of scaling factors, traversal scaling which generates moves to a non-dominated 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.

2.4 A New Multiobjective Simulated Annealing Based Technique: AMOSA

2.4.1 Introduction

Simulated Annealing (SA) [108] is a popular search algorithm. However there have been only a few attempts in extending SA to multiobjective optimization, primarily because of its search-from-a-point nature. In most of the earlier attempts, a single objective function is constructed by combining the different objectives into one using a weighted sum approach [52][82][143][194][200][201]. In addition to the earlier aggregating approaches of multiobjective SA, there have been a few techniques that incorporate the concept of Pareto dominance. Some such methods are proposed in [181][188], which use Pareto domination based acceptance criterion in multiobjective SA. A good review of several multiobjective simulated annealing algorithms and their comparative performance analysis can be found in [192].

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 [181] [188]. In this chapter, a new multiobjective SA is proposed, hereafter referred to as AMOSA (Archived Multiobjective Simulated Annealing), which incorporates a concept of amount of dominance in order to determine the acceptance of a new solution [27]. The Pareto optimal (PO) solutions are stored in an archive. A complexity analysis of the proposed AMOSA has been provided. The performance of the newly proposed AMOSA has been compared to the two other well-known MOEA's, namely NSGA-II [61] and PAES [110] (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* [61], *Purity* [22][94], *Spacing* [176] and *MinimalSpacing* [22]. Another measure called *displacement* [52][95], that 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 where we can estimate if the solution set is able to approximate all the sub-fronts. Many existing measures are unable to achieve this.

It may be noted that the multiobjective SA methods developed in [181][188] are on lines similar to ours. The concept of archive or a set of potentially PO solutions is also utilized in [181][188] for storing the non-dominated 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 determines the acceptance probability of x in the earlier attempts, in the present chapter 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 [181][188] where a single form of acceptance probability is considered, the present chapter deals with different forms of acceptance probabilities depending on the domination status, the choice of which are explained intuitively later on.

In [181] an unconstrained archive is maintained. Note that theoretically, the number of Pareto optimal solutions can be infinite. Since the ultimate purpose of an 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 [181] 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 the proposed AMOSA. Clustering has also been used earlier in [212].

For comparing the performance of real-coded AMOSA with that of the multiobjective SA (MOSA) [181], six three objective test problems, namely, DTLZ1-DTLZ6 are used. Results demonstrate that the performance of

AMOSA is comparable to, 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 specially 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 [61] do not work well on many-objective optimization problems as pointed out in some recent studies [91][93].

2.4.2 Archived Multiobjective Simulated Annealing (AMOSA) [27]

AMOSA incorporates the concept of an *Archive* where the non-dominated solutions seen so far are stored. In [70], the use of unconstrained *Archive* size to reduce the loss of diversity is discussed in detail. In our approach we have kept the archive size limited since finally only a limited 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 non-dominated solutions are stored in the *Archive* as and when they are generated until the size of the *Archive* increases to *SL*. Thereafter if more non-dominated 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 proposed simulated annealing based AMOSA is shown in Figure 2.4. 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 non-dominated 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.

Algorithm AMOSA

```
Set $Tmax$, $Tmin$, $HL$, $SL$, $iter$, $\alpha$, 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*/
      $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k}(\Delta dom_{i,new-pt})+\Delta dom_{current-pt,new-pt}\right)}{(k+1)}$.
      /* $k$=total-no-of points in the *Archive* which dominate *new-pt*, $k \geq 0$. */
      Set *new-pt* as *current-pt* with probability calculated using Equation 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by $\Delta dom_{avg}$.

    if (*current-pt* and *new-pt* are non-dominating 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)*/
        $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k}\Delta dom_{i,new-pt}\right)}{k}$.
        Set *new-pt* as *current-pt* with probability calculated using Equation 2.3
        with $(-(E(q,T) - E(s,T))$ replaced by $\Delta dom_{avg}$.
      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)*/
        $\Delta dom_{min}$ = 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 $\Delta dom_{min}$ as *current-pt* with probability=$prob$.
        else set *new-pt* as *current-pt*
      if (*new-pt* is non-dominating 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.
```

Figure 2.4: The AMOSA Algorithm

- *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 non-dominated solutions (*ND*) that are obtained are stored in the *Archive*, up to a maximum of *HL*. In case the number of non-dominated solutions exceeds *HL*, clustering is applied to reduce the size to *HL* (the clustering procedure is explained below). That means 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 [181], 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 future we intend 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 the diversity of the non-dominated solutions. In general, the size of 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 upto *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 atmost *HL* non-dominated solutions and reduces excessive calls to clustering in the initial stages of the AMOSA process.

For clustering, the well-known Single linkage algorithm [96] 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 [212], except that they have used average linkage method [96]. 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 = 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 these 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 Figure 2.5 for a two objective case. $\Delta dom_{a,b}$ is used in AMOSA while computing the probability of acceptance of a newly generated solution.

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*. The *current-pt* is perturbed to generate a new solution called *new-pt*. The domination status of *new-pt* is checked with respect to the *current-pt* and solutions in the *Archive*.

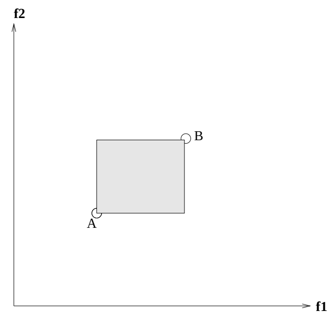


Figure 2.5: Total amount of domination between the two solutions A and B = the area of the shaded rectangle

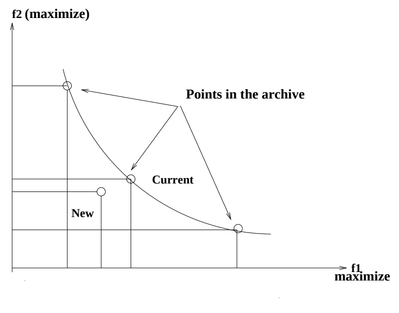
Based on the domination status between *current-pt* and *new-pt* three different cases may arise. These are enumerated below.

- Case 1: *current-pt* dominates the *new-pt* and k ($k \geq 0$) points from the *Archive* dominate the *new-pt*.
  This situation is shown in Figure 2.6. Here Figure 2.6(a) and 2.6(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, the *new-pt* is selected as the *current-pt* with a probability calculated using Equation 2.3 with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg} where

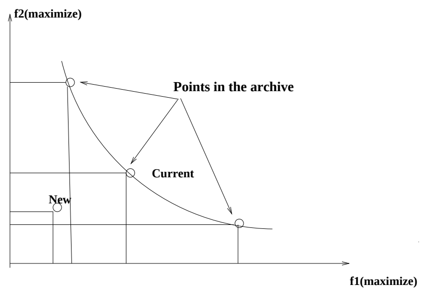
  $$\Delta dom_{avg} = (\sum_{i=1}^{k}(\Delta dom_{i,new-pt}) + \Delta dom_{current-pt,new-pt})/(k+1).$$

  Note that Δdom_{avg} denotes the average amount of domination of the *new-pt* by $(k+1)$ points, namely, the *current-pt* and k points of the *Archive*. Also, as k increases, Δdom_{avg} will increase since here the dominating points that are further away from the *new-pt* are contributing to its value.
  Lemma: When $k = 0$, the *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 the *new-pt*, by transitivity, A will also dominate the *new-pt*. However, we have considered that



(a)



(b)

Figure 2.6: Different cases when *New* is dominated by *Current* (a) *New* is non-dominating with respect to the solutions of *Archive* except *Current* if it is in the *Archive* (b) Some solutions in the *Archive* dominate *New*

no other point in the *Archive* dominates the *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 Figure 2.7(a). The *new-pt* is selected as the *current-pt* with a probability calculated using Equation 2.3 with $(-(E(q,T) - E(s,T))$ replaced by Δdom_{avg} where

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

     Note that here the *current-pt* may or may not be on the archival front.

  2. *new-pt* is non-dominating with respect to the other points in the *Archive* as well. In this case the *new-pt* is on the same front as the *Archive* as shown in Figure 2.7(b). So the *new-pt* is selected

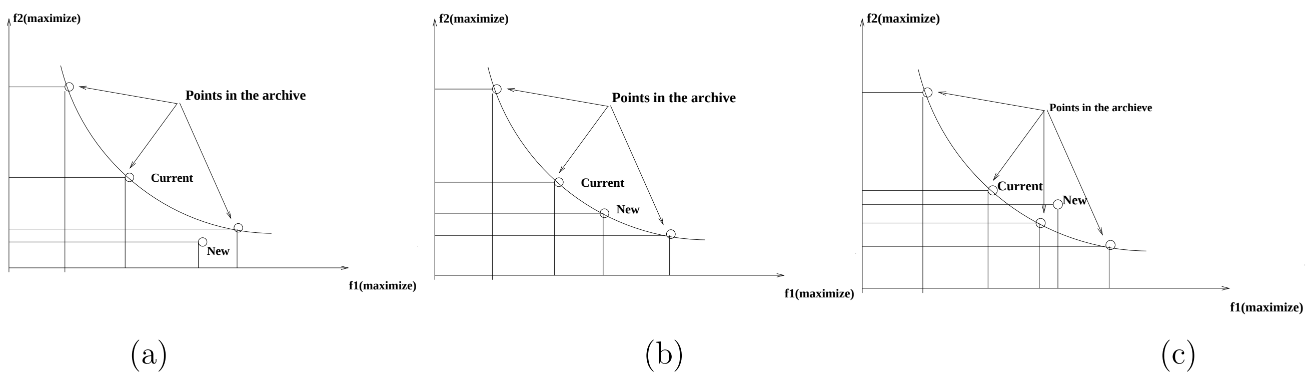


Figure 2.7: Different cases when *New* and *Current* are non-dominating (a) Some solutions in *Archive* dominates *New* (b) *New* is non-dominating with respect to all the solutions of *Archive* (c) *New* dominates k ($k \geq 1$) solutions in the *Archive*

.

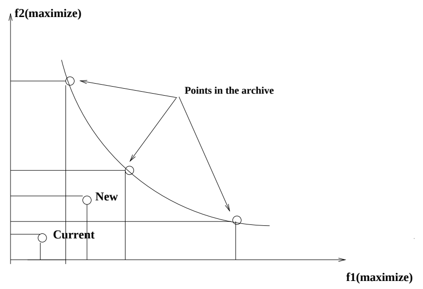
as the *current-pt* and added to the *Archive*. In case the *Archive* becomes overfull (i.e., the *SL* is exceeded), clustering is performed to reduce the number of points to *HL*.

3. *new-pt* dominates k ($k \geq 1$) points of the *Archive*. This situation is shown in Figure 2.7(c). Again, the *new-pt* is selected as the *current-pt*, and added to the *Archive*. All the k dominated points are removed from the *Archive*. Note that here too the *current-pt* may or may not be on the archival front.

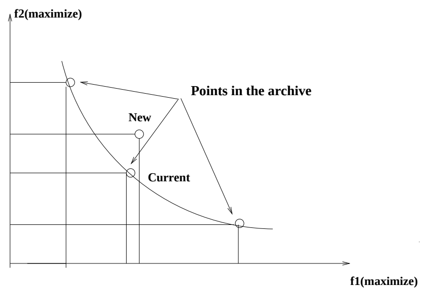
- Case 3: *new-pt* dominates *current-pt*

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

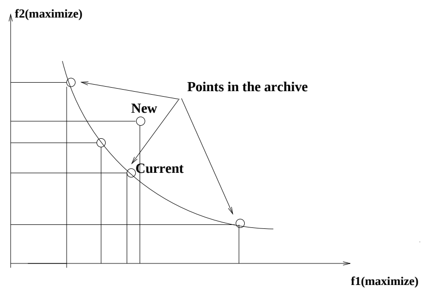
  1. *new-pt* dominates the *current-pt* but k ($k \geq 1$) points in the *Archive* dominate this *new-pt*. Note that this situation (shown in Figure 2.8(a)) may arise only if the *current-pt* is not a member of the *Archive*. Here, the minimum of the difference of domination amounts between the *new-pt* 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



(a)



(b)



(c)

Figure 2.8: Different cases when *New* dominates the *Current* (a) *New* is dominated by some solutions in *Archive* (b) *New* is non-dominating 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*

as the *current-pt* with $prob = \frac{1}{1+exp(-\Delta dom_{min})}$. Otherwise the *new-pt* is selected as the *current-pt*. Note that according to the SA paradigm, the *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 the *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 non-dominating with respect to the points in the *Archive* except the *current-pt* if it belongs to the *Archive*. This situation is shown in Figure 2.8(b). Thus *new-pt*, which is now accepted as the *current-pt*, can be considered as a new non-dominated solution that must be stored in *Archive*. Hence *new-pt* is added to the *Archive*. If the *current-pt* is in the *Archive*, then it is removed. Otherwise, if the number of points in the *Archive* becomes more than the *SL*, clustering is performed to reduce the number

of points to *HL*.
Note that here the *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 Figure 2.8(c)). Hence, the *new-pt* is selected as the *current-pt* and added to the *Archive*, while all the dominated points of the *Archive* are removed. Note that here the *current-pt* may or may not be on the archival front.

The above process is repeated *iter* times for each temperature (*temp*). 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 non-dominated solutions.

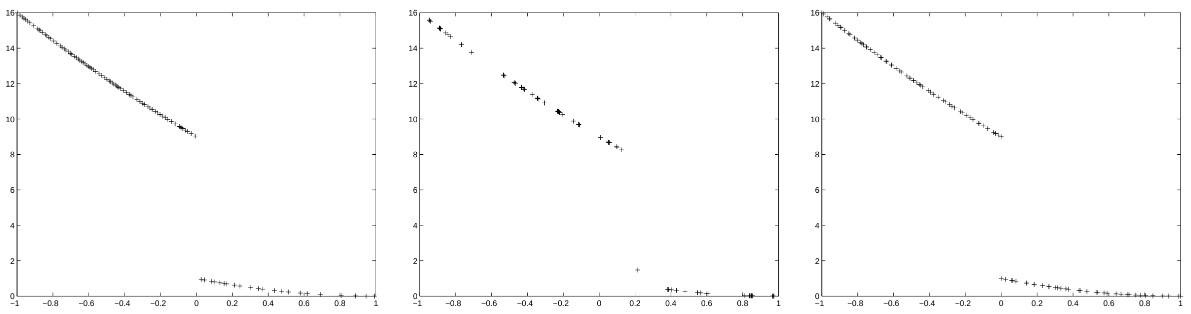
Note that in AMOSA, as in other versions of multiobjective SA algorithms, there is a possibility that a new solution worse than the current solution may be selected. In most other MOEAs, e.g., NSGA-II, PAES, if a choice needs to be made between two solutions $\overline{x}$ and $\overline{y}$, and if $\overline{x}$ 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 non-zero 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 characteristics 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.



(a) (b) (c)

Figure 2.9: The final non-dominated front for SCH2 obtained by (a) AMOSA (b) PAES (c) NSGA-II

3. Procedure to check the domination status between a particular solution and the *Archive* members: O($M \times SL$).

4. Single linkage clustering: O($SL^2 \times \log(SL)$) [199].

5. Clustering procedure is executed

   - once after initialization if $|\mathrm{ND}| > HL$
   - after each ($SL - HL$) number of iterations.
   - at the end if final $|Archive| > HL$

   So maximum number of times the Clustering procedure is called= ($TotalIter/(SL-HL)$)+2.
   Therefore, total complexity due to Clustering procedure is O(($TotalIter/(SL-HL)$) $\times$ $SL^2 \times \log(SL)$).

Total complexity of AMOSA becomes

$$(SL + M + M \times SL) \times (TotalIter) + \frac{TotalIter}{SL - HL} \times SL^2 \times \log(SL). \quad (2.4)$$

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 the

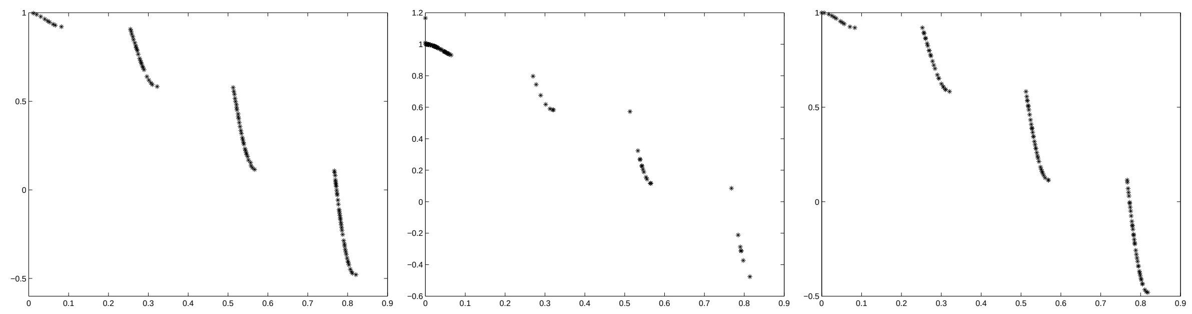

(a) (b) (c)

Figure 2.10: The final non-dominated front for Deb4 obtained by (a) AMOSA (b) PAES (c) NSGA-II

AMOSA becomes

$$(TotalIter) \times (\beta \times N + M + M \times \beta \times N + (\beta^2/(\beta - 1)) \\ \times N \times \log(\beta N)), \qquad (2.5)$$

or,

$$O(TotalIter \times N \times (M + \log(N))). \qquad (2.6)$$

Note that the total complexity of NSGA-II is O(*TotalIter*$\times M \times N^2$) and that of PAES is O(*TotalIter* $\times M \times N$). 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)).

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

Test	*Convergence*			*Purity*		
Problem	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)

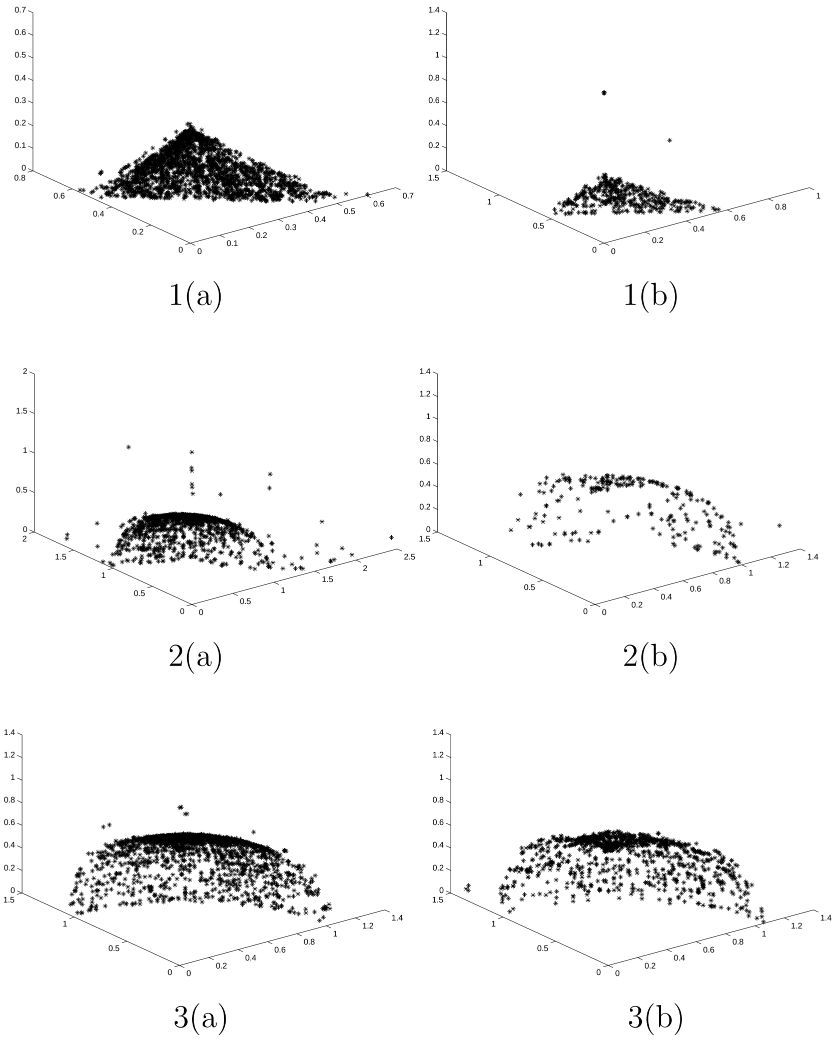


Figure 2.11: The final non-dominated front obtained by (a) AMOSA (b) MOSA for the test problems (1) DTLZ1 (2) DTLZ2 (3) DTLZ3

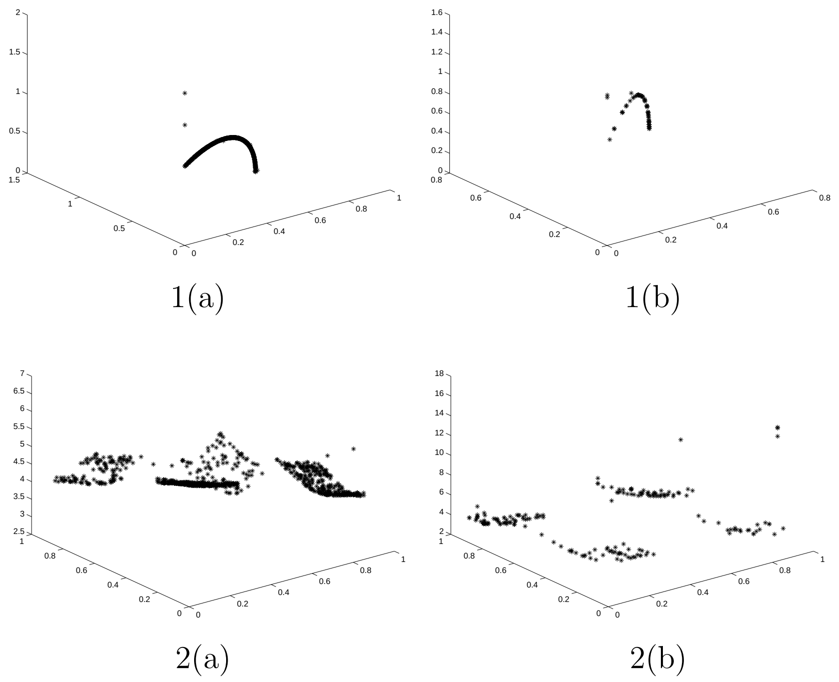


Figure 2.12: The final non-dominated front obtained by (a) AMOSA (b) MOSA for the test problems (1) DTLZ5 (2) DTLZ6

2.5 Simulation Results

In this section, we first describe comparison metrics used for the experiments. The performance analysis of both the binary-coded AMOSA and the real-coded AMOSA are also provided in this section.

2.5.1 Comparison Measures

In multiobjective optimization, there are basically two functionalities that an MOO strategy must achieve regarding the obtained solution set [60]. 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 opti-

Table 2.2: *Spacing* and *MinimalSpacing* measures on the test functions for binary encoding

Test Problem	*Spacing*			*MinimalSpacing*		
	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

mal 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 we have mainly used three such performance measures. The first measure is the *Convergence* measure γ [61]. It measures the extent of convergence of the obtained solution set to a known set of PO solutions. Lower the value of γ, better is the convergence of the obtained solution set to the true PO front. The second measure called *Purity* [22][94] 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.

Table 2.4: Time taken by different programs (in sec) 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

Table 2.5: *Convergence*, *Purity* and *Minimal Spacing* measures on the 3 objective test functions while *Archive* is bounded to 100

Test	$Convergence$			$Purity$			$MinimalSpacing$		
Problem	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

The third measure named *Spacing* was first proposed by Schott [176]. It reflects the uniformity of the solutions over the non-dominated front. It is shown in [22] that this measure will fail to give adequate result in some situations. In order to overcome the above limitations, a modified measure, named *MinimalSpacing* is proposed in [22]. Smaller values of *Spacing* and *MinimalSpacing* 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 extents, none of the existing measures, will be able to reflect this. In case of discontinuous PO front, this problem becomes severe when an algorithm totally misses a sub-front. Here a performance measure which is very similar to the measure used in [52] and [95] named *displacement* is used that is able to overcome this limitation. It measures how far the obtained solution set is from a known set of PO solutions. In order to compute *displacement* measure, a set P^* consisting of uniformly spaced solutions from the true PO front in the objective space is found (as

Table 2.6: *Convergence*, *Purity* and *Minimal Spacing* measures on the 3 objectives test functions by AMOSA and MOSA while *Archive* is unbounded

Test	*Convergence*		*Purity*		*MinimalSpacing*	
Problem	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

is done while calculating γ). Then *displacement* is calculated as

$$displacement = \frac{1}{|P^*|} \times \sum_{i=1}^{|P^*|} (min_{j=1}^{|Q|} d(i,j)) \qquad (2.7)$$

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. Lower the value of this measure, better is 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, we have compared the binary encoded AMOSA with the binary-coded NSGA-II and PAES algorithms. For AMOSA binary mutation is used. Seven test problems have been considered for the comparison purpose. These are SCH1 and SCH2 [60], Deb1 and Deb4 [59], ZDT1, ZDT2, ZDT6 [60]. All the algorithms are 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. We have chosen the values of *Tmax* (maximum value of the temperature), *Tmin* (minimum value of the temperature) and *iter* (number of iterations at each temperature) so that 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. Maximum iterations for NSGA-II and PAES are 500 and 50000 respectively. For AMOSA, $Tmax = 200$, $Tmin = 10^{-7}$, $iter = 500$. The parameter values were determined after extensive sensitivity studies, which are omitted here to restrict the size of the chapter.

Discussions of the Results

The *Convergence* and *Purity* values obtained using the three algorithms are shown in Table 2.1. AMOSA performs best for ZDT1, ZDT2, ZDT6 and Deb1 in terms of γ. For SCH1 all three are performing equally well. NSGA-II performs well for SCH2 and Deb4. Interestingly, for all the functions, AMOSA is found to provide more number of overall non-dominated solutions than NSGA-II. (This is evident from the quantities in parentheses in Table 2.1 where $\frac{x}{y}$ indicates that on an 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 to 32 seconds for NSGA-II in case of ZDT1. From Table 2.1 it is again clear that AMOSA and PAES are always giving more number of distinct solutions than NSGA-II.

Table 2.2 shows the *Spacing* and *MinimalSpacing* measurements. AMOSA is giving the best performance of *Spacing* most of the times while PAES performs the worst. This is also evident from Figures 2.9 and 2.10 which show the final PO fronts of SCH2 and Deb4 obtained by the three methods for the purpose of illustration. With respect to *MinimalSpacing* the performances of AMOSA and NSGA-II are comparable.

Table 2.3 shows the value of *displacement* for five problems, two with discontinuous and three with continuous PO fronts. AMOSA performs the best in almost all the cases. The utility of the new measure is evident in particular for Deb4 where PAES performs quite poorly (see Figure 2.10). 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 shows 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 smaller complexity. AMOSA takes less time than NSGA-II in 30 variable problems like ZDT1, ZDT2, 10 variable problem ZDT6. But for single and two variable problems SCH1, SCH2, Deb1 and Deb4, AMOSA takes more time than NSGA-II. This may be due to 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.

2.5.3 Comparison of Real-coded AMOSA with the Algorithm of Smith et al. [181] and Real-coded NSGA-II

The real-coded version of the proposed AMOSA has also been implemented. The mutation is done as suggested in [181]. 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, $p(\epsilon) \propto e^{-\frac{|\epsilon-\mu|}{\delta}}$, 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 equals to 0.1 is used for mutation. The initial temperature is selected by the procedure mentioned in [181]. 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 total number of generations is set such 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 investigations, the archive size is kept unlimited as in [181]. The results are compared to those obtained by MOSA [181] and provided in [1].) AMOSA is executed for a total of 5000, 1000, 15000, 5000, 1000, 5000 and 9000 run lengths respectively for DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ5 and DTLZ6. Total number of iterations, $iter$, per temperature is set accordingly. We have run real-coded NSGA-II (code obtained from KANGAL site: http://www.iitk.ac.in/kangal/codes.html). 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 [181] 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 that has an unlimited archive [181], the clustering procedure (adopted for AMOSA), is used to reduce the number of solutions of [1] to 100. Comparison is performed in terms of *Purity*, *Convergence* and *Minimal Spacing*. Table 2.5 shows the *Purity*, *Convergence*, *Minimal Spacing* measurements for DTLZ1-DTLZ6 problems 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 among all. Table 2.5 shows the *Minimal Spacing* values obtained by the 3 algorithms for DTLZ1-DTLZ6. AMOSA performs the best in all the cases.

As mentioned earlier, for comparing the performance of MOSA (by considering the results reported in [1]), a version of AMOSA without clustering and with unconstrained archive is executed. The results reported here are the averages over 10 runs. Table 2.6 shows the corresponding *Purity*, *Con-*

vergence and *Minimal Spacing* values. Again AMOSA performs much better than MOSA for all the test problems except DTLZ4. For DTLZ4, the MOSA performs better than that of AMOSA in terms of both *Purity* and *Convergence* values. Figure 2.11 shows the final Pareto optimal front obtained by AMOSA and MOSA for DTLZ1-DTLZ3 while Figure 2.12 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 to MOSA.

It was mentioned in [210] that for a particular test problem, almost 40% of the solutions provided by an algorithm with truncation of archive got dominated by the solutions provided by an algorithm without archive truncation. However, the experiments we conducted did not adequately justify this finding. Let us denote the set of solutions of AMOSA with and without clustering as S_c and S_{wc} respectively. We found that for DTLZ1, 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 are non-dominating to each other and the solutions are same. Because execution of AMOSA without clustering doesn't 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 are non-dominating to each other.

To have a look at the performance of the AMOSA on a four-objective problem, we apply AMOSA and NSGA-II to the 13-variable DTLZ2 test problem. 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 the algorithms are 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 shown in Table 2.7. AMOSA performs much better than NSGA-II in terms of all the three measures.

The proposed AMOSA and NSGA-II are 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 the algorithms are executed for a total of 1,00,000 function evaluations for these three test problems (for NSGA-II popsize=200, number of generations=500) and the corresponding *Purity*, *Convergence* and *Minimal Spacing* values are shown in Table 2.7. *Convergence* value indicates that NSGA-II doesn't converge to the true PO front where as AMOSA reaches the true PO front for all the 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 [91][93].

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	*Convergence*		*Purity*		*MinimalSpacing*	
Problem	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) initial value of temperature ($Tmax$), (ii) cooling schedule, (iii) number of iterations to be performed at each temperature and (iv) stopping criterion to terminate the algorithm. 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 [192]. In this chapter, as in [181], we have set the initial temperature to achieve an initial acceptance rate of approximately 50% on derogatory proposals. This is described in Section 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 [192]. The cooling schedule should be so chosen that it is able to strike a good balance between exploration and exploitation of the search space. In order to investigate the performance of AMOSA with another cooling schedule, the following is considered (obtained from http://members.aol.com/btluke/simanf1.htm):

$$T_i = T_0 \left(\frac{T_N}{T_0}\right)^{i/N}.$$

Here N is the total number of iterations, T_N is the final temperature and T_0 is the initial temperature. T_i is the temperature at iteration i. AMOSA with the above cooling schedule is applied on ZDT1. The *Convergence* and *Minimal Spacing* values obtained are 0.008665 and 0.017 respectively. Comparing with the corresponding values in Table 2.1 and Table 2.2 it is found that the results with this cooling schedule are somewhat poorer. However, an exhaustive sensitivity study needs to be performed for AMOSA.

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 [192], 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, where as in some other, the minimum value of the temperature is specified. Detailed discussion on this issue can be found in [192].

2.6 Discussion and Conclusions

This chapter first describes some existing single and multiobjective optimization techniques. Thereafter in this chapter a simulated annealing based multiobjective optimization algorithm has been proposed. The concept of amount of domination is used in solving the multiobjective optimization problems. 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) [61] and PAES [110] for a suite of seven 2-objective test problems having different complexity levels. In a part of the investigation, comparison of the real-coded version of the proposed algorithm is conducted with a very recent multiobjective simulated annealing algorithm MOSA [181] and real-coded NSGA-II for six 3-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 like *Convergence*, *Purity*, *MinimalSpacing*, and *Spacing*, and the time taken are used for the purpose of comparison. In this regard, a measure called *displacement* has also been used that 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 more than that of PAES but smaller than that of NSGA-II.

It is seen from the given results that the performance of the proposed 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 like 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 [61] do not work well on many-objective optimization problems as pointed out in some recent studies [91][93]. An interesting feature of AMOSA, as in other versions of multi-objective SA algorithms, is that it has a non-zero 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 in other MOO algorithms in order to improve the performance.

Clustering [96][197] is an important problem in data-mining and pattern recognition. It has applications in a large number of fields. In this thesis the problem of clustering a data set is posed as an optimization problem. Next chapter uses some of the single and multi objective approaches used in this chapter to solve the problem of clustering a data set.

Chapter 3

A Point Symmetry Based Distance Measure and its Application to Single and Multi Objective Clustering

3.1 Introduction

Clustering [96][197] is an important problem in data-mining and pattern recognition. It has applications in a large number of fields. 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. Symmetry is considered an important feature, use of which enhances the recognition of different structures [10]. Therefore incorporating symmetry while searching for cluster structures in the data appears to be natural.

Based on the above observations, a new point symmetry based distance (PS-distance) is proposed in this chapter which incorporates both the Euclidean distance as well as a measure of symmetry. For reducing the complexity of computing the PS-distance, use of Kd-tree [4] is proposed in this chapter. As already mentioned, K-means is a widely used clustering algorithm. However K-means is known to get stuck at sub-optimal solutions depending on the choice of the initial cluster centers. In order to overcome this limitation, genetic algorithms have been used as the underlying optimization technique [131]. Genetic Algorithms (GAs) [83] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and having a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In view of the advantages of the GA-based clustering method [131] over the standard K-means, the former has been used in this work. In the proposed GA with point symmetry distance (GAPS) based clustering technique, the assignment of the points to the different clusters is done based on the newly proposed point symmetry distance rather the Euclidean distance. This enables the proposed algorithm to detect both convex and non-convex clusters of any shape and sizes as long as the clusters do have some symmetry property. The convergence of the proposed GAPS clustering technique is also established in the present chapter.

Note that 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 the different data characteristics. In order to achieve this the problem of clustering a data set has been posed as one of multiobjective optimization. Based on this observation a multiobjective clustering technique for fixed number of clusters is also proposed in the present chapter. This technique again uses the point symmetry based distance for assignment of points to different clusters and the AMOSA algorithm, described in Chapter 2, as the underlying MOO technique.

3.2 Some Existing Symmetry Based Distance Measures

As discussed in Chapter 1, Section 1.3.3 a point symmetry distance was proposed by Su and Chou in [187]. This is defined as follows: Given N patterns, $\overline{x}_j$, $j = 1, \dots N$, 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

$$d_s(\overline{x}_j, \overline{c}) = \min_{i=1,\dots N \text{ and } i \neq j} \frac{\|(\overline{x}_j - \overline{c}) + (\overline{x}_i - \overline{c})\|}{\|(\overline{x}_j - \overline{c})\| + \|(\overline{x}_i - \overline{c})\|} \tag{3.1}$$

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 Equation 3.1 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 $\overline{x}_i = (2 \times \overline{c} - \overline{x}_j)$, 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 [187], 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 Figure 3.1 and the process of cluster assignment is clearly stated in step 3 of the figure.

It is evident from Equation 3.1 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 that cluster that is the farthest. This is intuitively unappealing. In the example shown in Figure 3.2, 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 K-means algorithm point $\overline{x}$ is assigned to the cluster 1. But when SBKM is applied on the result given by K-means algorithm, the following will happen. The symmetrical 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

$$d_s(\overline{x}, \overline{c}_1) = \frac{d_1}{d_e(\overline{x}, \overline{c}_1) + d_e(\overline{x}_1, \overline{c}_1)}, \tag{3.2}$$

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

$$d_s(\overline{x}, \overline{c}_2) = \frac{d_2}{d_e(\overline{x}, \overline{c}_2) + d_e(\overline{x}_2, \overline{c}_2)} \tag{3.3}$$

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 [187], and have suggested a modified measure d_c in [43] that is defined as follows:

$$d_c(\overline{x}_j, \overline{c}) = d_s(\overline{x}_j, \overline{c}) \times d_e(\overline{x}_j, \overline{c}) \tag{3.4}$$

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 [43] corresponding to this new measure. A little thought will show that even this modification will not work for the situation shown in Figure 3.2. Let $\overline{x}_j^*$ be the symmetrical point of $\overline{x}_j$ with respect to $\overline{c}$. Therefore from Equation 3.4 we obtain

$$d_c(\overline{x}_j, \overline{c}) = \frac{d_{symm}(\overline{x}_j, \overline{c})}{d_e(\overline{x}_j, \overline{c}) + d_e(\overline{x}_j^*, \overline{c})} d_e(\overline{x}_j, \overline{c}) \tag{3.5}$$

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 from Equation 3.5, we obtain

$$d_c(\overline{x}_j, \overline{c}) \propto d_{symm}(\overline{x}_j, \overline{c}). \tag{3.6}$$

As a result there is no impact of Euclidean distance, only 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. We refer to the clustering algorithm based on this modified measure as Mod-SBKM algorithm. It has been shown experimentally that Mod-SBKM with this measure will also fail for several data sets considered in Section 3.8.1 of this chapter.

The most limiting aspect of the measures suggested in [187] and [43] is that in cases where K-means provides reasonably good clusters, application of the fine-tuning phase (see algorithm in Figure 3.1) will destroy this structure. Another limitation of the SBKM is that it requires a 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 have chosen θ equals to 0.18. However we have 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 is provided in [187].

In the following section we propose a new definition of the PS-based distance that can overcome the limitations of both the measures d_s and d_c.

3.3 A New Definition of the Point Symmetry Distance [24]

As discussed in Section 3.2, both the 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, we propose a new point symmetry (PS) distance in this chapter which is called $d_{ps}(\overline{x}, \overline{c})$ associated with point $\overline{x}$ with respect to a center $\overline{c}$. The proposed 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 centre $\overline{c}$ 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 of d_is, $i = 1, 2, \ldots knear$ such that the d_is are all distinct. Then

$$\begin{aligned} d_{ps}(\overline{x}, \overline{c}) &= d_{sym}(\overline{x}, \overline{c}) \times d_e(\overline{x}, \overline{c}), && (3.7) \\ &= \frac{\sum_{i=1}^{knear} d_i}{knear} \times d_e(\overline{x}, \overline{c}), && (3.8) \end{aligned}$$

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 Equation 3.8 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 it 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 using Figure 3.3. Here a particular point is $\overline{x}$. The cluster center is denoted by

```
**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,
    $k^* = argmin_{k=1\ldots K} d_s(\overline{x}, \overline{c}_k)$
    where $d_s(\overline{x}, \overline{c}_k)$ is computed using Equation 3.1.
    If $d_s(\overline{x}, \overline{c}_{k^*}) < \theta$ /*$\theta$ is a user specified parameter*/
        assign $\overline{x}$ to the $k^*$th cluster.
    else, compute $k^* = argmin_{k=1,\ldots K} d_e(\overline{x}, \overline{c}_k)$
        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: Updation**: Compute the new centroids of the
    $K$ clusters as follows:
    $\overline{c}_k(t+1) = \frac{\sum_{\overline{x}_i \in S_k(t)} \overline{x}_i}{N_k},\ k = 1 \ldots K$
    where $S_k(t)$ is the set of elements that are assigned
    to the $k$th 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.
```

Figure 3.1: Steps of SBKM algorithm

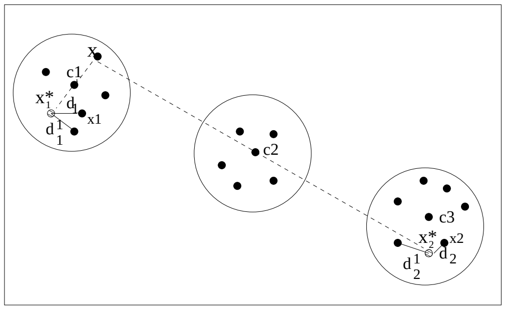


Figure 3.2: Example of a data set having some symmetrical interclusters

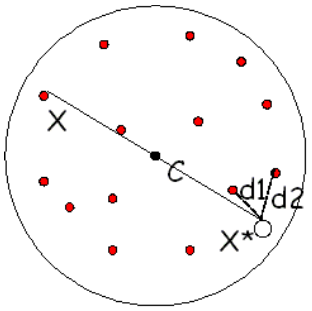


Figure 3.3: Example of point symmetry distance

$\overline{c}$. 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 an Euclidean distances of d_1 and d_2, respectively. Then the point symmetry based distance between $\overline{x}$ and $\overline{c}$ is calculated as $d_{ps}(\overline{x}, \overline{c}) = \frac{d_1+d_2}{2} \times d_e(\overline{x}, \overline{c})$.

The basic differences between the PS based distances in [187] and [43], and the proposed 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 Figure 3.2. Note that if only the nearest neighbor of $\overline{x}^*$ is considered as in d_s of [187], 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 Section 3.2.

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 though 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 [43][187]). This will depend on certain conditions discussed in detail in the next section.

4. We also provide a rough guideline of the choice of θ, the threshold value on the d_{ps} which is used for symmetry based cluster assignment. 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

$$d_{NN}^{max} = \max_{i=1,\ldots N} d_{NN}(\overline{x}_i), \tag{3.9}$$

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, we can bound d_1 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, we have kept the threshold θ equals to d_{NN}^{max}, making its computation automatic and without user intervention.

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

It is to be noted that $d_{ps}(\overline{x}, \overline{c})$ is a non-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 the newly proposed measure are established. For this purpose, some terms are also defined.

Definition 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 $i = argmin_v \Delta(\overline{x}, \overline{c}_v)$, 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 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 1: *The proposed 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

$$\frac{d_{sym}(\overline{x}, \overline{c}_1)}{d_{sym}(\overline{x}, \overline{c}_2)} < \frac{d_{e2}}{d_{e1}}, \tag{3.10}$$

the point $\overline{x}$ is assigned to center $\overline{c}_1$. Point $\overline{x}$ is assigned to 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 the d_{sym} satisfies the EDR property defined in Definition 1. ♣

It may be noted that for data sets having convex clusters of different densities, the newly proposed point symmetry based distance will detect the appropriate clustering as long as the condition in Equation 3.10 holds well for the points.

Definition 2: If two clusters are symmetrical to each other with respect to a third cluster center, then these clusters are called "symmetrical interclusters".

Observation 2: *The proposed d_{ps} measure is able to detect the 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 2nd 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, $\frac{d_{sym}(\overline{x},\overline{c}_1)}{d_{sym}(\overline{x},\overline{c}_2)} < \frac{d_e(\overline{x},\overline{c}_2)}{d_e(\overline{x},\overline{c}_1)} \Rightarrow d_{sym}(\overline{x},\overline{c}_1) \times d_e(\overline{x},\overline{c}_1) < d_{sym}(\overline{x},\overline{c}_2) \times d_e(\overline{x},\overline{c}_2) \Rightarrow d_{ps}(\overline{x},\overline{c}_1) < d_{ps}(\overline{x},\overline{c}_2)$. Thus point $\overline{x}$ will be assigned to cluster of $\overline{c}_1$. This will happen for all the points of cluster 1. Similarly points which should belong to cluster 3 will form cluster 3. Thus, the proposed d_{ps} measure is able to detect the symmetrical interclusters properly.$\clubsuit$

The above observation is also evident from Figure 3.2. In Figure 3.2, the first and the third clusters are "symmetrical interclusters" with respect to the middle one. As explained in the above example, though 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 Figure 3.2 are identified properly. Thus it is proved that the proposed 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 the computational complexity, an approximate nearest neighbor search using the Kd-tree approach is adopted in this chapter.

3.5 Kd-tree Based Nearest Neighbor Computation

A space-partitioning data structure used for arranging points in K dimensional space is Kd-tree or K-dimensional tree [4]. Splitting planes used by Kd-tree are only those which are perpendicular to one of the coordinate axes. In case of 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++ . The usage of the data structures and algorithms improvised for finding precise as well as comparative nearest neighbors in arbitrary high dimensional space are supported by ANN. In this chapter ANN is used to find d_1 and d_2 in Equation 3.8 efficiently. ANN library devices 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 $\epsilon \geq 0$ may be supplied. If so, then ith nearest neighbor is $(1+\epsilon)$ 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 $(1+\epsilon)$. If ϵ is omitted then the nearest neighbors will be computed exactly.

For computing $d_{ps}(\overline{x}, \overline{c})$ in Equation 3.8, 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 computation intensive task that can be speeded up by using the Kd-tree based nearest neighbor search. For the purpose of this chapter, the exact nearest neighbor is computed; so the 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 the proposed $d_{ps}(\overline{x}, \overline{c})$, for optimization.

3.6 GAPS: The Genetic Clustering Scheme with the Proposed PS Distance [24]

A genetic algorithm based clustering technique which uses the point symmetry based distance is proposed in this section. The method is referred to as GAPS clustering [24]. Here the number of clusters is assumed to be known *apriori*, 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 Figures 3.4 and 3.5.

3.6.1 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 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 is 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.

3.6.2 Fitness Computation

In order to compute the fitness of the chromosomes, the clustering procedure clustering-PS() (as shown in Figure 3.5) is called. Here a point $\overline{x}_i$, $1 \le i \le n$, is assigned to cluster k iff $d_{ps}(\overline{x}_i, \overline{c}_k) \le d_{ps}(\overline{x}_i, \overline{c}_j)$, $j = 1, \ldots, K$, $j \ne k$ and $(d_{ps}(\overline{x}_i, \overline{c}_k)/d_e(\overline{x}_i, \overline{c}_k)) \le \theta$. 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 $d_e(\overline{x}_i, \overline{c}_m) \le d_e(\overline{x}_i, \overline{c}_j)$, $j = 1, 2 \ldots K$, $j \ne m$. In other words, point $\overline{x}_i$ is assigned to that cluster with respect to whose center its d_{ps} is the 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 [16] 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, then 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 Section 3.3 of the present chapter. 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:

$$M = \sum_{i=1}^{K} \sum_{j=1}^{n_i} d_{ps}(\overline{x}_j^i, \overline{c}_k), \tag{3.11}$$

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.,

$$F(s_i) = \frac{1}{M} \tag{3.12}$$

This fitness function, $F(s_i)$, will be maximized by using genetic algorithm.

(Note that there could be other ways of defining the fitness function).

3.6.3 Selection

Roulette wheel selection [78] is used to implement the proportional selection strategy.

3.6.4 Crossover

Here, we have used the normal single point crossover [83]. Crossover probability is selected adaptively as in [184]. The expressions for 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:

$$\mu_c = k_1 \times \frac{(f_{max} - f^{'})}{(f_{max} - \overline{f})}, \quad \text{if } f^{'} > \overline{f}, \tag{3.13}$$

$$\mu_c = k_3, \quad \text{if } f^{'} \leq \overline{f}. \tag{3.14}$$

Here, as in [184], 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.

3.6.5 Mutation

Each chromosome undergoes mutation with a probability μ_m. The mutation probability is also selected adaptively for each chromosome as in [184]. The expression for mutation probability, μ_m, is given below:

$$\mu_m = k_2 \times \frac{(f_{max} - f)}{(f_{max} - \overline{f})} \quad \text{if } f > \overline{f}, \tag{3.15}$$

$$\mu_m \;=\; k_4 \quad \text{if} \quad f \leq \overline{f}. \tag{3.16}$$

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

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, $p(\epsilon) \propto e^{-\frac{|\epsilon - \mu|}{\delta}}$, 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.

3.6.6 Termination

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

```
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
```

Figure 3.4: 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
    $k^* = argmin_{k=1\ldots K} d_{ps}(\overline{x}_i, \overline{c}_k)$
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
    $k^* = argmin_{k=1\ldots K} d_e(\overline{x}, \overline{c_k})$
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 $j$th point of the $i$th 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 $k$th cluster at generation $t$ and $N_k = |S_k|$.
```

Figure 3.5: Main steps of clustering_PS() procedure

3.6.7 Complexity Analysis

Below we analyze the complexity of the proposed GAPS clustering.

1. As discussed in Section 3.5, Kd-tree data structure has been used in order to find the nearest neighbor of a particular point. The construction of Kd-tree requires $O(nlogn)$ time and $O(n)$ space [4].

2. Initialization of 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 we have to calculate the minimum symmetrical distance of that point with respect to all clusters. 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 $O(c^d + logn)$, where c is a constant depending on dimension and the point distribution. This is $O(logn)$ if the dimension d is a constant [30]. Friedman et al. [74] also reported $O(logn)$ expected time for finding the nearest neighbor. So in order to find the minimal symmetrical distance of a particular point, $O(Klogn)$ time is needed.
    For n points, the total complexity becomes $O(Knlogn)$.

    (b) The complexity for updating the centers is $O(K)$.

    So the overall complexity for fitness evaluation is$=O(Popsize \times Knlogn)$.

4. 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 $O(Knlogn \times Popsize)$ per generation. For maximum $maxgen$ number of generations, the overall complexity becomes $O(Knlogn \times Popsize \times maxgen)$. As $Popsize$ is constant, the overall complexity of GAPS clustering is $O(nKlogn \times maxgen)$.

3.7 On the Convergence Property of GAPS [165]

It has been shown using the finite Markov chain theory that the canonical genetic algorithms converge to the global optimum [159]. In [114], it has also been proved along the lines of [159] that 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 minimum symmetrical compactness is proved along similar lines by deriving some conditions on the parameters of GAPS-clustering that ensure the global convergence.

3.7.1 Preliminaries

Consider the process $\{\mathcal{P}(t)\},\ t \geq 0$, where $\mathcal{P}(t)$ represents the population maintained by GAPS at tth generation. The state space, $\mathcal{S}$, of this process refers to the space of all possible populations. The states of this state space can be numbered from 1 to $|\mathcal{S}|$. 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, $\mathcal{P}(t+1)$ can be completely determined by $\mathcal{P}(t)$ as the following:

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

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

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

then $p_{ij}(s) = p_{ij}(t)$ for all $p_i, p_j \in \mathcal{S}$ and for all $s, t \geq 1$. Thus it can be concluded that $\{\mathcal{P}(t)\},\ t \geq 0$ is a time-homogeneous finite Markov chain. Let

$\mathbf{P} = (p_{ij})$ be the *transition matrix* of the process $\{\mathcal{P}(t)\},\ t \geq 0$. The entries of the matrix $\mathbf{P}$ satisfy $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|\mathcal{S}|} p_{ij} = 1,\ \forall i \in \mathcal{S}$. Satisfaction of the above mentioned condition is the required qualification of a stochastic matrix. Since $\mathbf{P}$ satisfies it therefore it can be considered as a *stochastic matrix*. A few terms are defined below which will be further used 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

$$\mathbf{P} = \mathbf{K} \times \mathbf{C} \times \mathbf{M} \times \mathbf{S}, \tag{3.17}$$

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

Proposition 1: *Stochastic matrices form a group under matrix multiplication.*

Thus for the two stochastic matrices, $\mathbf{K}$ and $\mathbf{C}$, by proposition 1, $\mathbf{C}^{'} = \mathbf{K}\times\mathbf{C}$ is also a stochastic matrix. Therefore Equation 3.17 can be written as

$$\mathbf{P} = \mathbf{C}^{'} \times \mathbf{M} \times \mathbf{S}, \tag{3.18}$$

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

Proposition 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 before mentioned product of the stochastic matrices being positive, **M** needs to be positive and **S** needs to be column-allowable.

To Check Whether the Mutation Matrix is Positive

The matrix **M** is positive if any valid string $s \in \mathcal{S}$ can be obtained from any another valid string after application of the corresponding mutation operator. In GAPS-clustering technique, during mutation operation, a particular center is modified by a value generated using 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 more. 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 **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 Equation 3.12. 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:

$$\begin{aligned} s_{ii} &\geq \frac{F(s_1)}{\sum_{l=1}^{P} F(s_l)} \times \frac{F(s_2)}{\sum_{l=1}^{P} F(s_l)} \times \ldots \times \frac{F(s_P)}{\sum_{l=1}^{P} F(s_l)} \qquad (3.19) \\ &= \frac{\prod_{l=1}^{P} F(s_l)}{(\sum_{l=1}^{P} F(s_l))^P} > 0 \ \ \forall i \in S. \qquad (3.20) \end{aligned}$$

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 but still it is always strictly positive; hence selection matrix **S** is *column-allowable.*

3.7.2 Convergence Proof

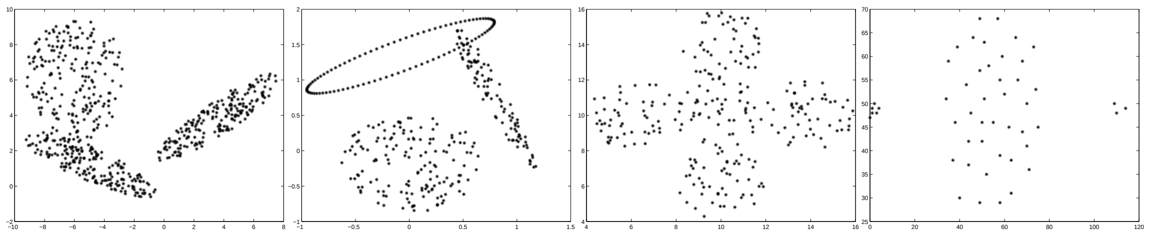
Theorem: *Let* $X(t) = F(s^*(t))$, *where* $s^*(t)$ *is the string with maximum* *fitness value, encountered during the evolution of GAPS-clustering till the time instant t. Let the mutation operator be the same as defined in Section 3.6.5, and the fitness function be as defined in Equation 3.12. Then*

$$\lim_{t\to\infty} Pr\{X(t) = \mathcal{S}^*\} = 1 \tag{3.21}$$

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

Proof: According to the proof provided in [Ref. [159], 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 Equation 3.21. It is proved in Proposition 2 that the transition matrix of the GAPS-clustering with the mutation operator defined in Section 3.6.5 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 upto the present time. Thus, the theorem follows from [Ref. [159], 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.



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

Figure 3.6: (a) *Mixed_3_2* (b) *Sym_3_2* (c) *AD_5_2* (d) *Bensaid_2_2*

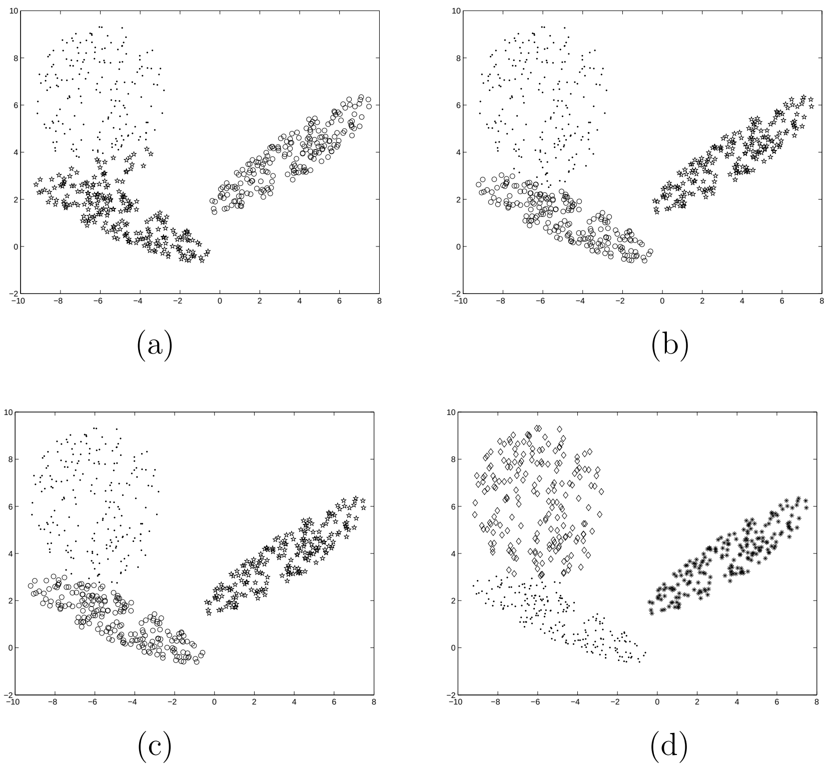


Figure 3.7: Clustering of *Mixed_3_2* for $K = 3$ after application of (a)K-means (b) SBKM (c) Mod-SBKM (d) GAPS

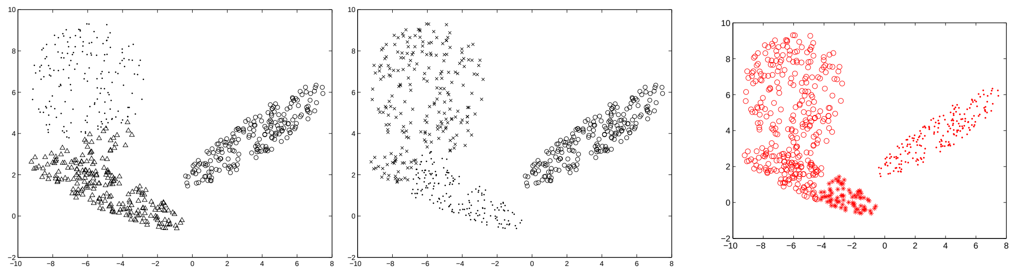
3.8 Experimental Results of GAPS

3.8.1 Data Sets Used

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

1. *Artificial Data Sets*: Four artificial data sets are used.

   (a) *Mixed_3_2*: This data set consists of 600 data points distributed over three clusters as shown in Figure 3.6(a), where each cluster consists of 200 data points.



(a) (b) (c)

Figure 3.8: Clustered *Mixed_3_2* for $K = 3$ after application of (a) GAK-means clustering technique (b) Average Linkage clustering technique (c) Expectation Maximization clustering technique

(b) *Sym_3_2*: This data set is a combination of ring-shaped, compact and linear clusters shown in Figure 3.6(b). The total number of points in it is 350.

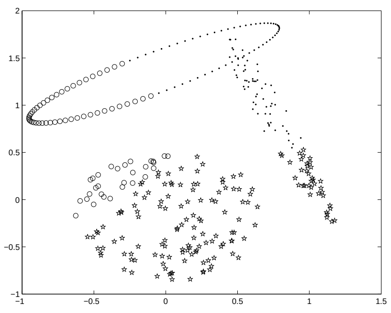
(c) *AD_5_2*: This data set consists of 250 data points distributed over 5 spherically shaped clusters as shown in Figure 3.6(c). The clusters present here are highly overlapping, each consisting of 50 data points.

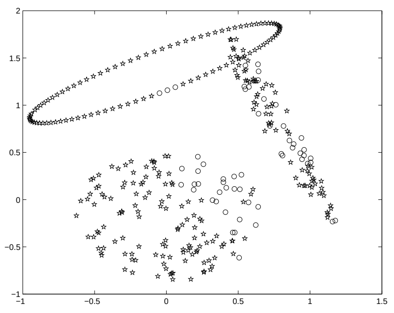
(d) *Bensaid_3_2*: This is a two-dimensional data set consisting of 49 points distributed in three clusters as shown in Figure 3.6(d). This data set, used in Ref. [29], consists of two small clusters (one has six elements and the other has three) separated by a large (40 element) cluster.

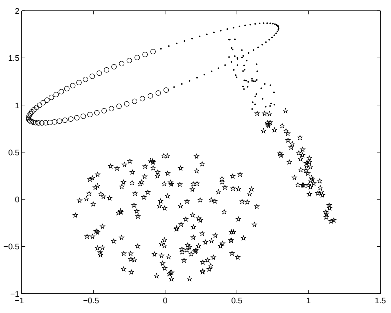
2. Real-life data sets: The four real life data sets were obtained from [2].

(a) *Iris*: *Iris* data set consists of 150 data points distributed over 3 clusters. Each cluster consists of 50 points. This data set represents different categories of irises characterized by four feature values [71]. 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) *Breast Cancer*: This *Wisconsin Breast Cancer* data set consists


(a)


(b)


(c)

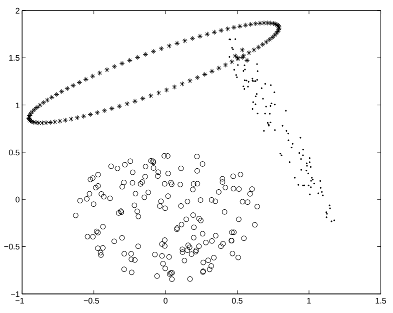
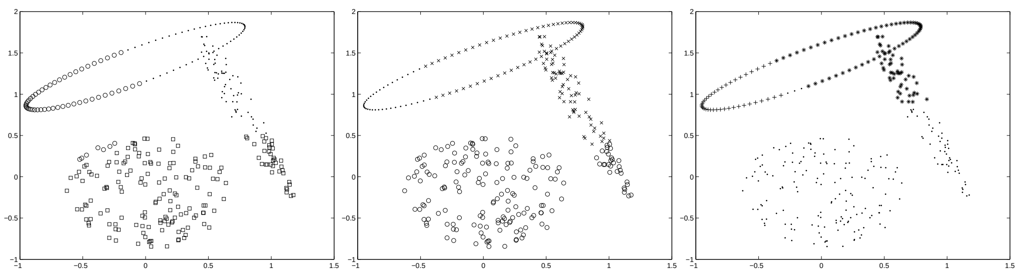

(d)

Figure 3.9: Clustering of *Sym_3_2* for $K = 3$ after application of (a)K-means (b) SBKM (c) Mod-SBKM (d) GAPS

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) *Newthyroid*: The original database from where it has been collected is titled as Thyroid gland data ('normal', 'hypo' and 'hyper' functioning). Five laboratory tests are used to predict whether a patient's thyroid belongs to the class euthyroidism, hypothyroidism or hyperthyroidism. There are a total of 215 instances and the number of attributes is five.



(a) (b) (c)

Figure 3.10: Clustered *Sym_3_2* for $K = 3$ after application of (a) GAK-means clustering technique (b) Average Linkage clustering technique (c) Expectation Maximization clustering technique

(d) *LungCancer*: This data consists of 32 instances having 56 features each. The data describes 3 types of pathological lung cancers.

3.8.2 Discussion on Parameter Values

The proper choice of parameters in genetic algorithm, e.g., population size, number of generations, probabilities of crossover and mutation, etc., is crucial for its good performance. Different parameter values might yield very different results. A good setting may result in convergence of the algorithm to the best solution within a reasonable time period. In contrast, a poor setting might cause the algorithm to execute for a very long time before finding a good solution. Sometimes it may so happen that it is not able to find a good solution at all. Theoretical results indicating the optimal values of the parameters and their best combination have proved difficult to derive in the past. Below we provide a short discussion on the choice of parameter values in genetic algorithm.

1. Population size: Selecting the appropriate population size is a very crucial problem in genetic algorithm. The population size is mainly related to the problem's difficulty. For a more difficult problem, larger population size should be used in order to reliably achieve a good solution. It is also intuitive to spend more resources for GA in order to

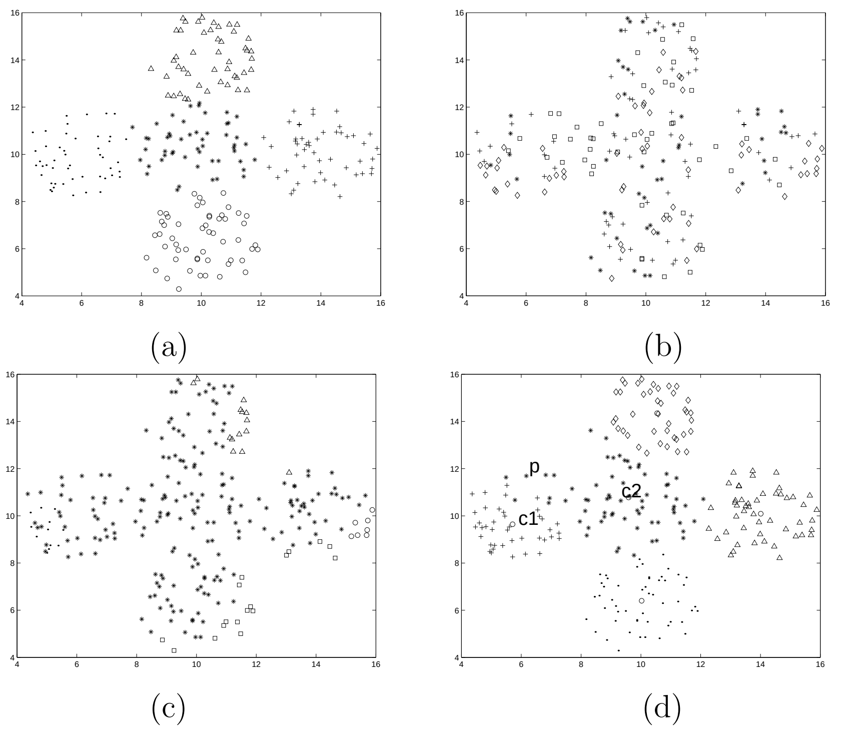
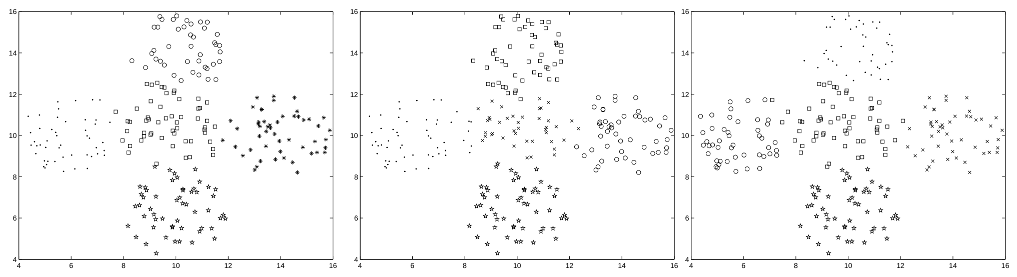


Figure 3.11: Clustering of AD_5_2 for $K = 5$ after application of (a)K-means (b) SBKM (c) Mod-SBKM (d) GAPS

solve the larger problems. Thus, in general, larger population size is necessary when the search space grows.

Goldberg [78] has theoretically analyzed the population size in GAs. According to his analysis the optimal population size increases exponentially and is rather large for even moderate chromosome lengths. It has been shown in [78] that the number of schemata processed effectively is proportional to n^3, where n is the population size. This seems to justify the selection of large population size. It is also evident from nature that large populations are more stable and resist evolution more than small populations perhaps founded by only a few colonists. However, the larger the population size, the longer the genetic algorithm takes to compute each generation. A large population size is also less appealing if fast convergence and great divergence are aimed at[78].



(a) (b) (c)

Figure 3.12: Clustered *AD_5_2* for $K = 5$ after application of (a) GAK-means clustering technique (b) Average Linkage clustering technique (c) Expectation Maximization clustering technique

2. Number of generations: A burning issue in GAs is the number of generations to execute before terminating the algorithm. A simple GA generally converges within a few generations. The pure selection convergence times are $O(\log N)$ generations, where N is the size of the population [78]. Thus GA generally searches fairly quickly. This can be thought of as a strength of GA if the population size is made larger and/or GA is restarted with a new random initial population upon detection of population convergence.

   In [78] it is mentioned that for a given adequate population size if some linkage knowledge is incorporated into the chromosomes then it is expected that "mixing" of good building blocks can take place before convergence. Thus it is important to detect near-uniformity of the population and terminate the GA, before wasting function evaluations on an inefficient, mutation-based search. Useful diversity can also be maintained in the population using some niching method which can balance convergence. There exists a number of estimates of population convergence. One most commonly used measure is "bit-wise average convergence measure" which is used in some of the experiments in [78]. Generally, this convergence is measured at each loci (i.e., the percentage of the population with a single allele at that locus) and averaged over all loci. When this average exceeds some threshold, say 90%, the algorithm is terminated.

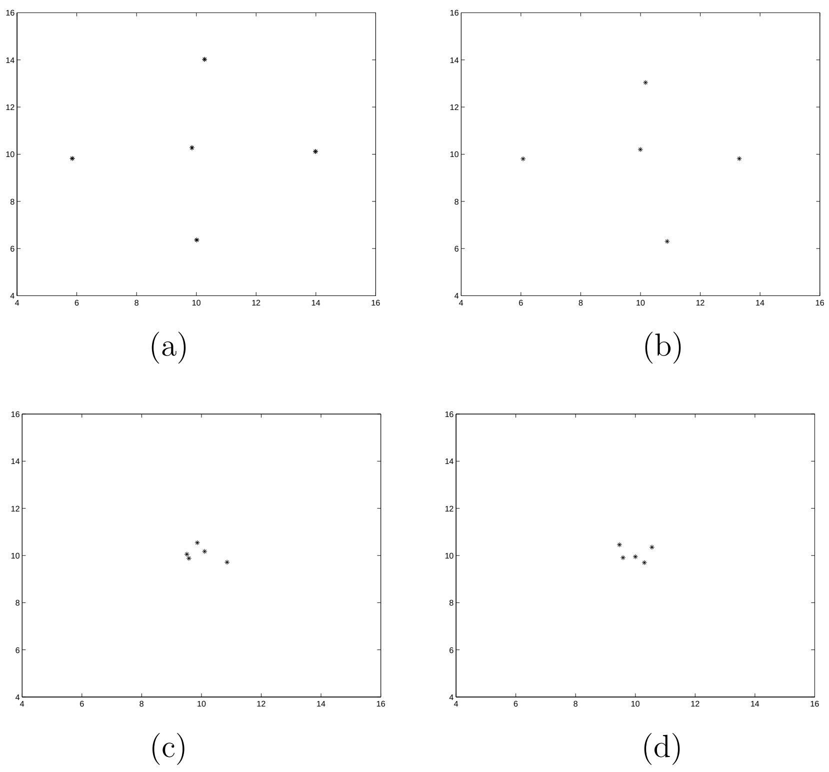


Figure 3.13: Change of the cluster centers obtained by SBKM on *AD_5_2* after (a) application of the K-means algorithm (b) 1 iteration (c) 10 iterations (d) 20 iterations.

3. Initialization of population: It is customary to initialize genetic algorithm with a population of random individuals. But sometimes previously known (good) solutions can be used to initialize a fraction of the population and this results in faster convergence of GA. In the proposed GAPS, after randomly generating the cluster centers, some iterations of K-means algorithm are executed to separate the cluster centers as much as possible.

4. Selection of crossover and mutation probabilities: These are two basic parameters of GA.

   Crossover probability (μ_c) determines how often the crossover operation can be performed. If $\mu_c = 0$, then offspring is the exact copy of parents. If $\mu_c > 0$ then the offspring is made from parts of parents' chromosome. If $\mu_c = 100\%$, then all offspring are made by crossover.

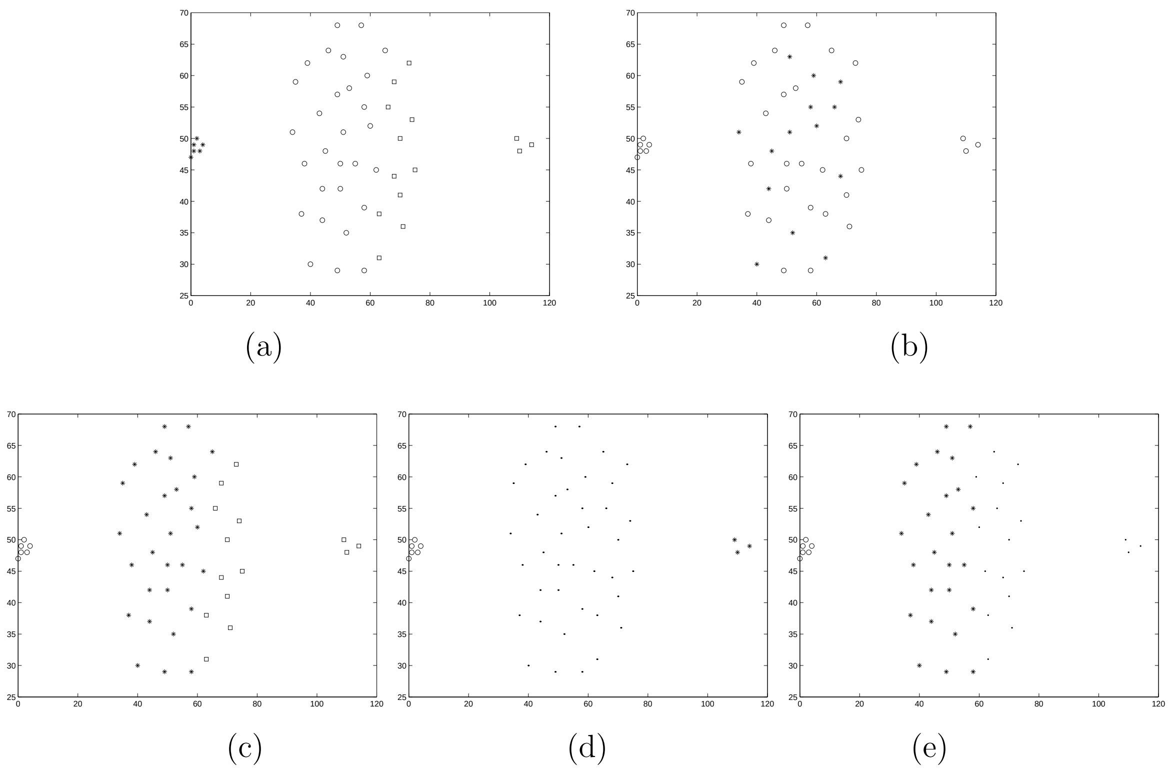


Figure 3.14: Clustered *Bensaid_3_2* for $K = 3$ after application of (a)K-means (b) SBKM (c) Mod-SBKM (d) GAPS/Average Linkage/ Expectation Maximization clustering techniques (e) GAK-means clustering technique

Crossover operation is executed in the hope that good building blocks of the parent chromosomes get combined in the offspring to result in potentially improved solutions. However, it is good to leave some parts of population to survive for the next generation.

Mutation probability (μ_m) determines how often parts of a chromosome are mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed (i.e., $\mu_m > 0$), a part of a chromosome is changed. If mutation probability is 100%, the whole chromosome is changed, if it is 0%, nothing is changed. Mutation is made to prevent GA from falling into a local extrema. But it should not occur very often; otherwise GA will change to random search.

Table 3.1: Best *Minkowski Score* obtained by the seven algorithms for all data sets. Here EM and AL denote 'Expectation Maximization' and 'Average Linkage' clustering techniques, respectively.

Data set	K-means	SBKM	Mod-SBKM	GAK-means	EM	AL	GAPS
Mixed_3_2	0.40	0.11	**0.08**	0.42	0.76	0.52	0.18
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**
Newthyroid	0.94	0.63	0.76	0.81	0.85	0.88	**0.59**
LungCancer	1.46	1.09	1.09	1.24	0.96	0.96	**0.89**

5. Dependencies among the parameter values: The parameters of GAs are interdependent on each other. With a very small population size, the required number of generations is too large to solve the problem with comparable number of function evaluations needed for moderate population size. This supports the observation that micro-GAs work nicely on simpler problems with a population size of 5 or more, but does not work as well with smaller population size. Whereas for a GA with a very large population size, the number of allowed generations is not enough to find the optimum.

   John J. Grefenstelle has used genetic algorithm to determine the optimal parameters of genetic algorithms [78]. According to his investigations, the best parameter values for GA are: population size = 30, number of generations = not specified, crossover type = typically two point, crossover rate of 0.9, mutation type = bit flip, mutation rate of 0.01. However, the selection of optimal parameters in GA is domain-dependent and relies on the specific application areas. Determining the appropriate parameter settings for GAs is still an open problem.

6. Parameters in GAPS: Inspired by the above discussed studies, and af-

Table 3.2: Estimated marginal means and pairwise comparisons of different algorithms on *Minkowski Score* obtained by ANOVA testing for *Iris* data (GAPS:GP, K-means:KM, Mod-SBKM:MSB, SBKM:SB, GAK-means:GAK, Expectation Maximization:EM, Average Linkage:AL)

Algo. Name (I)	Comparing Algo.(J)	Mean Difference (I-J)	Significance Value
GP	KM	$-9E-02 \pm 0.00163$	< 0.01
	MSB	$-6.0E-02 \pm 0.004$	< 0.01
	SB	-0.37 ± 0.023	< 0.01
	GAK	$-1.00E-02 \pm 0.0001$	0.54
	EM	$-1.00E-02 \pm 0.02$	0.59
	AL	-0.11 ± 0.015	< 0.01
KM	GP	$9E-02 \pm 0.00163$	< 0.01
	MSB	$3.73E-02 \pm 0.004$	< 0.01
	SB	-0.27864 ± 0.01	< 0.01
	GAK	$8.0E-02 \pm 0.021$	< 0.01
	AL	$-2.0E-02 \pm 0.015$	0.45
	EM	0.08 ± 0.025	< 0.01
MSB	GP	$6.0E-02 \pm 0.004$	< 0.01
	KM	$-3.73E-02 \pm 0.004$	< 0.01
	SB	-0.3132 ± 0.0012	< 0.01
	GAK	$5.0E-02 \pm 0.005$	< 0.01
	AL	$-5.0E-02 \pm 0.0041$	< 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	$1.00E-02 \pm 0.0001$	0.54
	KM	$-8.0E-02 \pm 0.021$	< 0.01
	SB	-0.36 ± 0.021	< 0.01
	MSB	$-5.0E-02 \pm 0.005$	< 0.01
	AL	0.28 ± 0.02	< 0.01
	EM	0.00 ± 0.001	1.00
EM	GP	$1.00E-02 \pm 0.02$	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	$2.0E-02 \pm 0.015$	0.45
	SB	-0.26 ± 0.018	< 0.01
	MSB	$5.0E-02 \pm 0.0041$	< 0.01
	GAK	-0.28 ± 0.02	< 0.01
	EM	0.10 ± 0.02	< 0.01

ter thorough experimentation, we set the following parameter values of GAPS: population size = 100, number of generations = 50 (executing GAPS further did not improve its performance). Initially the mutation probability, μ_m, and crossover probability, μ_c, were kept fixed. Good results were obtained with some combinations of $[\mu_c, \mu_m]$, e.g., [0.8,0.02], [0.8, 0.05] [0.8, 0.008], while the performance was not up to the mark for other combinations like [0.99, 0.02], [0.7, 0.02] etc. Moreover, the appropriate combination was also dependent on the data set being considered. Consequently, we decided to keep these two probabilities adaptive in this thesis as described in Section 3.6.4 and Section 3.6.5, respectively.

3.8.3 Implementation Results

The experimental results comparing the performance of GAPS, SBKM [187], Mod-SBKM [43], K-means algorithm, genetic algorithm based K-means clustering technique (GAK-means) [131], average linkage clustering technique (AL) [97] and Expectation Maximization clustering technique (EM) [97] are provided for the four artificial and four real-life data sets. For SBKM algorithm, θ is set equal to 0.18 as suggested in [187]. 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 Section 3.3.

The eight data sets used for comparison are divided into four groups. In order to compare the obtained clustering results quantitatively, the *Minkowski Scores* [98] 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 we wish to measure. 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. *Minkowski Score* (MS) is then defined as:

$$\text{MS(T,S)} = \sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}. \tag{3.22}$$

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 3.1 for all data sets.

1. *Mixed_3_2* and *Sym_3_2*: The clusters present in these two data sets are internally symmetrical but clusters themselves are not symmetrical with respect to any intermediate point.

   The final clustering results obtained by K-means, SBKM, Mod-SBKM, GAPS, GAK-means, AL, EM clustering techniques for *Mixed_3_2* are given in Figures 3.7(a), 3.7(b), 3.7(c), 3.7(d), 3.8(a), 3.8(b) and 3.8(c) respectively. All the above mentioned algorithms except K-means, GAK-means, AL and EM are able to find out the proper clustering for this data. As expected K-means and GAK-means show poor performance for this data since the clusters are not hyperspherical in nature. Average Linkage performs poorly as clusters are overlapping with each other. These results are also evident from the MS values reported in Table 3.1.

   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 Figures 3.9(a), 3.9(b), 3.9(c), 3.9(d), 3.10(a), 3.10(b) and 3.10(c), respectively. Both SBKM and Mod-SBKM (Figures 3.9(b) and 3.9(c)) are not able to detect the appropriate partitioning from this data set. In contrast, the proposed GAPS (Figure 3.9(d)) groups the points that are lying inside the ring, but actually belong to the elongated elliptic cluster, with those of the ring. In absence of any class information about the points, such a grouping is not really surprising. 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 3.1.

   It is to be noted that for the above data sets, K-means is unable to provide the correct clustering. However, the use of PS-distance in GAPS enables it to detect such clusters properly.

2. *AD_5_2*: The clusters present in this data set, used earlier in [16],

are internally symmetrical and clusters are also symmetrical with respect to some intermediate point. Figures 3.11(a), 3.11(b), 3.11(c), 3.11(d), 3.12(a), 3.12(b), 3.12(c) show the clustering results for K-means, SBKM, Mod-SBKM, 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 left cluster but after application of GAPS it is included in the central cluster (it is shown in Figure 3.11(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 $d_e(p, c1)$ but due to less 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 Figures 3.13(a)-(d) that 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 at the center of the whole data set thereby providing very poor performance. GAK-means and average linkage clustering techniques perform moderately for this data set (refer to Table 3.1).

3. *Bensaid_3_2*: This data set, used in [29], is taken in order to show that the proposed 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 3.14(a), 3.14(b), 3.14(c), 3.14(d) and 3.14(e) show the partitionings obtained by K-means, SBKM, Mod-SBKM and

GAPS/AL/EM and GAK-means, respectively. GAPS along with AL and EM clustering techniques are able to find out the proper partitioning, while the other four algorithms fail (refer to Table 3.1).

4. Real-life Data Sets: This category consists of four real life data sets: *Iris*, *Cancer*, *Newthyroid* 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 combined MS values of the seven algorithms when each is executed five times. ANOVA results are reported in detail for *Iris* (Table 3.2) for the purpose of illustration.

   (a) *Iris*: As seen from Table 3.1, the MS-scores of GAPS is the best for *Iris*, followed by GAK-means and EM. However, it can be seen from Table 3.2 that the differences in the means of the MS scores of GAPS with GAK-means and EM are not significant indicating their similar performances. The performance of SBKM algorithm is found to be the poorest.

   (b) *Cancer*: As can be seen from Table 3.1, 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) *Newthyroid* (or, Thyroid gland data): From Table 3.1, it is evident that GAPS performs the best (providing the lowest MS score), while K-means performs the worst. The improvement in performance obtained by GAPS as compared to each of the other six clustering techniques is statistically significant. SBKM is also found to provide improved performance over K-means and Mod-SBKM, and these improvements are also significant.

   (d) *LungCancer*: The MS scores, reported in Table 3.1, demonstrate the superior performance GAPS clustering technique. ANOVA statistical analysis is also done here. The analysis shows that the mean MS differences of all the algorithms are statistically significant.

3.8.4 Summary of Results

It can be seen from the above results that proposed GAPS is able to find out the proper clustering where SBKM and Mod-SBKM succeed while K-means fails as well as where 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 four 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 eight 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 newly proposed 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 [43][187], which resulted in some serious problems as discussed in Section 3.2 and displayed pictorially in Figure 3.2.

The 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 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 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 is 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.

3.9 MOPS: Multiobjective Clustering Using Point Symmetry Distance [160]

The newly proposed point symmetry based clustering technique, GAPS, optimizes only the total symmetrical compactness for clustering. However, as already mentioned in Section 3.1, 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 the different data characteristics. In order to achieve this, in this chapter the problem of clustering a data set is posed as one of multiobjective optimization (MOO) [60], where search is performed over a number of, often conflicting, objective functions. The newly developed simulated annealing based multiobjective optimization technique AMOSA [27], described in detail in Chapter 2, is used to determine the K cluster centers and the corresponding partitioning. The resulting clustering technique, ia 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 Section 3.6.2 of this chapter. 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 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

AMOSA based multiobjective clustering technique is defined as follows:

$$totalSym(K) = \sum_{i=1}^{K} E_i, \tag{3.23}$$

where K is the number of clusters. Here,

$$E_i = \sum_{j=1}^{n_i} d_{ps}(\overline{x}_j^i, \overline{c}_i), \tag{3.24}$$

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 Equation 3.8. Note that $totalSym(K)$ measures the within cluster total symmetrical distance. For clusters which have good symmetrical structure, 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:

$$\sigma(K) = \sum_{i=1}^{K} \sum_{k=1}^{n_i} d_e(\overline{c}_i, \overline{x}_k^i),$$

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 non-dominated solutions [60] 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. Consequently, in this chapter a method of selecting a single solution from the set of solutions, is now developed.

3.9.1 Selection of a Solution from the Archive

This method is a semi-supervised one, where we assume that the class labels of some (p%)of the points (denoted as *test patterns*) are known to us. The proposed 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 Equation 3.22). 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 the same. Here p is chosen equal to 10.

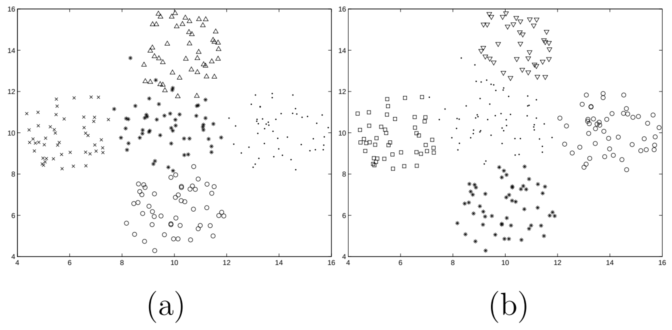


Figure 3.15: Clustered *AD_5_2* for $K = 5$ after application of (a) MOPS clustering technique (b) GAPS clustering technique

3.9.2 Experimental Results

The parameters of the proposed AMOSA with 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 non-dominated solutions on the final non-dominated front. The best solution is identified by the method proposed earlier. The performance of MOPS is compared with that of GAPS for the four artificial and four real-life data sets (described in Section 3.8.1 of this chapter). In order to compare the performance of the algorithms quantitatively, the *Minkowski Scores* (MS) [98] corresponding to their final partitionings are also computed. These are

Table 3.3: Best *Minkowski Scores* (MS) obtained by the proposed AMOSA with symmetry based multiobjective clustering technique (MOPS) and GAPS for all the data sets used here for experiment.

Data set	*Minkowski Score*	
	MOPS	GAPS
Mixed_3_2	0.16	0.18
Sym_3_2	0.12	0.12
AD_5_2	0.37	0.51
Bensaid_3_2	0	0
Iris	0.55	0.59
Cancer	0.31	0.36
Newthyroid	0.52	0.59
LungCancer	0.78	0.89

reported in Table 3.3.

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 was overestimating the central cluster. Consideration of the two objectives in MOPS reduces this problem. The clustering results are shown in Figures 3.15(a) and 3.15(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 demonstrates the effectiveness of using MOO for optimizing both Euclidean compactness and symmetrical compactness of a partitioning simultaneously.

3.10 Discussion and Conclusions

A new point symmetry based distance is proposed in this chapter that overcomes the limitations of some existing measures. It incorporates both the

Euclidean distance as well as a measure of symmetry with respect to a point in its computation. Kd-tree based nearest neighbor search is used to reduce the complexity of symmetry based distance computation. A genetic clustering technique (GAPS) is also proposed here that incorporates the new point symmetry distance while performing cluster assignments of the points and in the fitness computation. Experimental results on different data sets demonstrate the superiority of GAPS as compared to SBKM [187], its modified version Mod-SBKM [43], the K-means algorithm, genetic algorithm based K-means clustering technique (GAK-means), average linkage clustering technique (AL) and expectation maximization clustering technique (EM). GAPS is found to provide satisfactory performance for four artificial data sets, both where K-means fails but SBKM succeeds as well as SBKM fails but K-means succeeds. Results on *Bensaid_3_2* demonstrate that GAPS is able to detect symmetric clusters of any size where most of the algorithms fail. The superiority of GAPS is also established on four real-life data sets. These real-life data sets are of different characteristics, with the number of dimensions varying from 4 to 56. The performance of the different algorithms are compared using the statistical test, ANOVA. Results on the eight artificial and real-life data sets establish the fact that GAPS is well-suited to detect clusters of widely varying characteristics.

Thereafter in this chapter a multiobjective clustering technique based on the new point symmetry based distance, d_{ps}, has also been developed. Two cluster quality measures, one reflecting the total symmetry present in a partitioning calculated using the newly developed point symmetry based distance, and another reflecting the total compactness calculated using the well-known Euclidean distance, are optimized simultaneously. This enables the algorithm to detect clusters that are well characterized by Euclidean compactness as also those which are not compact in the conventional sense, but are symmetric about a point. The newly developed multiobjective simulated annealing based technique, AMOSA, has been used to optimize the two objective functions simultaneously. The superiority of the proposed multiobjective clustering technique (MOPS) over GAPS is shown for four artificial and four real-life data sets.

The definition of d_{ps} in Equation 3.7 involves a term, *knear* in its computation. It may be noted that the proper value of *knear* largely depends on the distribution of the data set. For instance, for very large clusters (with too many points), a small value of *knear* may not be proper as it is very likely that a few neighbors would have a distance close to zero. On the other hand, clusters with too few points are more likely to be scattered, and the distance of the two neighbors may be too large. A large value of *knear* may not be desirable in this case. Thus a proper choice of *knear* is an important issue that needs to be addressed in the future.

The choice of the crossover probability, μ_c, and mutation probability, μ_m, are important considerations in genetic algorithms [78]. Experiments with different values of μ_c and μ_m show that the performance of GAPS also depends on these values. Hence we kept μ_c and μ_m adaptive, as described in Section 3.6.4 and Section 3.6.5, respectively, so that these parameter values can be adjusted automatically based on the population status.

The major advantages of GAPS are as follows. In contrast to K-means, use of GA enables the algorithm to come out of local optima, making it less sensitive to the choice of the initial cluster centers. Moreover, use of Kd-tree makes the computation of the point symmetry distance significantly faster than both SBKM and Mod-SBKM as demonstrated in [24]. Again, the proposed GAPS is able to detect clusters that may be of widely varying sizes, where most of the other algorithms fail. Such situations may arise in several real-life domains, e.g., medical images, satellite images, fraud detection. Incorporation of the proposed d_{ps} enables GAPS to detect symmetric clusters, both convex and non-convex, even if the clusters are symmetrical with respect to some intermediate point. This is in contrast to SBKM and Mod-SBKM, which fail in such situations. In other words, GAPS can detect clusters of different shapes and sizes that is a superset of the types captured by most of the other methods considered. Moreover, the improved performance of MOPS demonstrates the effectiveness of using MOO for optimizing both Euclidean compactness and symmetrical compactness of a partitioning simultaneously.

Some important areas of further research in this regard involve the development of new cluster validity indices based on the newly developed d_{ps} as well as automatic clustering methods based on the proposed point symmetry distance. Chapters 4 and 5 of this thesis will deal with these.

Chapter 4

Validity Index Based on Symmetry

4.1 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 [68] 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 [68] work well for non-overlapping clusters of any shape, but fail if the clusters overlap. The Gaussian Mixture models [37] are considered to be the appropriate partitioning techniques for data sets whose type of the distributions (e.g., Gaussian) are known. Clustering methods like SBKM [187], Mod-SBKM [43], GAPS and MOPS exploit the symmetry property within the clusters. These methods are found to be superior to several other techniques when the clusters do offer a symmetric structure. Thus given a wide choice of methods, determining an appropriate clustering algorithm invokes a challenge.

The task of determining the number of clusters and also the validity of the clusters formed [132] are generally addressed by providing several definitions of validity indices. Several cluster validity indices have been proposed in the literature. These are Davies-Bouldin (DB) index [132], Dunn's index [64], Xie-Beni (XB) index [206], I-index [132], CS-index [44], 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. Such an attempt has been made in the present chapter.

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 properties, these indices are often found to fail. In this chapter we propose a symmetry based cluster validity index named *Sym*-index that 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 its shape, size or convexity as long as they possess the property of point symmetry.

In a part of this investigation, the newly defined d_{ps} is incorporated in the definitions of several cluster validity indices to develop the symmetry based versions of these indices. The performance of these symmetry based validity indices are 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.

4.2 *Sym*-index: The Proposed Symmetry Based Cluster Validity Index [162][168]

In this section a new cluster validity index is proposed that is based on d_{ps}. This is followed by an explanation of the interaction among the different components of the index so that it can indicate the proper partitioning of the data. A theoretical analysis of the index is also provided.

4.2.1 The Proposed Cluster Validity Measure

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 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 Equation 3.8 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 2: Total compactness of the partitioning in terms of symmetry is denoted by $\mathcal{E}_K$ and is given by:

$$\mathcal{E}_K = \sum_{i=1}^{K} E_i. \tag{4.1}$$

Definition 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.,

$$D_K = max_{i,j=1}^{K} \|\overline{c}_i - \overline{c}_j\|. \tag{4.2}$$

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. Given the above two criteria, at the same time for a proper partitioning the value of the number of clusters, K, should be minimized. Motivated by these observations, in the following we have provided a cluster validity index.

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. This is inspired by the I-index developed in [132]. The new cluster validity function *Sym* is defined as:

$$Sym(K) = \left(\frac{1}{K} \times \frac{1}{\mathcal{E}_K} \times D_K\right), \tag{4.3}$$

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.

Explanation

As formulated in Equation 4.3, *Sym*-index is a composition of three factors, $1/K$, $1/\mathcal{E}_K$ 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, $\mathcal{E}_K$ value is less. Note that as K increases, in general, the clusters tend to become more symmetric. Moreover, as $d_e(\overline{x}, \overline{c})$ in Equation 3.8 also decreases, $\mathcal{E}_K$ decreases, resulting in an increase in the value of the *Sym*-index. Since *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, 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, so they are expected to compete with and balance each other critically for determining the proper partitioning.

The use of D_K, as the measure of separation, requires further elaboration [168]. 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 largely with increase in the value of K. This might lead to the formation of maximum possible number of clusters equal to the number of elements in the data set. If D_K was the average inter cluster distance then 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 was 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 and then 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 Equation 4.2, 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 can not 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 [145].

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 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 $r_1, r_2, \ldots, r_d$ be the radii of a cluster along these directions, then $r = (r_1 + r_2 + \ldots + r_d)/d$ is considered to be the radius of the spherically approximated cluster. Let the exact reflected point of every point with respect to the cluster centre 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 centre is given by $\sum(\sum_{i=2}^{knear} \frac{d_i}{knear} \times d_e)$ since here $d_1 = 0$. Assuming that the $\sum_{i=2}^{knear} \frac{d_i}{knear}$ value 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 $\approx n\alpha\frac{r}{2}$.

If such a cluster of radius r having n number of elements be divisible 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 $\mathcal{E}_K$ denote the sum of 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 within cluster symmetrical distances for this configuration is $\mathcal{E}_K = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2n\alpha r = \frac{n\alpha r}{2}(K+3)$.

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 $\mathcal{E}_{K+1} = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + n\alpha\frac{r}{2} + n\alpha\frac{r}{2} = n\alpha\frac{r}{2}(K+1)$. Thus $\frac{\mathcal{E}_K}{\mathcal{E}_{K+1}} = \frac{K+3}{K+1}$ 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, $\mathcal{E}_{K+1} = \sum_{i=1}^{K-1} \frac{n\alpha r}{2} + 2(\frac{2d-1}{d} \times \frac{n\alpha r}{2})$.

Thus $\frac{\mathcal{E}_K}{\mathcal{E}_{K+1}} = \frac{K+3}{(K-1)+2(2d-1)/d}$ 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 at 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 like K^p ($p > 1$), $\log(K)$, $\exp(K)$, etc. Let us assume that $Sym_p(K) = \frac{D_K}{\mathcal{E}_K \times K^p}$, 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 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 sub parts where a division is indeed suggested.

Now if the form of *Sym*-index is $Sym_{exp}(K) = \frac{D_K}{\mathcal{E}_K \times \exp(K)}$, 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., $Sym_{log}(K) = \frac{D_K}{\mathcal{E}_K \times \log(K)}$, 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., $Sym(K) = \frac{D_K}{\mathcal{E}_K}$, then in the first case where the candidate cluster may have a clear tendency of 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 of 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. So it would break compact clusters into sub-parts where no division is needed. Thus this form of *Sym*-index is not suitable.

4.2.2 Mathematical Justification [169]

In this section, we mathematically justify the new validity index by establishing its relationship to the well-known validity measure proposed by Dunn [64] for hard partitions. This is inspired by a proof of optimality of the Xie-Beni index [206].

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 [64]. 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 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

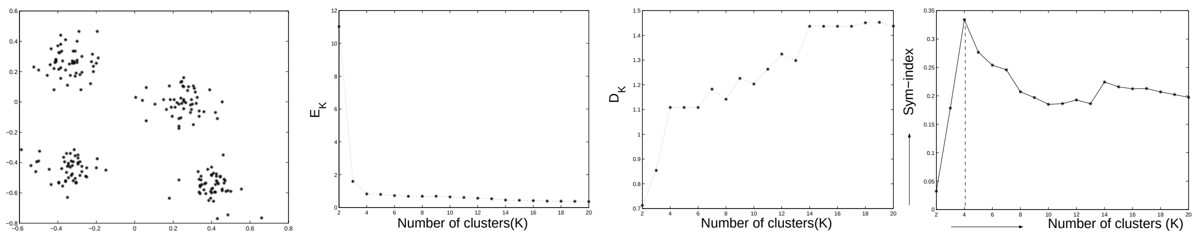
$$Sym \geq \frac{D_1}{n \times K \times 0.5 \times knear \times d_{NN}^{max}}$$

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 Equation 3.8 of Chapter 3. d_{NN}^{max} is the maximum nearest neighbor distance in the data set. That is

$$d_{NN}^{max} = \max_{i=1,\ldots N} d_{NN}(\overline{x}_i), \tag{4.4}$$

where $d_{NN}(\overline{x}_i)$ is the nearest neighbor distance of $\overline{x}_i$.



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

Figure 4.1: (a) Normal_2_4 (b) Variation of $\mathcal{E}_K$ 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

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 $\mathcal{E}_K$ of the optimal hard K-partition is defined in Equation 4.1. Thus,

$$\mathcal{E}_K = \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} d_{ps}(\overline{x}_j, \overline{c}_i) = \sum_{i}^{K} \sum_{\overline{x}_j \in u_i} \frac{\sum_{ii=1}^{knear} d_{ii}}{knear} d_e(\overline{x}_j, \overline{c}_i). \tag{4.5}$$

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

$$\frac{\sum_{ii=1}^{knear} d_{ii}}{knear} \leq \frac{d_{NN}^{max}}{2knear}\left(\sum_{ii=1}^{knear} (2 \times ii - 1)\right). \tag{4.6}$$

The right hand side of the inequality may be written as

$$\frac{d_{NN}^{max}}{2 \times knear} \times \frac{(knear \times (2 \times 1 + (knear - 1)2))}{2} = \frac{knear \times d_{NN}^{max}}{2}. \tag{4.7}$$

So, combining Equations 4.5, 4.6 and 4.7, we can write,

$$\mathcal{E}_K \leq \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} 0.5 \times knear \times d_{NN}^{\max} \times d_e(\overline{x}_j, \overline{c}_i) \leq 0.5 \times knear \times d_{NN}^{\max} \sum_{i=}^{K} \sum_{x_j \in u_i} d_e(\overline{x}_j, \overline{c}_i).$$

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

$$\begin{aligned}\mathcal{E}_K &\leq 0.5 \times knear \times d_{NN}^{\max} \sum_{i=1}^{K} \sum_{\overline{x}_j \in u_i} dia(u_i) \leq 0.5 \times knear \times d_{NN}^{\max} \sum_{i=1}^{K} n_i dia(u_i) \\ &\leq 0.5 \times knear \times d_{NN}^{\max} \times n \times \max_i dia(u_i).\end{aligned}$$

Here n_i denotes the total number of data points in cluster i. So,

$$\frac{1}{\mathcal{E}_K} \geq \frac{1}{0.5 \times knear \times d_{NN}^{\max} \times n \times \max_i dia(u_i)}.$$

We also have that $\min_{i,j,i \neq j} dis(u_i, u_j) \leq D_K$ 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 Equation 4.2. Thus,

$$Sym(K) = \frac{D_K}{K \times \mathcal{E}_K} \geq \frac{min_{i,j} dis(u_i, u_j)}{K \times 0.5 \times knear \times d_{NN}^{\max} \times n \times max_i dia(u_i)},$$

i.e.,

$$Sym(K) \geq \frac{\min_{1 \leq i \leq K} \{\min_{i+1 \leq j \leq K-1} \{\frac{dis(u_i, u_j)}{max_{1 \leq k \leq K} dia(u_k)}\}\}}{K \times 0.5 \times knear \times d_{NN}^{\max} \times n} \tag{4.8}$$

The separation index D_1 (Dunn [64]) is defined as

$$D_1 = \min_{1 \leq i \leq K} \{ \min_{i+1 \leq j \leq K-1} \{ \frac{dis(u_i, u_j)}{max_{1 \leq k \leq K} dia(u_k)} \} \} \tag{4.9}$$

So, combining Equations 4.8 and 4.9, we get

$$Sym(K) \geq \frac{D_1}{K \times 0.5 \times knear \times d_{NN}^{\max} \times n}. \tag{4.10}$$

Since the denominator of right hand side is constant for a given K, *Sym* increases as D_1 grows without bound. As mentioned earlier, it has been proved by Dunn [64] 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 Equation 4.10.

4.2.3 Interaction Between the Different Components of *Sym*-index

In order to show how the different components of the newly proposed *Sym*-index compete with each other to determine the proper model order from a data set, the variations of different components of *Sym*-index along with 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 Figure 4.1(a). It consists of 200 points in 2-d space and has four clusters.

• *Normal_2_10*: Shown in Figure 4.2(a). It consists of 500 points in 2-d space and has ten clusters.

The variations of values of different components of the *Sym*-index along with the number of clusters for the above two data sets are shown in Figures 4.1 and 4.2, 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 *Normal_2_10* data set are also shown in Figure 4.2. It is clearly seen from the given figures that *Sym*-index with K in the denominator performs the best compared to the other three.

PS-index [43], which is also based on a point symmetry based distance [43], is unable to identify the proper number of clusters from data sets like Normal_2_3 (shown in Figure 4.3(a)). The variation of the value of PS-index along with the number of clusters is shown in Figure 4.3(g). It is easy to see that PS-index gets its minimum value for $K = 2$. This is because d_{min} attains its maximum value for $K = 2$. Thus PS-index prefers the merging of 3 clusters into 2 clusters. The variation of minimum separation between two cluster centers (d_{min}) with respect to number of clusters is shown in Figure 4.3(f). The variation of *Sym*-index along with the number of clusters (shown in Figure 4.3(d)) reveals that it obtains its optimum value for $K = 3$.

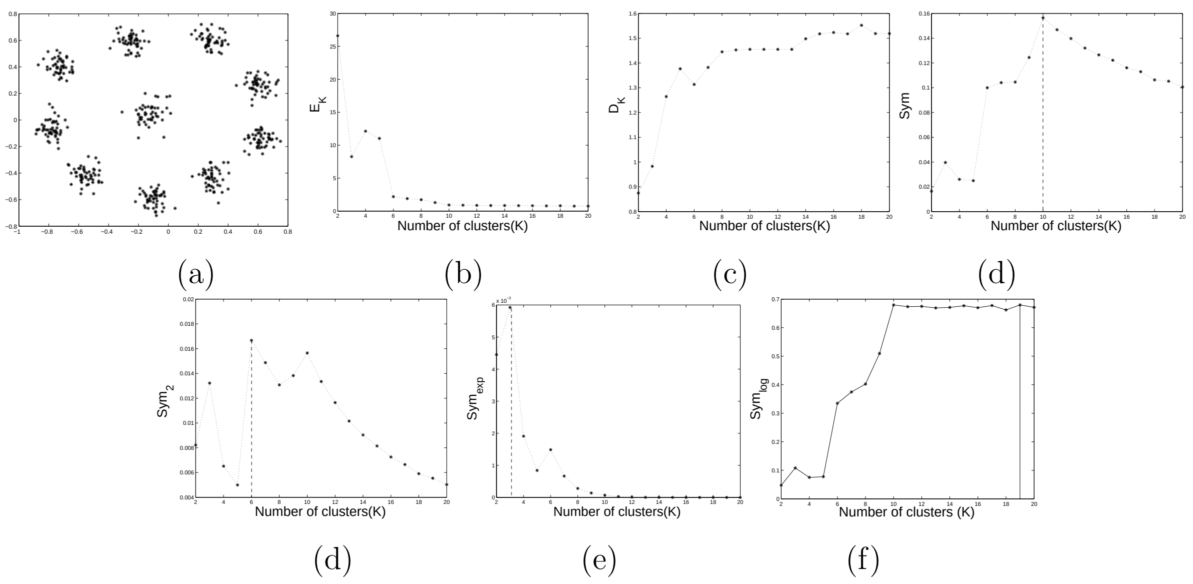


Figure 4.2: (a) Normal_2_10 (b) Variation of $\mathcal{E}_{\mathcal{K}}$ value with number of clusters (c) Variation of D_K value with number of clusters (d) Variation of *Sym*-index value with number of clusters (e) Variation of Sym_2-index where in the denominator K is replaced by K^2 with number of 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

4.3 Experimental Results

4.3.1 Data Sets

The data sets that are used for the experiments are as follows.

1. Group 1: This group consists of two 2-dimensional data sets: *Mixed_3_2* and *Sym_3_2*.
   (a) *Mixed_3_2*: This data set is described in Section 3.8.1 of Chapter 3.
   (b) *Sym_3_2*: This data set is described in Section 3.8.1 of Chapter 3.

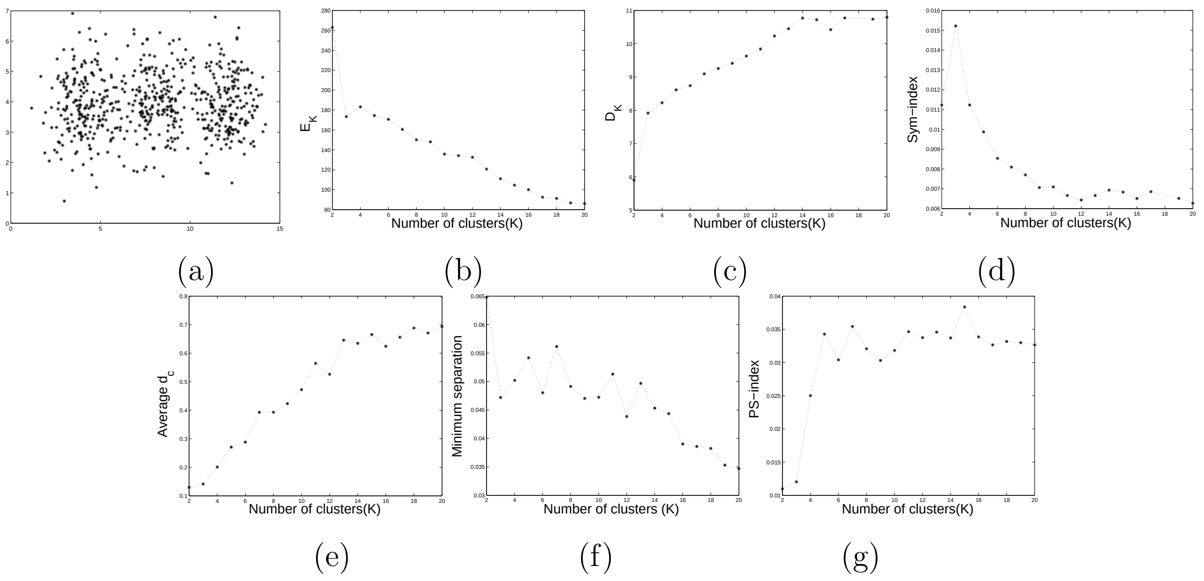


Figure 4.3: (a) Normal_2_3 (b) Variation of $\mathcal{E}_\mathcal{K}$ value with number of clusters (c) Variation of D_K value with number of clusters (d) Variation of *Sym*-index value with number of clusters (e) Variation of the numerator of PS-index with number of clusters (f) Variation of minimum separation (denominator of PS-index) between any two cluster centers with number of clusters (g) Variation of PS-index with number of clusters

2. Group 2: This group consists of two data sets used in [16]. The clusters present in these data sets are internally symmetrical about their centers and the clusters themselves are symmetrical with respect to some third cluster center. These are *AD_5_2* and *AD_4_3*.
   (a) *AD_5_2*: This data set is described in Section 3.8.1 of Chapter 3.
   (b) *AD_4_3*: This data set consists of 400 data points in 3 dimensional space distributed over 4 hyperspherical disjoint clusters where each cluster contains 100 data points. This data set is shown in Figure 4.4.

3. Group 3: This group consists of three real life data sets. These are *Iris*, *Cancer* and *Newthyroid* data sets described in Section 3.8.1 of Chapter

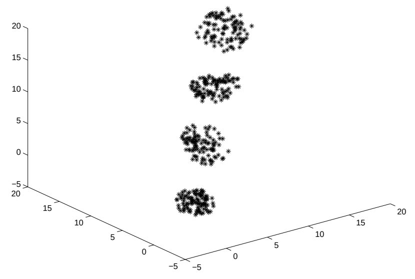


Figure 4.4: *AD_4_3*

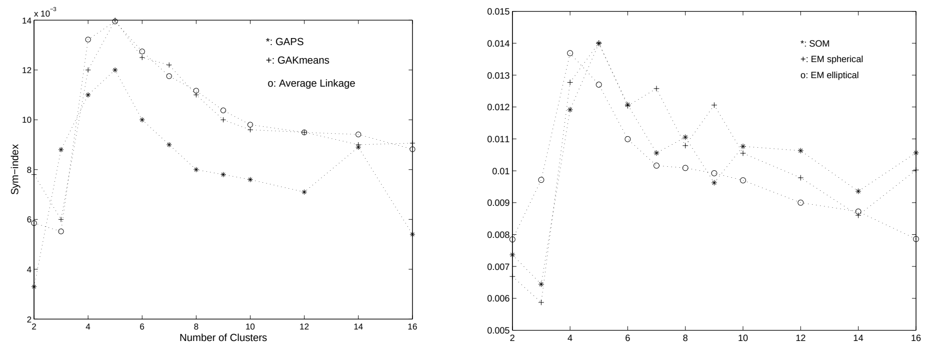
3.

4.3.2 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 seven 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) [24] described in Chapter 3 of this thesis.
- GAK-means algorithm [131].
- Average-linkage clustering algorithm [68] (source code was obtained from http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/5/659).
- Self Organizing Map (SOM) [111] (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) [37] (matlab source codes are obtained from http://www.mathworks.com/matlabcentral/fileexchange/).

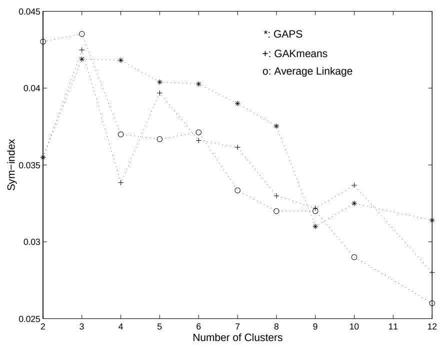
The number of clusters, K, is varied from 2 to $\sqrt{n}$, where n is the number of data points, and the variation of the *Sym*-index is noted. Its maximum value across different algorithms and different values of K, indicates the appropriate algorithm and the appropriate $K = K^*$. Comparison is also made with four existing cluster validity indices, namely, a point symmetry based PS-index [43], I-index [132], a recently proposed CS-index [44] and the well-known Xie-Beni (XB)-index [206], in terms of their ability in providing the appropriate number of clusters and the partitioning. The parameters of the genetic clustering algorithms (GAPS and GAK-means) are 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 [184]. For GAK-means, the crossover and mutation probabilities are chosen as 0.8 and 0.001 (as specified in [131]), 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.

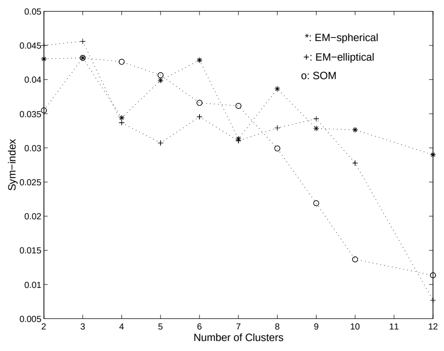


(a) (b)

Figure 4.5: Variation of *Sym*-index with number of clusters for *AD_5_2* using (a) GAPS, GAK-means, Average Linkage (b) SOM, EM-spherical, EM-elliptical

Figures 4.5-4.6 show the variations of *Sym*-index with the number of clusters for *AD_5_2* and *Iris* data sets, respectively, for the purpose of illustration.





(a) (b)

Figure 4.6: Variation of *Sym*-index with number of clusters for *Iris* using (a) GAPS, GAK-means, Average Linkage (b) SOM, EM-spherical, EM-elliptical

Similar figures were obtained using the other data sets and other indices as well. Based on the results in the figures, Table 4.1 shows 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 4.2 shows the overall (A^*, K^*) values obtained using the different indices for all the data sets.

4.3.3 Analysis of Results

This section analyzes the experimental results mentioned in the previous section.

1. *Mixed_3_2*: As the clusters present here are symmetric with elliptical shape, as expected both the symmetry based validity indices, i.e., *Sym*-index and PS-index, when used with GAPS and EM-elliptical can find

out the proper clustering for this data set (shown in Figure 4.7). This is also reflected in Table 4.2 where it is found that only the symmetry based validity indices, i.e., *Sym*-index and PS-index, are able to find the proper clustering, the proper cluster number and indicate the proper clustering algorithms. I-index and XB-index are not able to do so for any of the clustering algorithms (refer to Table 4.1). Although, CS-index is able to indicate the proper cluster number with EM-elliptical (refer to Table 4.1) but its optimum value corresponds to $K^* = 2$ with EM-spherical (refer to Table 4.2), wrongly indicating 2 as the proper cluster number.

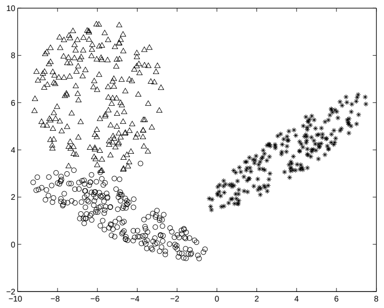


Figure 4.7: Clustered *Mixed_3_2* after the application of GAPS/EM-elliptical for the best value of *Sym*-index and PS-index providing $K^* = 3$

2. *Sym_3_2*: EM-elliptical along with both the symmetry distance based indices, *Sym*-index and PS-index, is able to detect the proper cluster number (see Table 4.2). The corresponding partitioning is shown in Figure 4.8(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 4.1, partitioning is shown in Figure 4.8(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 $\mathcal{E}_3 = 8.1059$ and $\frac{1}{\mathcal{E}_3} = 0.12337$. 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 $\mathcal{E}_3 = 8.9666$ and $\frac{1}{\mathcal{E}_3} = 0.11152$. 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 to 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 4.1) but its optimum value corresponds to $K^* = 4$ with EM-spherical (refer to Table 4.2).

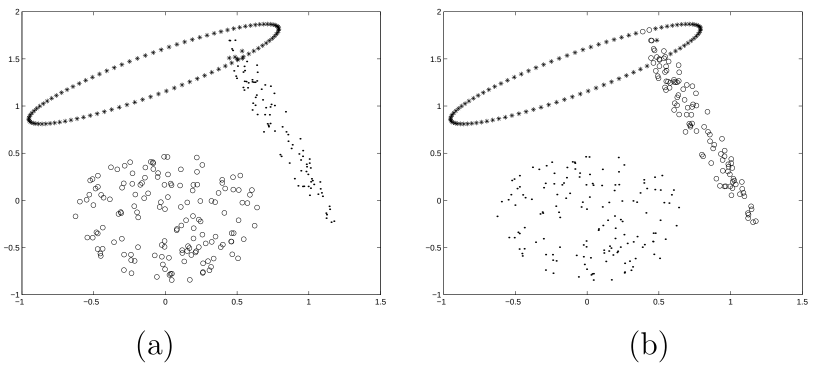


Figure 4.8: Clustered *Sym_3_2* after application of (a) GAPS for $K = 3$ (b) EM-elliptical for the best value of *Sym*-index and PS-index providing $K^* = 3$

3. *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 4.9(a), 4.9(b), 4.9(c), 4.10(a) and 4.10(b) respectively show the corresponding partitionings. It is evident from the figures that although the algorithms indicate 5 clusters as the optimum one, the resultant partitionings, spe-

cially 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 4.1). 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 4.2). Although it can be seen that *I*-index finds out the proper cluster number with GAK-means, EM-elliptical and SOM, (see Table 4.1), its optimum value over all the algorithms is obtained with $K = 9$ for EM-elliptical (see Table 4.2). CS-index is able to find out the proper number of clusters with GAK-means and SOM (refer to Table 4.1) but its optimum value over all the algorithms is obtained with $K = 4$ for EM-spherical (refer to Table 4.2).

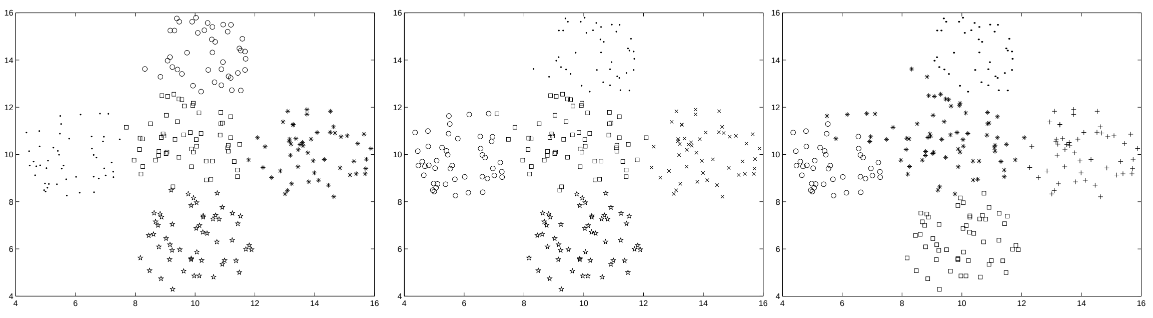
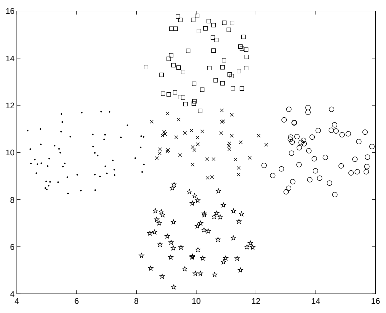
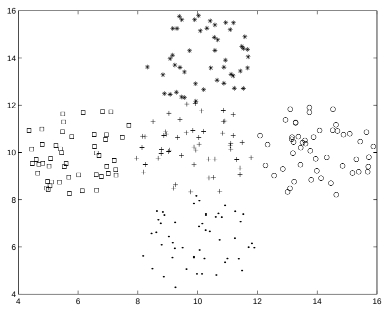


Figure 4.9: Clustered *AD_5_2* after application of (a) GAK-means for $K = 5$ (b) EM-spherical for the best value of *Sym*-index and PS-index providing $K^* = 5$ (c) GAPS for $K = 5$

4. *AD_4_3*: As the four clusters present here are symmetrical, spherical and nonoverlapping, all the algorithms and most of the indices except PS-index are able to find out the proper partitioning of this data (shown in Figure 4.11(a)). PS-index is known to fail in situations where clusters are symmetrical with respect to some intermediate center (symmetrical interclusters as defined in Chapter 3) (as for this data). This is because



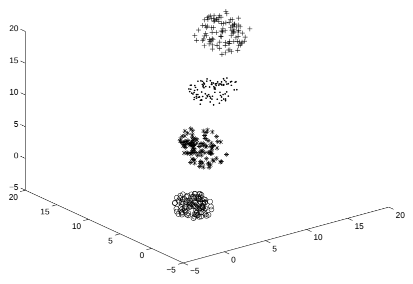
(a)



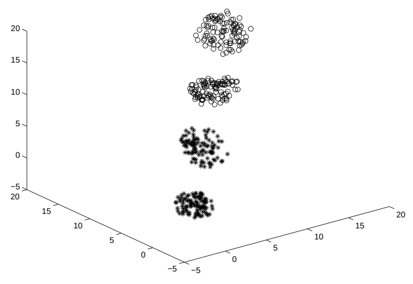
(b)

Figure 4.10: Clustered *AD_5_2* after application of (a) Average Linkage for $K = 5$ (b) SOM for the best value of XB-index providing $K^* = 5$

PS-index uses d_c of Equation 3.4 in Chapter 3. Thus it fails to detect symmetrical interclusters properly. The partitioning identified by PS-index for this data set is shown in Figure 4.11(b).



(a)



(b)

Figure 4.11: Clustered *AD_4_3* after application of GAK-means/GAPS/Average Linkage/EM-spherical/EM-elliptical/SOM (a) for the best value of *Sym*-index, *I*-index, CS-index and XB-index providing $K^* = 4$ (b) for the best value of PS-index providing $K^* = 2$

5. *Iris*: *Sym*-index and *I*-index are able to indicate the proper cluster number with all the algorithms (refer to Table 4.1). *Sym*-index attains its optimum value for the partitioning obtained by EM-elliptical for

Table 4.1: Optimal values of the five indices for *Mixed_3_2*, *Sym_3_2*, *AD_5_2*, *AD_4_3*, *Iris*, *Cancer* and *Newthyroid* 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.

Data	Method	*Sym*-index	PS-index	*I*-index	CS-index	XB-index
Set		Value (K^*)	Value (K^*)	Value (K^*)	Value (K^*)	Value (K^*)
Mixed_3_2	A_1	**0.012701(3)**	**0.010735(3)**	9750.092502(6)	0.901180(2)	0.202252(7)
	A_2	0.0100824(3)	0.035871(6)	10946.432227(8)	0.899795(2)	0.124709(2)
	A_3	0.008552(9)	0.014748(2)	**12656.378613(7)**	0.848677(2)	**0.123878(2)**
	A_4	0.009873(6)	0.012660(5)	10047.393375(6)	**0.798227(2)**	0.131021(2)
	A_5	**0.012701(3)**	**0.010735(3)**	9533.197699(9)	0.980731(3)	0.151893(2)
	A_6	0.01016(3)	0.012525(2)	10979.0445(10)	0.901180(2)	0.125006(2)
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)**
AD_4_3	$A_1 - A_6$	**0.013723(4)**	**0.012142(2)**	**636054.282343(4)**	**0.498988(4)**	**0.052002(4)**
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)
New	A_1	**0.001393(3)**	0.064669(8)	2521890.873937(7)	0.975621(8)	0.168048(2)
thyroid	A_2	0.001320(11)	0.108723(4)	1567291.137185(5)	1.395576(5)	0.194599(4)
	A_3	0.001190(3)	**0.025630(3)**	**4601698.925019(3)**	**0.631758(3)**	**0.100727(3)**
	A_4	0.001320(10)	0.111036(10)	2104572.222224(9)	0.956283(10)	0.284915(3)
	A_5	0.001057(5)	0.116310(5)	1233206.753436(9)	1.621346(5)	0.295940(2)
	A_6	0.001280(5)	0.114840(3)	1567291.137185(5)	1.395576(5)	0.245661(3)

Table 4.2: 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
Mixed_3_2	3	A_1, A_5	$(3, \{A_1, A_5\})$	$(3, \{A_1, A_5\})$	$(7, A_3)$	$(2, A_4)$	$(2, A_3)$
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	$(5, A_4)$	$(5, A_4)$	$(9, A_4)$	$(4, A_4)$	$(5, A_6)$
AD_4_3	4	$A_1 - A_6$	$(4, \{A_1 - A_6\})$	$(2, \{A_1 - A_6\})$	$(4, \{A_1 - A_6\})$	$(4, \{A_1 - A_6\}$	$(4, \{A_1 - A_6\})$
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	$(2, \{A_1, A_6\})$	$(3, A_3)$	$(3, A_2)$	$(2, A_4)$	$(2, \{A_1, A_2\})$
Newthyroid	3	A_1, A_6	$(3, A_1)$	$(3, A_3)$	$(3, A_3)$	$(3, A_3)$	$(3, A_3)$

$K = 3$ (refer to Table 4.2). I-index obtains its optimum value with GAPS for $K = 3$ (refer to Table 4.2). PS-index, CS-index and XB-index are unable to indicate three clusters with any of the algorithms (refer to Table 4.1 and Table 4.2). 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* [98] (defined in Equation 3.22 of Chapter 3) 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 are executed ten times, the *Minkowski Scores* are computed and ANOVA [5] statistical analysis is performed. The results for *Iris* data set are reported in Table 4.3. From Table 4.3 it can be seen that EM-elliptical finds the best clustering among the six algorithms. It also reveals from this table that although the mean MS value of GAPS is slightly better than those of GAK-means, EM-spherical and SOM but the differences of the values over ten runs are not statistically significant, i.e., GAPS, GAK-means, EM-spherical and SOM clustering algorithms perform similarly for this data set. From Table 4.2 we find that only the *Sym*-index indicates that EM-elliptical is the most appropriate algorithm, and that three clusters

Table 4.3: 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	$-1.0E-02\pm0.004$	0.54
		A_3	$-1.1E-01\pm0.004$	< 0.01
		A_4	$-1.0E-02\pm0.004$	0.59
		A_5	0.23 ± 0.004	< 0.01
		A_6	$-1.0E-02\pm0.004$	0.59
A_2	0.60 ± 0.003	A_1	$1.0E-02\pm0.004$	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	$1.1E-01\pm0.004$	< 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	$1.0E-02\pm0.004$	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	$1.0E-02\pm0.004$	0.59
		A_2	0.00 ± 0.004	1
		A_3	$-0.10E-02\pm0.004$	< 0.01
		A_4	0.00 ± 0.004	1
		A_5	0.24 ± 0.004	< 0.01

are present in the data. Therefore, combining the results of Table 4.2 and Table 4.3 reveals that the partitioning indicated by *Sym*-index is the best.

6. *Cancer*: For this data set, like the previous one, *Minkowski Score* (MS) [98] of the partitionings obtained after application of all six algorithms with $K = 2$ are calculated. The MS scores are 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 was observed that GAPS, GAK-means and SOM algorithms perform

equally for this data (the difference between the mean MSs 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 4.2, *Sym*-index indicates GAPS and SOM as the most appropriate choices. XB-index is also successful in correctly identifying GAPS and GAK-means algorithms with $K^* = 2$ as the most appropriate clustering algorithms. Even though I-index is able to detect the proper cluster number with GAPS, EM-elliptical and SOM (see from Table 4.1) but it attains its optimum value for GAK-means with $K = 3$ (refer to Table 4.2). CS-index attains its optimum value corresponding to the partitioning obtained by EM-spherical for $K = 2$ (see Table 4.2). PS-index is also able to find the proper cluster number with GAPS, EM-elliptical and SOM (refer to Table 4.1) but the optimum value of it indicates Average Linkage with $K = 3$ as the proper choice (refer to Table 4.2).

7. *Newthyroid*: *Sym*-index identifies GAPS as the appropriate clustering algorithm with the correct cluster number (refer to Table 4.2). Optimum values of PS-index, I-index, CS-index and XB-index indicate Average Linkage as the appropriate clustering algorithm for partitioning this data with the correct number of clusters (see Tables 4.1 and 4.2). The MS values attained by the six clustering techniques for this data set are 0.59, 0.81, 0.88, 0.85, 0.84 and 0.63, respectively for actual number of clusters present in the data set. It was observed from ANOVA analysis that GAPS performs the best (providing the lowest MS score), while average linkage performs the worst. The improvement in performance obtained by GAPS as compared to the other techniques is significant. This again highlights the utility of *Sym*-index which correctly identified GAPS as the proper algorithmic choice. Performance of EM-spherical and EM-elliptical are almost similar (the difference in their MS values over ten runs is not significant).

Interestingly, it was observed that for all the data sets, *Sym*-index was able to detect the proper number of clusters as well as atleast 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 *Mixed_3_2* which has symmetrical clusters, *Sym*-index (as well as PS-index) correctly identifies 3 clusters and GAPS/EM-elliptical as the appropriate clustering algorithms. However, the other three indices are neither able to identify the correct number of clusters nor the appropriate algorithm. Again for *AD_4_3*, PS-index fails in identifying the appropriate number of clusters, while the *Sym*-index (along with *I*-index, XB-index and CS-index) succeeds. Overall, *Sym*-index is able to detect the proper number of clusters and the appropriate clustering algorithms for all the data sets. PS-index is able to detect the proper number of clusters in four out of the seven cases and is able to indicate the appropriate clustering algorithms in only three cases. *I*-index is able to indicate the proper number of clusters in three cases among which in only one case it is also able to detect the proper clustering algorithm. CS-index and XB-index are able to detect the proper number of clusters in three and four data sets, respectively; among these data sets they are able to identify the appropriate clustering algorithms for only one and three data sets, respectively.

4.4 Incorporating d_{ps} in Some Existing Cluster Validity Indices [172]

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 [24] 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 proposed point symmetry based distance, rather than the Euclidean distance, to develop symmetry-based versions of Davies-Bouldin index (DB-index) [56], Dunn's index [64], Generalized Dunn's index [32], PS-index [43], Xie-Beni index (XB index) [206], FS-index [75], K-index [118] and SV-index [103]. The 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 $K^* = \text{argopt}_{i=K_{min} \cdots K_{max}}[V_i]$. Therefore, according to 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 $< U^*_{K^*}, K^* >$ is presented as the solution to the clustering problem.

The effectiveness of the newly proposed point symmetry based cluster validity indices namely, *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 *Sym* index performs the best compared to 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 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.

4.5 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.

4.5.1 Symmetry Based Davies-Bouldin index (***Sym-DB*** index)

This index is developed along the lines of the popular Davies-Bouldin (DB) index [56]. 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 Equation 3.8 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

$$\textit{Sym-DB} = \frac{\sum_{i=1}^{K} R_i}{K}.$$

where $R_i = \max_{j, j \neq i}\{\frac{S_i + S_j}{d_{ij}}\}$. The objective is to minimize the *Sym-DB* index for achieving the proper clustering.

4.5.2 Symmetry Based Dunn's Index (***Sym-Dunn*** index)

This index is developed along the lines of the popular Dunn's index [64]. Let S and T be two nonempty subsets of R^N. Then the radius $\triangle$ of S is defined as

$$\triangle(S) = \max_{x \in S}\{d^*_{ps}(\overline{x}, \overline{z})\},$$

where $\overline{z}$ represents the center of set S and $d^*_{ps}(\overline{x}, \overline{z})$ is computed using Equation 3.8. 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

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

Here, $d_e(\overline{x}, \overline{y})$ indicates the Euclidean distance between points $\overline{x}$ and $\overline{y}$. For any partition, *Sym-Dunn* index is defined as follows

$$\textit{Sym-Dunn} = \min_{1 \le i \le K} \min_{1 \le j \le K, j \ne i}\{\frac{\delta(C_i, C_j)}{\max_{1 \le k \le K} \triangle(C_k)}\}.$$

Larger values of *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.

4.5.3 Symmetry Based Generalized Dunn's Index (***Sym-GDunn*** index)

This index is developed along the lines of the Generalized Dunn's index [32]. The generalized Dunn's index was developed after demonstrating the sensitivity of the original Dunn's index [64], 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

$$\textit{Sym-GDunn} = \min_{1 \le s \le K}\{\min_{1 \le t \le K, t \ne s}\{\frac{\delta(C_s, C_t)}{\max_{1 \le k \le K} \triangle(C_k)}\}\}.$$

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 Equation 3.8 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.

4.5.4 Newly Proposed Symmetry Distance Based PS-index (***Sym-PS*** index)

This index is developed along the lines of PS-index [43]. The cluster validity index, *Sym-PS* index, is defined as

$$\begin{aligned} \textit{Sym-PS}(K) &= \frac{1}{K} \sum_{i=1}^{K} \frac{1}{n_i} \sum_{\overline{x} \in S_i} \frac{d^*_{ps}(\overline{x}, \overline{z}_i)}{min_{m,n=1,\ldots,K,\ m \neq n} d_e(\overline{z}_m, \overline{z}_n)} && (4.11) \\ &= \frac{1}{K} \sum_{i=1}^{K} \frac{1}{n_i} \sum_{\overline{x} \in S_i} \frac{d^*_{ps}(\overline{x}, \overline{z}_i)}{d_{min}} && (4.12) \end{aligned}$$

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 Equation 3.8 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-PS*(K^*) indicates a valid optimal partition with the optimal cluster number K^*.

4.5.5 Symmetry Based Xie-Beni index (*Sym-XB* index)

This index is developed along the lines of the popular XB-index [206]. It is defined as follows:

$$\textit{Sym-XB} = \frac{\sum_{i=1}^{K}(\sum_{\overline{x}\in C_i} d_{ps}^{*2}(\overline{x}, \overline{z}_i))}{n(\min_{i,k=1,\dots K, i\neq k} d_e^2(\overline{z}_i, \overline{z}_k))}.$$

$d_{ps}^*(\overline{x}, \overline{z_i})$ is computed by Equation 3.8. 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 *Sym-XB* index over $K = 2, 3, \dots K_{max}$.

4.5.6 Symmetry Based FS index (*Sym-FS* index)

This index is developed along the lines of the FS-index proposed in [75]. It is defined as follows:

$$\textit{Sym-FS} = \sum_{i=1}^{K}\sum_{\overline{x}\in C_i} d_{ps}^{*2}(\overline{x}, \overline{z}_i) - \sum_{i=1}^{K}\sum_{\overline{x}\in C_i} d_e^2(\overline{z}_i, \overline{z}).$$

where $\overline{z}$ is the center of the entire data set. $d_{ps}^*(\overline{x}, \overline{z_i})$ is computed by Equation 3.8. 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 *Sym-FS* index value with respect to $K = 2, 3, \dots K_{max}$.

4.5.7 Symmetry Based K index (*Sym-K* index)

Kwon extended the index given by Xie and Beni [206] to eliminate its tendency to monotonically decrease when the number of clusters approaches to the number of data points [118]. To achieve this, a punishing function was introduced to the numerator of the Xie and Beni's original validity index. The resulting index is named as K-index. Here we have developed a new validity index along the lines of K-index but using point symmetry based distance. This is named as *Sym-K* index. This is 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 Equation 3.8. 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.

4.5.8 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 [103]. Here, the newly developed *Sym-SV* index is defined along the lines of SV index proposed by Kim et al.[103].

$$\begin{aligned} Sym\text{-}SV &= v_{under}(Z : X) + v_{over}(Z) \\ &= \frac{1}{K} \sum_{i=1}^{K} \sum_{\overline{x} \in C_i} \frac{d_{ps}^{*}(\overline{x}, \overline{z}_i)}{n_i} + \frac{K}{\min_{i \neq j} d_e(\overline{z}_i, \overline{z}_j)}. \end{aligned}$$

A minimum value of *Sym-SV* index indicates the optimal number of clusters.

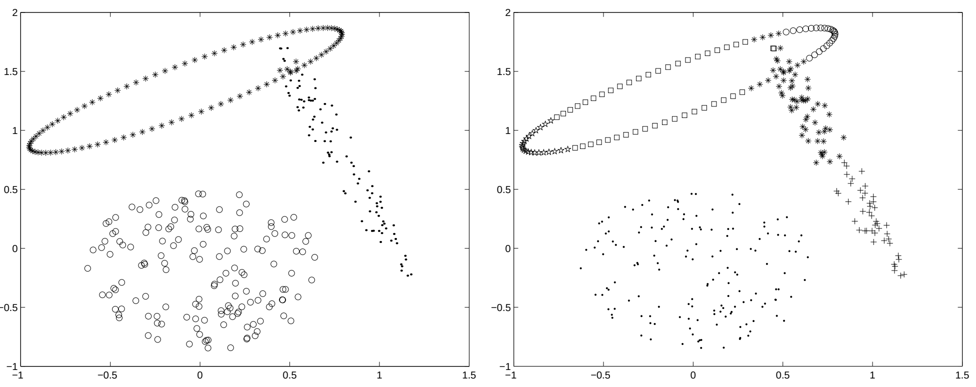
4.6 Experimental Results

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 are 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 Section 3.8.1 of Chapter 3).

The three real-life data sets are *Iris*, *Cancer*, *LungCancer* (described in Section 3.8.1 of Chapter 3).

Table 4.4: Optimal number of clusters identified by the newly proposed symmetry version and the original version of eight cluster validity indices and *Sym*-index for five data sets, segmented using 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 Section 4.6.1) of two different versions of 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)


(a) (b)

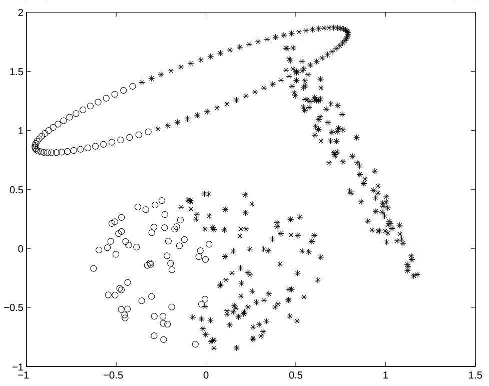

(c)

Figure 4.12: Clustered *Sym_3_2* after application of GAPS for (a) $K = 3$ (b) $K = 6$ (c) $K = 2$

4.6.1 Discussion of Results

The parameters of GAPS-clustering are set as detailed in Section 3.8.3 of Chapter 3. 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 a 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 $K^* = \arg\mathrm{opt}_{i=2,\ldots,\sqrt{n}}[V_i]$.

Table 4.4 shows the optimum number of clusters identified by the nine newly proposed point symmetry distance based cluster validity indices, namely,

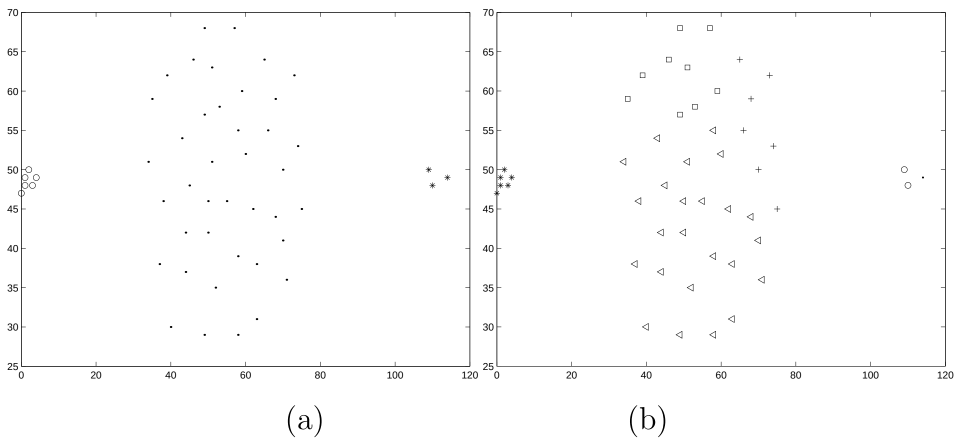


Figure 4.13: Clustered *Bensaid_3_2* after application of GAPS for (a) $K = 3$ (b) $K = 6$

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 experiment. Figures 4.12(a) and 4.13(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 Figure 4.12(a)). Optimum values of *Sym-DB* 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 Figures 4.12(b) and 4.12(c), respectively. For *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 Figure 4.13(a)). *Sym-SV* wrongly indicates $K^* = 6$. The corresponding partitioning is shown in Figure 4.13(b).

For the higher-dimensional three real-life data sets, *Iris*, *Cancer* and *Lung-Cancer*, the *Minkowski Scores* (defined in Equation 3.22 of Chapter 3) are calculated after application of GAPS-clustering algorithm. For *Iris* data set, MS value corresponding to the partitioning obtained by GAPS-clustering for $K = 3$ is 0.59 ± 0.00. As can be seen from Table 4.4, only *Sym* index is able to detect the proper number of partitions for this data set. Optimum values of *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*. *Sym-Dunn* and *Sym-FS* indices perform poorly for this data set. For *Cancer* dataset, MS value corresponding to the partitioning obtained by GAPS-clustering for $K = 2$ is 0.36 ± 0.00. *Sym-DB*, *Sym-GDunn*, *Sym-PS*, *Sym*, *Sym-XB*, *Sym-K* and *Sym-SV* indices are all able to indicate this partitioning. But again for this data set, the performance of both *Sym-Dunn* and *Sym-FS* indices are poor. For *LungCancer* data set, *Sym-DB*, *Sym-PS*, *Sym*, *Sym-XB* and *Sym-K* indices are able to detect the proper number of clusters along with GAPS-clustering (see Table 4.4). 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, *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 proposed *Sym* index performs the best than the other eight indices for detecting the proper number of clusters and the proper partitioning from data sets having symmetrical clusters.

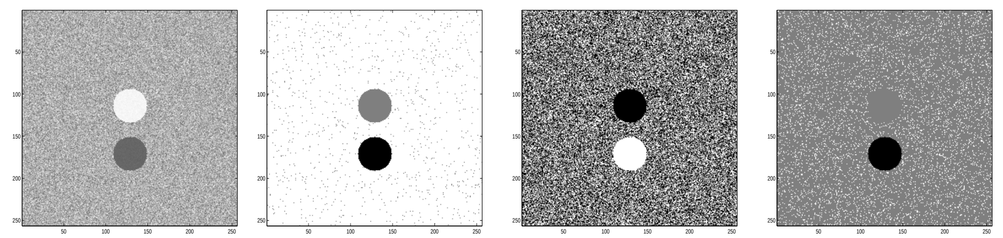
Table 4.4 also provides the performance results of the original versions of the validity indices. The success rates of the two versions of eight cluster validity indices (original version and the symmetry version) in detecting the proper number of partitions and the proper partitioning are reported. Here $SuccessRate(i) = \frac{A}{\text{total number of data sets}}$, where A=Number of data sets for which index i succeeds in determining the appropriate number of clusters. From the results provided in Table 4.4, it is easy to conclude that incorporation of point symmetry based distance in the definitions of existing cluster validity indices make 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 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 point symmetry distance impart the property of characterizing these non-compact, symmetric clusters to them.

4.7 Application to Remote Sensing Imagery [169]

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 [133], 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 are used here for automatically determining the appropriate number of clusters from different image data sets. 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 newly proposed cluster validity index, *Sym*-index, in conjunction with the GAPS-clustering [24] for automatically detecting different types of regions is demonstrated on one simulated and two satellite images. Segmentation results are compared with those obtained by two other recently proposed cluster validity indices, namely, PS-index [43] and I-index [132], and another well-known XB index [206]. For each image K is varied from 2 to 16.

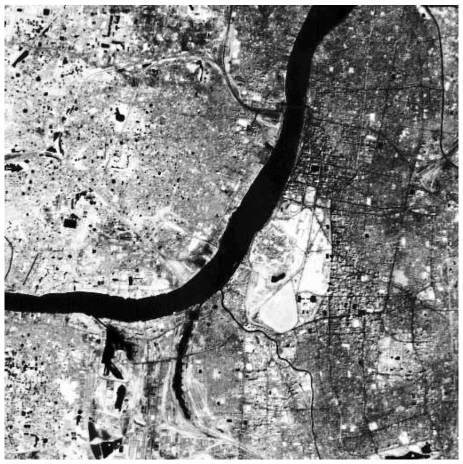


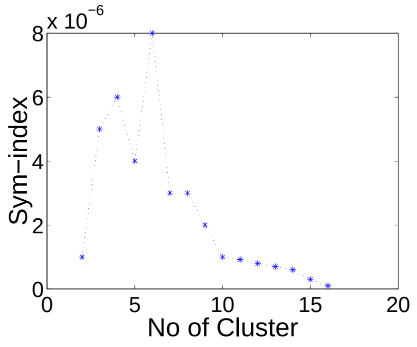
(a) (b) (c) (d)

Figure 4.14: (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 $K = 3$

4.7.1 Simulated Circle Image (SCI)

In order to show the effectiveness of the proposed *Sym*-index in identifying small clusters from much larger ones where there is a significant overlap of the small clusters with the bigger one, we first generate an artificial image of size 256×256 shown in Figure 4.14(a). There are two small circles of radius 20 each, 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 4.14(b) shows the segmented image using GAPS-clustering with *Sym*-index, when 3 clusters were automatically found. We have calculated *Minkowski Score* (MS) [98] (defined in Equation 3.22 of Chapter 3) 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, 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 out the proper clustering from this data set (shown in Figure 4.14(c)). MS value in this case is 0.806444. EM algorithm is also not able to find out the proper clustering from this overlapping dataset (Figure 4.14(d)). MS value in this case is 0.82.





(a) (b)

Figure 4.15: (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

4.7.2 SPOT Image of Kolkata

The French satellites SPOT (Systems Probataire d'Observation de la Terre) [157], 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:
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 infra red band of wavelength 0.79 - 0.89 μm.
Thus, here feature vector of each image pixel composed of three intensity values at different bands. The distribution of the pixels in the feature space of this image is shown in Figure 4.16. It can be easily seen from the Figure 4.16 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

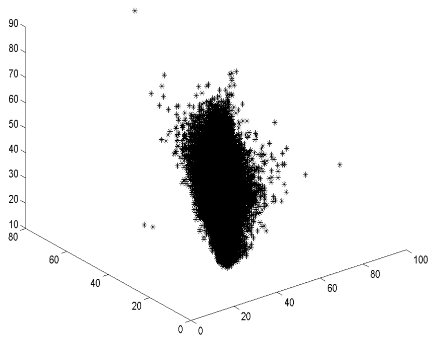


Figure 4.16: Data distribution of SPOT image of Kolkata in the Feature Space

can be identified, from a knowledge about the area, more easily in the near infra-red band of the input image (Fig. 4.15(a)). These are the following: The prominent black 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, the *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 is 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) are calculated. *Sym*-index obtained its optimal value for $K^* = 6$. The cor-

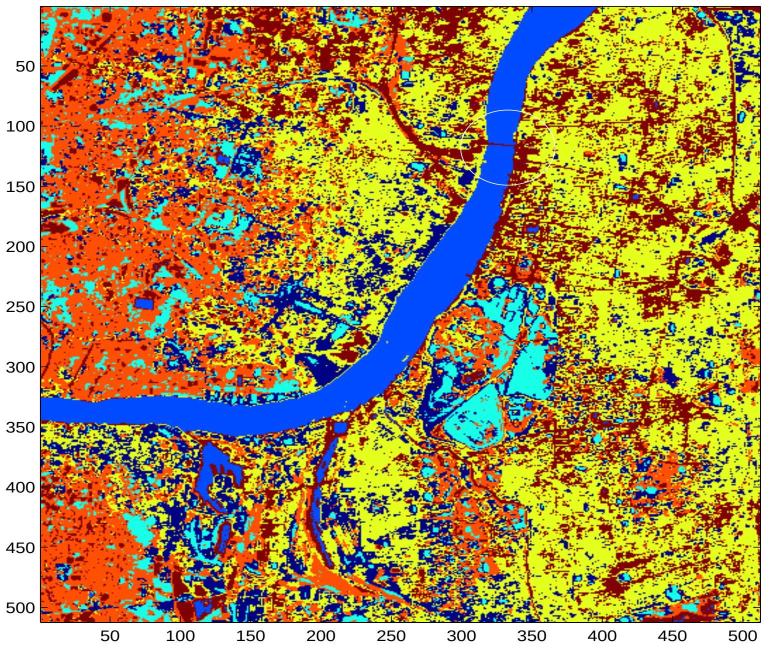


Figure 4.17: Segmented Kolkata image obtained by GAPS-clustering with *Sym*-index (provides $K^* = 6$)

responding segmented image is shown in Figure 4.17 (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 Figures 4.18, 4.19 and 4.20, respectively. The segmentations corresponding to the optimum values of *Sym*-index and I-index are able to separate almost all the regions equally well (Figures 4.17 and 4.18). Even the thin outline of the bridge on the river has been automatically identified (encircled in Figure 4.17). This again illustrates the superiority of symmetry based distance for detecting a small 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) [98] (defined in Equation 3.22 of Chapter 3) is calculated after application of GAPS-clustering for the

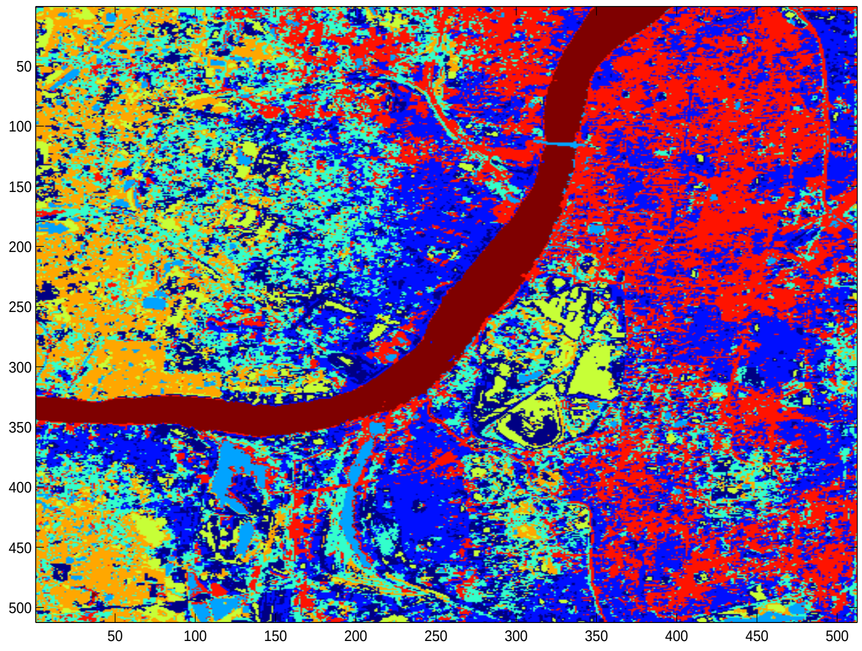


Figure 4.18: Segmented Kolkata image obtained by GAPS-clustering with I-index (provides $K^* = 8$)

optimal cluster number indicated by each of the indices. The MS scores corresponding to *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, Davies Bouldin (DB) index [56] has been calculated corresponding to the optimal values of *Sym*-index, I-index, PS-index and XB-index. The values are listed in Table 4.5. As smaller values of DB are preferable, it again signifies that segmentation corresponding to *Sym*-index is the best. Figure 4.15(b) shows the variations of the values of *Sym*-index with the number of clusters for this data set.

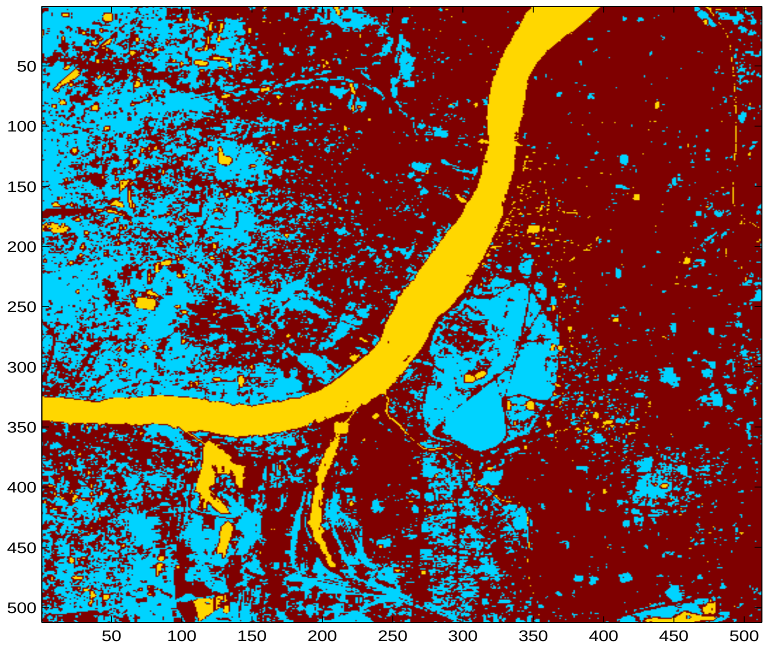


Figure 4.19: Segmented Kolkata image obtained by GAPS-clustering with PS-index (provides $K^* = 3$)

4.7.3 IRS Image of Mumbai

The IRS image of Mumbai was obtained using the LISS-II sensor. It is available in four bands, viz., blue, green, red and near infra-red. Fig. 4.21(a) shows the *IRS* image of a part of the city of Mumbai in the near infra red band. As can be seen, the elongated city area is surrounded on three sides by the Arabian sea. Towards the bottom right of the image, there are several islands, including the well known *Elephanta island.* The dockyard is situated on the south eastern part of Mumbai, which can be seen as a set of three finger like structure. This image has been classified into seven clusters [133].

Here also number of clusters is varied from 2 to 16. GAPS-clustering with *Sym*-index and PS-index get their optimal values for $K^* = 6$ where as GAPS-

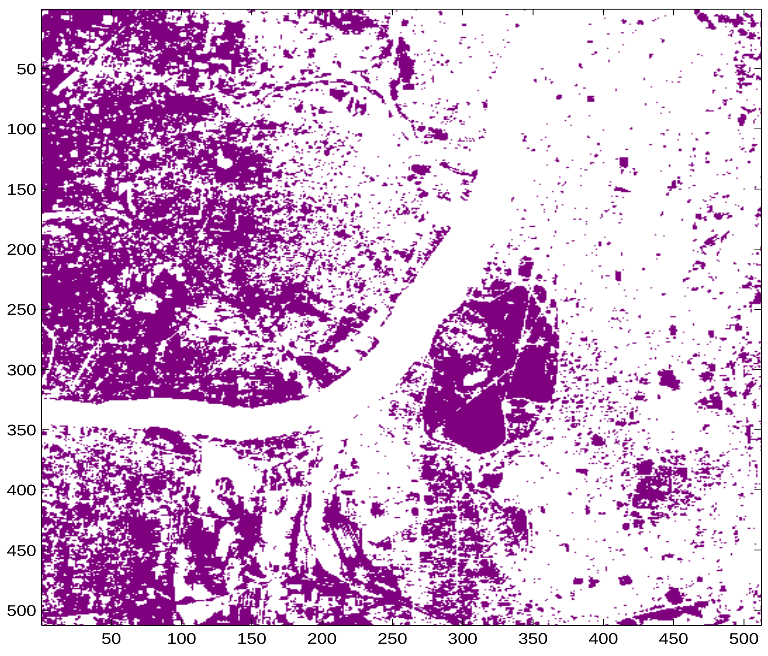
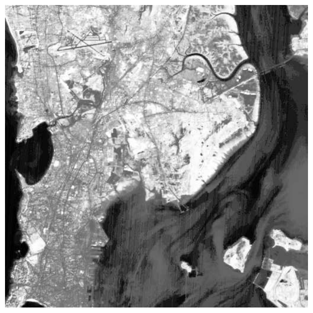
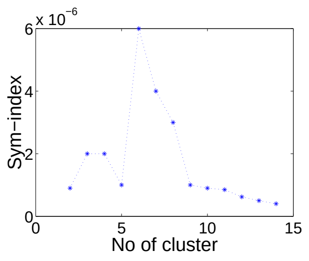


Figure 4.20: Segmented Kolkata image obtained by GAPS-clustering with XB-index (provides $K^* = 2$)

clustering with I-index and XB-index get their optimum values for $K^* = 5$ and $K^* = 3$, respectively. The segmented images corresponding to the optimum values of *Sym*-index and I-index are shown in Figures 4.22 and 4.23, respectively. In case of the former, the water (Arabian sea) surrounding Mumbai gets differentiated into two distinct regions, based on the difference in their spectral properties. The other landmarks e.g., the river above the bridge (north) and dockyard (south) (encircled in Figure 4.22) are detected reasonably well. In the segmentation obtained by GAPS-clustering using I-index, some landmarks, e.g., the river just above the bridge on its left end are not so well delineated. The DB index [56] values corresponding to the segmented images of Mumbai given by the optimal values of *Sym*-index/PS-index, I-index and XB-index are listed in Table 4.5. As smaller values of DB indicates better clustering, the result signifies that the segmentation





(a) (b)

Figure 4.21: (a) IRS image of Mumbai in the NIR band with histogram equalization (b) Variation of *Sym*-index with the number of clusters for IRS image of Mumbai using GAPS

Table 4.5: DB-index values of the segmented Kolkata and Mumbai satellite images corresponding to the optimal values of four cluster validity indices

Validity index	SPOT image of Kolkata	IRS image of Mumbai
Sym-index	0.669	1.586
I index	0.775	1.979
PS-index	0.800	1.586
XB-index	0.724	5.196

corresponding to *Sym*-index is the best. Figure 4.21(b) shows the variation of *Sym*-index with the number of clusters for this data set.

4.8 Discussion and Conclusions

Identifying the appropriate model and the model order are two crucial issues in unsupervised classification. A new symmetry based cluster validity function, *Sym*-index, is proposed in this chapter that exploits the property of point 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

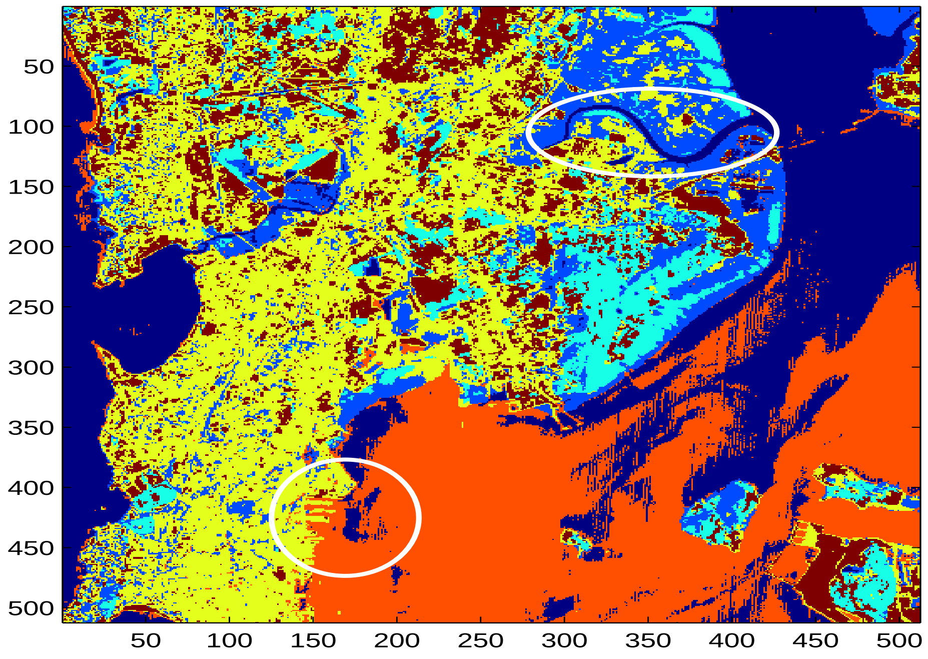


Figure 4.22: Segmented Mumbai image obtained by GAPS-clustering with *Sym*-index/PS-index (provides $K^* = 6$)

with each other to identify a proper clustering are provided. A mathematical justification of the newly proposed *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 *Sym*-index is demonstrated for four artificially generated and three 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 newly proposed *Sym*-index as compared to 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.

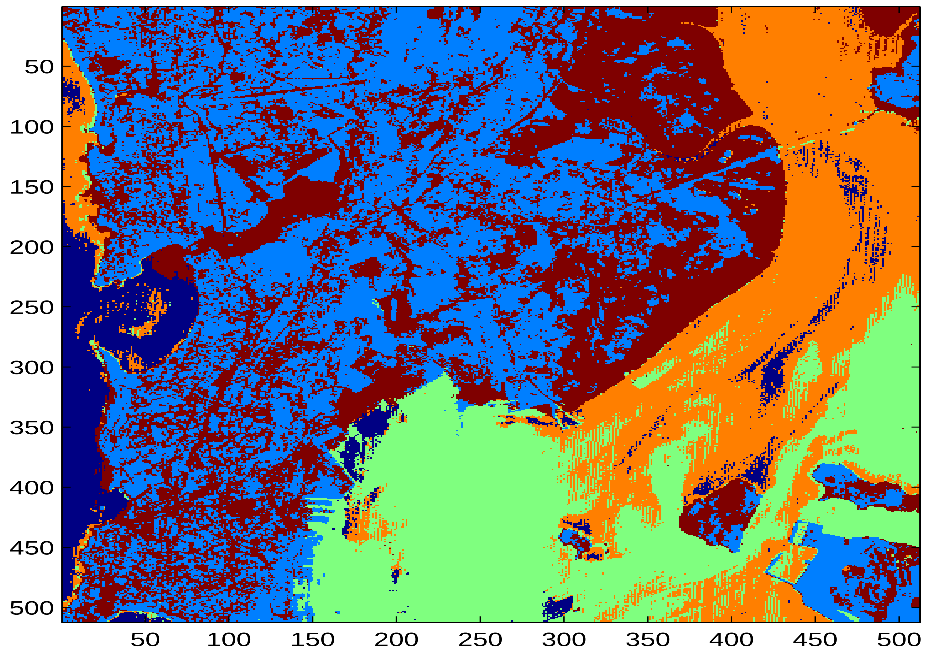


Figure 4.23: Segmented Mumbai image obtained by GAPS-clustering with I-index (provides $K^* = 5$)

Thereafter the point symmetry based distance is incorporated in 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 the incorporation of point symmetry distance in the definitions of existing eight cluster validity indices make 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 [172], results of the newly proposed 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 proposed 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-a-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 two multispectral satellite images. 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 out the proper clustering from the data sets like the synthetic image. In contrast, GAPS-clustering [24] 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 5

Symmetry Based Automatic Clustering

5.1 Introduction

In the last chapter, the appropriate algorithm and the number of clusters from a data set are determined in an iterated fashion. In this chapter an attempt has been made to automate this process, i.e., to determine the number of clusters and the appropriate partitioning in an one-shot process.

In this chapter, a variable string length GA (VGA) based clustering method is used as the underlying segmentation technique. Here assignment of points to different clusters is done based on the PS distance described in Chapter 3. The *Sym*-index is used as the optimizing criterion. The characteristic features of the proposed 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. The *Sym*-index, used as the fitness function, provides the most appropriate partitioning even when the number of clusters, K, is varied. Again use of GA enables the algorithm to come out of local optima, a typical problem associated with local search methods like the K-means. 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 a globally optimal partition is also established. Thereafter this single objective automatic clustering technique is extended to develop a multiobjective clustering technique by utilizing the search capability of AMOSA described in Chapter 2.

5.2 Description of VGAPS [26]

In this section a new clustering technique based on the optimization of *Sym*-index using genetic algorithms is described in detail. It includes determination of the number of clusters as well as the appropriate clustering of the data set. This genetic clustering technique is subsequently referred to as variable string length genetic clustering technique with point symmetry based distance (VGAPS).

5.2.1 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 $<$ 12.3 1.4 5.6 22.1 0.01 10.2 0.0 5.3 15.3 13.2 10.2 7.5 $>$ encodes 4 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 $K_i = (rand() \text{mod}(K_{max} - 1)) + 2$. 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.

5.2.2 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 Section 3.6.2 of Chapter 3). Next, the cluster validity index, *Sym*-index (defined in Section 4.2.1 of Chapter 4), is computed and used as a measure of the fitness of the chromosome. This fitness function is maximized using the genetic algorithm.

5.2.3 Genetic Operations and Terminating Criterion

The following genetic operations are performed on the population of strings for a number of generations.

Selection

The selection operator randomly selects a chromosome from the previous population according to the distribution given by

$$P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)} \tag{5.1}$$

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.

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 the offspring encode the centers of at least two clusters. For this purpose, the operator is defined as follows [133]: Let parent chromosomes P_1 and P_2 encode M_1 and M_2 cluster centers, respectively. The crossover point, τ_1, in P_1 is generated as τ_1=rand() mod 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
$\mathrm{LB}(\tau_2) = \min[2, \max[0, 2-(M_1-\tau_1)]]$ and $\mathrm{UB}(\tau_2) = [M_2 - \max[0, 2-\tau_1]]$.
Therefore τ_2 is given by

$$\begin{aligned}
\tau_2 &= \mathrm{LB}(\tau_2) + \mathrm{rand()mod}(\mathrm{UB}(\tau_2) - \mathrm{LB}(\tau_2)), \mathrm{if}(UB(\tau_2) \geq LB(\tau_2)), \\
\tau_2 &= 0 \text{ otherwise.}
\end{aligned}$$

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 less than two clusters.

Crossover probability, p_c, is selected adaptively as in GAPS described in Section 3.6.4 of Chapter 3.

Mutation

Three types of mutations are considered here.

1. Each cluster center encoded in a chromosome is replaced with a random variable drawn from a Laplacian distribution, $p(\epsilon) \propto e^{-\frac{|\epsilon-\mu|}{\delta}}$, 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. 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. 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.

Any one of the above mentioned types of mutation is applied on each chromosome of the population with a probability of mutation, p_m. The mutation probability is selected adaptively for each chromosome as in GAPS (described in Section 3.6.5 of Chapter 3).

Termination Criterion

In VGAPS, 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. We have implemented elitism at each generation by preserving the best string seen up to that generation in a location outside the population and also inside the population by replacing the string with lowest fitness value. Thus on termination, this location contains the centers of the final clusters.

5.2.4 On The Convergence Property of VGAPS [26]

In this chapter the global convergence of VGAPS to the optimum value of *Sym*-index will be proved along the similar lines that of GAPS (derived in Section 3.7 of Chapter 3) by deriving some conditions on the parameters of VGAPS that ensure the global convergence.

Here the state space comprises the populations containing strings representing partitions with K clusters where $K \in [K_{min}, K_{max}]$. To prove the following theorem $\mathbf{P}$ is needed to be a primitive matrix. Therefore the first step of our investigation contributes in finding the conditions on the operators necessary for the matrix $\mathbf{P}$ to be primitive. The transition matrix $\mathbf{P}$ reflects the probabilistic changes of the chromosomes in the population introduced due to the usage of operators in VGAPS. This transition matrix, $\mathbf{P}$ can be evaluated as a product of four different stochastic matrices as

$$\mathbf{P} = \mathbf{K} \times \mathbf{C} \times \mathbf{M} \times \mathbf{S}. \tag{5.2}$$

$\mathbf{K}$, $\mathbf{C}$, $\mathbf{M}$ and $\mathbf{S}$ in Equation 5.2 describe intermediate transitions due to K-means like update center operator, crossover operator, mutation and selection operators, respectively. It is easy to consider that all these matrices are stochastic matrices.

Propositions 1 and 2 of Chapter 3 together state that in order to show that the matrix $\mathbf{P}$ is primitive, it would be sufficient to find the conditions necessary to be imposed, for $\mathbf{M}$ to be positive and $\mathbf{S}$ to be column-allowable.

To Check Whether the Mutation Matrix is Positive

If the resultant of a mutation operation yields a valid string $s \in \mathcal{S}$ ($\mathcal{S}$ denotes the state space as mentioned in Section 3.7 of Chapter 3) when works upon any other valid string then the matrix $\mathbf{M}$ is positive. The mutation operator defined in Section 5.2.3 assures the fulfillment of 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 more. The second type is for decreasing the value of K i.e., from a chromosome consisting of K_1 number of centers, another chromosome of 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 in number of clusters.

Therefore it can be concluded that the mutation operation can change any valid string to any other valid string in the search space with nonzero probability, making the transition matrix, **M**, corresponding to the above mentioned mutation operator 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**. It can not be assured that **S** is column allowable or not if the fitness function is defined as only the *Sym*-index value of that particular partition. The following modification in the fitness function is incorporated to make **S** a column allowable matrix. Let

$$F(s) = c_s \times Sym_{max} + Sym(s). \tag{5.3}$$

Here, Sym_{max} represents the maximum *Sym*-index value that has been found till the present generation and $c_s \geq 1$. *Sym(s)* is the *Sym*-index value of the sth string. This will make every chromosome of the population to possess a strictly positive value. Therefore, the probability that the present state remains same after the selection, s_{ii} can be bounded as follows:

$$s_{ii} \geq \frac{F(s_1)}{\sum_{l=1}^{N} F(s_l)} \times \frac{F(s_2)}{\sum_{l=1}^{N} F(s_l)} \cdots \times \frac{F(s_N)}{\sum_{l=1}^{N} F(s_l)}$$

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

Here s_l denotes the lth string of the current population. Even though this bound changes with the generation, it is always strictly positive; hence selection matrix **S** is *column-allowable.*

Convergence Proof

Theorem: 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 subsection 5.2.3, and the fitness function be as defined in Equation 5.3. Then

$$\lim_{t\to\infty} Pr\{X(t) = Sym^*\} = 1 \tag{5.4}$$

where $Sym^* = max\{Sym(i)|i \in \mathcal{T}\}$, $\mathcal{T}$ is the set of all legal strings.

Proof: According to the proof provided in [Ref. [159], Theorem 6], a canonical GA whose transition matrix is primitive and which maintains the best solution found over generations converges to the global optimum in the sense given in Equation 5.4. It is proved in Proposition 2 that the transition matrix of VGAPS-clustering with mutation operator same as defined in subsection 5.2.3, and the fitness function as defined in Equation 5.3 is positive. Since every positive matrix is primitive, thus the transition matrix of VGAPS is also primitive. Moreover, VGAPS uses elitist model of GA, i.e., it preserves the best solution obtained upto the present time instant. Thus, the above theorem follows from ([Ref. [159], Theorem 6]).

The above theorem implies that $X(t)$, the maximum *Sym*-index value of the strings found by VGAPS upto the instant t, converges to the global optimum Sym^*, with probability 1 when t goes to infinity.

5.3 Data Sets Used and Implementation Results

This section provides a description of the data sets and the implementation results of the proposed algorithm. Five artificial and three real life data sets are used for the experiments.

5.3.1 Data Sets Used

1. Artificial data sets used: *Sym_3_2*, *AD_5_2*, *Bensaid_3_2* and two more 3-dimensional data sets.

   (a) *Sym_3_2*: This data set is described in Section 3.8.1 of Chapter 3.

   (b) *AD_5_2*: This data set is described in Section 3.8.1 of Chapter 3.

   (c) *Bensaid_3_2*: This data set is described in Section 3.8.1 of Chapter 3.

   (d) *3dsym_3_2*: This data set contains 398 points distributed on two non-overlapping ellipsoidal shells in three dimensions as shown in Figure 5.1(a).

   (e) *3dsym_3_3*: This data set contains 598 points distributed on two non-overlapping ellipsoidal shells and in one elliptical shaped cluster in three dimensions as shown in Figure 5.1(b).

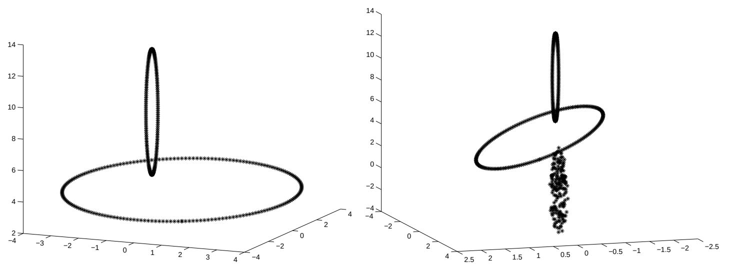


Figure 5.1: (a) *3dsym_3_2* dataset (b) *3dsym_3_3* dataset

2. Real-life data sets: The 3 real life data sets were obtained from [2]. These are *Iris*, *Cancer* and *Newthyroid* described in Section 3.8.1 of Chapter 3.

Table 5.1: Comparing the number of clusters found on 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 actual number of clusters present in the data and OC denotes the obtained number of clusters.

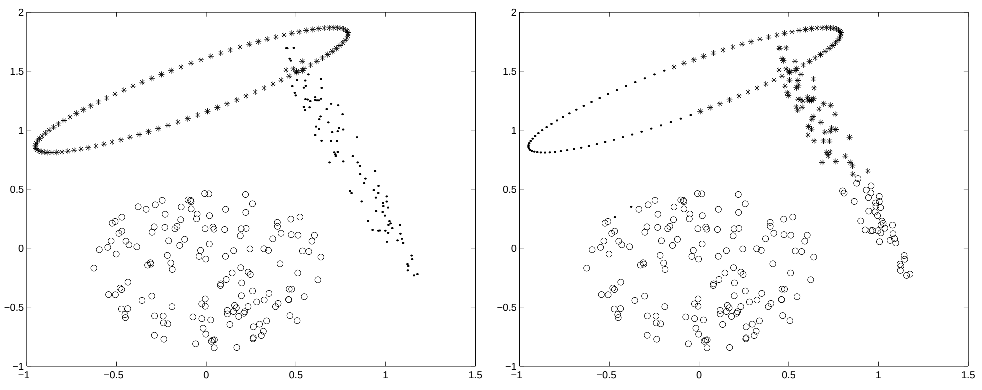
Data Set	AC	OC by VGAPS using			OC by different methods		
		Sym	I	PS	VGAPS	GCUK	HNGA
Sym_3_2	3	3	8	3	3	3	16
AD_5_2	5	5	6	4	5	5	5
Bensaid_3_2	3	3	3	3	3	2	3
3dsym_3_2	2	2	3	9	2	8	5
3dsym_3_3	3	3	4	9	3	8	17
Iris	3	3	3	2	3	2	2
Cancer	2	2	2	2	2	2	2
Newthy roid	3	3	7	8	3	8	5

5.3.2 Results and Discussions

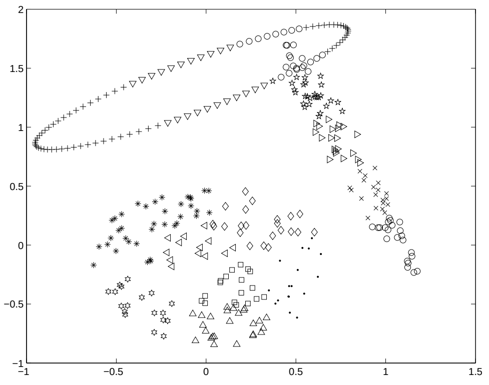
As already discussed in detail in Section 3.8.2 of Chapter 3, a proper choice of the parameters is crucial for good performance of genetic algorithms.

In VGAPS-clustering, as in GAPS, the population size is taken to be equal to 100. K_{min} (minimum number of clusters) and K_{max} (maximum number of clusters) are set equal to 2 and $\sqrt{n}$, respectively, where n is the total number of data points in the particular data set. VGAPS is executed for a total of 50 generations.

In order to evaluate the proposed method, we performed two types of experiments. At first we show that VGAPS optimizing *Sym*-index performs better than VGAPS optimizing two other indices, viz., PS-index [43] and I-index [132]. After that, we explore the properties of the VGAPS optimizing *Sym*-index and compare its performance with other genetic clustering methods,



(a) (b)



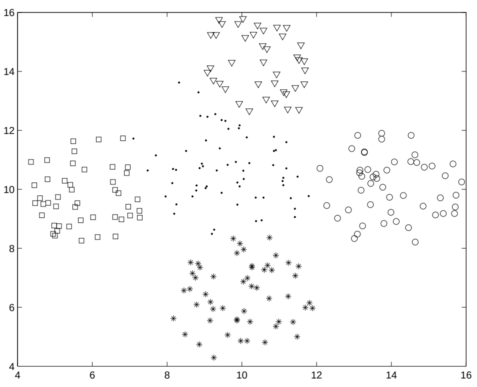
(c)

Figure 5.2: Clustered *Sym_3_2* after application of (a) VGAPS-clustering where 3 clusters are detected (b) GCUK-clustering where 3 clusters are detected (c) HNGA-clustering where 16 clusters are detected

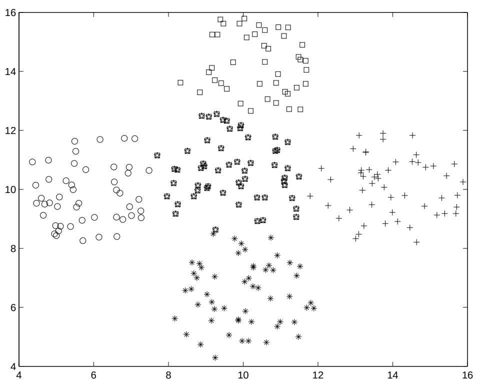
which do not need knowledge about the number of clusters *a priori*.

Exploring *Sym*-index as Fitness Function

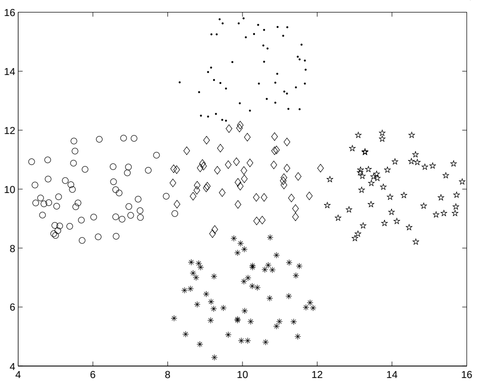
In the first experiment, we establish the effectiveness of using the *Sym*-index with VGAPS-clustering vis-a-vis another point symmetry based validity index, PS-index [43] and an Euclidean distance based cluster validity index, I-index [132]. The number of clusters obtained after applying VGAPS optimizing these three validity indices separately for all the data sets are shown in



(a)



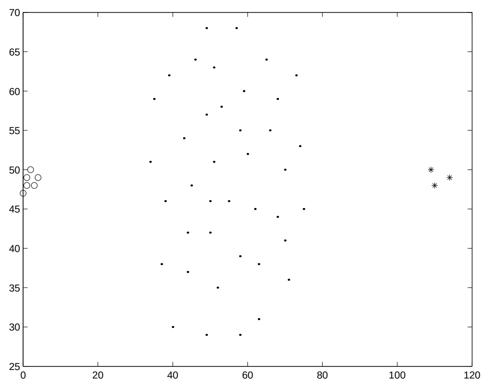
(b)



(c)

Figure 5.3: Clustered *AD_5_2* using (a) VGAPS-clustering where 5 clusters are detected (b) GCUK-clustering where 5 clusters are detected (c) HNGA-clustering where 5 clusters are detected

Table 5.1. It can be seen from the table that VGAPS-clustering with *Sym*-index is able to find out the proper cluster number from data sets having symmetrical shaped clusters. VGAPS-clustering with *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. It is because *I*-index essentially prefers hyperspherical clusters, which is not the case for *Sym_3_2*, *3dsym_3_2* and *3dsym_3_3*. VGAPS-clustering with PS-index is able to detect the proper clusters from those data sets where the clusters have strong point symmetry. However, as discussed in Section 3.2


(a)

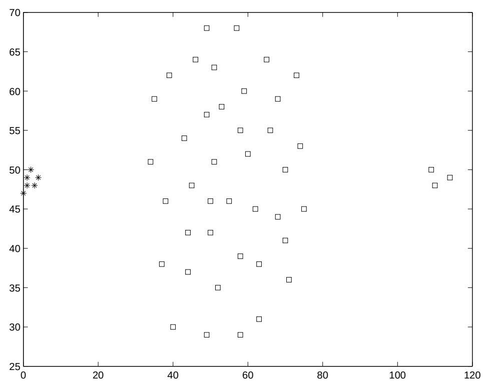

(b)

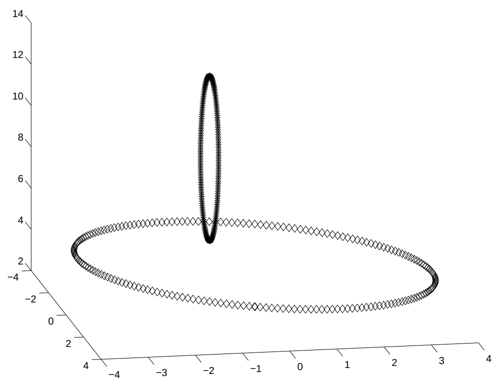
Figure 5.4: (a) Clustered *Bensaid_3_2* by VGAPS-clustering and HNGA-clustering where 3 clusters are detected (c) Clustered *Bensaid_3_2* by GCUK-clustering where 2 clusters are detected

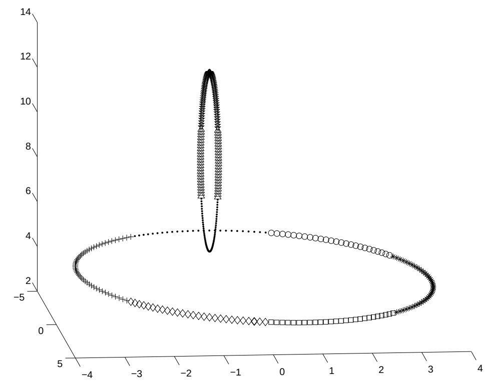
of Chapter 3 of this thesis, the definition of point symmetry distance in PS-index precludes the detection of symmetrical interclusters. Thus it fails for *AD_5_2* which has clearly symmetrical interclusters.

Exploring the VGAPS-clustering

In this section, we compare the performance of the VGAPS-clustering (in conjunction with *Sym*-index) with those of the GCUK-clustering [16] and a recently developed HNGA clustering [179]. GCUK clustering utilizes Davies-Bouldin [56] cluster validity index for computing the fitness of the chromosomes. In HNGA [179], a weighted sum validity function (WSVF), which is a weighted sum of several normalized cluster validity functions, is used for optimization.

Table 5.1 shows the number of clusters identified by the three clustering algorithms for all the data sets. As is evident from Table 5.1, VGAPS is able to find out the appropriate number of clusters and the proper partitioning for all the data sets. Figures 5.2(a), 5.3(a), 5.4(a), 5.5(a), and 5.6(a) show the final partitionings obtained after application of VGAPS on *Sym_3_2*, *AD_5_2*,


(a)


(b)

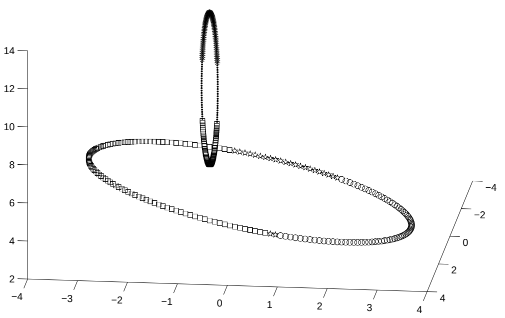

(c)

Figure 5.5: Clustered *3dsym_3_2* using (a) VGAPS-clustering where 2 clusters are detected (b) GCUK-clustering where 8 clusters are detected (c) HNGA-clustering where 5 clusters are detected

Bensaid_3_2, *3dsym_3_2*, and *3dsym_3_3*, 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).

The results on real-life data sets are quantitatively compared with respect to the *Minkowski scores* [98] described earlier in Section 3.8.3 of Chapter 3.

Final clustering results obtained after the application of GCUK algorithm on the five artificial data sets are also shown in Figures 5.2(b), 5.3(b), 5.4(b), 5.5(b) and 5.6(b), respectively. Results shown in Table 5.1 reveals that GCUK-clustering is able to determine the proper cluster number only for

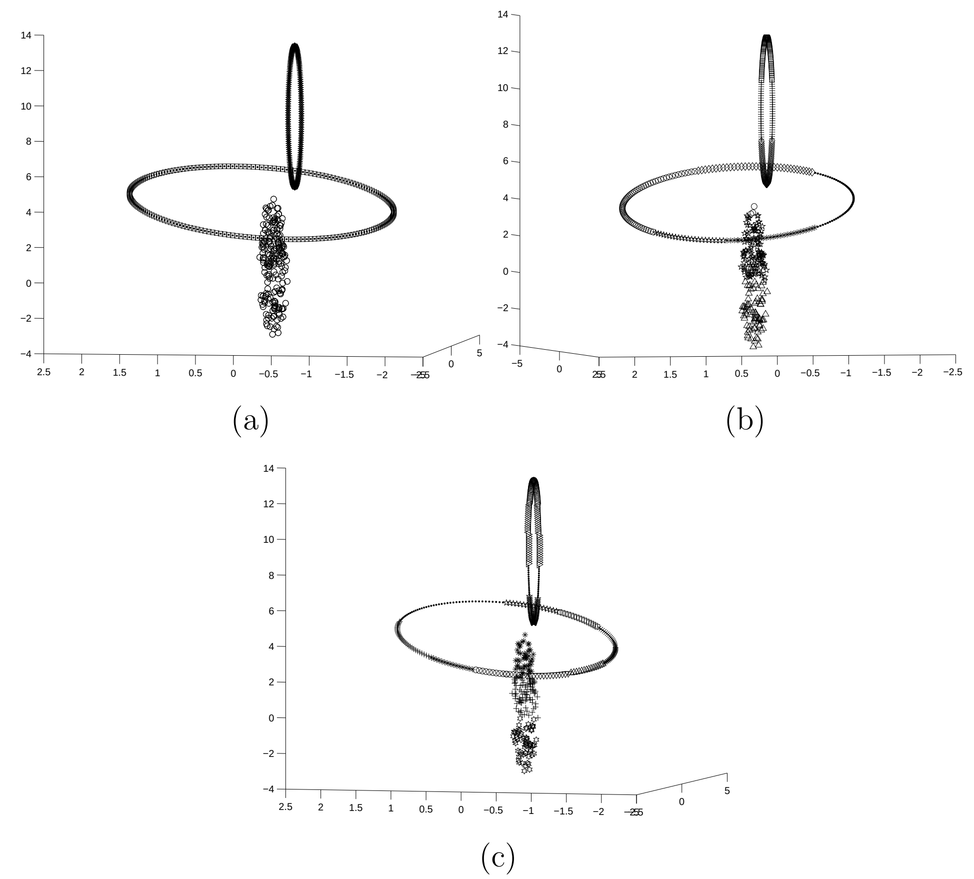


Figure 5.6: Clustered *3dsym_3_3* using (a) VGAPS-clustering where 3 clusters are detected (b) GCUK-clustering where 8 clusters are detected (c) HNGA-clustering where 17 clusters are detected

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 Figure 5.2(b)) is not proper. Figures 5.2(c), 5.3(c), 5.4(a), 5.5(c) and 5.6(c) show, respectively, the clustering results obtained after application of HNGA-clustering on five artificial data sets. Again, results shown in Table 5.1, reveal that HNGA-clustering is able to determine the proper cluster number only for *AD_5_2*, *Bensaid_3_2* and *Cancer* data sets. Thus it is easy to conclude that HNGA-clustering is only able to find out hyperspherical clusters from a data set but not any other

Table 5.2: *Minkowski Scores* obtained by three algorithms for all data sets used here for experiment

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
3dsym_3_2	0.0 ± 0.00	1.01 ± 0.02	1.12 ± 0.012
3dsym_3_3	0.0 ± 0.00	1.1 ± 0.03	1.21 ± 0.01
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
Newthyroid	0.58 ± 0.03	0.83 ± 0.021	0.84 ± 0.022

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.

Minkowski Score (MS) [98] (defined in Equation 3.22 of Chapter 3) of the resultant partitionings are calculated after application of all the three algorithms on both the artificial and real-life data sets used here for the experiments.

Each of the above mentioned three algorithms are executed ten times for 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 5.2. Except for *AD_5_2*, VGAPS-clustering is found to provide the lowest MS values for the other data sets, which indicate that the partitionings corresponding to VGAPS-clustering are the best among the three clustering algorithms. ANOVA [5] statistical analysis is 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 it is MS value) by a single independent variable (here it is the algorithm). Analysis of variance is used to test the hypothesis that several means are equal. From the statistical test ANOVA, it is found that the difference in the mean MS values obtained by VGAPS-clustering with those obtained by GCUK-clustering and HNGA-clustering algorithms, are statistically significant at the level of 0.05 for all data sets. This indicates the better performance of VGAPS as compared to GCUK-clustering and HNGA-clustering. For *AD_5_2*, HNGA-clustering performs the best in terms of MS score and the difference in the mean MS values obtained by HNGA and VGAPS clustering techniques is statistically significant with significance value $2.4603e-008$.

5.4 VAMOSA: Symmetry Based Multiobjective Clustering Technique for Automatic Evolution of Clusters [171]

The VGAPS clustering technique optimizes a single cluster validity measure, namely the point symmetry based cluster validity index, *Sym*-index. However, a single cluster validity measure like *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 the different data characteristics. In order to achieve this, in this chapter, the problem of clustering a data set is posed as one of multiobjective optimizations (MOO) [60], where search is performed over a number of, often conflicting, objective functions. The newly developed simulated annealing based multiobjective optimization technique, AMOSA [27], described in Chapter 2, is used here to determine the appropriate cluster centers and the corresponding partitioning from a data set. The resultant technique is referred to as VAMOSA. Here encoding of a variable number of cluster centers, and the assignment of the points to the different clusters are done as in VGAPS (discussed in Section 5.2.1 and 5.2.2 of this chapter). Two cluster validity measures are optimized simultaneously, the well-known XB-index [206] and the recently developed point symmetry distance based *Sym*-index (defined in Section 4.2.1 of Chapter 4). 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 cluster centers are extracted from a string in AMOSA. 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 [206] is defined as a function of the ratio of the total variation σ to the minimum separation *sep* of the clusters. For the crisp XB-index, 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, and $d_e(\overline{c}_i, \overline{x}_k^i)$ 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 crisp version of 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 therefore to minimize the XB-index for achieving the proper clustering. The second objective function is the newly defined point symmetry distance based *Sym*-index (defined in Section 4.2.1 of Chapter 4). Thus the two objective functions of VAMOSA are $f_1 = XB$ and $f_2 = \frac{1}{Sym}$. These two objective functions are minimized simultaneously in VAMOSA using the search capability of AMOSA.

Here mutation operation similar to VGAPS (described in detail in Section 5.2.3 of this Chapter) is used. Finally the best solution is selected from the final archive of AMOSA following the procedure mentioned in Section 3.9.1 of Chapter 3.

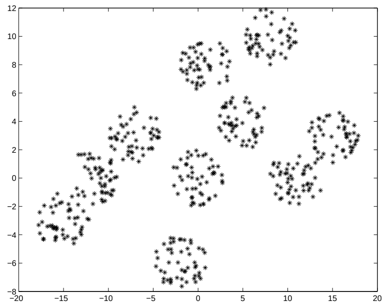
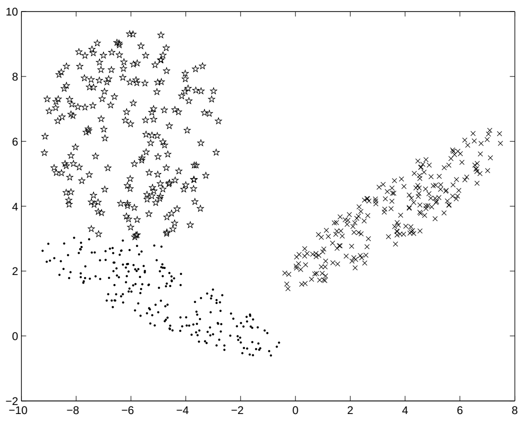
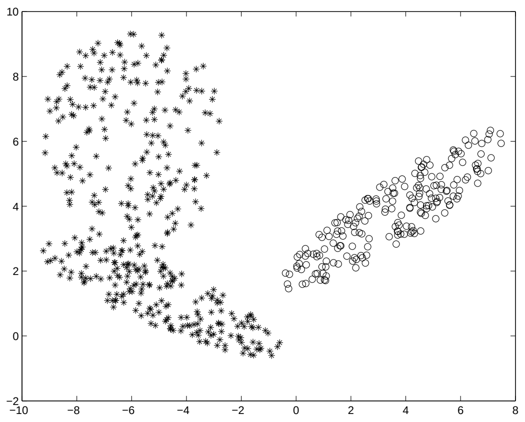


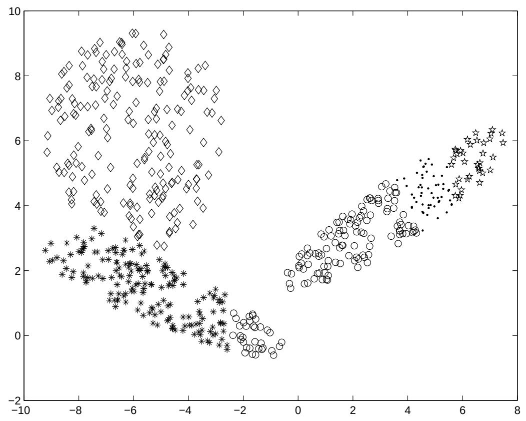
Figure 5.7: *AD_10_2*



(a)



(b)



(c)

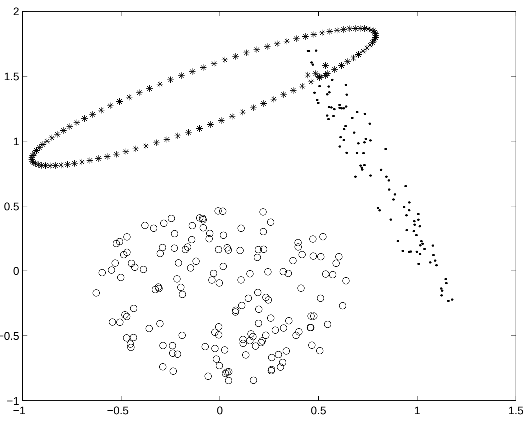
Figure 5.8: Automatically clustered *Mixed_3_2* after application of (a) VA-MOSA/VGAPS clustering technique for $K = 3$ (b) MOCK clustering technique for $K = 2$ (c) GCUK-XB clustering technique for $K = 5$.

5.4.1 Data Sets Used for the Experiment

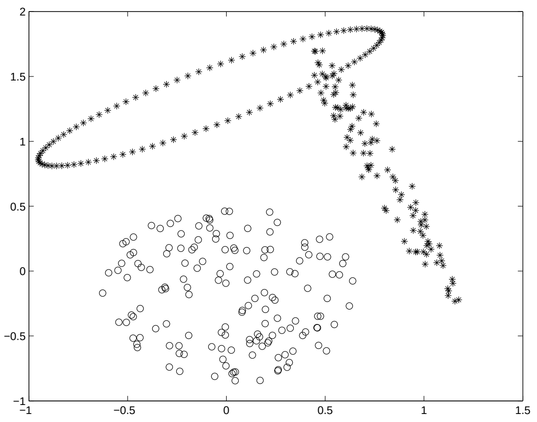
Eight data sets are used for the experiment: four of them are artificial data (*Mixed_3_2*, *Sym_3_2*, *AD_5_2* and *AD_10_2*) and four are real-life data sets (*Iris*, *Cancer*, *Newthyroid* and *LungCancer*). Real-life data sets are obtained from [2].

1. *Mixed_3_2*: This data set is described in Section 3.8.1 of Chapter 3.

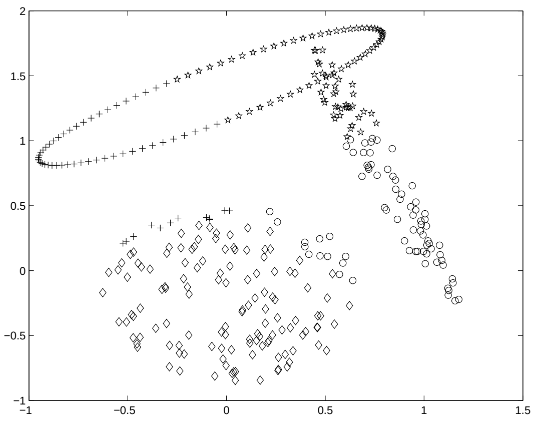
2. *Sym_3_2*: This data set is described in Section 3.8.1 of Chapter 3.



(a)



(b)



(c)

Figure 5.9: Automatically clustered *Sym_3_2* after application of (a) VAMOSA/VGAPS clustering technique for $K = 3$ (b) MOCK clustering technique for $K = 2$ (c) GCUK-XB clustering technique for $K = 4$.

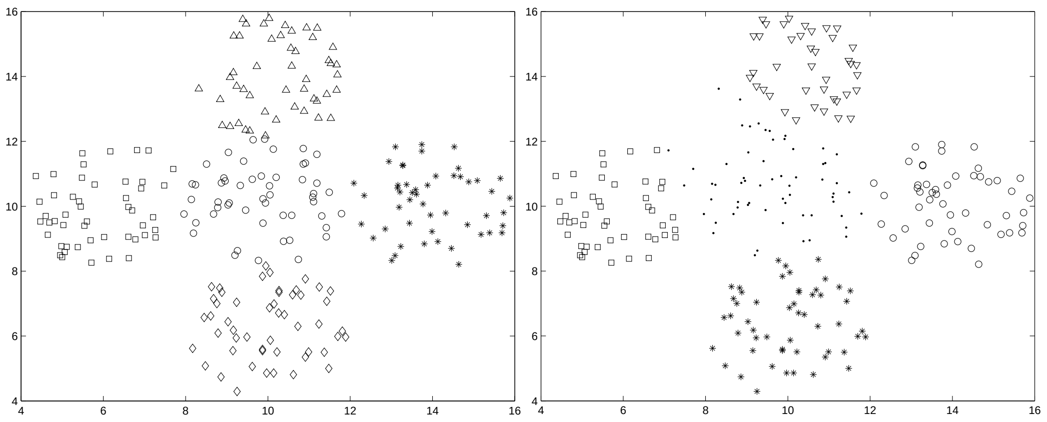
3. *AD_5_2*: This data set is described in Section 3.8.1 of Chapter 3.

4. *AD_10_2*: This data set, used in Ref. [21], 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 Figure 5.7.

5. *Iris*: This data set is described in Section 3.8.1 of Chapter 3.

6. *Cancer*: This data set is described in Section 3.8.1 of Chapter 3.

7. *Newthyroid*: This data set is described in Section 3.8.1 of Chapter 3.



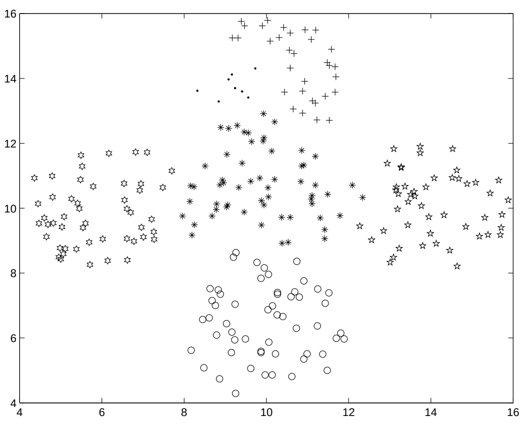
(a) (b)

Figure 5.10: Automatically clustered *AD_5_2* after application of (a) VAMOSA clustering technique for $K = 5$ (b) VGAPS clustering technique for $K = 5$.

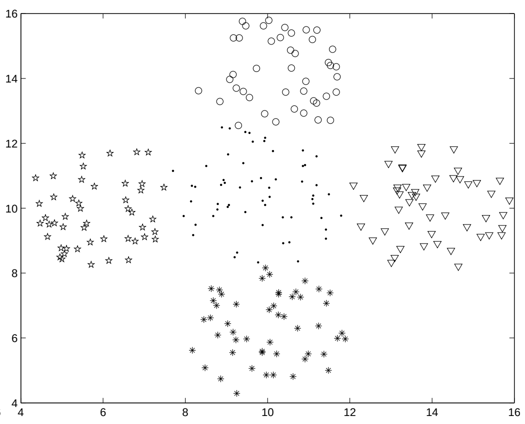
8. *LungCancer*: This data set is described in Section 3.8.1 of Chapter 3.

5.4.2 Experimental Results

The parameters of the proposed VAMOSA clustering technique are 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 [81] is also executed on the above mentioned data sets with default parameter settings. The source code for MOCK is obtained from (http://dbkgroup.org/handl/mock/). In MOCK, the best solution from the final Pareto optimal front is selected by GAP-statistics [196]. Note that for every data set used here for the experiment, the actual class labels of all the data points are available. Thus in order to quantify the quality of the obtained partitionings by different algorithms, their corresponding *Minkowski Score*s (MS) [98] (defined in Equation 3.22 of Chapter 3) are computed. The number of clusters identified by the best solution of the proposed VAMOSA clustering technique and MOCK clustering technique, and the *Minkowski Score* (MS) values of the corresponding partitionings are reported in Table 5.3.



(c)



(d)

Figure 5.11: Automatically clustered *AD_5_2* after application of (a) MOCK clustering technique for $K = 6$ (b) GCUK-XB clustering technique for $K = 5$.

In order to show the efficacy of the proposed 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) [16] and VGAPS clustering (described in Section 5.2 of Chapter 5), are also executed on the above mentioned eight data sets. These single objective automatic clustering techniques provide a single solution after their execution. GCUK clustering technique optimizes an Euclidean distance based cluster validity index, XB-index [206], by using the search capability of genetic algorithms to automatically determine the appropriate partitioning from data sets. The parameters of the GCUK-XB clustering technique are as follows: population size=100, number of generations=40, probability of mutation=0.2 and probability of crossover=0.8 (as specified in [16]). The parameters of the VGAPS clustering technique are set as detailed in Section 5.3.2 of Chapter 5. The number of clusters automatically determined by these clustering techniques for the eight data sets are also reported in Table 5.3. The MS values are also calculated for the partitionings obtained by these two single objective clustering techniques for these eight data sets. These are also reported in Table 5.3. Here the best MS values obtained by the algorithms over five runs for all data sets are reported.

1. *Mixed_3_2*: As can be seen from Table 5.3, both the proposed VA-

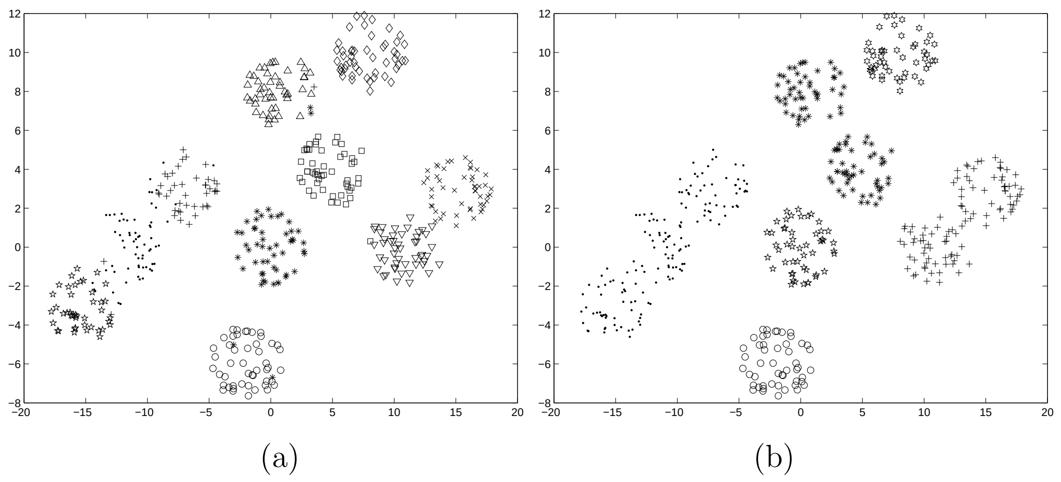
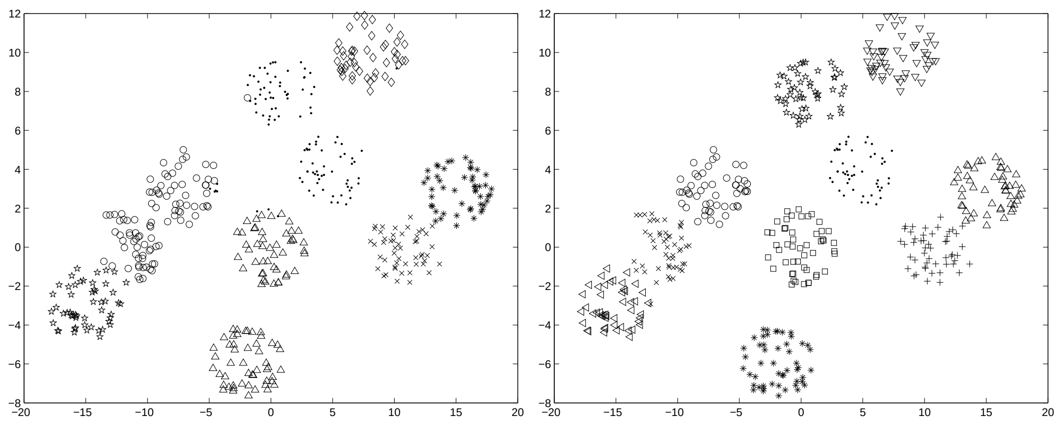


Figure 5.12: Automatically clustered *AD_10_2* after application of (a) VAMOSA clustering technique for $K = 10$ (b) MOCK clustering technique for $K = 6$.

MOSA and the existing point symmetry based clustering technique, VGAPS, are able to detect the appropriate number of clusters and the proper partitioning from this data set. The corresponding partitioning is shown in Figure 5.8(a). MOCK fails to detect the appropriate number of clusters and the appropriate partitioning. It merges two overlapping clusters into a single cluster. The corresponding partitioning is shown in Figure 5.8(b). GCUK optimizing XB-index is also not able to detect the appropriate partitioning. The partitioning provided by GCUK-XB is shown in Figure 5.8(c).

2. *Sym_3_2*: As seen from Table 5.3, both the symmetry based clustering techniques, proposed 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 Figure 5.9(a). MOCK again merges the two overlapping clusters into one cluster and provides $K = 2$ as the optimal number of clusters. The corresponding partitioning is shown in Figure 5.9(b). GCUK-XB clustering technique identifies total $K = 4$ number of clusters from this data set. The corresponding partitioning is shown in Figure 5.9(c). The MS scores reported in Table 5.3 also show the poorer performance of both MOCK and GCUK-XB



(a) (b)

Figure 5.13: Automatically clustered *AD_10_2* after application of (a) VGAPS clustering technique for $K = 7$ (b) GCUK-XB clustering technique for $K = 10$.

clustering techniques for this data set.

3. *AD_5_2*: As can be seen from Table 5.3, for this data set the proposed VAMOSA clustering technique performs much better than VGAPS clustering technique. The corresponding partitionings are shown in Figures 5.10(a) and 5.10(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 Figure 5.11(a). GCUK clustering optimizing 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 5.3). The corresponding partitioning is shown in Figure 5.11(b).

4. *AD_10_2*: For this data set, GCUK-XB clustering provides the best partitioning (shown in Figure 5.13(b)) and the corresponding MS value is also the minimum (refer to Table 5.3). VAMOSA clustering technique is also able to detect the appropriate number of clusters from this data set but the corresponding MS value is slightly higher than that of GCUK-XB clustering (refer to Table 5.3). The corresponding

Table 5.3: Number of clusters and the *Minkowski Score* (MS) values obtained by VAMOSA clustering technique, another automatic MO clustering technique, MOCK, two single objective clustering techniques, VGAPS clustering optimizing *Sym*-index and GCUK clustering optimizing XB-index, for all the data sets used here for experiment.

DataSet	AC	VAMOSA		MOCK		VGAPS		GCUK-XB	
		OC	MS	OC	MS	OC	MS	OC	MS
Mixed_3_2	3	**3**	**0.18**	2	0.82	**3**	**0.18**	5	0.62
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
AD_10_2	10	10	0.43	6	1.01	7	0.84	**10**	**0.09**
Iris	3	2	0.80	2	0.82	**3**	**0.62**	2	0.84
Cancer	2	**2**	**0.32**	2	0.39	2	0.36	2	0.38
Newthyroid	3	5	**0.57**	2	0.82	3	0.58	6	0.65
LungCancer	3	**3**	**0.85**	7	0.97	2	0.97	6	0.94

partitioning is shown in Figure 5.12(a). But both MOCK and VGAPS clustering techniques are not able to detect the appropriate number of clusters from this data set. The partitionings identified by both MOCK and VGAPS clustering techniques for this data set are shown in Figures 5.12(b) and 5.13(a), respectively.

5. *Iris*: For this real-life data set, only VGAPS clustering technique is able to determine the appropriate number of clusters. The corresponding MS score is also the minimum (refer to Table 5.3). As this is a higher dimensional data set, no visualization is possible. Other three clustering algorithms, newly proposed VAMOSA, MOCK and GCUK-XB, provide $K = 2$ as the optimal number of clusters, which is also often obtained for many other methods for *Iris* data set. But the MS score corresponding to VAMOSA for $K = 2$ is the minimum among these three clustering techniques (refer to Table 5.3).

6. *Cancer*: For this data set all the four clustering techniques are able to detect the appropriate number of clusters ($K = 2$ for this case).

But the MS value obtained by VAMOSA clustering technique is the minimum (refer to Table 5.3).

7. *Newthyroid* : For this real-life data set only the VGAPS clustering technique is able to detect the appropriate number of clusters ($K = 3$ in this case). VAMOSA provides $K = 5$ as the optimal number of clusters. But the MS value obtained by the final solution provided by VAMOSA clustering technique is lesser than that obtained by VGAPS (refer to Table 5.3). Both MOCK and GCUK-XB clustering techniques are not able to determine the appropriate number of clusters from this data set. MOCK attains the highest MS value compared to other three comparing algorithms. For the purpose of comparison the final Pareto optimal front obtained by VAMOSA clustering technique is shown in Figure 5.15(a). The boxplots of the *Minkowski Score* values of the solutions on the final Pareto optimal front provided by both VAMOSA and MOCK clustering techniques are shown in Figure 5.14 for the purpose of illustration. This figure reveals that the MS values over the final Pareto optimal front provided by VAMOSA are much lesser than those of MOCK clustering technique.

8. *LungCancer*: For this high dimensional data set only the proposed 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 5.3). The final Pareto optimal front obtained by the proposed VAMOSA clustering technique is shown in Figure 5.15(b).

Summary of Results

It can be seen from the above results that the proposed VAMOSA clustering technique is able to detect the appropriate partitioning and the appropriate number of clusters from most of the data sets used here for the experiments. It outperforms another MO clustering technique, MOCK, and two single objective genetic algorithm based clustering techniques. The superiority of VAMOSA is also established on four real-life data sets which are of different characteristics with the number of dimensions varying from 4 to 56. Results on the eight artificial and real-life data sets establish the fact that VAMOSA is well-suited to detect the number of clusters automatically from data sets having clusters of widely varying characteristics as long as they possess the property of point symmetry.

We have also experimented with MOCK using our proposed method of selecting a single solution from the final non-dominated solution set (as described in Section 3.9.1 of Chapter 3). We found that for *AD_5_2* and *AD_10_2*, MOCK was able to detect the appropriate number of clusters. (The use of Gap Statistic however did not select these solutions as the best ones.) Even then, while for *AD_10_2* the MS value was 0.13 (better than VAMOSA), for *AD_5_2* it was 0.33 (which is poorer than 0.25 provided by VAMOSA).

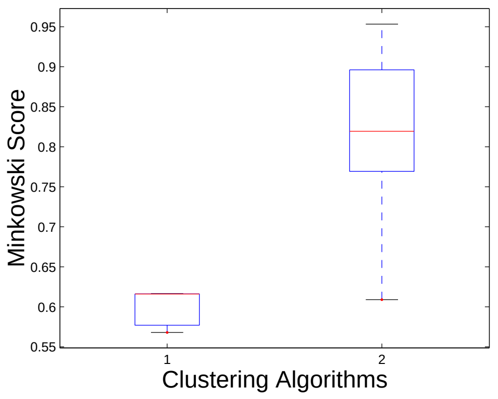
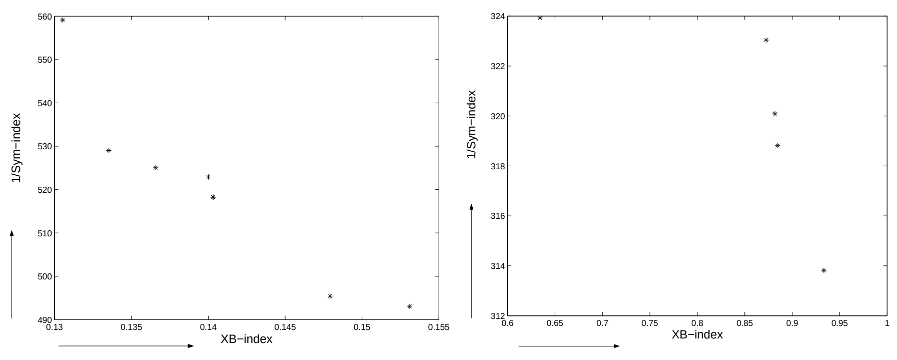


Figure 5.14: Boxplots of the *Minkowski Scores* of the Pareto optimal solutions obtained by VAMOSA clustering Technique and MOCK clustering Technique for *Newthyroid* data set. Here column '1' denotes the VAMOSA clustering technique and column '2' denotes the MOCK clustering technique.

5.5 Discussion and Conclusions

Most of the clustering methods make prior assumptions about the structure of the clusters. For example, GCUK-clustering, which is a genetic K-means clustering technique for automatic determination of clusters, can only detect



(a) (b)

Figure 5.15: Pareto optimal front obtained by the proposed VAMOSA clustering technique for (a) *Newthyroid* data set (b) *LungCancer* data set

equisized hyperspherical clusters from a data set. In this chapter the newly defined 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. It optimizes the *Sym*-index, which is capable of detecting both the proper partitioning and the proper number of clusters present in a data set. 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 the proposed VGAPS-clustering is also established. The effectiveness of the VGAPS-clustering, as compared to two recently proposed automatic clustering techniques, namely, GCUK-clustering and HNGA-clustering, is demonstrated on five artificially generated and three real-life data sets of different characteristics. Results on these eight data sets establish the fact that VGAPS-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, or overlap or size, as long as they possess the property of symmetry. 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. Note that optimizing *Sym*-index is not inherent to a GA framework. Any other optimization technique, such as Simulated Annealing [13], may have been used instead.

Thereafter in this chapter, VGAPS-clustering technique is extended to develop a multiobjective clustering technique. Two cluster validity measures, one based on the newly developed point symmetry based distance, *Sym*-index, and another based on the Euclidean distance, *XB*-index, are optimized simultaneously. In this regard, the newly developed multiobjective simulated annealing based technique, AMOSA, has been used in this chapter. The effectiveness of the proposed clustering technique in detecting the proper number of partitions and the proper partitioning is shown for four artificial and four real-life data sets and the results are compared with those obtained by another MO clustering technique, MOCK [81], two single objective automatic genetic clustering techniques, GCUK clustering optimizing XB-index [16] and VGAPS clustering technique.

VGAPS as well as VAMOSA seek for clusters which are point symmetric with respect to their centers. Thus they will fail if the clusters do not have this property. Much further work is needed to investigate using different and more objectives, and to test the VAMOSA approach still more extensively. Selecting the best solution(s) from the Pareto optimal front is an important problem in multiobjective clustering. A semi-supervised method of selecting a single solution from the Pareto optimal front is used here. But this method assumes that the labeling of the partial data points is known beforehand. Thus some new unsupervised methods to choose the best solution from the final Pareto optimal front have to be developed.

VGAPS and VAMOSA are able to detect the appropriate number of clusters and the appropriate partitioning from data sets having symmetrical shaped clusters. Thus these will fail for clusters having other shapes. Thus developing a clustering technique which can automatically determine the appropriate number of clusters and the appropriate partitioning from data sets with clusters having different shapes is another important research problem. The next chapter deals with this.

Chapter 6

A Generalized Automatic Clustering Algorithm in a Multiobjective Framework

6.1 Introduction

The newly developed single and multiobjective clustering techniques like VGAPS and VAMOSA are able to automatically detect the number of clusters and the appropriate partitioning from data sets having clusters of symmetrical shapes. But these algorithms fail for data sets having clusters other than symmetrical shapes.

In this chapter we have developed a new multiobjective clustering technique with encoding of cluster centers as in [131], which can detect the appropriate number of clusters and the appropriate partitioning from data sets with many different types of cluster structures. A newly developed simulated annealing based multiobjective optimization technique, AMOSA, is used as the underlying optimization strategy. The concept of "multiple centers" corresponding to each cluster is used in this chapter. Each cluster is divided into several non-overlapping small hyperspherical sub-clusters and the centers of these sub-clusters are encoded in a state 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 the proposed *GenClustMOO* in comparison with MOCK-clustering technique [81], VAMOSA clustering technique (described in Section 5.4 of Chapter 5) and a single objective genetic clustering technique, VGAPS-clustering (described in Section 5.2 of Chapter 5), is shown for seven artificial data sets (including most of the data sets used in [81]) and five 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, PESA2 [50].

6.2 Proposed Method of Multiobjective Clustering [25]

This section describes the newly proposed multiobjective clustering technique, *GenClustMOO*, in detail.

6.2.1 State Representation and Archive 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 non-overlapping hyperspherical sub-clusters. Then each cluster is represented by the centers of these individual sub-clusters. Suppose a particular state encodes the centers of K number of clusters and each cluster is divided into C number of sub-clusters. If the data set is of dimension d, then the length of the state will be $C \times K \times d$. This concept of representing one cluster using multi-centers is shown in Figure 6.1. Suppose a particular state contains $K = 2$ number of clusters. Each cluster is divided into 10 smaller sub-clusters, i.e., here $C = 10$. Let the dimension (d) of the data set be 2. Suppose the center of the jth sub-cluster of the ith cluster is denoted by $\overline{c}_j^i = (cx_j^i, cy_j^i)$. Then this state will look like: $< cx_1^1, cy_1^1, cx_2^1, cy_2^1, \ldots, cx_{10}^1, cy_{10}^1, cx_1^2, cy_1^2, \ldots, cx_{10}^2, cy_{10}^2 >$. Each

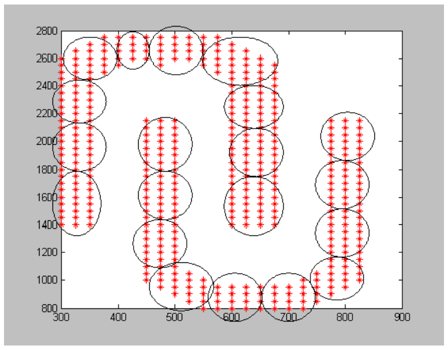


Figure 6.1: Example of representing clusters using multiple cluster centers.

state i in the archive initially contains K_i number of clusters, such that $K_i = (rand()\text{mod}(K^{max} - 1)) + 2$. 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 two and K^{max}. Our initialization procedure is motivated by that of Ref. [81]. 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 are initialized after running Single Linkage clustering algorithm [67] for different values of K. These solutions perform well when clusters present in the data set are well-separated. Another one third of the solutions in the archive are generated using the K-means algorithm. These solutions perform well when clusters present in the data set are hyperspherical in shape. The last one third of the solutions are generated randomly, i.e., for these states the K_i centers encoded in a state 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 state.

6.2.2 Assignment of Points

For the purpose of assignment, each sub-cluster 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 sub-cluster where

$$k = argmin_{i=1}^{K \times C} d_e(\overline{c}_i, \overline{x}_j).$$

Here $d_e(\overline{c}_i, \overline{x}_j)$ denotes the Euclidean distance between sub-cluster center $\overline{c}_i$ and the data point $\overline{x}_j$. Thereafter, the partition matrix is formed in the following way: $u(kj) = 1$ and $u(ij) = 0,\ \forall i = 1 \ldots K \times C,\ k \neq i$.

6.2.3 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 validity index 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 cluster validity index to be optimized is the newly proposed symmetry based cluster validity index, *Sym*-index defined in Chapter 4.

6.2.4 Newly proposed Connectivity Based Cluster Validity Index: *Con*-index

In this chapter a new cluster validity index based on the concept of connectedness of the clusters is developed. 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 well-separated. The concept of relative neighborhood graph (RNG) [198] has been successfully applied for solving several pattern recognition problems. An unsupervised clustering technique based on the concepts of RNG is developed in Ref. [12]. 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.

Relative Neighborhood Graph [198]

Let $\overline{p}$, $\overline{q}$ be two points in r-dimensional Euclidean space. Let r be an integer. Then the lune of $\overline{p}$ and $\overline{q}$, denoted by $lun(\overline{p},\overline{q})$ or $lun(\overline{pq})$, is the set of points

$$\{z \in R^r : d(\overline{p},\overline{z}) < d(\overline{p}.\overline{q}) \text{ and } d(\overline{q},\overline{z}) < d(\overline{p},\overline{q})\}.$$

Here d denotes the Euclidean distance. In other words, $lun(\overline{p},\overline{q})$ is the interior of the region formed by the intersection of two r-dimensional hyperspheres

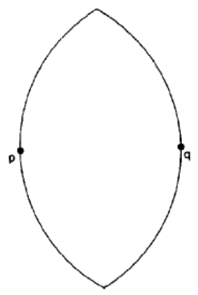


Figure 6.2: The lune of two points $\overline{p}$ and $\overline{q}$ is the region between the two arcs, not including the boundary.

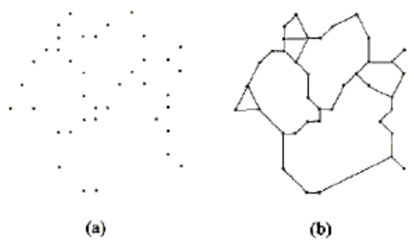


Figure 6.3: (a) A set of points in the plane [198] (b) RNG of the points in (a)

of radius $d(\overline{p}, \overline{q})$, one of the hyperspheres being centered at $\overline{p}$ and the other at $\overline{q}$. This is illustrated in Figure 6.2 which shows the lune of two points $\overline{p}$, $\overline{q}$ in the plane. If V is a set of n points in r-space, then the relative neighborhood graph of V (denoted $RNG(V)$) is the undirected graph with vertices V such that for each pair $\overline{p}, \overline{q} \in V$, $\overline{pq}$ is an edge of $RNG(V)$ if and only if $\text{lun}(\overline{p}, \overline{q}) \cap V = \emptyset$. Weight of a particular edge ($\overline{pq}$) is kept equal to $d(\overline{p}, \overline{q})$. Here $d(\overline{p}, \overline{q})$ is the Euclidean distance between the points $\overline{p}$ and $\overline{q}$.

Figure 6.3(a) demonstrates a set V of points in the plane. The corresponding RNG of this set of points V is shown in Figure 6.3(b). The RNG problem is: Given a set V, find $RNG(V)$.

Measuring the Connectivity Among a Set of Points

Here we propose a novel way of measuring the connectivity among a set of points using the above discussed RNG. The distance between a pair of points is measured in the following way.

- Construct the relative neighborhood graph of the whole data set.
- The distance between any two points, $\overline{x}$ and $\overline{y}$, denoted as $d_{short}(\overline{x}, \overline{y})$, is measured along the relative neighborhood graph. Find all possible paths among these two points along the RNG. Suppose there are total p paths between $\overline{x}$ and $\overline{y}$, 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 $\overline{x}$ and $\overline{y}$ is defined as follows:

$$d_{short}(\overline{x}, \overline{y}) = \min_{i=1}^{p} \max_{j=1}^{n_i} w(ed_j^i). \tag{6.1}$$

In order to improve the efficiency of computing d_{short}, we adopt the following pruning strategy. 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.

Definition of the Proposed Cluster Validity Index

The proposed 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,

$$\text{minindex} = argmin_{i=1}^{n_k} \frac{\sum_{j=1}^{n_k} d_e(\overline{x}^k_i, \overline{x}^k_j)}{n_k},$$

where n_k is the total number of points in the kth cluster and $\overline{x}^k_i$ 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:

$$Con = \frac{\sum_{i=1}^{K} \sum_{j=1}^{n_k} d_{short}(\overline{m}_i, \overline{x}^i_j)}{n \times \min_{i,j=1 \bigwedge i \neq j}^{K} d_{\text{S}hort}(\overline{m}_i, \overline{m}_j)},$$

where $d_{short}(\overline{m}_i, \overline{x}^i_j)$ is the shortest distance along the relative neighborhood graph between the two points $\overline{m}_i$ and $\overline{x}^i_j$, the jth point of the ith cluster. It is calculated using the procedure mentioned in Section 6.2.4. n denotes the total number of points present in the data set. Intuitively smaller values of *Con*-index corresponds 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 well-separated, this distance is the maximum and this 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 well-connected then the shortest distance between the medoid and any point of that particular cluster is small and thus numerator of the *Con*-index also takes a very small value. Thus *Con*-index obtains its minimum value when clusters are connected as well as separated too.

6.2.5 Euclidean Distance Based Cluster Validity Index: *I*-index

The third objective function used here is an Euclidean distance based cluster validity index, *I*-index [132]. It is defined as follows:

$$I(K) = (\frac{1}{K} \times \frac{E_1}{E_K} \times D_K)^p,$$

where K is the number of clusters. Here $E_K = \sum_{k=1}^{K} \sum_{j=1}^{n_k} d_e(\overline{c}_k, \overline{x}_j^k)$ and $D_K = \max_{i,j=1}^{K} d_e(\overline{c}_i, \overline{c}_j)$ where $\overline{c}_j$ denotes the center of the jth cluster and $\overline{x}_j^k$ denotes the jth point of the kth cluster. n_k is the total number of points present in the kth cluster. The value of K for which I-index takes its maximum value is considered as the appropriate number of clusters.

The index I is a composition of three factors, namely, $\frac{1}{K}$, $\frac{E_1}{E_K}$, and D_K. The first factor will try to reduce index I as K is increased. The second factor consists of the ratio of E_1, which is constant for a given data set, and E_K, which decreases with increase in K. Hence, because of this term, index I increases as E_K decreases. This, in turn, indicates that formation of more numbers of clusters, which are compact in nature, would be encouraged. Finally, the third factor, D_K (which measures the maximum separation between two clusters over all possible pairs of clusters), will increase with the value of K. However, note that this value is upper bounded by the maximum separation between two points in the data set. Thus, the three factors are found to compete with and balance each other critically. The power p is used to control the contrast between the different cluster configurations. Here, we have taken $p = 2$.

6.2.6 Sub-cluster Merging for Objective Function Calculation

Before computing the above mentioned three objective functions for each state, first the total $C \times K$ number of sub-clusters encoded in a particular state 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.,

$$distance_{short} = [d_{short}(\overline{c}_i, \overline{c}_j)]_{i,j=1\ldots C\times K}.$$

Thereafter single linkage clustering technique [67] 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 state. Thus the objective functions for a particular state 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 state. Here K denotes the number of clusters present in that particular state. These three objective functions are simultaneously optimized by using the simulated annealing based MOO algorithm, AMOSA.

6.2.7 Mutation Operation

A new state is generated from the current one by adopting one of the following three types of mutations.

1. Each cluster center encoded in a state is replaced with a random variable drawn from a Laplacian distribution, $p(\epsilon) \propto e^{-\frac{|\epsilon-\mu|}{\delta}}$, 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. A total of C sub-cluster centers are removed from the state, i.e., the total number of clusters present in that state is decreased by 1.

3. The total number of clusters present in that chromosome is increased by 1. C randomly chosen points from the data set are encoded as the new sub-cluster centers.

Any one of the above mentioned types of mutation is applied randomly on a particular state.

6.2.8 Selection of a Solution from the Archive

Here a solution is selected from the final archive using the procedure described in Section 3.9.1 of Chapter 3.

Table 6.1: Results on different data sets by *GenClustMOO*, MOCK, VGAPS, *GenClustPESA2*, VAMOSA clustering algorithms.

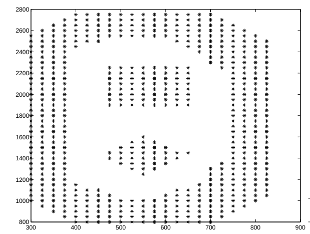
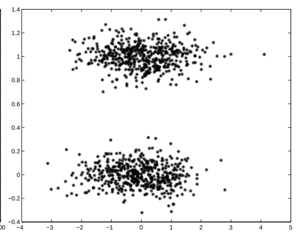
Data Set	*N*	*d*	*K*	*GenClustMOO*		MOCK		VGAPS		*GenClustPESA2*		VAMOSA	
				OC	MS	OC	MS	OC	MS	OC	MS	OC	MS
AD_10_2	500	2	10	**10**	**0.13**	6	1.01	7	0.84	11	0.32	10	0.43
Pat1	557	2	3	**3**	**0.00**	10	0.89	4	0.93	**3**	**0.00**	2	0.83
Long1	1000	2	2	**2**	**0.00**	2	0.00	3	1.00	**2**	**0.00**	8	0.73
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**
Newthyroid	215	5	3	**3**	**0.55**	2	0.82	3	0.58	9	0.85	5	0.57
LungCancer	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

6.3 Experimental Results

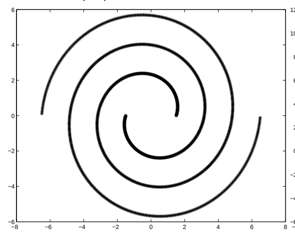
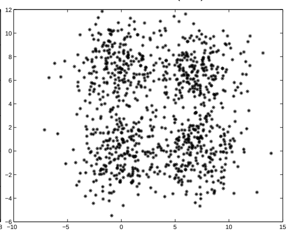
6.3.1 Data Sets Used

Seven artificial data sets and five real-life data sets are used for the experiment. The artificial data sets are *AD_10_2*, *Pat1*, *Long1*, *Spiral*, *Square4*, *Sizes5* and *Twenty*. Real-life data sets were obtained from [2]. These are *Iris*, *Cancer*, *Newthyroid*, *LungCancer* and *Glass*. A description of the data sets in terms of the number of points present, dimension of the data set, number of clusters is presented in Table 6.1.

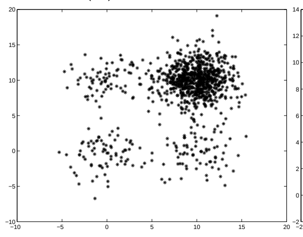
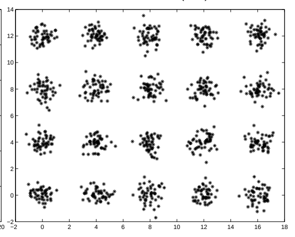
1. *AD_10_2*: This data set is described in Section 5.4.1 of Chapter 5.
2. *Pat1*: This data, used in Ref. [146], consists of 880 patterns . There are three non convex clusters present in this data set. This is shown in Figure 6.4(a).
3. *Long1*: This data set, used in Ref. [81], consists of 1000 data points distributed over 2 long clusters. This is shown in Figure 6.4(b).

(a) (b)

(c) (d)

(e) (f)

Figure 6.4: (a) *Pat1* (b) *Long1* (c) *Spiral* (d) *Square4* (e) *Sizes5* (f) *Twenty*

4. *Spiral*: This data set, used in Ref. [81], consists of 1000 data points distributed over 2 spiral clusters. This is shown in Figure 6.4(c).

5. *Square4*: This data set, used in Ref. [81], consists of 1000 data points distributed over four squared clusters. This is shown in Figure 6.4(d).

6. *Sizes5*: This data set, used in Ref. [81], consists of 1000 data points distributed over four clusters. The densities of these clusters are not uniform. This is shown in Figure 6.4(e).

7. *Twenty*: This data set, used in Ref. [81], consists of 1000 data points distributed over 20 small clusters. This is shown in Figure 6.4(f).

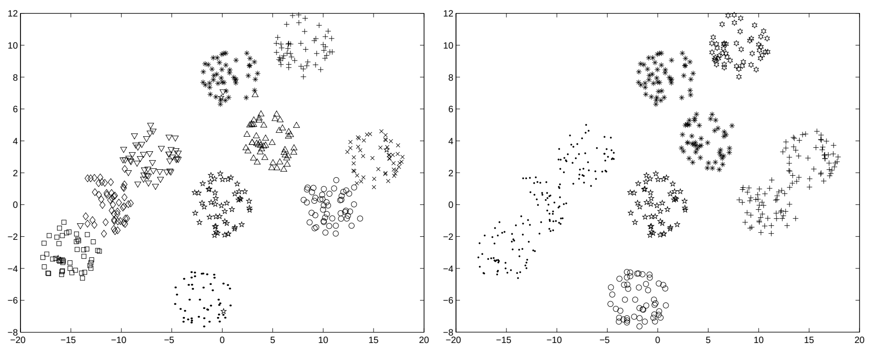
8. *Iris*: This data set is described in Section 3.8.1 of Chapter 3.

9. *Cancer*: This data set is described in Section 3.8.1 of Chapter 3.

10. *Newthyroid*: This data set is described in Section 3.8.1 of Chapter 3.

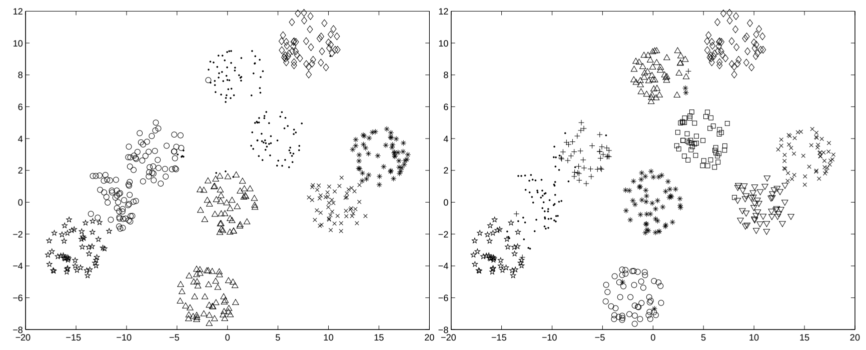
11. *LungCancer*: This data set is described in Section 3.8.1 of Chapter 3.

12. *Glass*: This is a glass identification data consisting of 214 instances having 9 features (an Id# feature has been removed). Criminological investigation inspires the study of the classification of the types of glass. At the scene of the crime, the glass left can be used as evidence, if it is correctly identified. There are 6 categories present in this data set.



(a) (b)

Figure 6.5: Automatically clustered *AD_10_2* after application of (a) *GenClustMOO* clustering technique for $K = 10$ (b) MOCK clustering technique for $K = 6$.



(a) (b)

Figure 6.6: Automatically clustered *AD_10_2* after application of (a) VGAPS clustering technique for $K = 7$ (b) *VAMOSA* clustering technique for $K = 10$.

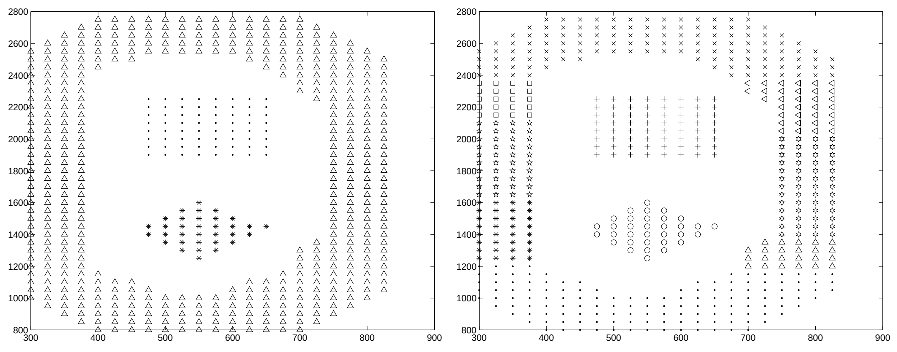

(a) (b)

Figure 6.7: Automatically clustered *Pat1* after application of (a) *GenClust-MOO* clustering technique for $K = 3$ (b) MOCK clustering technique for $K = 10$.

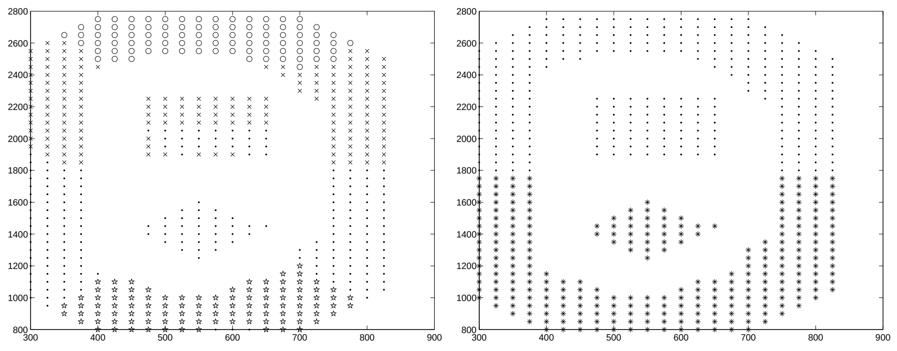
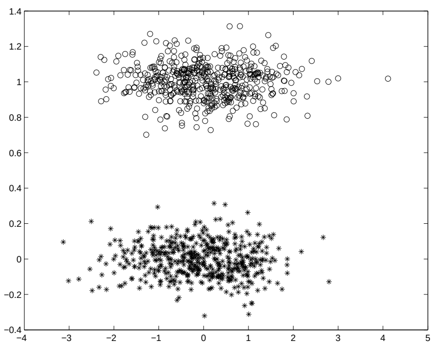

(a) (b)

Figure 6.8: Automatically clustered *Pat1* after application of (a) VGAPS clustering technique for $K = 4$ (b) *VAMOSA* clustering technique for $K = 2$.


(a)

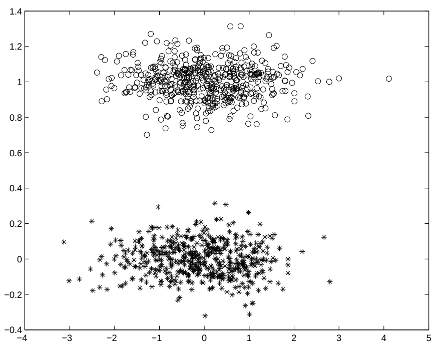
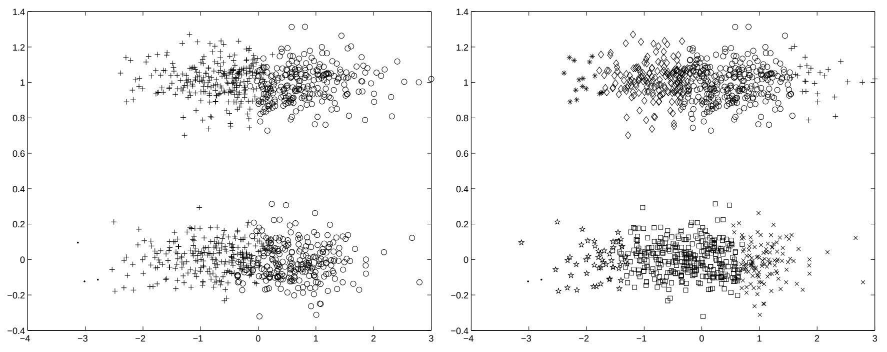
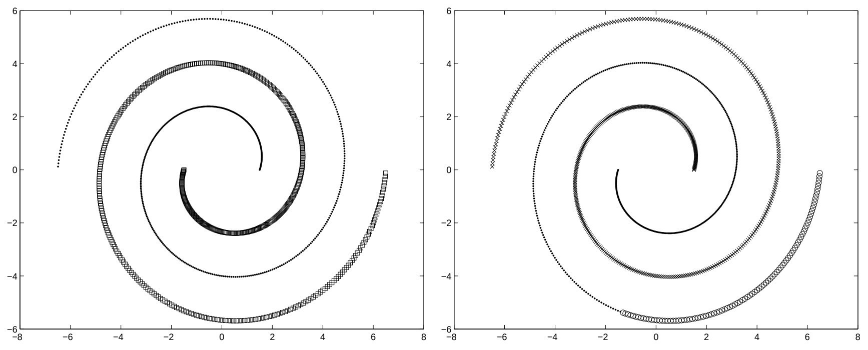

(b)

Figure 6.9: Automatically clustered *Long1* after application of (a) *GenClust-MOO* clustering technique for $K = 2$ (b) MOCK clustering technique for $K = 2$.



(a) (b)

Figure 6.10: Automatically clustered *Long1* after application of (a) VGAPS clustering technique for $K = 3$ (b) *VAMOSA* clustering technique for $K = 8$.

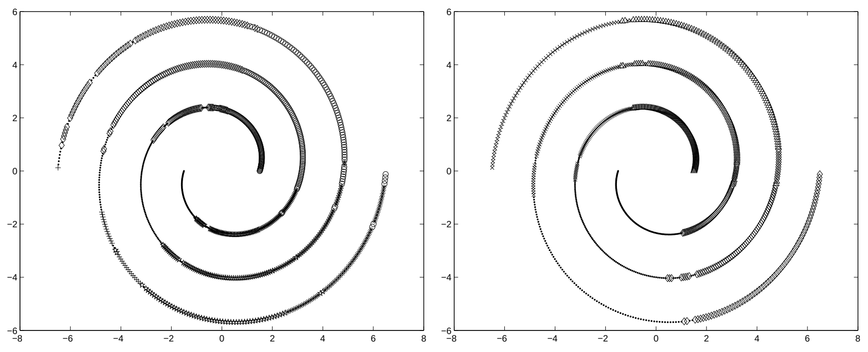


(a) (b)

Figure 6.11: Automatically clustered *Spiral* after application of (a) *GenClustMOO* clustering technique for $K = 2$ (b) MOCK clustering technique for $K = 3$.

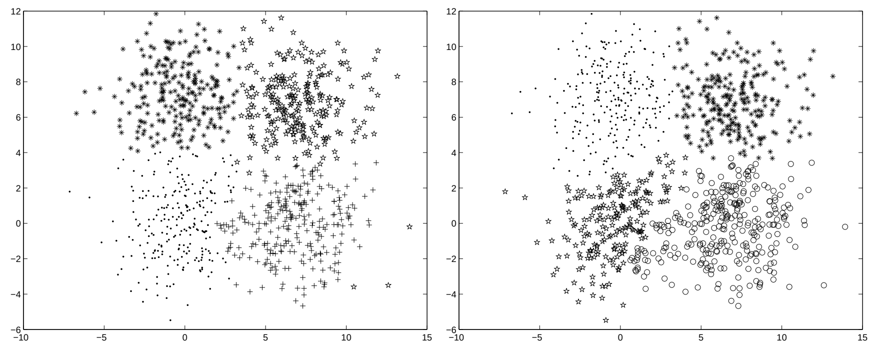
6.3.2 Discussion of Results

In *GenClustMOO*, the newly developed simulated annealing based MOO technique, AMOSA is used as the underlying optimization strategy. The parameters of the proposed *GenClustMOO* clustering technique are as follows: *SL*=100 *HL*=50, *iter*=50, *Tmax*=100, *Tmin*=0.00001 and cooling rate, $\alpha = 0.9$. *GenClustMOO* has been executed on all the data sets used in the previous chapters. For all the data sets having symmetrical shaped clusters



(a) (b)

Figure 6.12: Automatically clustered *Spiral* after application of (a) VGAPS clustering technique for $K = 6$ (b) VAMOSA clustering technique for $K = 4$.



(a) (b) (c)

Figure 6.13: Automatically clustered *Square4* after application of (a) *GenClustMOO* clustering technique for $K = 4$ (b) MOCK clustering technique for $K = 4$.

(e.g., *Mixed_3_2*, *Sym_3_2*) it performs similar to those of the VAMOSA and VGAPS. For *AD_5_2* data set it performs similar to 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 [81], is also executed on the above mentioned data sets. The source code for MOCK is obtained from [3] and the default parameter values are used. In MOCK the final best solution is selected by GAP-statistics [196]. The number of clusters automatically determined by

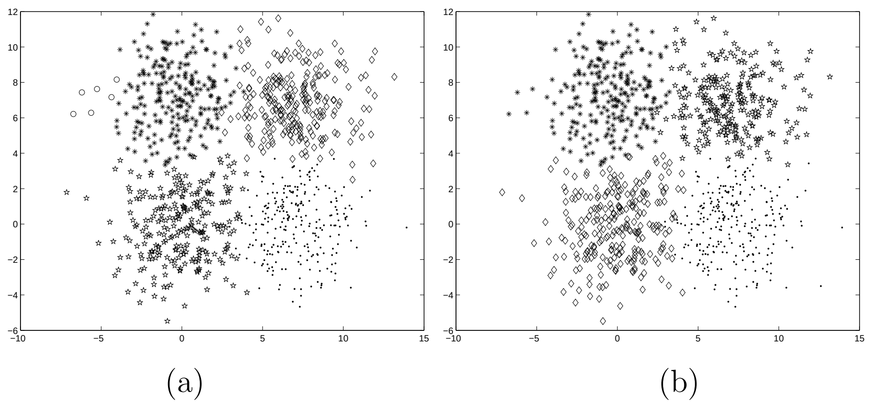


Figure 6.14: Automatically clustered *Square4* after application of (a) VGAPS clustering technique for $K = 5$ (b) VAMOSA clustering technique for $K = 4$.

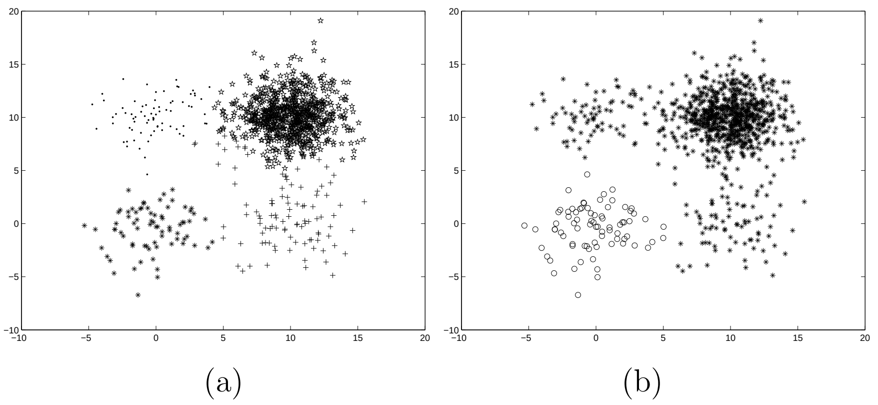


Figure 6.15: Automatically clustered *Sizes5* after application of (a) *GenClustMOO* clustering technique for $K = 4$ (b) MOCK clustering technique for $K = 2$.

the proposed *GenClustMOO* and MOCK clustering techniques for all the above mentioned data sets are shown in Table 6.1. This table also contains the *Minkowski Score* [98] (defined in Equation 3.22 of Chapter 3) values of the final partitionings identified by these two algorithms. Here we have reported the best *Minkowski Score* (MS) values obtained by the algorithms over five runs for all data sets. In [81] it has already been established that the performance of 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 omit-

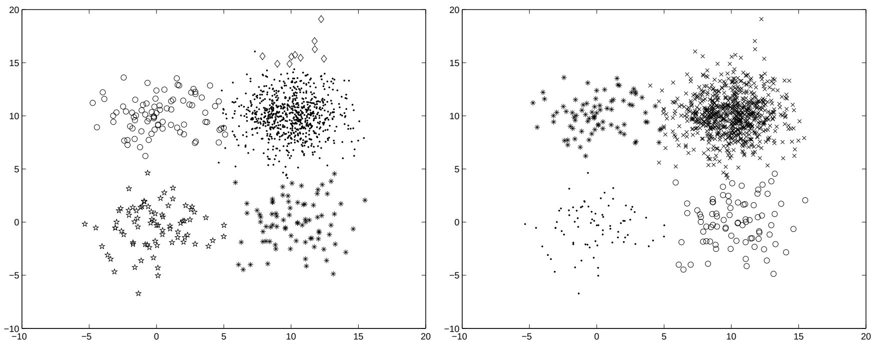


Figure 6.16: Automatically clustered *Sizes5* after application of (a) VGAPS clustering technique for $K = 5$ (b) VAMOSA clustering technique for $K = 4$.

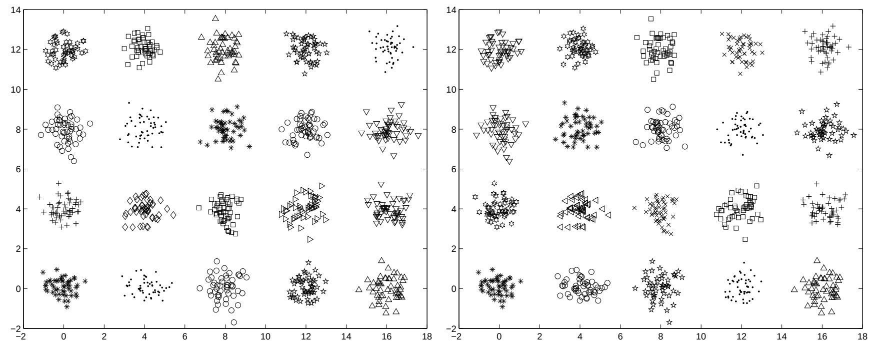


Figure 6.17: Automatically clustered *Twenty* after application of (a) *GenClustMOO* clustering technique for $K = 20$ (b) MOCK clustering technique for $K = 20$.

ted from this chapter. In order to show the effectiveness of AMOSA as the underlying optimization technique in *GenClustMOO*, we have also shown the results for all the data sets obtained by *GenClustPESA2* that has exactly the same approach of *GenClusMOO*, with the underlying MOO strategy replaced by PESA2 [50]. The number of clusters and the corresponding *Minkowski Score* values are reported in Table 6.1. The performance of *GenClustMOO* is also compared with those of VAMOSA clustering technique. The number of clusters obtained by this technique along with the corresponding *Minkowski Score* values are also reported in Table 6.1.

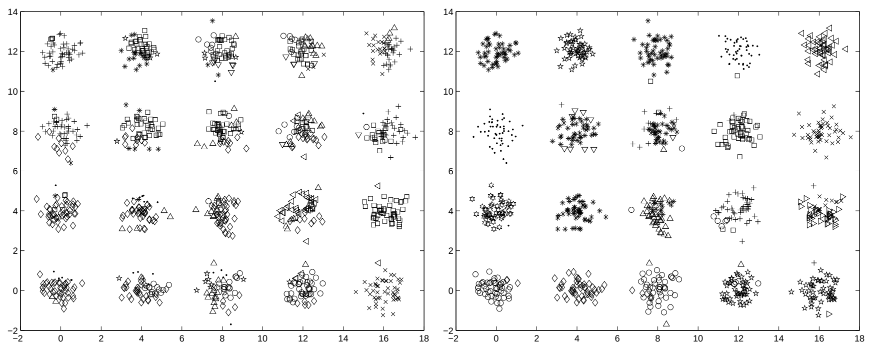

(a) (b)

Figure 6.18: Automatically clustered *Twenty* after application of (a) VGAPS clustering technique for $K = 20$ (b) VAMOSA clustering technique for $K = 20$.

In order to show that the proposed multiobjective clustering technique (*GenClustMOO*) performs better than a single objective version, VGAPS-clustering [26] is also executed on the above mentioned data sets used here for experiment. The parameter values of VGAPS-clustering technique are set as detailed in Section 5.3.2 of Chapter 5.

The clusters present in *AD_10_2* are hyperspherical in shape. The proposed *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 Figure 6.5(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 is shown in Figure 6.5(b). The MS value obtained by *GenClustMOO* is also less than that obtained by MOCK (refer to Table 6.1). VGAPS is not able to detect the proper partitioning for *AD_10_2* data set (corresponding partitioning is shown in Figure 6.6(a)). *GenClustPESA2* clustering technique fails to detect the proper number of partitions from *AD_10_2* data set. Partitioning obtained by VAMOSA for *AD_10_2* is shown in Figure 6.6(b).

The clusters present in *Pat1*, *Long1*, *Spiral*, *Sizes5*, *Square4* and *Twenty* are well-separated having any shape, size or convexity. These data sets are used to show the performance of the algorithms for detecting some well-separated clusters. Our proposed *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 Figures 6.7(a), 6.9(a), 6.11(a), 6.13(a), 6.15(a), and 6.17(a), respectively. MOCK is able to detect the appropriate number of partitions from three out of six data sets. The partitionings are shown in Figures 6.7(b), 6.9(b), 6.11(b), 6.13(b), 6.15(b) and 6.17(b), respectively. VGAPS-clustering is able to detect the the proper partitioning and the proper number of clusters from only one out of six data sets. The partitionings are shown in Figures 6.8(a), 6.10(a), 6.12(a), 6.14(a), 6.16(a) and 6.18(a), respectively. *GenClustPESA2* clustering technique performs poorly for *Sizes5* and *Twenty* data sets (refer to Table 6.1). For other data sets of this group, *GenClustPESA2* and *GenClustMOO* clustering techniques perform similarly. The partitionings identified by VAMOSA for all these six data sets are shown in Figures 6.8(b), 6.10(b), 6.12(b), 6.14(b), 6.16(b) and 6.18(b), respectively. Table 6.1 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 non-symmetrical shapes but well-separated structures.

For the real-life data sets no visualization is possible as these are higher dimensional data sets. For *Iris* data set both *GenClustMOO* and VGAPS clustering techniques are able to detect the appropriate number of partitions. But the *Minkowski Score* value attained by *GenClustMOO* clustering technique is slightly smaller than that obtained by VGAPS (refer to Table 6.1). MOCK and VAMOSA automatically identify $K = 2$ number of clusters for this data set which is also often obtained for many other methods of *Iris* [131]. For *Cancer* data set all the five algorithms are able to detect the appropriate number of clusters ($K = 2$) for this data set. But the MS value obtained by *GenClustMOO* is smaller than those corresponding to MOCK, *GenClustPESA2*, and VGAPS. This in turn indicates that the proposed algorithm provides more better partitioning for this data set than MOCK, *GenClustPESA2* and VGAPS. For *Newthyroid* data set, proposed *GenClustMOO* is able to detect the appropriate number of partitions from this data set but

MOCK, *GenClustPESA2* and VAMOSA fail to do so. The *Minkowski Score* value of the partitioning identified by *GenClustMOO* for this data set is also much smaller than those obtained by MOCK, *GenClustPESA2*, VAMOSA and VGAPS (refer to Table 6.1). For *LungCancer* data set, only VAMOSA is able to detect the appropriate number of partitions from this data set. But the MS value attained by *GenClustMOO* is the smallest among all other algorithms. For *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 6.1).

Summary of Results

Results on a wide variety of data sets show that the proposed *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 capable to identify various symmetrical shaped clusters (hyperspheres, linear, ellipsoidal, 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 capable to detect partitioning from real-life data sets of varying characteristics. The results on seven artificial and five real-life data sets establish the fact that *GenClustMOO* is well-suited to detect clusters of 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 non-overlapping. The proposed *GenClustMOO* clustering technique is able to find out the proper clustering automatically where MOCK succeeds while VGAPS fails as well as where VGAPS succeeds while MOCK fails. Experimental results also show that VAMOSA clustering technique is only able to detect the appropriate partitioning automatically from data sets having symmetrical shaped clusters. 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 have also compared the effectiveness of the underlying multiobjective optimization techniques, AMOSA and PESA2, in the proposed clustering algorithm, *GenClustMOO*. *GenClustPESA2* clustering technique, utilizing PESA2 [50] as the underlying optimization technique in *GenClust-MOO* framework, performs similarly as *GenClustMOO* clustering technique using AMOSA for data sets with equisized, equi-density small number of clusters.

The improved performance of *GenClustMOO* can be attributed to the following facts. Use of multi-center 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. Use of relative neighborhood graph to compute the *Con*-index enables it to detect any shaped clusters as long as they are well-separated. AMOSA, the underlying optimization technique makes it capable of optimizing three cluster validity indices efficiently.

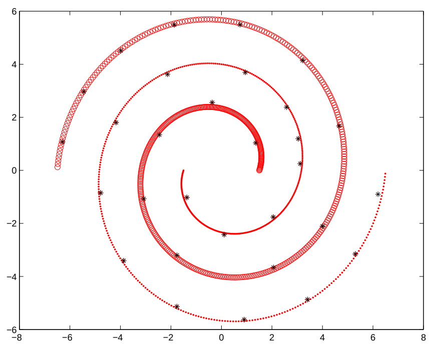


Figure 6.19: Automatically clustered *Spiral* after application of *GenClust-MOOV* where 2 clusters are detected. Number of sub-cluster centers per cluster are 14 and 16, respectively.

6.4 Varying the Number of Sub-clusters per Cluster

In the previous section, the number of sub-clusters for each cluster was kept fixed apriori in order to keep the approach simple. Evidently, this number needs to be varied depending on the cluster type. An initial attempt in this direction is reported in this section. Here the number of sub-clusters is varied over a range. The proposed *GenClustMOOV* (generalized MOO clustering technique with variable number of sub-cluster centers per cluster) is capable of determining the appropriate number of sub-cluster centers per cluster.

Suppose a particular state encodes the centers of K clusters. Initially each cluster i is represented by C_i number of sub-cluster centers. Here C_i varies within a range 2 to C^{max}. These C_i number of sub-centers are randomly selected points from each cluster. Therefore $\sum_{i=1}^{K} C_i$ number of sub-centers are encoded in that particular state. If the data set is of dimension d, then the length of the state will be $\sum_{i=1}^{K}(C_i \times d)$. Suppose a particular state contains $K = 2$ clusters. The first cluster is divided into 10 smaller sub-clusters and the second cluster is divided into 8 smaller sub-clusters, i.e., here $C_1 = 10$, $C_2 = 8$. Let the dimension (d) of the data set be 2. Suppose the center of the jth sub-cluster of the ith cluster is denoted by $\overline{c}_j^i = (c1_j^i, c2_j^i)$. Then this state will look like: $< c1_1^1, c2_1^1, c1_2^1, c2_2^1, \ldots, c1_{10}^1, c2_{10}^1, c1_1^2, c2_1^2, \ldots, c1_8^2, c2_8^2 >$.

Mutation operator has been modified accordingly. Results reveal that number of sub-cluster centers per cluster depends on the size and shape of the clusters. Experiments have been carried out for all the data sets used earlier in this chapter. Results show that for *Pat1* data set, the number of sub-clusters for the three clusters are 11, 5 and 3, respectively. Similarly for *Spiral*, *Sizes5*, *Square4* data sets number of sub-centers per clusters are (14, 16), (10, 2, 2, 2) and (2, 2, 3, 4), respectively. The partitioning obtained by *GenClustMOOV* clustering technique for *Spiral* data set is shown in Figure 6.19. The sub-centers automatically detected by *GenClustMOOV* is also shown in this figure. This work is just in its initial stage. Future works include detailed evaluations and more extensive analysis of *GenClustMOOV*.

6.5 Discussion and Conclusions

In this chapter, a new multiobjective clustering technique is proposed. This uses a newly developed simulated annealing based multiobjective optimization technique, AMOSA, as the underlying optimization strategy. Center based encoding is used. Multiple cluster centers are used to encode a particular cluster. Three cluster validity indices, an Euclidean distance based cluster validity index, a point symmetry distance based cluster validity index, and a connectivity based cluster validity index are optimized simultaneously. Relative neighborhood graph [198] is utilized to compute the connectivity index. The performance of the proposed algorithm named *GenClustMOO* is compared with the existing multiobjective clustering techniques, MOCK and VAMOSA, one single objective clustering technique, VGAPS, for several data sets having different characteristics. Results show that the proposed technique is well-suited to detect the appropriate partitioning from data sets having either the point symmetric clusters or well-separated clusters. 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, PESA2.

Much further work is needed to investigate the utility of having different and many more objectives, and to test the approach still more extensively. Extensive results have to be taken varying the number of sub-centers per cluster. Selecting the best solution(s) from the Pareto optimal front is an important problem in multiobjective clustering. One method of selecting a single solution from the Pareto optimal front is used here. But this method *apriori* assumes that the labeling of a small set of points is known beforehand. Thus some new methods to choose the best solution from the Pareto optimal front have to be developed. A recently proposed approach of combining the Pareto optimal solutions proposed in [135] may be used in future.

Chapter 7

Conclusions and Scope for Further Research

7.1 Conclusions

In this thesis some single and multiobjective clustering techniques which exploit the property of symmetry present in the clusters are proposed. In this regard a new symmetry based distance measure and a validity index based on it are proposed. For single objective clustering techniques search and optimization capabilities of genetic algorithms are used to obtain the appropriate partitioning from a data set. In order to solve the multiobjective clustering problem a new multiobjective optimization technique based on simulated annealing is first developed in the present thesis.

Many real-life problems involve the optimization of several, often conflicting, objectives simultaneously. Chapter 2 first discusses some existing single and multiobjective optimization techniques. Thereafter a simulated annealing based multiobjective optimization algorithm, archived multiobjective simulated annealing based technique (AMOSA) [27] has been proposed. In contrast to most other MOO algorithms, AMOSA [27] 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) [61] and PAES [110] for a suite of seven 2-objective test problems having different complexity levels. In a part of the investigation, comparison of the real-coded version of the proposed algorithm is conducted with a very recent multiobjective simulated annealing algorithm, MOSA [181], and the real-coded NSGA-II for six 3-objective test problems. Real-coded AMOSA is also compared with the real-coded NSGA-II for some 4, 5, 10 and 15 objective test problems. Several different comparison measures like *Convergence*, *Purity*, *MinimalSpacing*, and *Spacing*, and the time taken are used for the purpose of comparison. In this regard, a measure called *displacement* has also been used that 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 has been performed. It has been found that its complexity is more than that of PAES but smaller than that of NSGA-II.

It is seen from the given results that the performance of the proposed 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 like 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 [61] do not work well on many-objective optimization problems as pointed out in some recent studies [91], [93]. An interesting feature of AMOSA, as in other versions of multiobjective SA algorithms, is that it has a non-zero probability of allowing a dominated solution to be chosen as the current solution in favour 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 in other MOO algorithms in order to improve their performance.

Chapter 3 deals with the problem of clustering that is an important analysis tool in data-mining. The property of symmetry, commonly observed in nature, is exploited in this regard. Here, a symmetry based similarity measurement [24] is first defined. 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 clustering technique named GAPS (genetic algorithm with point symmetry based clustering technique) [24] which is able to detect any types of clusters possessing the property of point symmetry. The major advantages of GAPS are as follows. In contrast to K-means, use of GA enables the algorithm to come out of local optima, making it less sensitive to the choice of the initial cluster centers. Again, the proposed GAPS is able to detect clusters that may be of widely varying sizes, where K-means fails. Such situations may arise in several real-life domains, e.g., medical images, satellite images, fraud detection. Incorporation of the proposed PS-distance enables GAPS to detect symmetric clusters, both convex and non-convex, even if the clusters are symmetrical with respect to some intermediate point. This is in contrast to SBKM and Mod-SBKM, which fail in such situations. Moreover, use of Kd-tree makes the computation of the point symmetry distance significantly faster than both SBKM and Mod-SBKM. The global convergence proof of GAPS-clustering technique has also been established in this chapter.

Experimental results on four artificial data sets and four real-life data sets demonstrate the superiority of GAPS as compared to SBKM [187], Mod-SBKM [43], K-means algorithm, genetic algorithm based K-means clustering technique (GAK-means), average linkage clustering technique (AL) and Expectation Maximization clustering technique (EM). GAPS is found to provide satisfactory performance both where K-means fails but SBKM succeeds as well as where SBKM fails but K-means succeeds. Results on *Bensaid_3_2* demonstrate that GAPS is able to detect symmetric clusters of any size where most of the other algorithms fail. Comparison of the obtained results by different algorithms are performed by statistical tests like ANOVA.

In the later part of this chapter, the problem of clustering a data set is posed as one of multiobjective optimization. The aforementioned simulated annealing based multiobjective optimization technique, AMOSA, has been used as the underlying optimization technique. Two cluster quality measures, total Euclidean compactness and total symmetrical compactness are optimized simultaneously using the search capability of AMOSA. This enables the algorithm to detect clusters that are well characterized by Euclidean compactness as also those which are not compact in the conventional sense, but are symmetric about a point. The performance of MOPS is compared with GAPS in order to establish its effectiveness.

The newly proposed symmetry based distance is then used to develop a cluster validity index called *Sym*-index [162][168] in the next chapter (Chapter 4). Experimental results show that *Sym*-index is not only able to find the proper number of clusters from a given data set (model order) but is also able to detect the proper clustering algorithm (or, the appropriate model) suitable for that data set. 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 newly proposed *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 *Sym*-index is demonstrated for four artificially generated and three real life data sets. GAPS, a newly proposed symmetry distance based genetic clustering technique, 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 newly proposed *Sym*-index as compared to the four existing validity indices, namely, PS index, *I*-index, CS-index and XB-index for the data sets.

The concept of point symmetry is thereafter introduced in eight well-known cluster validity indices [172]. Results show that incorporation of point symmetry distance in the definitions of existing eight cluster validity indices make 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. Thereafter, the application of the newly proposed symmetry based cluster validity index, *Sym*-index and GAPS-clustering technique are described for image segmentation [169].

In the earlier chapters, the number of clusters was assumed to be known *apriori*. However, in many real-life situations this value may not be known. In order to overcome this limitation, a variable string length GA based clustering technique [26], VGAPS-clustering, has been then proposed in chapter 5. The newly proposed 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 the proposed VGAPS-clustering is also established. The effectiveness of the VGAPS-clustering, as compared to two recently proposed automatic clustering techniques, namely, GCUK-clustering and HNGA-clustering, is demonstrated on five artificially generated and three real-life data sets of different characteristics. Results on the eight data sets establish the fact that VGAPS-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, or overlap or size, as long as they possess the property of symmetry.

The corresponding MO version of VGAPS clustering technique [171] utilizing the aforementioned AMOSA algorithm is also developed in this chapter. It simultaneously optimizes *Sym*-index and an Euclidean distance based cluster validity index, XB-index [206]. The effectiveness of the proposed clustering technique (VAMOSA) in detecting the proper number of partitions and the proper partitioning is shown for four artificial and four real-life data sets and the results are compared with those obtained by another recent MO clustering technique, MOCK [81] and two single objective automatic genetic clustering techniques- GCUK clustering optimizing the XB-index [16] and VGAPS clustering [26].

Finally Chapter 6 deals with the development of a multi-center based multiobjective clustering approach [25] which can automatically determine any type of clusters having either symmetrical (may be convex or non-convex and/or overlapping or non-overlapping) or non-overlapping but connected shapes from a data set. Here each cluster is divided into several nonoverlapping small hyperspherical clusters and the centers of these sub clusters are encoded in the states. For the assignment of points all these sub clusters are considered individually. However, during fitness computation, the sub clusters corresponding to each cluster are identified based on a proximity measure and these are merged together. Three cluster validity indices, an Euclidean distance based index, a point symmetry distance based index, and a connectivity based index are optimized simultaneously. Relative neighborhood graph is utilized to compute the connectivity index. The performance of the proposed algorithm named *GenClustMOO* is compared with the existing multiobjective clustering technique, MOCK, another multiobjective clustering technique developed in this thesis, VAMOSA and a 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 to another evolutionary MO algorithm, PESA2. Results show that the proposed technique is well-suited to automatically detect the appropriate partitioning from data sets having either the point symmetric clusters or well-separated clusters.

For the data sets having symmetrical overlapping clusters (e.g, *Sym_3_2*, *Mixed_3_2*), none of the solutions in the final non-dominated set of MOCK correspond to the best ones. However for data sets like *AD_10_2* and *AD_5_2*, the best solutions are present in the final non-dominated set of MOCK, though the GAP statistic is not able to identify them. If our proposed semi-supervised approach of selecting the best solution is adopted in MOCK, then these solutions may be identified as discussed in Section 5.4.2 of Chapter 5.

The papers corresponding to these works are either published or accepted or under revision or communicated.

7.2 Scope for Further Research

Though we have made all possible attempts to make the work as complete as possible, as in any research there are still several areas where a lot of potential work may be carried out. We highlight some such future research directions in this section.

There are several ways in which the proposed AMOSA algorithm [27] may be extended in future. The main time consuming procedure in AMOSA is the clustering part. Some other more efficient clustering techniques or even the PAES [110] like grid based strategy, can be incorporated for improving its performance.

In [181][182] an unconstrained archive is maintained. Note that theoretically, the number of Pareto optimal solutions can be infinite. Since the ultimate purpose of an 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. (It has been mentioned in [70] that it is sometimes desirable not to truncate the archive as it was found in an earlier study [210] that almost 40% of the solutions provided by an algorithm with truncation of archive got dominated by the solutions provided by an algorithm without archive truncation. However the experiments we conducted did not adequately justify this finding. This is explained more in detail in Chapter 2.) Though maintaining unconstrained archives as in [181][182] 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 the proposed AMOSA. Clustering has also been used earlier in [212]. It will be interesting to investigate, in future, the effect of incorporating the concept of unconstrained archive as in [181][182] in the proposed AMOSA.

An algorithm, unless analyzed theoretically, is good for only the experiments conducted. Thus a theoretical analysis of AMOSA needs to be performed in future in order to study its convergence properties. We are currently trying to develop a proof for the convergence of AMOSA in the lines of the proof for single objective SA given by Geman and Geman [77]. As has been mentioned in [192], there are no firm guidelines for choosing the parameters in an SA-based algorithm. Thus, an extensive sensitivity study of AMOSA with respect to its different parameters, notably the annealing schedule, needs to be performed. Finally, application of AMOSA to several real-life domains e.g., VLSI system design [175], remote sensing imagery [17], Bioinformatics [140], needs to be demonstrated.

GAPS clustering algorithm [24] based on the proposed d_{ps} distance (described in Chapter 3) does not require a point and its symmetrical point (or its nearest neighbor) to belong to the same cluster. This may prove to be a disadvantage as was evident for *AD_5_2* where the central cluster got overestimated. Further research needs to be carried out to rectify this limitation. We are currently working in this direction. A detailed sensitivity study of different parameters of GAPS constitutes an important direction of future research. In the newly proposed symmetry based cluster validity indices in Chapter 4 we have retained whatever normalization was used in the original version. However, appropriate normalization is important and a study in this direction needs to be conducted in future.

As a part of future work, the utility of the proposed d_{ps} distance needs to be studied on more real life problems such as brain tissue classification from magnetic resonance images (MRI) which is of great importance for research and clinical study of much neurological pathology. Since some of the tissue types in the brain constitute small clusters while some others may constitute much larger clusters, application of the proposed schemes appears to be appropriate for such applications. Some works have been initiated for automatic segmentation of brain MR images. VGAPS and VAMOSA clustering techniques have been used to automatically partition these MR normal and MS lesion magnetic resonance brain images [163][173]. The obtained segmentation results by VGAPS and VAMOSA clustering techniques need to be compared with those obtained by other well-known segmentation algorithms. As a part of the future work, some spatial information inherently available in the pixel locations will also be incorporated while segmenting the brain MR images.

The current work concentrates only on a particular form of symmetry viz., point-based symmetry. Other forms of symmetry e.g., line-based symmetry, polynomial symmetry etc. may exist in the clusters. Techniques for detecting clusters with these higher forms of symmetry are also some interesting areas of future research. Some work in this area has already been initiated [161][164][166] though a lot more remains to be done.

It is important to investigate systematically the different objectives that are to be optimized in MOPS, VAMOSA and finally in the *GenClustMOO* clustering techniques. Selecting the best solution(s) from the Pareto optimal front is an important problem in multiobjective clustering. In MOO, the algorithms produce a large number of non-dominated solutions on the final Pareto optimal front. 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. Thus selecting the best solution(s) from the Pareto optimal front or combining them into one is an important problem in multiobjective clustering. In this regard, a new semi-supervised method is proposed to select the best solution in MOPS, VAMOSA and *GenClustMOO* clustering techniques. Another approach recently proposed in [135] integrates the learning capability of a support vector machine based classifier. Thus several alternate approaches need to be developed and tested in this regard.

GenClustMOO clustering technique has been extended to automatically evolve variable number of sub-cluster centers per cluster. This work has just been initiated. Extensive comparison results have to be taken in future.

Most of the clustering techniques developed in this thesis are only able to detect symmetrical shaped clusters. Thus these will fail for non-symmetrical shaped clusters. The tolerance of these approaches have to be studied on some more data sets varying the amount of symmetry within data sets. Changing the amount of symmetry within data sets in a parametric way is also another important future research work.

The use of symmetry based distance and AMOSA for solving partitioning problems in several different domains have to be investigated in future. In VLSI, the problem of equi-partitioning a set of flipflops in order to design the clock trees efficiently is a very important problem. This problem is posed as one of multiobjective optimization in [174][175]. A popular MOO technique, NSGA-II has been used to solve this problem. AMOSA can be used as an alternative MOO technique for solving these problems. The newly developed point symmetry based distance, d_{ps}, can also be used for segmentation purpose.

Developing a supervised classification technique based on point symmetry is another area of future research. Some work has been initiated in this direction [167].

Developing some semi-supervised clustering techniques based on point symmetry based distance, d_{ps}, is another important direction of future work. Some works have also been initialized in this direction [170].

Bibliography

[1] http://www.dcs.ex.ac.uk/people/kismith/mosa/results/tec/.

[2] http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[3] http://dbkgroup.org/handl/mock/.

[4] M. R. Anderberg. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[5] T. W. Anderson and S. L. Scolve. *Introduction to the Statistical Analysis of Data*. Houghton Mifflin, 1978.

[6] S. Asharaf and M. N. Murty. An adaptive rough fuzzy single pass algorithm for clustering large data sets. *Pattern Recognition*, 36(12):3015–3018, 2003.

[7] S. Asharaf and M. N. Murty. Scalable non-linear support vector machine using hierarchical clustering. In *ICPR (1)*, pages 908–911, 2006.

[8] S. Asharaf, M. N. Murty, and S. K. Shevade. Cluster based core vector machine. In *ICDM*, pages 1038–1042, 2006.

[9] S. Asharaf, S. K. Shevade, and M. N. Murty. Rough support vector clustering. *Pattern Recognition*, 38(10):1779–1783, 2005.

[10] F. Attneave. Symmetry information and memory for pattern. *Am. J. Psychology*, 68:209–222, 1995.

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

[12] S. Bandyopadhyay. An automatic shape independent clustering technique. *Pattern Recognition*, 37(1):33–45, 2004.

[13] S. Bandyopadhyay. Simulated annealing using reversible jump markov chain monte carlo algorithm for fuzzy clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):479–490, 2005.

[14] S. Bandyopadhyay and U. Maulik. Non-parametric genetic clustering : Comparison of validity indices. *IEEE Transactions on Systems, Man and Cybernetics Part-C*, 31(1):120–125, 2001.

[15] S. Bandyopadhyay and U. Maulik. An evolutionary technique based on K-means algorithm for optimal clustering in R^N. *Information Sciences*, 146(1-4):221–237, 2002.

[16] S. Bandyopadhyay and U. Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6):1197–1208, 2002.

[17] S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions Geoscience and Remote Sensing*, 45(5):1506–1511, 2007.

[18] S. Bandyopadhyay, U. Maulik, and M. K. Pakhira. Clustering using simulated annealing with probabilistic redistribution. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(2):269–285, 2001.

[19] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal. Pattern classification using genetic algorithms. *Pattern Recognition Letters*, 16(8):801–808, 1995.

[20] S. Bandyopadhyay and S. K. Pal. *Classification and Learning Using Genetic Algorithms Applications in Bioinformatics and Web Intelligence.* Springer-Verlag, Hiedelberg, Germany, 2007.

[21] S. Bandyopadhyay and S. K. Pal. *Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence.* Springer, Heidelberg, 2007.

[22] S. Bandyopadhyay, S. K. Pal, and B. Aruna. Multi-objective GAs, quantitative indices and pattern classification. *IEEE Transactions on Systems, Man and Cybernetics-Part B*, 34(5):2088–2099, 2004.

[23] S. Bandyopadhyay, S. K. Pal, and C. A. Murthy. Simulated annealing based pattern classification. *Information Sciences*, 109(1-4):165–184, 1998.

[24] S. Bandyopadhyay and S. Saha. GAPS: A clustering method using a new point symmetry based distance measure. *Pattern Recognition*, 40(12):3430–3451, 2007.

[25] S. Bandyopadhyay and S. Saha. A generalized automatic clustering algorithm in a multiobjective framework. *IEEE Transactions on Knowledge and Data Engineering (communicated)*, March, 2009.

[26] S. Bandyopadhyay and S. Saha. A point symmetry based clustering technique for automatic evolution of clusters. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1–17, November, 2008.

[27] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing based multi-objective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283, June, 2008.

[28] A. Bargiela, W. Pedrycz, and K. Hirota. Granular prototyping in fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 12(5):697–709, 2004.

[29] A. M. Bensaid, L. O. Hall, J. C. Bezdek, L. P. Clarke, M. L. Silbiger, J. A. Arrington, and R. F. Murtagh. Validity-guided (re)clustering with applications to image segmentation. *IEEE Transactions on Fuzzy Systems*, 4(2):112–123, 1996.

[30] J. L. Bentley, B. W. Weide, and A. C. Yao. Optimal expected-time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, 6(4):563–580, 1980.

[31] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum, New York, 1981.

[32] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics*, 28(3):301–315, 1998.

[33] S. M. Bhandarkar and H. Zhang. Image segmentation using evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 3(1):1–21, 1999.

[34] J. N. Bhuyan, V. V. Raghavan, and V. K. Elayavalli. Genetic algorithm for clustering with an ordered representation. In *Proc. Int. Conf. on Genetic Algorithm '91*, pages 408–415, San Mateo, CA, 1991. Morgan Kaufman.

[35] A. Bouchachia and W. Pedrycz. Data clustering with partial supervision. *Data Mining & Knowledge Discovery*, 12(1):47–78, 2006.

[36] A. Bouchachia and W. Pedrycz. Enhancement of fuzzy clustering by mechanisms of partial supervision. *Fuzzy Sets and Systems*, 157(13):1733–1759, 2006.

[37] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling EM (expectation maximization) clustering to large databases. Technical report, Microsoft Research Center, 1998.

[38] R. B. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Comm. in Stat.*, 3:1–27, 1974.

[39] R. Caves, S. Quegan, and R. White. Quantitative comparison of the performance of SAR segmentation algorithms. *IEEE Transactions on Image Processing*, 7(11):1534–1546, 1998.

[40] D. Charalampidis. A modified K-means algorithm for circular invariant clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1856–1865, December, 2005.

[41] A. Chipperfield, J. Whidborne, and P. Fleming. Evolutionary algorithms and simulated annealing for MCDM. In *Multicriteria Decision*

Making -Advances in MCDM Models, Algorithms, Theory and Applications, pages 16.1–16.32. Kluwer Acedemic Publishing, Boston, Massachusetts, 1999.

[42] J.-N. Choi, S.-K. Oh, and W. Pedrycz. Structural and parametric design of fuzzy inference systems using hierarchical fair competition-based parallel genetic algorithms and information granulation. *Int. J. Approx. Reasoning*, 49(3):631–648, 2008.

[43] C.-H. Chou, M.-C. Su, and E. Lai. Symmetry as a new measure for cluster validity. In *2nd WSEAS Int. Conf. on Scientific Computation and Soft Computing*, pages 209–213. Crete, Greece, 2002.

[44] C.-H. Chou, M.-C. Su, and E. Lai. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7(2):205–220, 2004.

[45] K.-L. Chung and J.-S. Lin. Faster and more robust point symmetry-based K-means algorithm. *Pattern Recognition*, 40(2):410–422, 2007.

[46] K.-L. Chung and K.-S. Lin. An efficient line symmetry-based K-means algorithm. *Pattern Recognition Letters*, 27(7):765–772, 2006.

[47] C. A. Coello Coello, D. Van Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Boston: Kluwer Academic Publishers, 2002.

[48] C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

[49] R. M. Cole. Clustering with genetic algorithms. Master's thesis, Department of Computer Science, University of Western Australia, Australia, 1998.

[50] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*

(GECCO-2001), pages 283–290, San Francisco, California, USA, 2001. Morgan Kaufmann.

[51] M. C. Cowgill, R. J. Harvey, and L. T. Watson. A genetic algorithm approach to cluster analysis. *Computational Mathematics and its Application*, 37(7):99–108, 1999.

[52] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998.

[53] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[54] R. N. Dave. Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. *Intell. Robots Compt. Vision VIII*, 1192:600–611, 1989.

[55] R. N. Dave and K. Bhaswan. Adaptive fuzzy c-shells clustering and detection of ellipses. *IEEE Transactions on Neural Networks*, 3(5):643–662, September 1992.

[56] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.

[57] L. Davis, editor. *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann Publishers Inc., Los Altos, California, 1987.

[58] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[59] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[60] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms.* John Wiley and Sons, Ltd, England, 2001.

[61] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[62] K. DeJong. Learning with genetic algorithms: an overview. *Mach. Lang.*, 3(2-3):121–138, 1988.

[63] R. C. Dubes and A. K. Jain. Clustering techniques : The user's dilemma. *Pattern Recognition*, 8(4):247–260, 1976.

[64] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[65] P. Engrand. A multi-objective approach based on simulated annealing and its application to nuclear fuel management. In *5th International Conference on Nuclear Engineering*, pages 416–423. Nice, France, 1997.

[66] V. Estivill-Castro and A. T. Murray. Spatial clustering for data mining with genetic algorithms. In *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems*, pages 317–323. 1997.

[67] B. S. Everitt. *Cluster Analysis*. Halsted Press, third edition, 1993.

[68] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. London: Arnold, 2001.

[69] E. Falkenauer. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[70] J. Fieldsend, R. Everson, and S. Singh. Using unconstrained elite archives for multi-objective optimisation. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, 2003.

[71] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 3:179–188, 1936.

[72] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[73] P. Fränti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen. Genetic algorithms for large scale clustering problem. *Comput. J*, 40:547–554, 1997.

[74] J. H. Friedman, J. L. Bently, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[75] Y. Fukuyama and M. Sugeno. A new method of choosing the number of clusters for the fuzzy c-means method. In *Proc. of the fifth Fuzzy Systems Symposium*, pages 247–250. 1989.

[76] I. Gath and A. B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781, 1989.

[77] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[78] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, New York, 1989.

[79] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16:122–128, 1986.

[80] D. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *Proc. IEEE Conf. Decision Contr.*, pages 761–766, 1979.

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

[82] M. Hapke, A. Jaszkiewicz, and R. Slowinski. Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, 6(3):329–345, 2000.

[83] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, AnnArbor, 1975.

[84] F. Höppner. Fuzzy shell clustering algorithms in image processing: Fuzzy c-rectangular and 2-rectangular shells. *IEEE Transactions on Fuzzy Systems*, 5(4):599–613, 1997.

[85] J. Horn, J. Horn, N. Nafpliotis, N. Nafpliotis, D. E. Goldberg, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87, 1994.

[86] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro. Evolutionray algorithms for clustering gene-expression data. In *Proc. 4th IEEE Int. Conference on Data Mining*, pages 403–406. 2004.

[87] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro. Improving the efficiency of a clustering genetic algorithm. In *Proc. 9th Ibero-American Conference on Artificial Intelligence*, volume LNCS 3315, pages 861–870. 2004.

[88] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro. Evolving clusters in gene-expression data. *Information Sciences*, 176(13):1898–1927, 2006.

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

[90] E. R. Hruschka and N. F. F. Ebecken. A genetic algorithm for cluster analysis. *Intelligent Data Analysis*, 7(1):15–25, 2003.

[91] E. J. Hughes. Evolutionary many-objective optimization: Many once or one many? In *Proceedings of 2005 Congress on Evolutionary Computation*, pages 222–227, Edinburgh, Scotland, UK, September 2-5, 2005.

[92] L. Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8):967–973, 1989.

[93] H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. *In Parallel Problem Solving from Nature IX (PPSN-IX)*, 4193:493–502, September 2006.

[94] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, 28(3):392–403, August, 1998.

[95] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic serach and local serach in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 6(6):721–741, 1984.

[96] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[97] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[98] N. Jardine and R. Sibson. *Mathematical Taxonomy*. John Wiley and Sons, 1971.

[99] A. Jaszkiewicz. Comparison of local search-based metaheuristics on the multiple objective knapsack problem. *Foundations of computer and decision sciences*, 26(1):99–120, 2001.

[100] L. Kankanala and M. N. Murty. Hybrid approaches for clustering. In *PReMI*, pages 25–32, 2007.

[101] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient K-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[102] L. Kaufman and P. J. Roussenw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, NY, US, 1990.

[103] D. J. Kim, Y. W. Park, and D. J. Park. A novel validity index for determination of the optimal number of clusters. *IEICE Transactions on Information and Systems*, D-E84(2):281–285, 2001.

[104] D.-W. Kim, K. H. Lee, and D. Lee. Fuzzy cluster validation index based on inter-cluster proximity. *Pattern Recognition Letters*, 24(15):2561–2574, 2003.

[105] M. Kim and R. S. Ramakrishna. New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15):2353–2363, 2005.

[106] Y.-I. Kim, D.-W. Kim, D. Lee, and K. H. Lee. A cluster validation index for GK cluster analysis based on relative degree of sharing. *Information Sciences*, 168(1-4):225–242, 2004.

[107] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5/6):975–986, 1984.

[108] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[109] S. Kirpatrick and M. P. Vecchi. Global wiring by simulated annealing. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):215–222, October,1983.

[110] J. D. Knowles and D. W. Corne. Approxmating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[111] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, New York, Berlin, 3rd edition, 1989.

[112] E. E. Korkmaz, J. Du, R. Alhajj, and K. Barker. Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intell. Data Anal.*, 10(2):163–182, 2006.

[113] B. Kövesi, J. M. Boucher, and S. Saoodi. Stochastic K-means algorithm for vector quantization. *Pattern Recognition Letters*, 22(6-7):603–610, 2001.

[114] K. Krishna and M. N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, And Cybernetics-Part B*, 29(3):433–439, June,1999.

[115] R. Krovi. Genetic algorithms for clustering: A preliminary investigation. In *Proceedings of the 25th Hawaii Int. Conference on System Sviences*, volume 4, pages 540–544, 1992.

[116] L. I. Kuncheva and J. C. Bezdek. Selection of cluster prototypes from data by a genetic algorithm. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, pages 1683–1688, 1997.

[117] S. Kwanghoon, K. H. Jung, and W. E. Alexander. A mean field annealing approach to robust corner detection. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 28(1):82–90, 1998.

[118] S. H. Kwon. Cluster validity index for fuzzy clustering. *Electron. Lett.*, 34(22):2176–2177, 1998.

[119] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, 2004.

[120] M. Laszlo and S. Mukherjee. A genetic algorithm using hyper-quadtrees for low-dimensional K-means clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):533–543, April, 2006.

[121] E. Leon, O. Nasraoui, and J. Gomez. ECSAGO: Evolutionray clustering with self adaptive genetic operators. In *Proc. IEEE Congress on Evolutionray Computation*, pages 1768–1775. July 16-21, 2006.

[122] J.-Y. Lin, H. Peng, J.-M. Xie, and Q.-L. Zheng. Novel clustering algorithm based on central symmetry. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1329– 1334. 26-29 Aug. 2004.

[123] V. Loia, W. Pedrycz, and S. Senatore. Semantic web content analysis: A study in proximity-based collaborative clustering. *IEEE Transactions Fuzzy Systems*, 15(6):1294–1312, 2007.

[124] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown. FGKA: a fast genetic K-means clustering algorithm. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 622–623, New York, NY, USA, 2004. ACM.

[125] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown. Incremental genetic K-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics*, 5:172, 2004.

[126] C. B. Lucasius, A. D. Dane, and G. Kateman. On K-medoid clustering of large data sets with the aid of a genetic algorithm: Backgroud, feasibility and comparison. *Analytica Chimica Acta*, 282:647–669, 1993.

[127] P. C. H. Ma, K. C. C. Chan, X. Yao, and D. K. Y. Chiu. An evolutionray clustering algorithm for gene expression microarray data analysis. *IEEE Transactions on Evolutionray Computation*, 10(3):296–314, 2006.

[128] Y. Man and I. Gath. Detection and separation of ring-shaped clusters using fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):855–861, 1994.

[129] J. Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering. *IEEE Transactions on Neural Networks*, 7(1):16–29, 1996.

[130] N. Matake, T. Hiroyasu, M. Miki, and T. Senda. Multiobjective clustering with automatic k-determination for large-scale data. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 861–868, New York, NY, USA, 2007. ACM.

[131] U. Maulik and S. Bandyopadhyay. Genetic algorithm based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.

[132] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[133] U. Maulik and S. Bandyopadhyay. Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE Transactions on Geoscience and Remote Sensing*, 41(5):1075–1081, 2003.

[134] U. Maulik, S. Bandyopadhyay, and J. Trinder. SAFE: An efficient feature extraction technique. *Journal of Knowledge and Information Systems*, 3(3):374–387, 2001.

[135] U Maulik, A. Mukhopadhyay, and S. Bandyopadhyay. Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC Bioinformatics*, 10(27):1197–1208, 2009.

[136] P. Merz and A. Zell. Clustering gene expression profiles with memetic algorithms. In *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, pages 811–820, London, UK, 2002. Springer-Verlag.

[137] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbloth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21(6):10871092, 1953.

[138] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, New York, 1992.

[139] G. W. Milligan and C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[140] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. Efficient two-stage fuzzy clustering of microarray gene expression data. In *Proceedings of 9th International Conference on Information Technology (ICIT 2006)*, pages 11–14. IEEE Computer Society, December, 2006.

[141] C. A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, 1996.

[142] D. Nam and C. H. Park. Pareto-based cost simulated annealing for multiobjective optimization. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 2, pages 522–526, Nanyang Technical University, Orchid Country Club, Singapore, November, 2002.

[143] D. K. Nam and C. H. Park. Multiobjective simulated annealing: a comparative study to evolutionary algorithms. *Int. J. Fuzzy Systems*, 2(2):87–97, 2000.

[144] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, Heidelberg, 2007.

[145] M. K. Pakhira, U. Maulik, and S. Bandyopadhyay. Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487–501, 2004.

[146] S. K. Pal and S. Mitra. Fuzzy versions of Kohonen's net and MLP-based classification: Performance evaluation for certain nonconvex decision regions. *Information Sciences*, 76:297–337, 1994.

[147] S. K. Pal and P. P. Wang. *Genetic Algorithms for Pattern Recognition*. CRC press, Boca Raton, 1996.

[148] Y. J. Park and M. S. Song. A genetic algorithm for clustering problems. In *Proc. 3rd Annual Conference on Genetic Programming*, pages 568–575. Paris, France, 1998.

[149] W. Pedrycz. Fuzzy clustering with a knowledge-based guidance. *Pattern Recognition Letters*, 25(4):469–480, 2004.

[150] W. Pedrycz. Knowledge-based clustering in computational intelligence. In *Challenges for Computational Intelligence*, pages 317–341. 2007.

[151] W. Pedrycz. A dynamic data granulation through adjustable fuzzy clustering. *Pattern Recognition Letters*, 29(16):2059–2066, 2008.

[152] W. Pedrycz, A. Amato, V. D. Lecce, and V. Piuri. Fuzzy clustering with partial supervision in organization and classification of digital images. *IEEE Transactions on Fuzzy Systems*, 16(4):1008–1026, 2008.

[153] W. Pedrycz and K. Hirota. A consensus-driven fuzzy clustering. *Pattern Recognition Letters*, 29(9):1333–1343, 2008.

[154] W. Pedrycz, V. Loia, and S. Senatore. P-FCM: a proximity – based fuzzy clustering. *Fuzzy Sets and Systems*, 148(1):21–41, 2004.

[155] W. Pedrycz and P. Rai. Collaborative clustering with the use of fuzzy c-means and its quantification. *Fuzzy Sets and Systems*, 159(18):2399–2427, 2008.

[156] I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[157] J. A. Richards. *Remote Sensing Digital Image Analysis : An Introduction.* Springer-Verlag, New York, 1993.

[158] K. S. N. Ripon, C.-H. Tsang, S. Kwong, and M.-K. Ip. Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 1200–1203, Washington, DC, USA, 2006. IEEE Computer Society.

[159] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[160] S. Saha and S. Bandyopadhyay. A new multiobjective simulated annealing based clustering technique using symmetry. *Pattern Recognition Letters (accepted).*

[161] S. Saha and S. Bandyopadhyay. A genetic clustering technique using a new line symmetry based distance measure. In *Proceedings of Fifth International Conference on Advanced Computing and Communications (ADCOM'07)*, pages 365–370, Guwahati, India, 2007. IEEE Computer Society.

[162] S. Saha and S. Bandyopadhyay. A validity index based on cluster symmetry. In *Proceedings of IEEE International Conference SMC 2007*, pages 462–467, Montreal, 2007.

[163] S. Saha and S. Bandyopadhyay. Application of a new symmetry based multiobjective clustering technique for automatic MR brain image segmentation. *Australian Journal of Intelligent Information Processing Systems (AJIIPS)*, 10.2:19–28, 2008.

[164] S. Saha and S. Bandyopadhyay. A new principal axis based line symmetry measurement and its application to clustering. In *Proceedings of ICONIP 2008*, pages 179–180, New Zealand, 2008. Springer Verlag.

[165] S. Saha and S. Bandyopadhyay. Application of some symmetry based cluster validity indices to satellite image segmentation. *Applied Soft Computing (communicated)*, 2009.

[166] S. Saha and S. Bandyopadhyay. A new line symmetry distance and its application to data clustering. *Journal of Computer Science and Technology*, 24(3):544–556, 2009.

[167] S. Saha and S. Bandyopadhyay. Some symmetry based classifiers. *Fundamenta Informaticae*, 90(1-2):107–123, 2009.

[168] S. Saha and S. Bandyopadhyay. A validity index based on symmetry and stability: Selection of model and model order. *IEEE Transactions on Pattern Analysis and Machine Intelligence (under revision)*, 2009.

[169] S. Saha and S. Bandyopadhyay. Application of a new symmetry based cluster validity index for satellite image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 5(2):166–170, April, 2008.

[170] S. Saha and S. Bandyopadhyay. Semi-GAPS: A semi-supervised clustering technique using a point symmetry based distance. *Fundamenta Informaticae (accepted)*, DOI 10.3233/FI-2009-0000, 2009.

[171] S. Saha and S. Bandyopadhyay. A new symmetry based multiobjective clustering technique for automatic evolution of clusters. *Pattern Recognition*, doi:10.1016/j.patcog.2009.07.004, 2009.

[172] S. Saha and S. Bandyopadhyay. Performance evaluation of some symmetry based cluster validity indices. *IEEE Transactions on Systems Man and Cybernetics, Part C*, 39(4):420–425, July, 2009.

[173] S. Saha and S. Bandyopadhyay. MRI brain image segmentation by fuzzy symmetry based genetic clustering technique. In *Proceedings of International Conference IEEE CEC 2007*, pages 4417–4424, Singapore, September 2007. IEEE Computer Society.

[174] S. Saha, S. Sur-Kolay, S. Bandyopadhyay, and P. Dasgupta. Multiobjective genetic algorithm for k-way equipartitioning of a point set with application to CAD-VLSI. In *Proceedings of the IEEE 9th International Conference on Information Technology (ICIT, 2006)*, pages 281–284, Bhubaneswar, 2006. IEEE Computer Society.

[175] S. Saha, S. Sur-Kolay, P. Dasgupta, and S. Bandyopadhyay. MAkE: Multiobjective algorithm for k-way equipartitioning of a point set. *Applied Soft Computing*, 9(2):711–724, 2009.

[176] J. R. Schott. Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics, Massachussets Institute of Technology, Boston,MA, 1995.

[177] P. Serafini. Simulated annealing for multiple objective optimization problems. In *Proceedings of the tenth international conference on mul-*

tiple criteria decision making: expand and enrich the domains of thinking and application, volume 1, pages 283–292. Berline, Springer-Verlag, 1994.

[178] W. Sheng and X. Liu. A hybrid algorithm for k-medoid clustering of large data sets. In *Proceedings of IEEE Congress on Evolutionray Computation*, pages 77–82, 2004.

[179] W. Sheng, S. Swift, L. Zhang, and X. Liu. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 35(6):56–67, December, 2005.

[180] V. S. Slves, R. J. G. B. Campello, and E. R. Hruschka. Towards a fast evolutionray algorithm for clustering. In *Proc. IEEE Congress on Evolutionray Computation*, pages 6240–6247. 2006.

[181] K. I. Smith, R. M. Everson, and J. E. Fieldsend. Dominance measures for multi-objective simulated annealing. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'04)*, pages 23–30. 2004.

[182] K. I. Smith, R. M. Everson, J. E. Fieldsend, C. Murphy, and R. Misra. Dominance-based multi-objective simulated annealing. *IEEE Transactions on Evolutionary Computation*, 12(3):323–342, 2008.

[183] E. Sontag and H. Sussman. Image restoration and segmentation using the annealing algorithm. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):215–222, October,1983.

[184] M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernatics*, 24(4):656–667, April, 1994.

[185] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3):221–248, 1994.

[186] A. Staiano, R. Tagliaferri, and W. Pedrycz. Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering. *Neurocomputing*, 69(13-15):1570–1581, 2006.

[187] M.-C. Su and C.-H. Chou. A modified version of the K-means algorithm with a distance based on cluster symmetry. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 23(6):674–680, 2001.

[188] B. Suman. Study of self-stopping PDMOSA and performance measure in multiobjective optimization. *Computers and Chemical Engineering*, 29(5):1131–1147, 15 April 2005.

[189] B. Suman. Multiobjective simulated annealing-a metaheuristic technique for multiobjective optimization of a constrained problem. *Foundations of Computer and Decision Science*, 27(3):171–191, 2002.

[190] B. Suman. Simulated annealing based multiobjective algorithm and their application for system reliability. *Engineering Optimization*, 35(4):391–416, 2003.

[191] B. Suman. Study of simulated annealing based multiobjective algorithm for multiobjective optimization of a constrained problem. *Computers and Chemical Engineering*, 28(9):1849–1871, 2004.

[192] B. Suman and P. Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57(10):1143–1160, 2006.

[193] A. Suppapitnarm, K. Seffen, G. Parks, and P. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1):59–85, 2000.

[194] A. Suppapitnarm, K. A. Seffen, G. T. Parks, and P. J. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1):59–85, 2000.

[195] H. H. Szu and R. L. Hartley. Fast simulated annealing. *Physics Letter A*, 122(3-4):157–162, 1987.

[196] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters via the gap statistics. *J. Royal Statistical Society*, 63:411–423, 2001.

[197] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Reading, 1974.

[198] G. T. Toussaint. The realtive neighborhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

[199] G. T. Toussaint. Pattern recognition and geometrical complexity. In *Proc. Fifth International Conf. on Pattern Recognition*, pages 1324–1347. Miami Beach, December 1980.

[200] D. Tuyttens, J. Teghem, and N. E.-Sherbeny. A particular multiobjective vehicle routing problem solved by simulated annealing. *Metaheuristics for multiobjective optimization*, 535:133–152, 2003.

[201] E. L. Ulungu, J. Teghaem, P. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial decision problems. *J. multi-criteria decision analysis*, 8(4):221–236, 1999.

[202] S. Varma, S. Asharaf, and M. N. Murty. Rough core vector clustering. In *PReMI*, pages 304–310, 2007.

[203] P. A. Vijaya, M. N. Murty, and D. K. Subramanian. Leaders - subleaders: An efficient hierarchical clustering algorithm for large data sets. *Pattern Recognition Letters*, 25(4):505–513, 2004.

[204] P. A. Vijaya, M. N. Murty, and D. K. Subramanian. An efficient hybrid hierarchical agglomerative clustering (HHAC) technique for partitioning large data sets. In *PReMI*, pages 583–588, 2005.

[205] W. Wang and Y. Zhang. On fuzzy cluster validity indices. *Fuzzy Sets Syst.*, 158(19):2095–2117, 2007.

[206] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[207] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16:645–678, 2005.

[208] X. Yao. A new simulated annealing algorithm. *International Journal of Computer Mathematics*, 56:161–168, 1995.

[209] Y. Zhang, W. Wang, X. Zhang, and Y. Li. A cluster validity index for fuzzy clustering. *Information Sciences*, 178(4):1205–1218, 2008.

[210] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[211] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[212] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 1998.

LIST OF RELATED PUBLICATIONS OF THE AUTHOR

1. S. Bandyopadhyay and S. Saha. GAPS: A New Symmetry Based Genetic Clustering Technique. *Pattern Recognition*, 40 (12): 3430-3451, December 2007.

2. S. Bandyopadhyay, S. Saha, U. Maulik and K. Deb. A Simulated Annealing Based Multi-objective Optimization Algorithm: AMOSA. *IEEE Transaction on Evolutionary Computation*, 12 (3): 269-283, June 2008.

3. S. Saha and S. Bandyopadhyay. Application of a New Symmetry Based Cluster Validity Index for Satellite Image Segmentation. *IEEE Geoscience and Remote Sensing Letters*, 5 (2):166-170, April 2008.

4. S. Bandyopadhyay and S. Saha. A Point Symmetry Based Clustering Technique for Automatic Evolution of Clusters. *IEEE Transactions on Knowledge and Data Engineering*, 20 (11):1-17, November 2008.

5. S. Saha and S. Bandyopadhyay. Performance Evaluation of Some Symmetry Based Cluster Validity Indices. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39(4):420-425, July, 2009.

6. S. Saha, S. Bandyopadhyay. A New Symmetry Based Multiobjective Clustering Technique for Automatic Evolution of Clusters. *Pattern Recognition* (accepted), doi:10.1016/j.patcog.2009.07.004.

7. S. Saha, S. Bandyopadhyay. A New Multiobjective Simulated Annealing Based Clustering Technique Using Symmetry. *Pattern Recognition Letters* (accepted), 2009.

8. S. Bandyopadhyay and S. Saha. A Validity Index Based on Symmetry and Stability: Selection of Model and Model Order. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (under revision).

9. S. Saha and S. Bandyopadhyay. Application of Some Symmetry Based Cluster Validity Indices to Satellite Image Segmentation. *Applied Soft Computing* (communicated).

10. S. Bandyopadhyay and S. Saha. A Generalized Automatic Clustering Algorithm in a Multiobjective Framework. *IEEE Transactions on Knowledge and Data Engineering* (communicated).

11. S. Saha, S. Sur-Koley, S. Bandyopadhyay and P. Dasgupta. Multiobjective Genetic Algorithm for k-way Equipartitioning of a Point Set with Application to CAD-VLSI. *Proceedings of the IEEE 9th International Conference on Information Technology (ICIT)*, Bhubaneswar, pp. 281-284, 2006.

12. S. Saha and S. Bandyopadhyay. MRI Brain Image Segmentation by Fuzzy Symmetry Based Genetic Clustering Technique. *IEEE International Congress on Evolutionary Computation*, Singapore, pp. 4417-4424, 2007.

13. S. Saha and S. Bandyopadhyay. A Validity Index Based on Cluster Symmetry. *IEEE International Conference SMC 2007*, Montreal, pp. 462-467, 2007.

14. S. Saha and S. Bandyopadhyay. A Genetic Clustering Technique Using a New Line Symmetry Based Distance Measure. *15th International Conference on Advanced Computing & Communication (ADCOM 2007)*, Guwahati, 2007, pp. 365-370 (IEEE CS Press).

15. S. Bandyopadhyay and S. Saha. A New Principal Axis Based Line Symmetry Measurement and its Application to Clustering. In the Proceedings of ICONIP 2008, New Zealand, pp. 179-180.

——————————- xxx ——————————-