Applied Soft Computing 13 (2013) 89–108

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A generalized automatic clustering algorithm in a multiobjective framework

Sriparna Saha [a,*], Sanghamitra Bandyopadhyay [b]

[a] Department of Computer Science and Engineering, Indian Institute of Technology Patna, India
[b] Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 30 December 2011
Received in revised form 3 July 2012
Accepted 2 August 2012
Available online 6 September 2012

Keywords:
Clustering
Multiobjective optimization (MOO)
Symmetry
Relative neighborhood graph
Multi-center
Automatic determination of number of clusters

ABSTRACT

In this paper a new multiobjective (MO) clustering technique (GenClustMOO) is proposed which can automatically partition the data into an appropriate number of clusters. Each cluster is divided into several small hyperspherical subclusters and the centers of all these small sub-clusters are encoded in a string to represent the whole clustering. For assigning points to different clusters, these local sub-clusters are considered individually. For the purpose of objective function evaluation, these sub-clusters are merged appropriately to form a variable number of global clusters. Three objective functions, one reflecting the total compactness of the partitioning based on the Euclidean distance, the other reflecting the total symmetry of the clusters, and the last reflecting the cluster connectedness, are considered here. These are optimized simultaneously using AMOSA, a newly developed simulated annealing based multiobjective optimization method, in order to detect the appropriate number of clusters as well as the appropriate partitioning. The symmetry present in a partitioning is measured using a newly developed point symmetry based distance. Connectedness present in a partitioning is measured using the relative neighborhood graph concept. Since AMOSA, as well as any other MO optimization technique, provides a set of Pareto-optimal solutions, a new method is also developed to determine a single solution from this set. Thus the proposed GenClustMOO is able to detect the appropriate number of clusters and the appropriate partitioning from data sets having either well-separated clusters of any shape or symmetrical clusters with or without overlaps. The effectiveness of the proposed GenClustMOO in comparison with another recent multiobjective clustering technique (MOCK), a single objective genetic algorithm based automatic clustering technique (VGAPS-clustering), *K*-means and single linkage clustering techniques is comprehensively demonstrated for nineteen artificial and seven 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.

1. Introduction

Clustering [1,2] is a popular unsupervised pattern classification technique which 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, $X = \{x_1, x_2, \ldots, x_n\}$) such that

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

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

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

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

Determining the appropriate number of clusters from a given data set is an important consideration in clustering. For this

* Corresponding author. Tel.: +91 8809559190.
E-mail addresses: sriparna.saha@gmail.com, sriparna@iitp.ac.in (S. Saha), sanghami@isical.ac.in (S. Bandyopadhyay).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.08.005

purpose, and also to validate the obtained partitioning, several cluster validity indices have been proposed in the literature. The measure of validity of the clusters should be such that it will be able to impose an ordering of the clusters in terms of their goodness. The classical approach of determining the number of clusters is to apply a given clustering algorithm for a range of K values and to evaluate a certain validity function of the resulting partitioning in each case. The partitioning exhibiting the optimal validity is chosen as the true partitioning. In [3] a genetic clustering technique is proposed which uses a cluster validity index as the objective function. Authors have experimented with several different cluster validity indices. In [4] authors have experimented with three clustering algorithms, hard K-Means, single linkage, and a simulated annealing (SA) based technique, in conjunction with four cluster validity indices, namely Davies–Bouldin index, Dunn's index, Calinski–Harabasz index, and a recently developed index I. In [5] a new cluster validity index named CS-index is developed which is able to detect clusters of different densities. In addition, authors also propose a modified K-means algorithm that can assign more cluster centers to areas with low densities of data than the conventional K-means algorithm does. Some fuzzy logic based cluster validity indices are proposed in [6]. A partitional clustering method based on graph theory and a clustering tendency index are proposed in [7]. The number of clusters and the partition that best fits the data set, are selected according to the optimal cluster tendency index value. In [8], authors have presented an analysis of design principles implicitly used in defining cluster validity indices and reviewed a variety of existing cluster validity indices in the light of these principles. After that authors proposed some remedies to overcome the limitations of the existing indices. Based on these remedies six new cluster validity indices are proposed.

The method of using cluster validity indices for searching the optimal number of cluster number depends on the selected clustering algorithm, whose performance may depend on several factors including the initial values, algorithm's parameters, optimization approach and assumptions regarding the cluster distributions. Similarly, most of the validity measures usually assume a certain geometrical structure in the cluster shapes. But if several different cluster structures exist in the same data set, these have often been found to fail.

The global optimum of these validity functions correspond to the most "valid" solutions. Thus Genetic Algorithms (GAs) have been applied to optimize the validity functions to determine the appropriate number of clusters and the appropriate partitioning of a data set simultaneously [3,9,10]. Simple GA (SGA) [11] or its variants are used as the genetic clustering techniques in [3,9,10]. In [12], a function called Weighted Sum Validity Function (WSVF), which is a weighted sum of the several normalized validity functions, is used for optimization along with a Hybrid Niching Genetic Algorithm (HNGA) to automatically evolve the proper number of clusters from a given data set. Within this HNGA, a niching method is developed to prevent premature convergence by preserving both the diversity of the population with respect to the number of clusters encoded in the individuals and the diversity of the subpopulation with the same number of clusters during the search. In [13], a variable string length GA (VGA) based clustering method (named VGAPS-clustering) is proposed which uses a newly developed point symmetry based distance [14] for assignment of points to different clusters and optimizes a newly developed point symmetry (PS) based cluster validity index, *Sym*-index [15,13]. Use of the PS-distance enables the proposed VGAPS-clustering to evolve the clusters of any shape and size as long as they possess the symmetry property.

1.1. Relevance of multiobjective optimization for 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 [16], 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. Although the objectives of [16] 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. Moreover MOCK uses locus-based adjacency representation proposed in [17]. Thus when the number of data points is too large the string length becomes high too and convergence becomes slow.

In this paper we have developed a new multiobjective clustering technique with encoding of cluster centers instead of data points. The technique 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 article. Each cluster is divided into several non-overlapping small hyperspherical sub-clusters and the centers of these sub-clusters are encoded in a string to represent a particular cluster. Three cluster validity indices are optimized simultaneously using the search capability of AMOSA. One of these cluster validity indices reflects the total compactness of a particular partitioning, another represents the total symmetry present in a particular partitioning and the last one measures, in a novel way, the degree of "connectedness" of a particular partitioning.

Any multiobjective optimization technique generates a large number of non-dominated solutions on its final Pareto optimal front. Each of these solutions provides a way of partitioning the particular data set. All these solutions are equally important from the algorithmic point of view, but sometimes the user wants a single solution. Thus in this article we have also developed a new semi-supervised method to identify a single best solution from the set of final Pareto-optimal solutions. The superiority of the proposed *GenClustMOO* in comparison with MOCK, a recently proposed MO clustering technique, a single objective genetic clustering technique VGAPS-clustering [13], K-means clustering technique and single linkage clustering techniques, is shown for nineteen artificial data sets (including most of the data sets used in [16]) and seven 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. In a part of the paper we have also experimented with a second criterion of selecting a single solution from the final Pareto optimal set.

2. The SA based MOO algorithm: AMOSA

Archived multiobjective simulated annealing (AMOSA) [18] is an efficient MO version of the simulated annealing (SA) algorithm. MOO is applied when dealing with the real-world problems where

there are several objectives that should be optimized simultaneously. Simulated annealing (SA) is a search technique for solving difficult optimization problems, which is based on the principles of statistical mechanics [19]. SA can not only replace exhaustive search to save time and resource, but also converge to the global optimum if annealed sufficiently slowly [20]. Although the single objective version of SA has been quite popular, its utility in the multiobjective case was limited because of its search-from-a-point nature. Recently Bandyopadhyay et al. developed an efficient multiobjective version of SA called AMOSA [18] that overcomes this limitation.

In AMOSA (archived multiobjective simulated annealing) [18], which is an multiobjective version of SA, several concepts have been newly integrated. It utilizes the concept of an archive where the non-dominated solutions seen so far are stored. Two limits are kept on the size of the archive: a hard or strict limit denoted by *HL*, and a larger, soft limit denoted by *SL*, where $SL > HL$. The non-dominated solutions are stored in the archive as and when they are generated. In the process, if some members of the archive get dominated by the new solutions, then these are removed. If at some point of time, the size of the archive exceeds a specified value, then the clustering process, described below, is invoked.

In AMOSA, the initial temperature is set to T_{max}. Then, one of the points is randomly selected from the archive. This is taken as the *current-pt*, or the initial solution. The *current-pt* is perturbed to generate a new solution called *new-pt*, and its objective functions are computed. The domination status of the *new-pt* is checked with respect to the *current-pt* and the solutions in the archive. A new quantity called the amount of domination, $\Delta dom(a, b)$, between two solutions a and b is defined as follows:

$$\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{M} \frac{|f_i(a) - f_i(b)|}{R_i}, \quad (1)$$

where $f_i(a)$ and $f_i(b)$ are the ith objective values of the two solutions and R_i is the corresponding range of the objective function computed from the individuals in the population. M is the number of objectives. Based on the domination status between the *new-pt*, *current-pt* and the points in the archive, different cases may arise. These are discussed in details in [18], and are briefly mentioned here for the sake of completeness.

Case 1: *new-pt* is either dominated by the *current-pt* or it is nondominating with respect to the *current-pt*, but some points in the archive dominate the *new-pt*. Suppose *new-pt* is dominated by a total of k points (including *current-pt* and points in the archive). This case is demonstrated in Fig. 1 (the points D–H in the figure signify the content of the archive at any instant, while the other points illustrate different cases that may arise with respect to the archive) where F represents the *current-pt* and B represents the *new-pt*. Then a quantity Δdom_{avg} is computed as $(\sum_{i=1}^{k}(\Delta dom_{i,new-pt}) + \Delta dom_{current-pt,new-pt})/(k+1)$. The *new-pt* is accepted as *current-pt* with a probability

$$p_{qs} = \frac{1}{1 + e^{((\Delta dom_{avg})/T)}}. \quad (2)$$

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 farther away from the *new-pt* are contributing to its value.

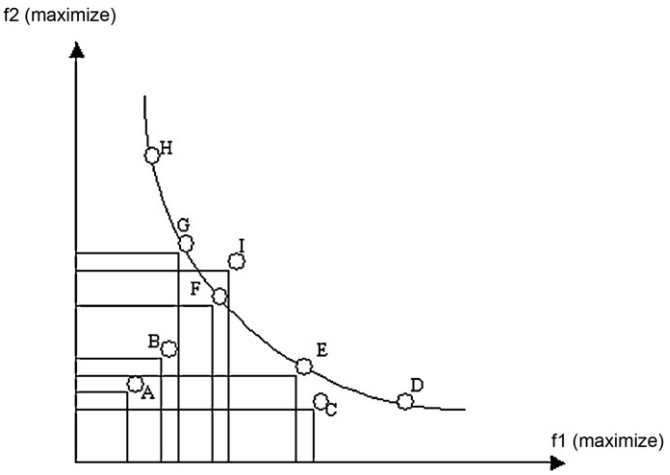

Fig. 1. Pareto-optimal front and different domination examples.

Case 2: Neither the *current-pt* nor the points in the archive dominate the *new-pt*. This can be demonstrated with different examples shown in Fig. 1, e.g., F represents the *current-pt* and E represents the *new-pt*, G represents the *current-pt* and I represents the *new-pt*, F represents the *current-pt* and I represents the *new-pt*. For all these cases, accept the *new-pt* as the *current-pt*. If there are any points in the archive which are dominated by *new-pt*, remove them from the archive. Add *new-pt* in the archive. If archive size crosses the *SL*, apply single linkage clustering to reduce its size to *HL*.

Case 3: *new-pt* dominates the *current-pt* but k points in the archive dominate the *new-pt*. This case can be demonstrated using Fig. 1 where A represents the *current-pt* and B represents the *new-pt*. Here the minimum of the differences 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 that corresponds to the minimum difference is selected as the *current-pt* with probability

$$probability = \frac{1}{1 + \exp(\Delta dom_{min})}. \quad (3)$$

Otherwise the *new-pt* is selected as the *current-pt*. This may be considered as an informed reseeding of the annealer only if the archive point is accepted.

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, T_{min}, is attained. The process thereafter stops, and the archive contains the final non-dominated solutions.

It has been demonstrated in [18] that the performance of AMOSA is better than that of NSGA-II [21] and some other well-known MOO algorithms. The pseudocode of the AMOSA algorithm is shown in Fig. 2.

Since the performance of AMOSA has been found to be as good as, or often better than, some other well-known MOO algorithms especially for 3 or more objectives [18], it is used as the underlying MOO technique in this article.

3. Proposed method of multiobjective clustering

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

Set *Tmax*, *Tmin*, *HL*, *SL*, *iter*, α, temp=*Tmax*.
Initialize the *Archive*.
current-pt = random(*Archive*). /* randomly chosen solution from *Archive**/
while (temp > *Tmin*)
  for (i=0; i< iter; i++)
    new-pt=perturb(*current-pt*).
    Check the domination status of *new-pt* and *current-pt*.
    /* Code for different cases */
    if (*current-pt* dominates *new-pt*) /* Case 1*/
      $\Delta dom_{avg} = \frac{\left(\sum_{i=1}^{k}(\Delta dom_{i,new-pt}) + \Delta dom_{current-pt,new-pt}\right)}{(k+1)}$.
      /* 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).
    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).
      if (*new-pt* is non-dominating w.r.t all the points in the *Archive*) /* Case 2(b)*/
        Set *new-pt* as *current-pt* and add *new-pt* to the *Archive*.
        if *Archive-size* > *SL*
          Cluster *Archive* to *HL* number of clusters.
      if (*new-pt* dominates k, ($k \geq 1$) points of the *Archive*) /* Case 2(c)*/
        Set *new-pt* as *current-pt* and add it to *Archive*.
        Remove all the k dominated points from the *Archive*.
    if (*new-pt* dominates *current-pt*) /* Case 3 */
      Check the domination status of *new-pt* and points in the *Archive*.
      if (*new-pt* is dominated by k ($k \geq 1$) points in the *Archive*) /* Case 3(a)*/
        Δdom_{min} = minimum of the difference of domination amounts between the *new-pt* and the k points
        $prob = \frac{1}{1+exp(-\Delta dom_{min})}$.
        Set point of the archive which corresponds to Δdom_{min} as *current-pt* with probability=*prob*.
        else set *new-pt* as *current-pt*
      if (*new-pt* is 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.

Fig. 2. The AMOSA algorithm [18].

Source code is available at: www.isical.ac.in/ sriparna_r.

3.1. String representation and population initialization

In *GenClustMOO*, a state of AMOSA comprises a set of real numbers which represents the coordinates of the centers of the clusters. 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 string 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 string will be $C \times K \times d$. This concept of representing one cluster using multi-centers is shown in Fig. 3. Suppose a particular string 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 string will look like: $\langle cx_1^1, cy_1^1, cx_2^1, cy_2^1, \ldots, cx_{10}^1, cy_{10}^1, cx_1^2, cy_1^2, \ldots, cx_{10}^2, cy_{10}^2\rangle$. Each string i in the archive initially contains K_i number of clusters, such that $K_i = (rand() \bmod (K^{max} - 1)) + 2$. Here, *rand*() is a function returning an integer, and K^{max} is a soft estimate of the upper bound of the number of clusters. The number of initial clusters will therefore lie between two and K^{max}. Our initialization procedure is motivated by that of [16]. Here the initialization procedure is partly random and partly based on two different single-objective algorithms in order to obtain a good initial spread of solutions. One third of the solutions in the archive is initialized after running single linkage clustering algorithm for different values of K. Let for the ith chromosome we execute single linkage clustering algorithm with $K = 3$. Then for each of these three clusters sub-cluster centers will be selected randomly from the points belonging to that particular cluster formed after application of single linkage

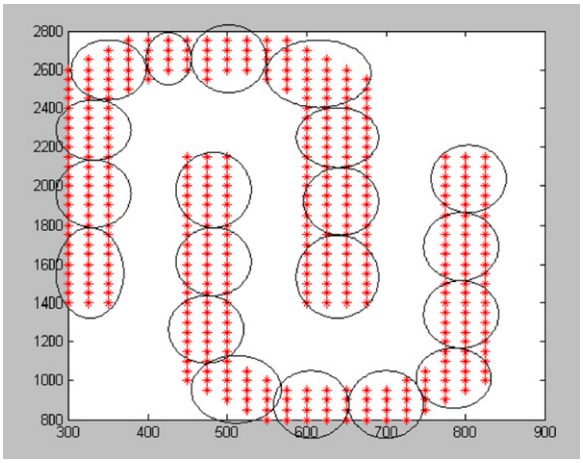

Fig. 3. Example of representing a single cluster using multiple cluster centers.

clustering algorithm. 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. Here again *K*-means is executed with different values of number of clusters, *K*. Let for *i*th cluster we run *K*-means algorithm with $K = 2$. Then for this chromosome $K_i = 2$. The sub-cluster centers for cluster *i* are chosen from points belonging to this particular cluster formed after application of *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 strings the K_i centers encoded in a string are randomly selected distinct points from the data set. The initial partitioning is obtained using a minimum center distance based criterion. For all the initial encoded solutions, *C* number of distinct points are selected from each cluster randomly. These $C \times K$ number of points are encoded in that particular string.

3.2. Assignment of points

For the purpose of assignment, each sub-cluster is considered as a separate cluster. Let the particular string contain *K* number of clusters. Here assignment is done based on the minimum Euclidean distance criterion. A data point $\bar{x}_j$ is assigned to the (i, l)th sub-cluster where

$$(i, l) = argmin\{d_e(\bar{c}_k^m, \bar{x}_j)\}, \quad \text{for} \quad k = 1 \ldots K, m = 1, \ldots, C.$$

Here *C* is equal to the total number of sub-cluster centers per cluster. Thereafter, the partition matrix is formed in the following way: $u[(i-1) \times C + l][j] = 1$ and $u[k][j] = 0, \forall\, k = 1 \ldots K \times C, k \neq i, l$.

3.3. Objective functions used

For the purpose of optimization, three different cluster validity indices are considered. These three objective functions reflect three different aspects of good clustering solutions. The first quantifies the amount of symmetry present in a particular partitioning. The second quantifies the connectedness of the clusters and the third measures the compactness of the partitionings in terms of the Euclidean distance. These indices are described below.

3.3.1. Cluster validity index based on symmetry: Sym-index [14]

This cluster validity index is based on a newly developed point symmetry based distance [14], $d_{ps}(\bar{x}, \bar{c})$, which is calculated as follows. Let a point be $\bar{x}$. The symmetrical (reflected) point of $\bar{x}$ with respect to a particular center $\bar{c}$ is $2 \times \bar{c} - \bar{x}$. Let us denote this by $\bar{x}^*$. Let *knear* unique nearest neighbors of $\bar{x}^*$ be at Euclidean distances of d_is, $i = 1, 2, \ldots knear$. Then

$$d_{ps}(\bar{x}, \bar{c}) = d_{sym}(\bar{x}, \bar{c}) \times d_e(\bar{x}, \bar{c}), \tag{4}$$

$$= \frac{\sum_{i=1}^{knear} d_i}{knear} \times d_e(\bar{x}, \bar{c}), \tag{5}$$

where $d_e(\bar{x}, \bar{c})$ is the Euclidean distance between the point $\bar{x}$ and $\bar{c}$, and $d_{sym}(\bar{x}, \bar{c})$ is a symmetry measure of $\bar{x}$ with respect to $\bar{c}$ and is defined as $(\sum_{i=1}^{knear} d_i)/(knear)$. Note that *knear* cannot be chosen equal to 1, since in this case if $\bar{x}^*$ exists in the data set then $d_{ps}(\bar{x}, \bar{c}) = 0$ 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. It may be noted that the proper value of *knear* largely depends on the distribution of the data set. The properties of $d_{ps}(\bar{x}, \bar{c})$ are thoroughly described in [13].

Sym-index [13] is a cluster validity function which measures the overall average symmetry with respect to the cluster centers. Consider a partition of the data set $X = \{\bar{x}_j : j = 1, 2, \ldots n\}$ into *K* clusters where the center of cluster $\bar{c}_i$ is computed by using $\bar{c}_i = \frac{\sum_{j=1}^{n_i} \bar{x}_j^i}{n_i}$ where n_i $(i = 1, 2, \ldots, K)$ is the number of points in cluster *i* and $\bar{x}_j^i$ denotes the *j*th point of the *i*th cluster. The new cluster validity function *Sym* is defined as:

$$Sym(K) = \left(\frac{1}{K} \times \frac{1}{\mathcal{E}_K} \times D_K\right). \tag{6}$$

Here,

$$\mathcal{E}_K = \sum_{i=1}^{K} E_i, \tag{7}$$

such that

$$E_i = \sum_{j=1}^{n_i} d_{ps}^*(\bar{x}_j^i, \bar{c}_i), \tag{8}$$

and

$$D_K = max_{i,j=1}^{K} \|\bar{c}_i - \bar{c}_j\|, \tag{9}$$

D_K is the maximum Euclidean distance between two cluster centers among all pairs of centers. $d_{ps}^*(\bar{x}_j^i, \bar{c}_i)$ is computed by Eq. (5) with some constraint. Here, the first *knear* nearest neighbors of $\bar{x}_j^* = 2 \times \bar{c}_i - \bar{x}_j^i$ will be searched among only those points which are in cluster *i*, i.e., the *knear* nearest neighbors of $\bar{x}_j^*$, the reflected point of $\bar{x}_j^i$ with respect to $\bar{c}_i$, and $\bar{x}_j^i$ should belong to the *i*th cluster. The objective is to maximize this index in order to obtain the actual number of clusters. The explanations of the interaction between different components of *Sym*-index are elaborately described in [13].

As formulated in Eq. (6), *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 $d_e(\bar{x}, \bar{c})$ in Eq. (5) 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 the value of D_K is bounded by the maximum separation between a pair of points in the data set. As these three factors are complementary in nature,

so they are expected to compete and balance each other critically for determining the proper partitioning.

3.3.2. Connectivity based cluster validity index: Con-index [22]

In this article a cluster validity index based on the concept of connectedness [22] of the clusters is used. 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) [23] has been successfully applied for solving several pattern recognition problems. An unsupervised clustering technique based on the concepts of RNG is developed in [24]. In this article, RNG is used to develop a cluster validity index, *Con*-index [22], that quantifies the degree of connectivity of well-separated clusters.

3.3.2.1. Measuring the connectivity among a set of points. Here a novel way of measuring the connectivity among a set of points using relative neighborhood graph is used [22]. 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, **x** and **y**, denoted as $d_{short}(\mathbf{x}, \mathbf{y})$, 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 **x** and **y**, and the number of edges along the ith path is $nedge_i$, for $i = 1, \ldots, p$. If the edges along the ith path are denoted as $ed^i_1, \ldots, ed^i_{nedge_i}$ and the corresponding edge weights are $w(ed^i_1), \ldots, w(ed^i_{nedge_i})$, then the shortest distance between **x** and **y** is defined as follows:

$$d_{short}(\mathbf{x}, \mathbf{y}) = \min_{i=1}^{p} \max_{j=1}^{nedge_i} w(ed^i_j). \tag{10}$$

In order to improve the efficiency of computing d_{short}, we adopt the following pruning strategy. The maximum value of $w(ed^i_j)$ 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.

3.3.2.2. Definition of the cluster validity index. The cluster validity index [22] 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 $\bar{x}^k_{\text{minindex}}$. Then,

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

where n_k is the total number of points in the kth cluster and $\bar{x}^k_i$ denotes the ith point of the kth cluster. Then

$$\overline{m}_k = \bar{x}^k_{\text{minindex}}.$$

The newly developed *Con*-index [22] is defined as follows:

$$Con = \frac{\sum_{i=1}^{K} \sum_{j=1}^{n_k} d_{short}(\overline{m}_i, \bar{x}^i_j)}{n \times \min_{i,j=1}^{K} \bigwedge_{i \neq j} d_{short}(\overline{m}_i, \overline{m}_j)}, \tag{11}$$

where $d_{short}(\overline{m}_i, \bar{x}^i_j)$ is the shortest distance along the relative neighborhood graph between the two points $\overline{m}_i$ and $\bar{x}^i_j$, the jth point of the ith cluster. It is calculated using the procedure mentioned in Section 3.3.2. 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.

3.3.3. Euclidean distance based cluster validity index: I-index

The third objective function used here is an Euclidean distance based cluster validity index, *I*-index [4]. It is defined as follows:

$$I(K) = \left(\frac{1}{K} \times \frac{\mathcal{E}_1}{\mathcal{E}_K} \times D_K\right)^p,$$

where K is the number of clusters. Here $\mathcal{E}_K = \sum_{k=1}^{K} \sum_{j=1}^{n_k} d_e(\bar{c}_k, \bar{x}^k_j)$ and $D_K = \max_{i,j=1}^{K} d_e(\bar{c}_i, \bar{c}_j)$ where $\bar{c}_j$ denotes the center of the jth cluster and $\bar{x}^k_j$ 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, $1/K$, $\mathcal{E}_1/\mathcal{E}_K$, and D_K. The first factor will try to reduce index I as K is increased. The second factor consists of the ratio of $\mathcal{E}_1$, which is constant for a given data set, and $\mathcal{E}_K$, which decreases with increase in K. Hence, because of this term, index I increases as $\mathcal{E}_K$ 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. In this article, we have taken $p = 2$.

3.4. Subcluster merging for objective function calculation

Before computing the above mentioned three objective functions for each string, first the total $C \times K$ number of sub-clusters encoded in a particular string are merged to form a total of K clusters. The merging operation is done in the following way. First, the shortest distance between each pair of $C \times K$ cluster medoids along the relative neighborhood graph is computed. This provides a distance matrix denoted as $distance_{short}$, i.e.,

$$distance_{short} = [d_{short}(c_i, c_j)]_{i,j=1 \ldots C \times K}.$$

Thereafter single linkage clustering technique [25] 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.

Note that after initialization each of the K clusters is formed by C subclusters, and that the representation contains all $K \times C$ subcluster centroids one after another. But after mutation operation some subcluster centers will be modified by some amount. It will break the order. Thus reshuffling is required. So we have merged subcluster centers according to their closeness to each other.

After the merging operation is done, the three cluster validity indices are computed for each string. Thus the objective functions for a particular string are:

$$obj = \{Sym(K), \frac{1}{Con(K)}, I(K)\},$$

where $Sym(K)$, $Con(K)$ and $I(K)$ are, respectively, the calculated *Sym*-index value, *Con*-index value and *I*-index value for that particular string. Here K denotes the number of clusters present in that particular string. These three objective functions are simultaneously maximized using the simulated annealing based MOO algorithm, AMOSA.

3.5. Mutation operation

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

(1) Each cluster center encoded in a string is replaced with a random variable drawn from a Laplacian distribution, $p(\epsilon) \propto e^{-((|\epsilon-\mu|)/\delta)}$, 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 string, i.e., the total number of clusters present in that string 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 with uniform probability on a particular string if it is selected for mutation.

3.6. Selection of the best solution

In MOO, the algorithms produce a large number of non-dominated solutions [26] 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. However the user may sometimes want only a single solution. Consequently, in this paper a semi-supervised method of selecting a single solution from the set of solutions, is now developed.

Here we assume that the class labels of some of the points (denoted as *test patterns*) are known to us. The proposed *GenClust-MOO* produces a set of Pareto optimal solutions. The clustering associated with each solution from the final Pareto optimal set is used to assign the cluster labels of the *test patterns* based on the nearest center criterion. The amount of misclassification is calculated by computing the Minkowski score values. *Minkowski score* is a measure of the quality of a solution given the true clustering [27]. 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 but in different clusters in S. *Minkowski score* is then defined as:

$$D_M(T, S) = \sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}. \quad (12)$$

In this case the optimum score is 0, with lower scores being "better".

The solution with the minimum *Minkowski score* value calculated over the *test patterns* is selected as the best solution.

4. Experimental results

4.1. Data sets used

Nineteen artificial data sets and seven real-life data sets are used for the experiments. A description of the data sets in terms of the number of points present, dimension of the data set and the number of clusters is presented in Table 1. These data sets are divided into four groups.

(1) Group 1: The first group of data sets contains symmetrical shaped clusters e.g., ring-shaped clusters, ellipsoidal clusters, etc.
   (a) *Sym_5_2*: This data set, used in [13], contains 850 data points distributed on five clusters, as shown in Fig. 4(a).
   (b) *Sym_3_2*: This data set contains 600 data points distributed on three clusters, as shown in Fig. 4(b).
   (c) *Ellip_2_2*: This data set, used in [13], contains 400 points distributed on two crossed ellipsoidal shells shown in Fig. 4(c).
   (d) *Ring_3_2*: This data set, used in [13], is a combination of ring-shaped, spherically compact and linear clusters shown in Fig. 4(d).
   (e) *Rect_3_2*: This data set, used in [14], is a combination of ring-shaped, compact and linear clusters shown in Fig. 4(e).
(2) Group 2: This group of data sets contains hyperspherical shaped clusters. Some of these data sets contain highly overlapping clusters.
   (a) *Sph_5_2*: This data set, used in [10], consists of 250 two dimensional data points distributed over 5 spherically shaped clusters. The clusters present in this data set are highly overlapping, each consisting of 50 data points. This data set is shown in Fig. 4(f).
   (b) *Sph_4_3*: This data set, used in [10], consists of 400 points distributed over 4 hyperspherical shaped clusters. This data set is shown in Fig. 4(g). The clusters present in this data set are well-separated, each consisting of 100 data points.
   (c) *Sph_6_2*: This data set, used in [10], consists of 300 data points distributed over 6 different clusters in two dimensions. The clusters are of same sizes. This data set is shown in Fig. 4(h).
   (d) *Sph_10_2*: This data set, used in [28], consists of 500 two dimensional data points distributed over 10 different clusters. Some clusters are overlapping in nature. Each cluster consists of 50 data points. This data set is shown in Fig. 4(i).
   (e) *Sph_9_2*: This data set, used in [28], consists of 900 points in 2-dimensional space distributed over 9 clusters. Each cluster contains 100 data points and the clusters are highly overlapping to each other. This data set is displayed in Fig. 4(j).
(3) Group 3: This group of data sets contains well-separated clusters of different shapes, sizes and convexities.
   (a) *Pat1*: This data, used in [29], consists of 880 patterns. There is one non convex cluster present in this data set. This is shown in Fig. 4(k).
   (b) *Pat2*: This data set, used in [30], consists of 2 non-linear, non-overlapping and non-symmetric clusters. The data set is shown in Fig. 4(l).
   (c) *Long1*: This data set, used in [16], consists of 1000 data points distributed over 2 long clusters. This is shown in Fig. 4(m).
   (d) *Sizes5*: This data set, used in [16], consists of 1000 data points distributed over 4 squares. This is shown in Fig. 4(n).

Table 1
Results on different data sets by *GenClustMOO*, *MOCK*, VGAPS, *GenClustPESA2*, KM (*K*-means), *SL* (single linkage) clustering algorithms. Here *d*, *K*, *OC*, *FM* denote respectively the dimension, the number of clusters, obtained number of clusters and *F*-measure, respectively. Here each algorithm is executed thirty times and the best results are presented.

Data set	# points	*d*	*K*	*GenClustMOO*		MOCK		VGAPS		*GenClustPESA2*	
				OC	FS	OC	FS	OC	FS	OC	FS
Sym_5_2	850	2	5	5	1.00	5	1.00	5	1.00	5	1.00
Sym_3_2	600	2	3	3	0.97	2	0.78	3	0.97	3	0.97
Ellip_2_2	400	2	2	2	0.98	4	0.67	2	1.00	2	0.98
Ring_3_2	350	2	3	3	0.97	2	0.81	3	0.97	3	0.97
Rect_3_2	400	2	3	3	1.00	3	1.00	6	0.74	3	1.00
Sph_5_2	250	2	5	5	0.97	6	0.91	5	0.55	5	0.94
Sph_4_3	400	3	4	4	1.00	4	1.00	4	1.00	4	1.00
Sph_6_2	300	2	6	6	1.00	6	1.00	6	1.00	6	1.00
Sph_10_2	500	2	10	10	0.99	6	0.72	7	0.76	12	0.94
Sph_9_2	900	2	9	9	0.69	9	0.73	9	0.49	8	0.66
Pat1	557	2	3	3	0.95	10	0.55	4	0.42	3	0.95
Pat2	417	2	2	2	1.00	11	0.55	4	0.59	2	1.00
Long1	1000	2	2	2	1.00	2	1.00	3	0.50	2	1.00
Sizes5	1000	2	4	4	0.97	2	0.80	5	0.82	3	0.88
Spiral	1000	2	2	2	1.00	3	0.95	6	0.38	2	1.00
Square1	1000	2	4	4	0.99	4	0.99	4	0.99	4	0.99
Square4	1000	2	4	4	0.92	4	0.90	5	0.93	4	0.88
Twenty	1000	2	20	20	1.00	20	1.00	20	0.48	24	0.95
Forty	1000	2	40	40	1.00	40	1.00	2	0.095	42	0.98
Iris	150	4	3	3	0.79	2	0.78	3	0.76	3	0.93
Cancer	683	9	2	2	0.969	2	0.82	2	0.95	2	0.97
Newthy.	215	5	3	3	0.86	2	0.74	5	0.66	9	0.69
Wine	178	13	3	3	0.71	3	0.73	6	0.62	13	0.44
LiverDis.	345	6	2	2	0.67	2	0.67	2	0.70	5	0.60
LungCan.	33	56	2	2	0.80	7	0.44	3	0.74	4	0.84
Glass	214	9	6	6	0.49	5	0.53	5	0.53	5	0.53

(e) *Spiral*: This data set, used in [16], consists of 1000 data points distributed over 2 spiral clusters. This is shown in Fig. 4(o).
(f) *Square1*: This data set, used in [16], consists of 1000 data points distributed over 4 squared clusters. This is shown in Fig. 4(p).
(g) *Square4*: This data set, used in [16], consists of 1000 data points distributed over 4 squared clusters. This is shown in Fig. 4(q).
(h) *Twenty*: This data set, used in [16], consists of 1000 data points distributed over 20 small clusters. This is shown in Fig. 4(r).
(i) *forty*: This data set, used in [16], consists of 1000 data points distributed over 40 small clusters. This is shown in Fig. 4(s).

(4) Group 4: This group consists of seven real-life data sets obtained from [31].
(a) *Iris*: This 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 [32]. It has three classes Setosa, Versicolor and Virginica. It is known that two classes (Versicolor and Virginica) have a large amount of overlap while the class Setosa is linearly separable from the other two.
(b) *Cancer*: Here we use the Wisconsin Breast *Cancer* data set, it consists of 683 sample points. Each pattern has nine features corresponding to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. There are two categories in the data: malignant and benign. The two classes are known to be linearly separable.
(c) *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.
(d) *Wine*: This is the Wine recognition data consisting of 178 instances having 13 features resulting from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
(e) *LiverDisorder*: This is the Liver Disorder data consisting of 345 instances having 6 features each. The data has two categories.
(f) *LungCancer*: This data consists of 32 instances having 56 features each. The data describes 3 types of pathological lung cancers.
(g) *Glass*: This is a glass identification data consisting of 214 instances having 9 features (an Id# feature has been removed). The study of the classification of the types of glass was motivated by criminological investigation. 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.

4.2. Performance measures

In order to evaluate the performance of all the clustering algorithms quantitatively, here we have used a measure which is usually used to quantify the performance of a classification model [16]. It assumes that the class labels of each point are known before hand. The measure is defined below [16]:

(1) Precision = The fraction of a cluster that consists of objects of a specified class. Precision of cluster i with respect to class j is $precision(i, j) = p_{ij} = m_{ij}/m_i$, where m_{ij} is the number of points which belong to cluster i and class j both, and m_i is the total number of points in cluster i.
(2) Recall = The extent of which a cluster contains all objects of a specified class.

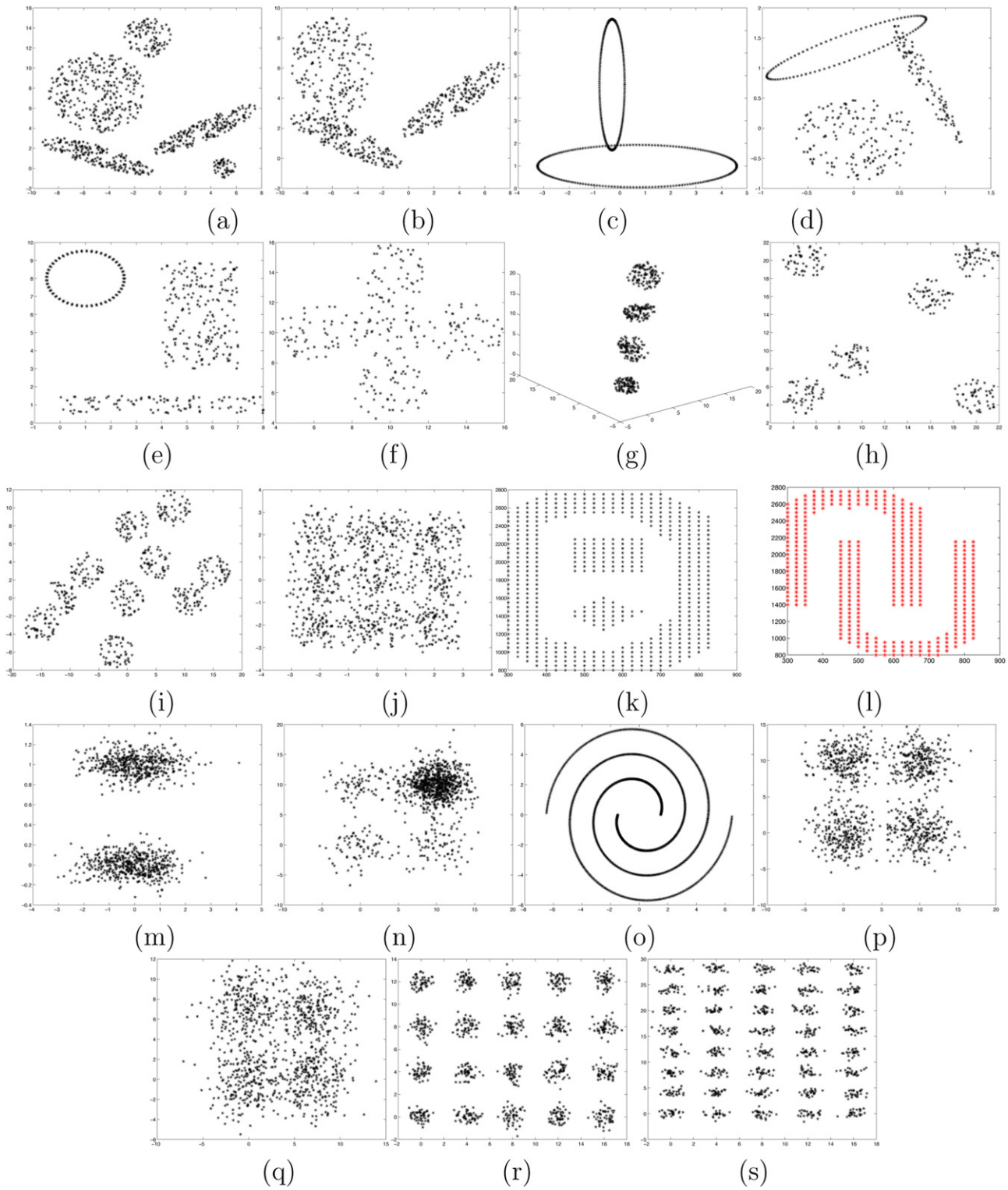

Fig. 4. (a) *Sym_5_2*, (b) *Sym_3_2*, (c) *Ellip_2_2*, (d) *Ring_3_2*, (e) *Rect_3_2*, (f) *Sph_5_2*, (g) *Sph_4_3*, (h) *Sph_6_2*, (i) *Sph_10_2*, (j) *Sph_9_2*, (k) *Pat1*, (l) *Pat2*, (m) *Long1*, (n) *Sizes5*, (o) *Spiral*, (p) *Square1*, (q) *Square4*, (r) *Twenty*, and (s) *forty*.

The recall of cluster i with respect to class j is $recall(i, j) = m_{ij}/m_j$, where m_j is the number of objects in class j.

(3) *F*-measure = A combination of both precision and recall that measures the extent to which a cluster contains only objects of a particular class and all objects of that class.

The *F*-measure of cluster i with respect to class j is $F(i, j) = (2 \times precision(i, j) \times recall(i, j))/(precision(i, j) + recall(i, j))$

(4) The overall *F*-measure of the whole partitioning is calculated as:

$$F = \sum_j \frac{m_j}{m} \max_i F(i, j),$$

Table 2
Final *F*-measure values on different data sets by *GenClustMOO*, *MOCK*, VGAPS, *GenClustPESA2*, KM (*K*-means), *SL* (single linkage) clustering algorithms. The best methods for each data set are marked in bold. Here each algorithm is executed thirty times and the average results with standard deviations are presented.

Data set	*GenClustMOO*	*GenClustMOO2*	MOCK	VGAPS	*GenClustPESA2*	KM	*SL*
Sym_5_2	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.815 ± 0.02	1.00 ± 0.00
Sym_3_2	**0.967** ± 0.01	0.561 ± 0.02	0.771 ± 0.012	**0.961** ± 0.021	**0.963** ± 0.011	0.951 ± 0.013	0.771 ± 0.0123
Ellip_2_2	0.971 ± 0.011	0.248 ± 0.012	0.667 ± 0.014	**1.00** ± 0.0101	0.968 ± 0.001	0.772 ± 0.008	0.723 ± 0.02
Ring_3_2	**0.964** ± 0.021	0.741 ± 0.017	0.801 ± 0.011	**0.961** ± 0.013	**0.961** ± 0.021	0.841 ± 0.011	0.808 ± 0.013
Rect_3_2	**1.00** ± 0.00	0.747 ± 0.012	**1.00** ± 0.00	0.736 ± 0.011	**1.00** ± 0.00	0.931 ± 0.021	**1.00** ± 0.00
Sph_5_2	**0.957** ± 0.021	0.841 ± 0.013	0.902 ± 0.0113	0.541 ± 0.011	0.936 ± 0.012	0.938 ± 0.015	0.661 ± 0.012
Sph_4_3	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00
Sph_6_2	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00
Sph_10_2	**0.981** ± 0.011	0.636 ± 0.021	0.717 ± 0.013	0.752 ± 0.0109	0.931 ± 0.021	0.891 ± 0.014	0.841 ± 0.011
Sph_9_2	0.681 ± 0.012	0.632 ± 0.011	**0.717** ± 0.009	0.481 ± 0.012	0.652 ± 0.018	0.683 ± 0.013	0.25 ± 0.014
Pat1	**0.946** ± 0.013	0.476 ± 0.012	0.547 ± 0.011	0.418 ± 0.014	**0.946** ± 0.009	0.618 ± 0.008	0.882 ± 0.011
Pat2	**1.00** ± 0.012	0.473 ± 0.009	0.545 ± 0.013	0.582 ± 0.021	**1.00** ± 0.00	0.754 ± 0.013	**1.00** ± 0.00
Long1	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.487 ± 0.021	**1.00** ± 0.00	0.5 ± 0.011	**1.00** ± 0.00
Sizes5	**0.968** ± 0.001	0.717 ± 0.011	0.791 ± 0.012	0.816 ± 0.013	0.883 ± 0.011	0.226 ± 0.021	0.181 ± 0.011
Spiral	**1.00** ± 0.00	0.553 ± 0.021	0.948 ± 0.011	0.373 ± 0.016	**1.00** ± 0.00	0.509 ± 0.011	0.504 ± 0.015
Square1	**0.999** ± 0.013	**0.999** ± 0.015	**0.999** ± 0.012	**0.999** ± 0.014	**0.99** ± 0.014	0.732 ± 0.021	0.368 ± 0.006
Square4	0.918 ± 0.014	0.919 ± 0.016	0.895 ± 0.011	**0.925** ± 0.013	0.878 ± 0.011	0.715 ± 0.015	0.368 ± 0.016
Twenty	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.479 ± 0.022	0.948 ± 0.015	0.809 ± 0.003	0.947 ± 0.009
Forty	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.095 ± 0.006	0.979 ± 0.015	0.798 ± 0.018	0.909 ± 0.023
Iris	0.788 ± 0.011	0.718 ± 0.016	0.775 ± 0.022	0.754 ± 0.013	**0.926** ± 0.015	0.887 ± 0.001	0.764 ± 0.009
Cancer	0.969 ± 0.009	0.755 ± 0.011	0.819 ± 0.014	0.953 ± 0.012	**0.979** ± 0.014	0.961 ± 0.013	0.688 ± 0.008
Newthy.	**0.863** ± 0.016	0.718 ± 0.0018	0.739 ± 0.014	0.659 ± 0.011	0.687 ± 0.015	0.677 ± 0.013	0.648 ± 0.009
Wine	0.709 ± 0.012	0.608 ± 0.006	**0.726** ± 0.002	0.617 ± 0.008	0.437 ± 0.012	0.709 ± 0.011	0.502 ± 0.013
LiverDis.	0.673 ± 0.002	0.672 ± 0.007	0.671 ± 0.012	**0.705** ± 0.009	0.603 ± 0.015	0.655 ± 0.013	0.672 ± 0.006
LungCan.	0.802 ± 0.014	0.664 ± 0.019	0.443 ± 0.011	0.741 ± 0.008	**0.843** ± 0.002	0.566 ± 0.005	0.236 ± 0.01
Glass	0.494 ± 0.012	0.495 ± 0.013	**0.534** ± 0.006	**0.534** ± 0.008	**0.534** ± 0.012	0.492 ± 0.014	0.422 ± 0.007

where the maximum is taken over all clusters i at all levels, m_j is the number of objects in class j, and m is the total number of objects. *F*-measure (*FM*) is a measure of the quality of a solution given the true clustering. For *F*-measure, the optimum score is 1, with higher scores being "better".

Here *F*-measure values are calculated and reported for the final solutions of *GenClustMOO*, MOCK, VGAPS, *GenClustPESA2*, *K*-means and single linkage clustering techniques.

5. Discussion of results

In *GenClustMOO*, a newly developed simulated annealing based MOO technique, AMOSA, is used as the underlying optimization technique. The parameters of the proposed *GenClustMOO* clustering technique are as follows: $SL = 100$ $HL = 50$, $iter = 50$, $T_{max} = 100$, $T_{min} = 0.00001$ and cooling rate, $\alpha = 0.9$. We have kept the number of sub-cluster centers per cluster $(C) = 10$. For the purpose of comparison, another automatic MOO clustering technique, MOCK [16] is also executed on the above mentioned data sets. The source code for MOCK is obtained from [33] and the default parameter values are used. In MOCK the final best solution is selected by GAP-statistics [34]. The number of clusters automatically determined by the proposed *GenClustMOO* and *MOCK* clustering techniques for all the above mentioned data sets are shown in Table 1. This table also contains the *F*-measure values of the final clusterings identified by these two algorithms. We have also compared our proposed technique with two traditional clustering techniques, *K*-means and single linkage clustering. These two algorithms are executed on all data sets with actual number of clusters. The final *F*-measure values are reported in Table 1. 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 *GenClustMOO*, with the underlying MOO strategy replaced by PESA2. The number of clusters and the corresponding *F*-measure values are reported in Table 1.

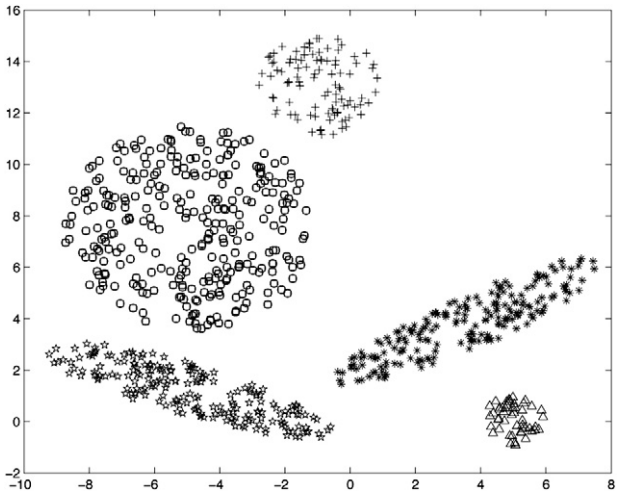

Fig. 5. Automatically clustered *Sym_5_2* after application of *GenClustMOO*/VGAPS clustering technique, and MOCK clustering technique for $K = 5$. Here we have used symbols '+', '□', '∗', '⋆' and '△' to represent different clusters.

In order to show that the proposed multiobjective clustering technique (*GenClustMOO*) performs better than a single objective version, variable string length point symmetry based clustering technique (VGAPS-clustering) [13], is also executed on the above mentioned data sets. The parameter values of VGAPS clustering are as follows: population size = 100, number of generations = 60 (executing the algorithm further did not improve the performance). Mutation and crossover probabilities are calculated adaptively. Note that here all the algorithms are executed for equal number of function evaluations. Total function evaluations performed by AMOSA based approach is equal to the total function evaluations of MOCK, VGAPS and *GenClustPESA2*. This is also same as the total number of iterations of *K*-means and *SL* algorithms. Each of the algorithms are executed thirty times and the average results along with the standard deviations are reported in Table 2. The best results of these thirty runs for all the algorithms are also reported in Table 1.

The clusters present in the data sets of group 1 are symmetrical in shape. Thus the proposed *GenClustMOO* is able to detect

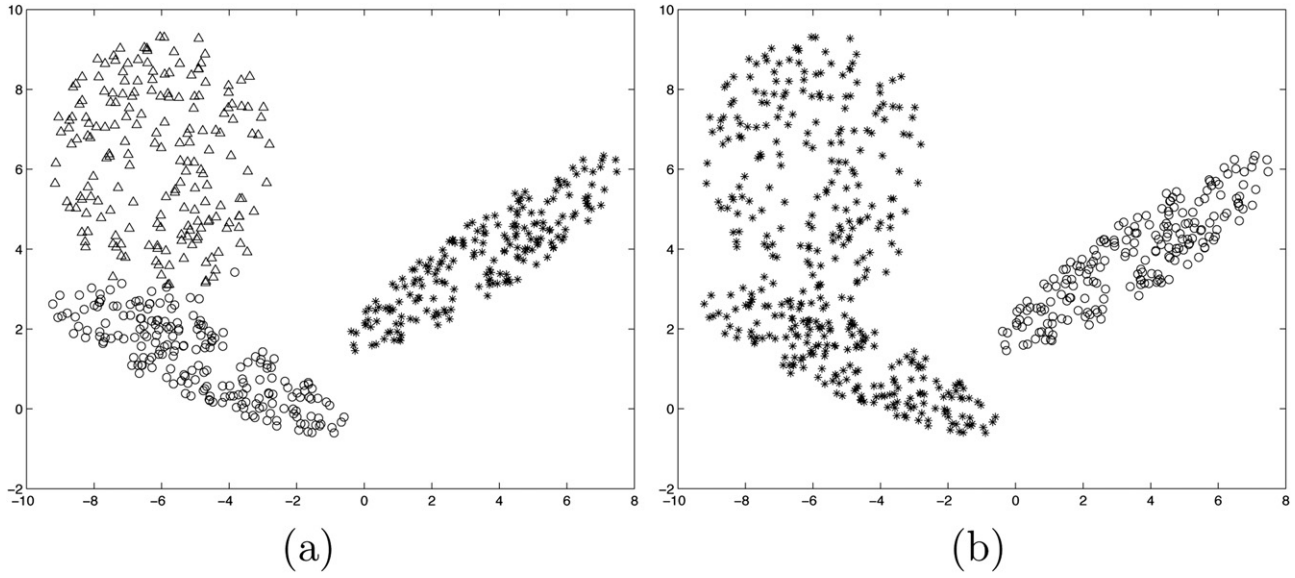

Fig. 6. Automatically clustered *Sym_3_2* after application of (a) *GenClustMOO* clustering technique/VGAPS clustering technique for $K = 3$; symbols used to denote different clusters: '△', '∗', '○'. (b) MOCK clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '○'.

the appropriate number of clusters and the appropriate clustering from all these data sets (refer to Table 1). The final clusterings identified by *GenClustMOO* for the data sets of this group are shown in Figs. 5(a), 6(a), 7(a), 8(a) and 9(a), respectively. MOCK [16] is also executed on these data sets. Except for data sets having well-separated clusters, e.g., *Sym_5_2* and *Rect_3_2*, MOCK fails to detect the appropriate clustering for the data sets in this group (refer to Table 1). The clusterings identified by MOCK for the five artificial data sets of this group are shown in Figs. 5(b), 6(b), 7(b), 8(b) and 9(b), respectively. VGAPS-clustering performs similarly to *GenClustMOO* clustering for most of the data sets of this group. It is able to identify the appropriate number of clusters and the appropriate clustering from data sets *Mixed_5_2*, *Ring_3_2*, *Ellip_2_2* and *Sym_3_2* (refer to Table 1). But it fails for data set *Rect_3_2*. The corresponding segmentation results are shown in Figs. 5(a), 6(a), 7(a), 8(c) and 9(a), respectively. For the data sets of this group, *GenClustPESA2* and *GenClustMOO* perform almost similarly (refer to Table 1). *K*-means fails to detect appropriate clusterings from these data sets. Single linkage succeeds in case of *Sym_5_2* and *Rect_3_2* but fails for other data sets. This is because single linkage can only detect well-separated clusters.

The clusters present in the data sets of group 2 are hyperspherical in shape. The proposed *GenClustMOO* is able to identify automatically the appropriate number of clusters and the appropriate clustering from these data sets (refer to Table 1). The clusterings obtained by *GenClustMOO* for these data sets are shown in Figs. 10(a), 11, 12, 13(a) and 14(a), respectively. MOCK is able to detect the appropriate number of clusters from three data sets of this group (refer to Table 1). The clusterings obtained by MOCK for the data sets of this group are shown in Figs. 10(b), 11, 12, 13(b) and 14(b), respectively. The *F*-measure values obtained by *GenClustMOO* for these data sets are also higher than or equal to

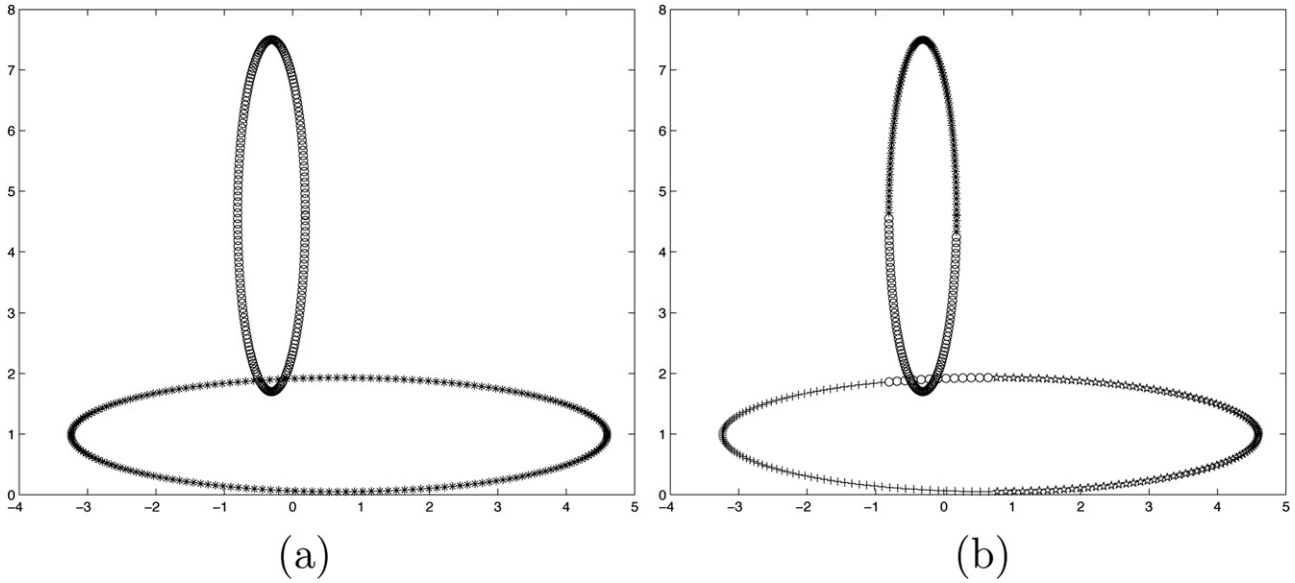

Fig. 7. Automatically clustered *Ellip_2_2* after application of (a) *GenClustMOO*/VGAPS clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '○'. (b) MOCK clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '○', '+', '⋆'.

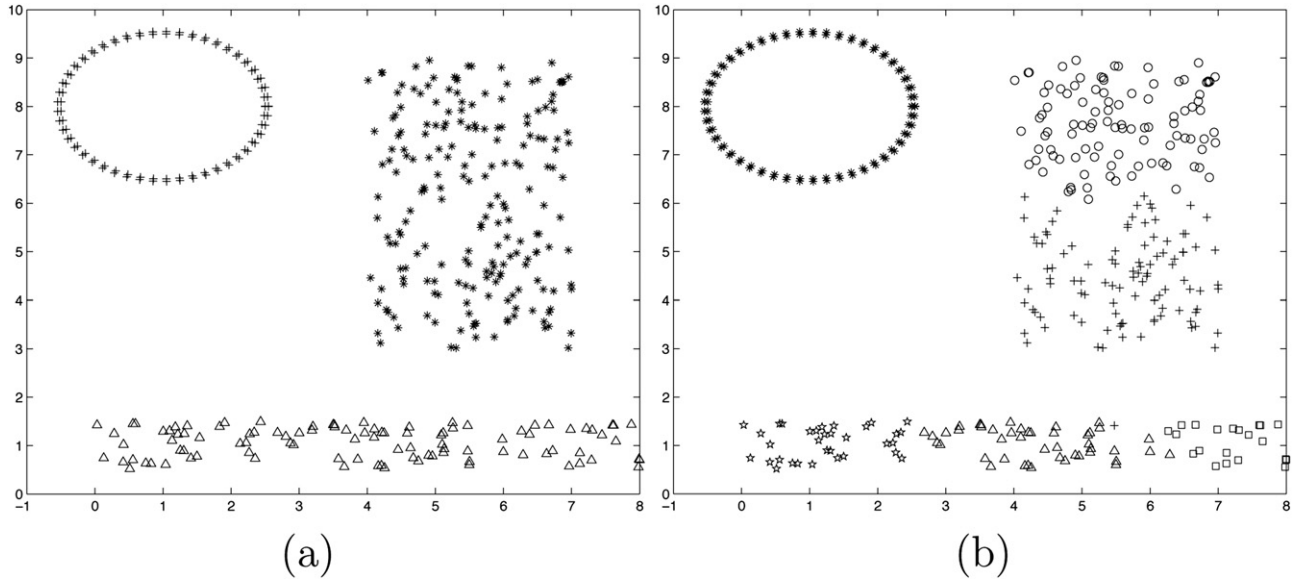

Fig. 8. Automatically clustered *Rect_3_2* after application of (a) *GenClustMOO* clustering technique/MOCK clustering technique for $K = 3$; symbols used to denote different clusters: '∗', '+', '△'. (b) VGAPS clustering technique for $K = 6$; symbols used to denote different clusters: '∗', '○', '★', '+', '△', '□'.

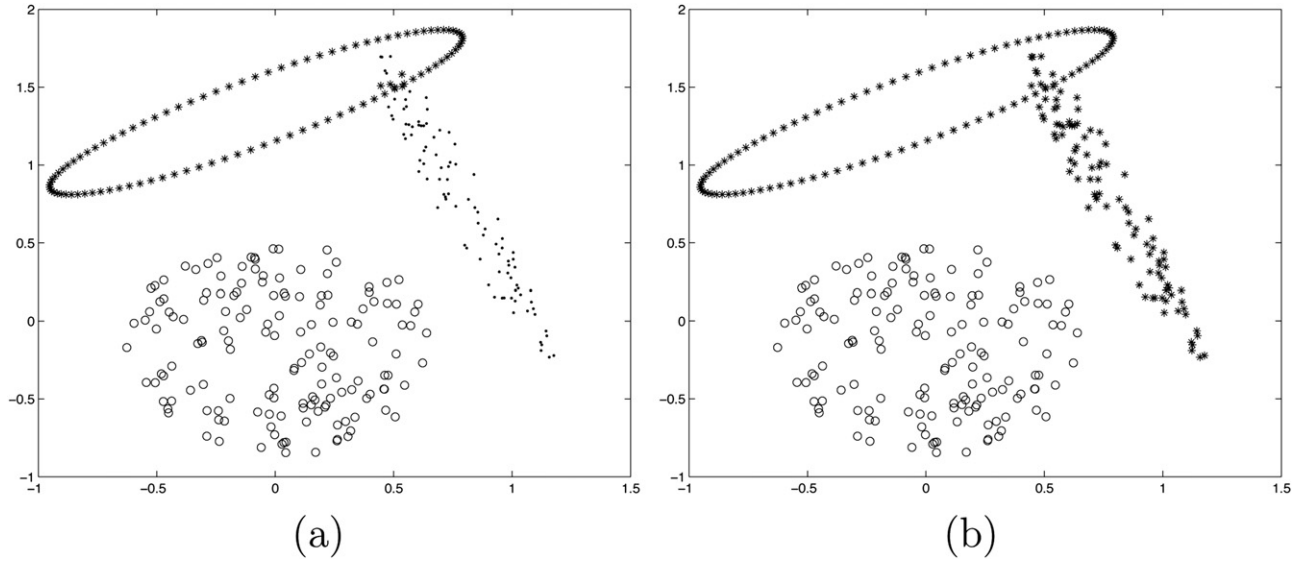

Fig. 9. Automatically clustered *Ring_3_2* after application of (a) *GenClustMOO*/VGAPS clustering technique for $K = 3$; symbols used to denote different clusters: '∗', '○', '·'. (b) MOCK clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '○'.

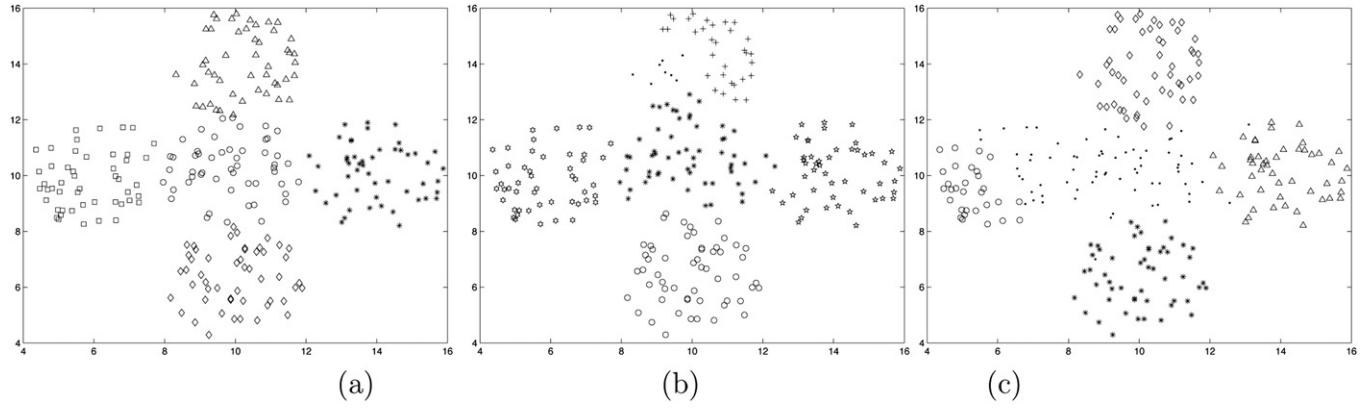

Fig. 10. Automatically clustered *Sph_5_2* after application of (a) *GenClustMOO* clustering technique for $K = 5$; symbols used to denote different clusters: '∗', '○', '◊', '□', '△'. (b) MOCK clustering technique for $K = 6$; symbols used to denote different clusters: '∗', '○', '+', '★', '·', 'hexagon'. (b) VGAPS clustering technique for $K = 5$; symbols used to denote different clusters: '∗', '○', '·', '△', '◊'.

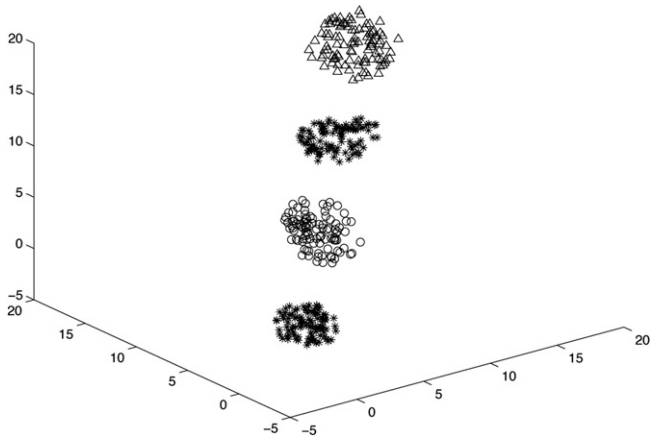

Fig. 11. Automatically clustered *Sph_4_3* after application of *GenClustMOO*, MOCK and VGAPS clustering techniques for $K = 4$; symbols used to denote different clusters: '∗', '○', '△', '★'.

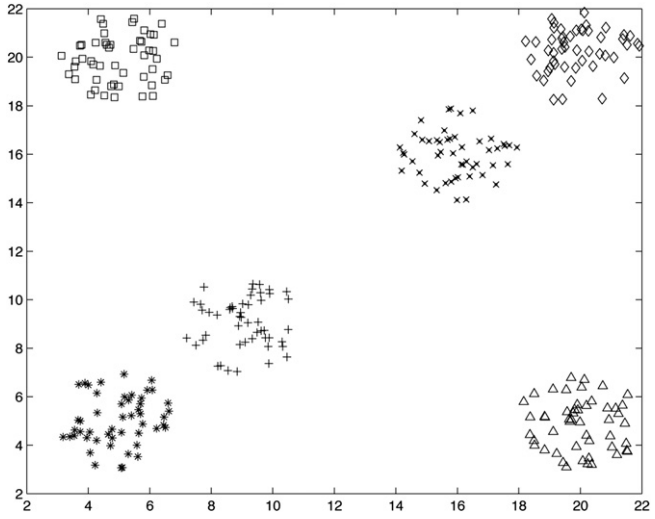

Fig. 12. Automatically clustered *Sph_6_2* after application of *GenClustMOO*, MOCK and VGAPS clustering techniques for $K = 6$; symbols used to denote different clusters: '∗', '△', '◊', '□', '×', '+'.

those obtained by MOCK (refer to Table 1). VGAPS-clustering is able to detect the appropriate number of clusters from *Sph_5_2*, *Sph_4_3*, *Sph_6_2* and *Sph_9_2* data sets. But the clusterings obtained by VGAPS for *Sph_5_2* and *Sph_9_2* data sets are not perfect. The clusterings obtained by VGAPS for *Sph_5_2*, *Sph_4_3*, *Sph_6_2*, *Sph_10_2* and *Sph_9_2* data sets are shown in Figs. 10(c), 11, 12, 13(c) and 14(c), respectively. VGAPS is not able to detect the proper clustering for *Sph_10_2* data set. *GenClustPESA2* clustering technique fails to detect the proper number of clusters from *Sph_10_2* and *Sph_9_2* data sets. But for the other three data sets, its performance is quite similar to that of *GenClustMOO*. For *Sph_9_2* data set, MOCK performs better than *GenClustMOO* (refer to Table 1). *K*-means is able to detect the appropriate clustering from *Sph_5_2*, *Sph_4_3*, *Sph_6_2* and *Sph_10_2* data sets. But it fails for *Sph_9_2* data set. Single linkage fails for most of these data sets except *Sph_4_3* and *Sph_6_2* which contain well-separated clusters.

The clusters present in the data sets of group 3 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 clusters and the appropriate clustering from all nine data sets of this group. The clusterings identified by *GenClustMOO* for all these nine data sets are shown in Figs. 15(a), 16(a), 17(a), 18(a), 19(a), 20(a), 21(a), 22(a) and 23(a), respectively. MOCK is able to detect the appropriate number of clusters from five out of nine data sets of this group. The clusterings are shown in Figs. 15(b), 16(b), 17(b), 18(b), 19(b), 20(b), 21(b), 22(b) and 23(b), respectively. VGAPS-clustering is able to detect the proper clustering and the proper number of clusters from only one out of nine data sets of this group. The clusterings are shown in Figs. 15(c), 16(c), 17(c), 18(c), 19(c), 20(c), 21(c), 22(c) and 23(c), respectively. *GenClustPESA2* clustering technique performs poorly for *Sizes5*, *Twenty* and *Forty* data sets (refer to Table 1). For other data sets of group 3, *GenClustPESA2* and *GenClustMOO* clustering techniques perform similarly. *K*-means in general fails for the data sets of these group. Single linkage is able to detect appropriate clustering from *Pat2*, *Long*, *Twenty* and *Forty* data sets (refer to Table 1).

For the real-life data sets no visualization is possible as these are higher dimensional data sets. Here we have reported the best *F*-measure values obtained by the six algorithms over five runs for all data sets. For *Iris* data set both *GenClustMOO* and VGAPS clustering techniques are able to detect the appropriate number of clusters from this data set. But the *F*-measure value attained by *GenClustMOO* clustering technique is slightly higher than that obtained by VGAPS (refer to Table 1). MOCK automatically identifies $K = 2$ number of clusters for this data set which is also often obtained for many other methods of *Iris* [35]. *F*-measure value obtained by *GenClustPESA2* is the best among all the clustering algorithms. For *Cancer* data set all the three algorithms are able to detect the appropriate number of clusters ($K = 2$) for this data set. But the *F*-measure value obtained by *GenClustMOO* is higher than those corresponding

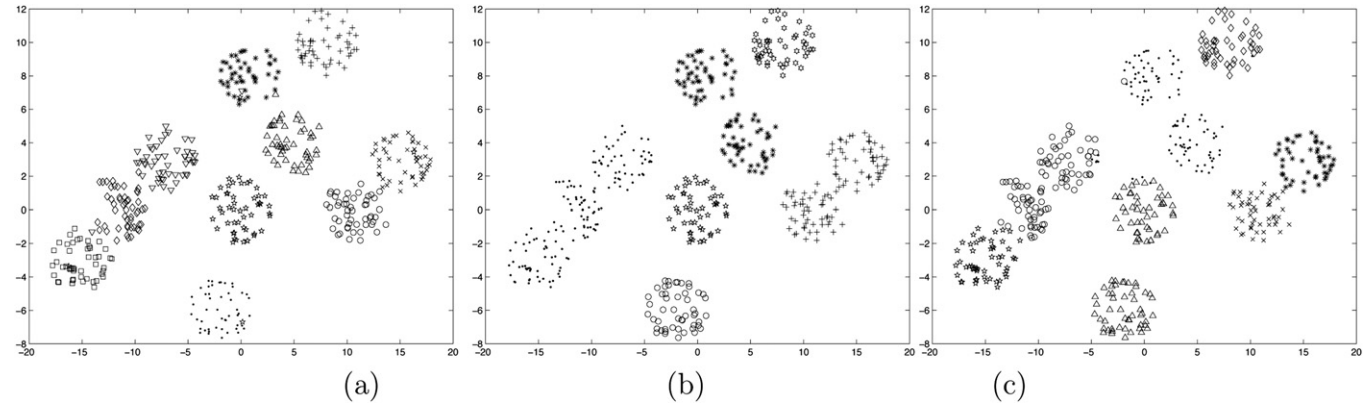

Fig. 13. Automatically clustered *Sph_10_2* after application of (a) *GenClustMOO* clustering technique for $K = 10$; symbols used to denote different clusters: '∗', '○', '□', '◊', '△', '+', '×', '·', '▽'. (b) MOCK clustering technique for $K = 6$; symbols used to denote different clusters: '∗', '○', '·', '+', '★', 'hexagon'. (c) VGAPS clustering technique for $K = 7$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△'.

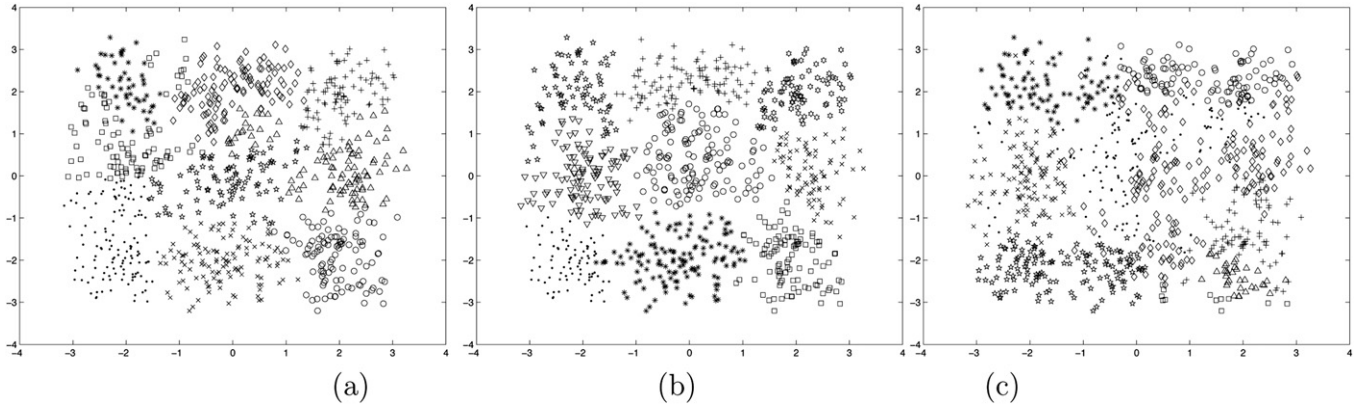

Fig. 14. Automatically clustered *Sph_9_2* after application of (a) *GenClustMOO* clustering technique for $K = 9$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '□', '+'. (b) MOCK clustering technique for $K = 9$; symbols used to denote different clusters: '∗', '○', '·', '+', '★', '×', '▽', '□', 'Hexagon'. (c) VGAPS clustering technique for $K = 9$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□'.

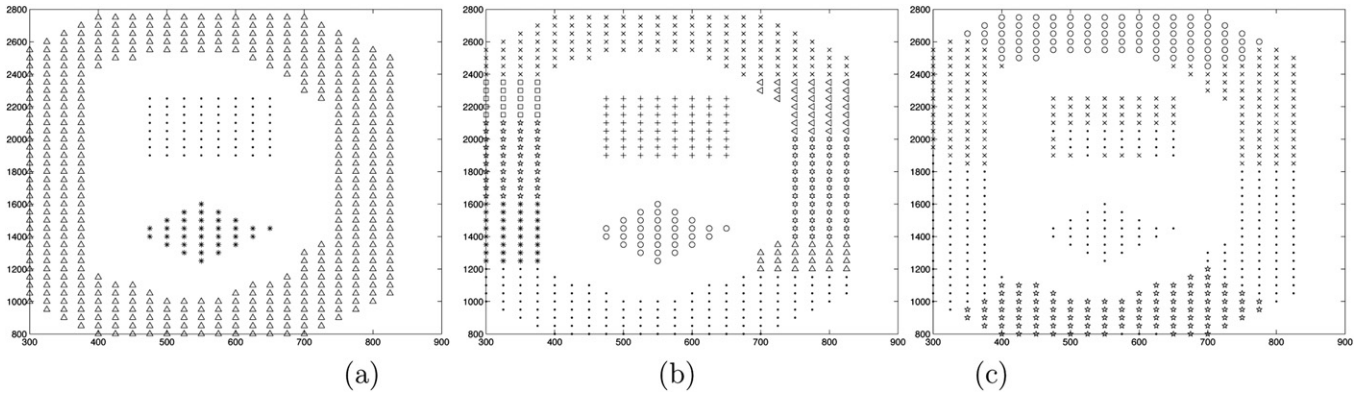

Fig. 15. Automatically clustered *Pat1* after application of (a) *GenClustMOO* clustering technique for $K = 3$; symbols used to denote different clusters: '∗', '·', '△'. (b) MOCK clustering technique for $K = 10$; symbols used to denote different clusters: '∗', '○', '·', '★', '×', '△', '+', '□', 'hexagon', '◁'. (c) VGAPS clustering technique for $K = 4$; symbols used to denote different clusters: '×', '○', '·', '★'.

to MOCK and VGAPS. This in turn indicates that the proposed algorithm provides more better clustering for this data set than MOCK and VGAPS. For *Newthyroid* data set, proposed *GenClustMOO* is able to detect the appropriate number of clusters from this data set but MOCK and VGAPS fail to do so. The *F*-measure value of the clustering identified by *GenClustMOO* for this data set is also much higher than those obtained by MOCK and VGAPS (refer to Table 1). For *Wine* and *LiverDisorder* data sets both *GenClustMOO* and MOCK clustering algorithms are able to detect the appropriate number of clusters. The *F*-measure values obtained by these two algorithms are also comparable. But VGAPS fails to identify the appropriate number of clusters for *Wine* data set. For *LungCancer* data set, only *GenClustMOO* is able to detect the appropriate number of clusters from this data set. Among the real-life data sets for only *Iris* and *LungCancer* data sets *GenClustPESA2* performs better than *GenClustMOO* (refer to Table 1). For *Newthyroid*, *Glass*, *Wine* and *LiverDisorder*, it performs poorly. Performance of *GenClustPESA2* for *Cancer* data set is quiet similar to that of *GenClustMOO*. *K*-means is able to detect the

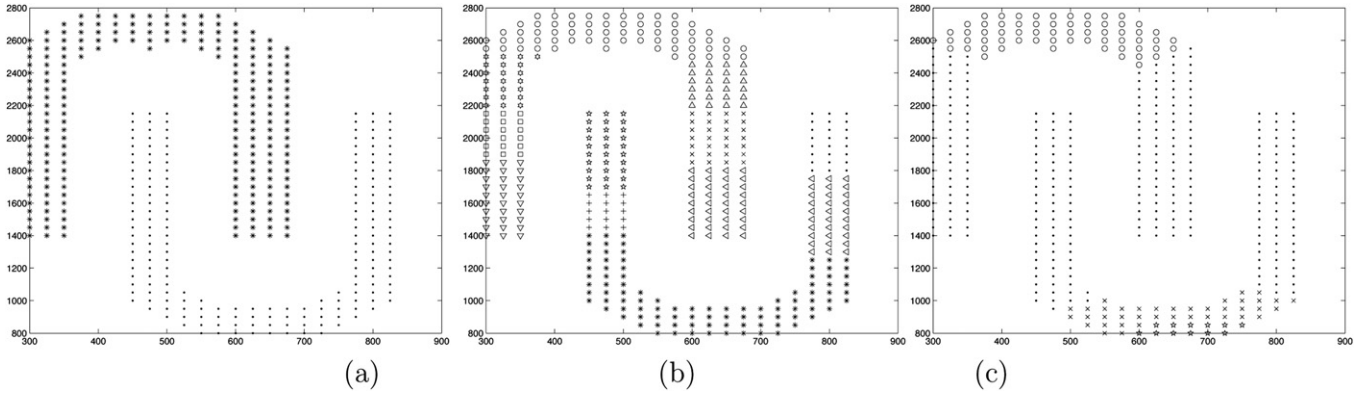

Fig. 16. Automatically clustered *Pat2* after application of (a) *GenClustMOO* clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '·'. (b) MOCK clustering technique for $K = 11$; symbols used to denote different clusters: '∗', '○', '·', '★', '×', '△', '+', '□', 'hexagon', '▽'. (c) VGAPS clustering technique for $K = 4$; symbols used to denote different clusters: '○', '·', '★', '×'.

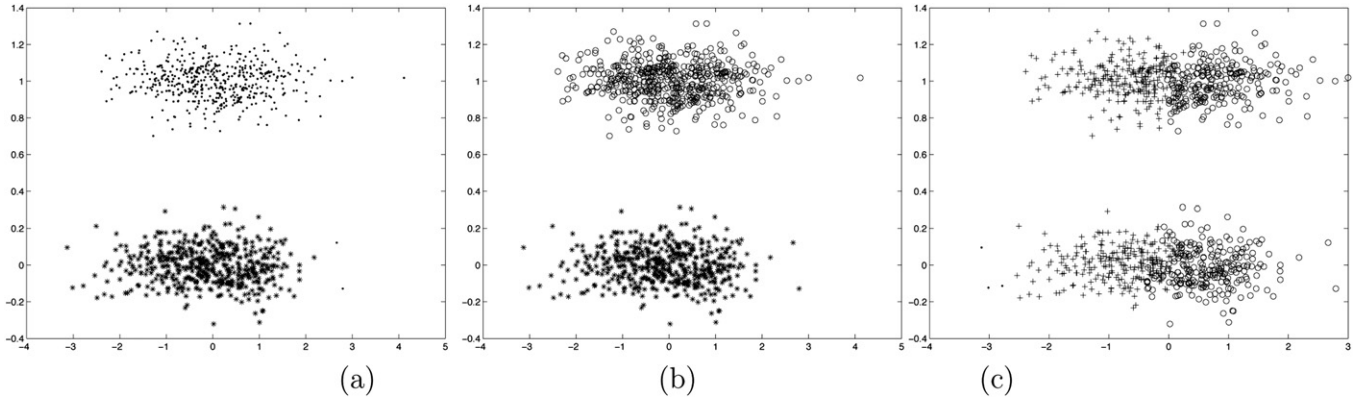

Fig. 17. Automatically clustered *Long1* after application of (a) *GenClustMOO* clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '·'. (b) MOCK clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '○'. (c) VGAPS clustering technique for $K = 3$; symbols used to denote different clusters: '+', '○', '·'.

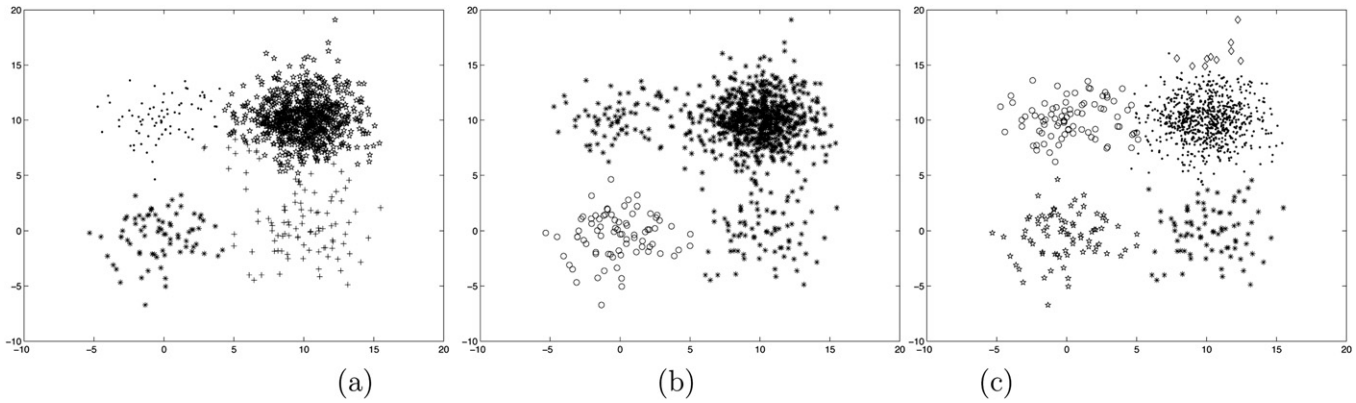

Fig. 18. Automatically clustered *Sizes5* after application of (a) *GenClustMOO* clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '·', '⋆', '+'. (b) MOCK clustering technique for $K = 2$; symbols used to denote different clusters: '○', '⋆'. (c) VGAPS clustering technique for $K = 5$; symbols used to denote different clusters: '∗', '○', '·', '⋆'.

appropriate clustering from *Iris* and *Cancer* data sets. single linkage in general is not able to detect appropriate clustering from any of these real-life data sets.

5.1. Results by spectral clustering technique [36] and DB-scan clustering technique [37,38]

In a part of the experiment we have also executed spectral clustering technique [36] and DB-scan clustering technique [37,38]. Spectral clustering toolbox is obtained from [36]. Here default parameter values are used. Spectral clustering technique is executed on *Sym_3_2*, *Ring_3_2*, *Ellip_2_2* and *Sph_4_3* data sets. The partitioning results are shown in Fig. 24(a)–(d), respectively. Results show that it fails for most of the data sets. Spectral clustering is only able to detect the appropriate partitioning from *Sph_4_3* data set but it fails for other three data sets. The code of DB-scan is downloaded from [39]. Here again default parameter values are used. The partitioning results obtained by this technique on *Sym_3_2*, *Ring_3_2*, *Ellip_2_2* and *Sph_4_3* data sets are shown in Fig. 25(a)–(d), respectively. These results again show that DB-scan fails for most of the data sets used here. It only succeeds for *Sph_4_3* data set but fails for other three data sets.

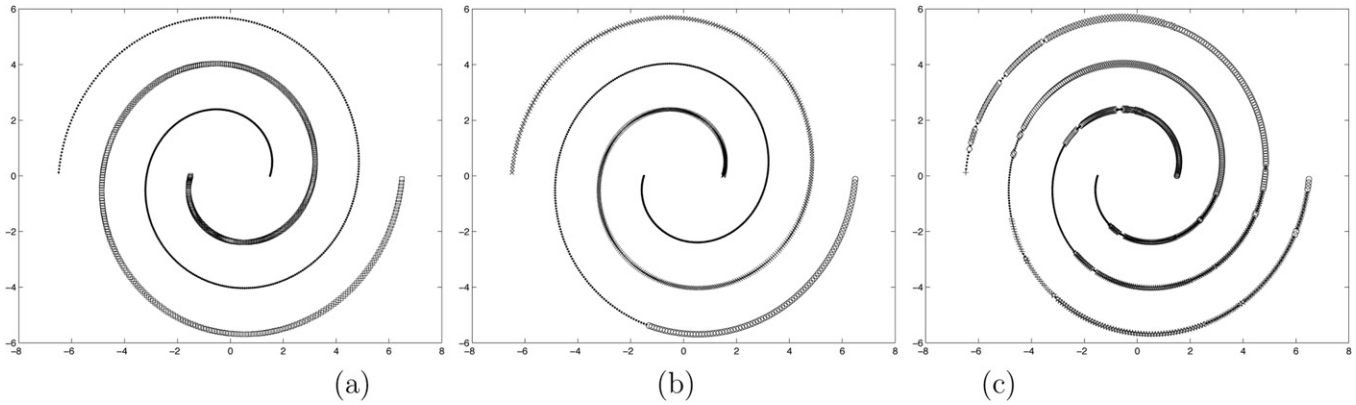

Fig. 19. Automatically clustered *Spiral* after application of (a) *GenClustMOO* clustering technique for $K = 2$; symbols used to denote different clusters: '○', '·'. (b) MOCK clustering technique for $K = 3$; symbols used to denote different clusters: '○', '·', '+'. (c) VGAPS clustering technique for $K = 6$; symbols used to denote different clusters: '∗', '○', '·', '⋆', '×', '+'.

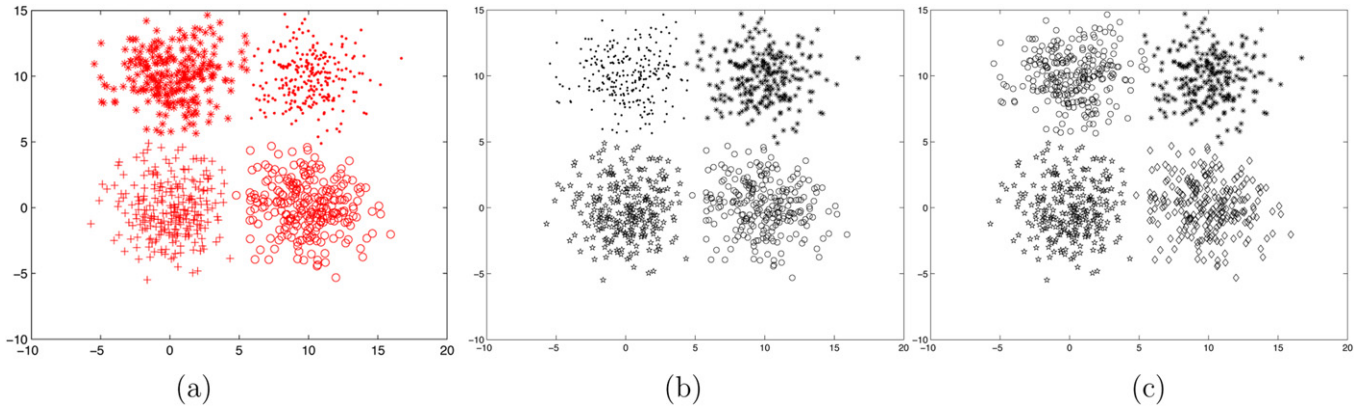

Fig. 20. Automatically clustered *Square1* after application of (a) *GenClustMOO* clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '○', '·', '+'. (b) MOCK clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '○', '·', '★'. (c) VGAPS clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '○', '◊', '★'.

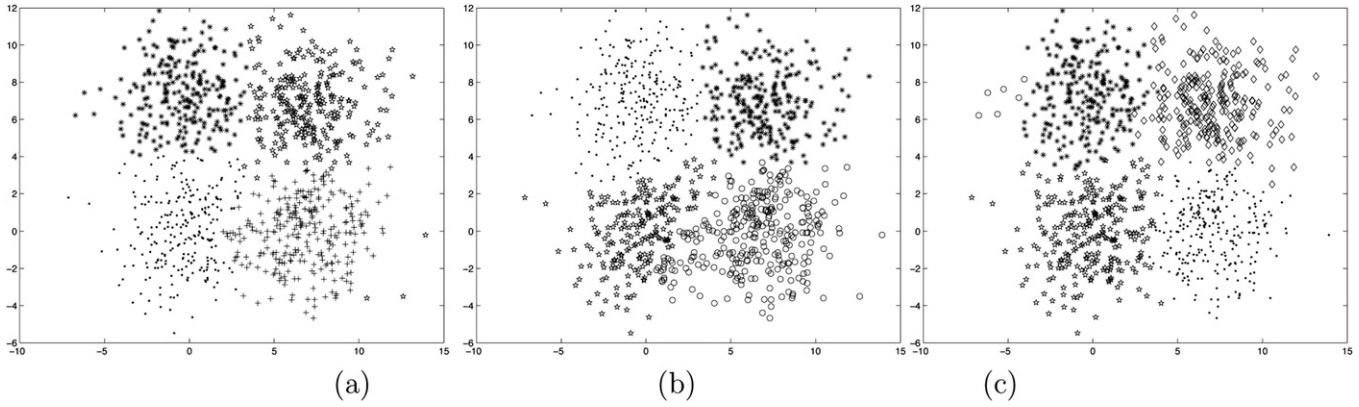

Fig. 21. Automatically clustered *Square4* after application of (a) *GenClustMOO* clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '·', '★', '+'. (b) MOCK clustering technique for $K = 4$; symbols used to denote different clusters: '∗', '○', '·', '★'. (c) VGAPS clustering technique for $K = 5$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★'.

5.2. Results using other methods of selecting a single solution from the archive

Here we have shown results using another approach of selecting the best solution from the *Archive* along with *GenClustMOO*. In this approach one single solution is selected from the *Archive* based on a cluster validity index, *Silhouette*-index [40]. To validate an obtained partitioning, several cluster validity indices [25] have been proposed in the literature. *Silhouette index* is a cluster validity index that is used to judge the quality of any clustering solution C. Suppose a represents the average distance of a point from the other points of the cluster to which the point is assigned, and b represents the minimum of the average distances of the point from the points of the other clusters. Now the silhouette width s of the point is defined as:

$$s = \frac{(b-a)}{\max\{a, b\}}. \tag{13}$$

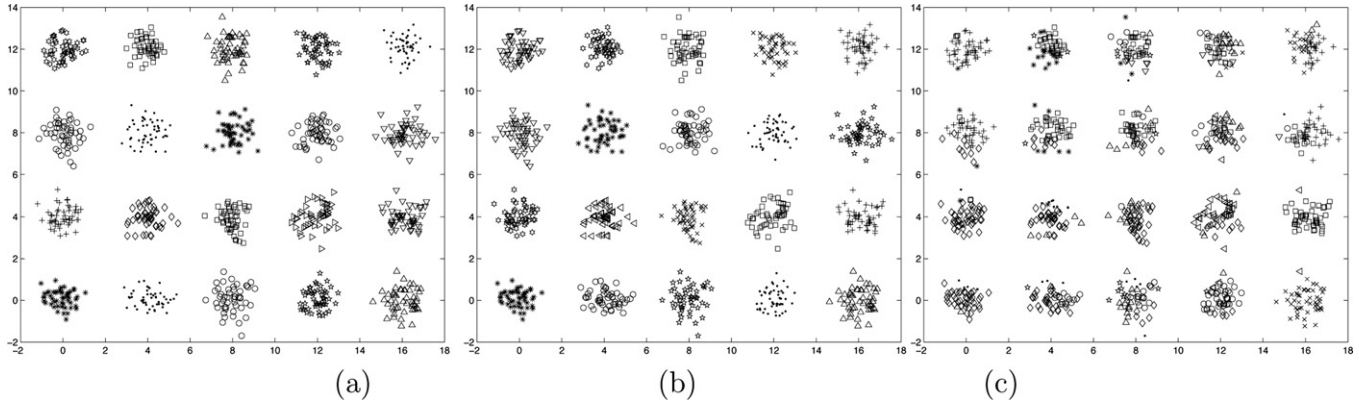

Fig. 22. Automatically clustered *Twenty* after application of (a) *GenClustMOO* clustering technique for $K = 20$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'. (b) MOCK clustering technique for $K = 20$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'. (c) VGAPS clustering technique for $K = 20$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'.

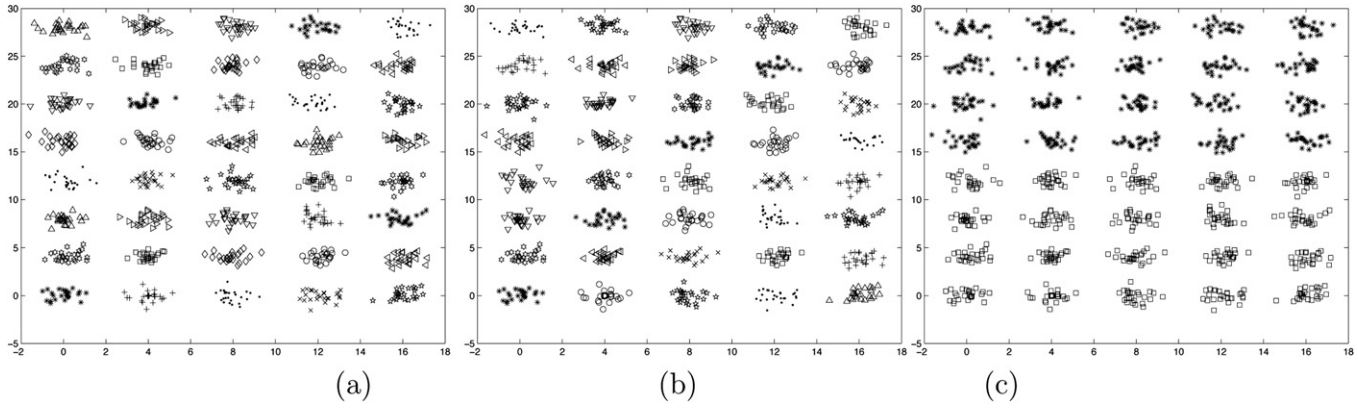

Fig. 23. Automatically clustered *Forty* after application of (a) *GenClustMOO* clustering technique for $K = 40$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'. (b) MOCK clustering technique for $K = 40$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'. (c) VGAPS clustering technique for $K = 2$; symbols used to denote different clusters: '∗', '○', '·', '◊', '★', '×', '△', '+', '□', 'hexagon', '∇', '◁', '▷'.

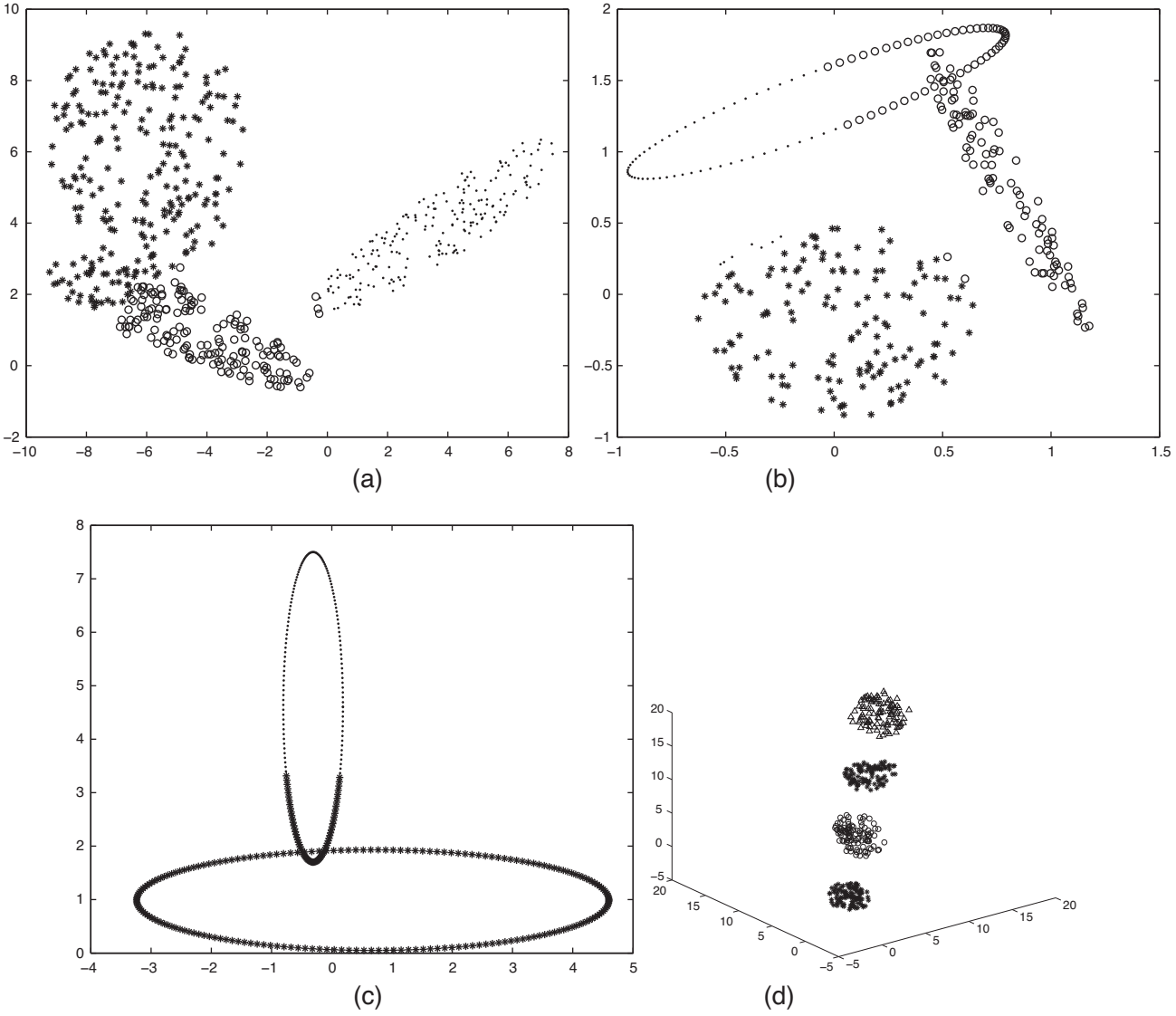

Fig. 24. Partitionings obtained by spectral clustering technique for (a) *Sym_3_2* (b) *Ring_3_2* (c) *Ellip_2_2* and (d) *Sph_4_3*. Symbols used to denote different clusters: (a) '∗', '○', '·' (b) '∗', '○', '·' (c) '∗', '·', (d) '∗', '○', '△', '∗'.

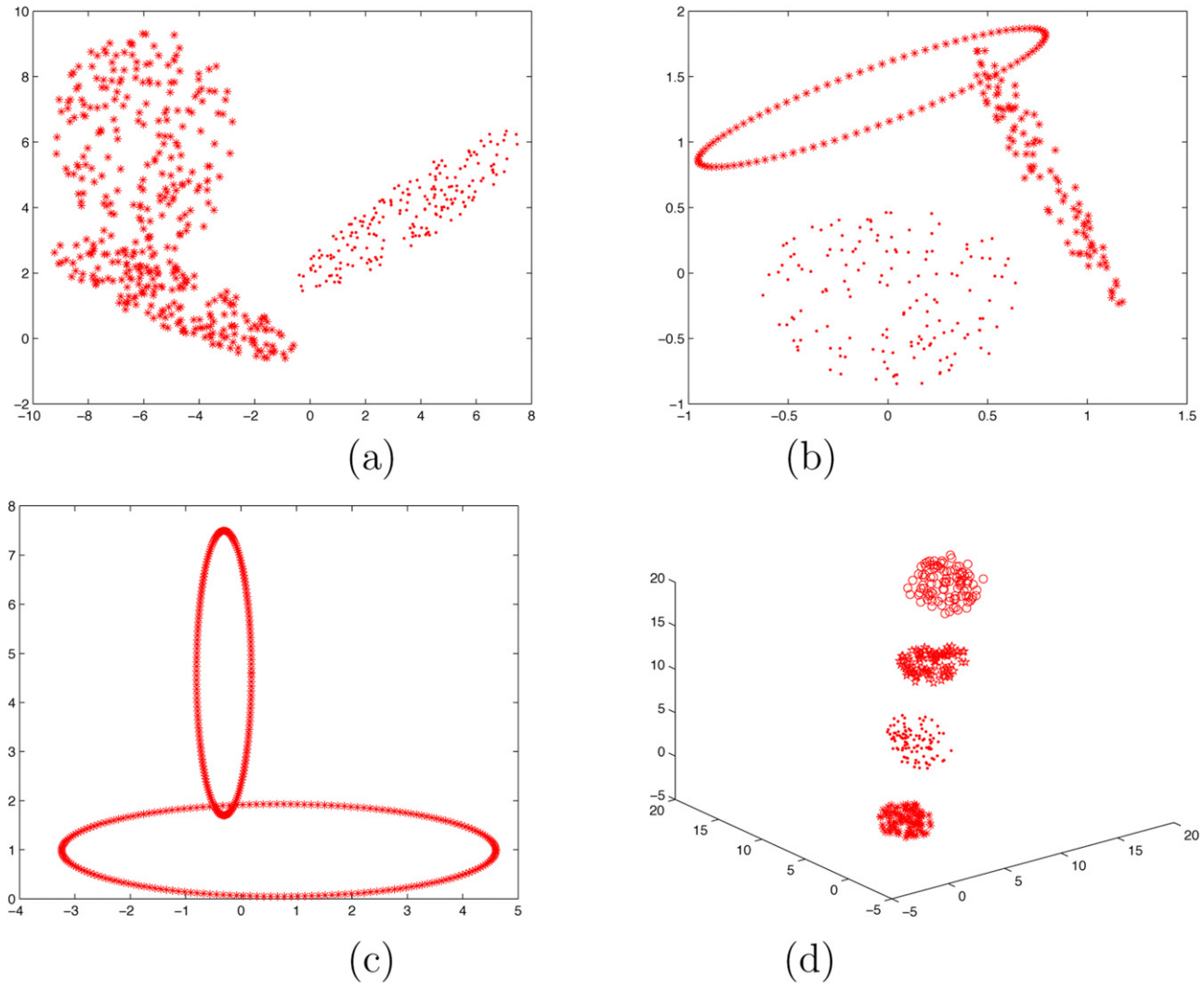

Fig. 25. Partitionings obtained by DB-scan clustering technique for (a) *Sym_3_2* (b) *Ring_3_2* (c) *Ellip_2_2* and (d) *Sph_4_3*. Symbols used to denote different clusters: (a) '∗', '·' (b) '∗', '·' (c) '∗' (d) '∗', '·', '○', '★'.

Silhouette index $s(C)$ is the average Silhouette width of all the data points (genes) and it reflects the compactness and separation of clusters. The value of Silhouette index varies from −1 to 1 and higher value indicates better clustering result.

Here for each solution in the *Archive* formed after execution of *GenClustMOO*, *Silhouette*-index value is computed. In order to obtain the proper partitioning, *Silhouette*-index value has to be maximized. Thus the solution of the *Archive* which corresponds to the maximum value of *Silhouette*-index is selected as the best solution. *GenClustMOO* with selection of the best solution using *Silhouette*-index is denoted as *GenClustMOO2*. *GenClustMOO* with selection of the best solution using the semi-supervised approach proposed in this paper in Section 3.6 is denoted as *GenClustMOO*.

Here the final *F-measure* values are reported for the solutions identified by *GenClustMOO*, and *GenClustMOO2* for all the data sets used here for experiment in Table 2. Results show that *GenClustMOO* performs the best for all the cases. This is because *GenClustMOO* selects that solution which performs best in terms of *Minkowski scores* for *test patterns* that constitute 10% of the whole data set. *Silhouette*-index is not able to detect the proper partitioning for many data sets. The solution with the optimal partitioning exists on the final Pareto optimal front. But *Silhouette*-index value for that solution is not optimum. Thus *GenClustMOO2* fails for most of these data sets.

5.3. Summary of results

Results on a wide variety of data sets show that the proposed *GenClustMOO* is able to detect the appropriate number of clusters and the appropriate partitioning from data sets having many different types of clusters. While results on data sets of group 1 and group 2 show that *GenClustMOO* is capable to identify various symmetrical shaped clusters (hyperspheres, linear, ellipsoidal, ring shaped, etc.) having overlaps, results on data sets of group 3 show its effectiveness on some well-separated clusters having any shape. Results on data sets of group 4 also show that *GenClustMOO* is capable to detect partitioning from real-life data sets of varying characteristics. The results on nineteen artificial and seven 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. The proposed *GenClustMOO* clustering technique is able to find out the proper clustering automatically where MOCK succeeds while VGAPS fails (data sets from Group 3) as well as where VGAPS succeeds while MOCK fails (data sets of Group 1). 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*

Table 3
Computation of the rankings for the four algorithms considered in the study over 26 data sets, based on the *F*-measure values obtained.

Data set	*GenClustMOO*	MOCK	VGAPS	*GenClustPESA2*
Sym_5_2	1.00**(1)**	1.00(1)	1.00(1)	1.00(1)
Sym_3_2	0.967(1)	0.771(4)	0.961(3)	0.963(2)
Ellip_2_2	0.971(2)	0.667(4)	1.00(1)	0.968(3)
Ring_3_2	0.964(1)	0.801(3)	0.961(2)	0.961(2)
Rect_3_2	1.00(1)	1.00(1)	0.736(2)	1.00(1)
Sph_5_2	0.957(1)	0.902(3)	0.541(4)	0.936(2)
Sph_4_3	1.00(1)	1.00(1)	1.00(1)	1.00(1)
Sph_6_2	1.00(1)	1.00(1)	1.00(1)	1.00(1)
Sph_10_2	0.981(1)	0.717(4)	0.752(3)	0.931(2)
Sph_9_2	0.681(2)	**0.717**(1)	0.481(4)	0.652(3)
Pat1	0.946(1)	0.547(2)	0.418(3)	0.946(1)
Pat2	1.00(1)	0.545(3)	0.582(2)	1.00(1)
Long1	1.00(1)	1.00(1)	0.487(2)	1.00(1)
Sizes5	0.968(1)	0.791(4)	0.816(3)	0.883(2)
Spiral	1.00(1)	0.948(2)	0.373(3)	1.00(1)
Square1	0.999(1)	**0.999**(1)	0.999(1)	0.999(1)
Square4	0.918(2)	0.895(3)	0.925(1)	0.878(4)
Twenty	1.00(1)	1.00(1)	0.479(3)	0.948(2)
Forty	1.00(1)	1.00(1)	0.095(3)	0.979(2)
Iris	0.788(2)	0.775(3)	0.754(4)	0.926(1)
Cancer	0.969(2)	0.819(4)	0.953(3)	0.979(1)
Newthyroid	0.863(1)	0.739(2)	0.659(4)	0.687(3)
Wine	0.709(2)	0.726(1)	0.617(3)	0.437(4)
LiverDisorder	0.673(2)	0.671(3)	0.705(1)	0.603(4)
LungCancer	0.802(2)	0.443(4)	0.741(3)	0.843(1)
Glass	0.494(2)	0.534(1)	0.534(1)	0.534(1)
average rank	1.346	2.69	2.385	1.846

clustering technique, utilizing PESA2 as the underlying optimization technique in *GenClustMOO* framework, performs similarly as *GenClustMOO* clustering technique using AMOSA for data sets with equisized, equi-density small number of clusters. *K*-means is able to detect hyperspherical shaped clusters. Single linkage is able to detect well-separated 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.

5.4. Statistical test

Here we have done some statistical tests guided by [41,42] to establish the superiority of the proposed clustering technique, *GenClustMOO1*. We have done Friedman statistical test [43] to detect whether the four clustering techniques, *GenClustMOO*, MOCK, VGAPS and *GenClustPESA2* used here for experiment perform similarly or not. It assigns ranks to each algorithm for each data set. It tests whether the measured average ranks are significantly different from the mean rank. Friedman test shows that measured average ranks and mean rank are different with a p value of 0.0166. The corresponding table is shown in Table 3. Finally Nemenyi's test [44] is performed to compare the clustering techniques pairwise. In each case $\alpha = 0.05$. Note that for all the cases the null hypotheses (the pairing algorithms perform similarly) are rejected as the corresponding p values are smaller than the α.

6. Conclusion

In this paper a new multiobjective (MO) clustering technique (*GenClustMOO*) is proposed which can automatically partition the data into an appropriate number of clusters. Each cluster is divided into several small hyperspherical subclusters and the centers of all these small sub-clusters are encoded in a string to represent the whole clustering. For assigning points to different clusters, these local sub-clusters are considered individually. For the purpose of objective function evaluation, these sub-clusters are merged appropriately to form a variable number of global clusters. Three objective functions, one reflecting the total compactness of the partitioning based on the Euclidean distance, the other reflecting the total symmetry of the clusters, and the last reflecting the cluster connectedness, are considered here. These are optimized simultaneously using AMOSA, a newly developed simulated annealing based multiobjective optimization method, in order to detect the appropriate number of clusters as well as the appropriate partitioning. The performance of the proposed algorithm named *GenClustMOO* is compared with the existing multiobjective clustering technique, MOCK, 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. Selecting the best solution(s) from the Pareto optimal front is an important problem in multiobjective clustering. Two methods of selecting a single solution from the Pareto optimal front is proposed here. Some new methods to choose the best solution from the Pareto optimal front have to be developed.

References

[1] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[2] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, 1974.
[3] S. Bandyopadhyay, U. Maulik, Nonparametric genetic clustering: comparison of validity indices, IEEE Transactions On Systems, Man and Cybernetics, Part C 31 (1) (2001) 120–125.

[4] U. Maulik, S. Bandyopadhyay, Performance evaluation of some clustering algorithms and validity indices, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (12) (2002) 1650–1654.
[5] H.C. Chou, M.C. Su, E. Lai, A new cluster validity measure and its application to image compression, Pattern Analysis and Applications 7 (July) (2004) 205–220.
[6] W. Wang, Y. Zhang, On fuzzy cluster validity indices, Fuzzy Sets and Systems 158 (October (19)) (2007) 2095–2117.
[7] P.B. Helena Brás Silva, J.P. da Costa, A partitional clustering algorithm validated by a clustering tendency index based on graph theory, Pattern Recognition 39 (May (5)) (2006) 776–788.
[8] M. Kim, R. Ramakrishna, New indices for cluster validity assessment, Pattern Recognition Letters 26 (November (15)) (2005) 2353–2363.
[9] R.H. Eduardo, F.F.E. Nelson, A genetic algorithm for cluster analysis, Intelligent Data Analysis 7 (2003) 15–25.
[10] S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition 2 (2002) 1197–1208.
[11] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, AnnArbor, 1975.
[12] W. Sheng, S. Swift, L. Zhang, 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 (December (6)) (2005) 1156–1167.
[13] S. Bandyopadhyay, S. Saha, A point symmetry based clustering technique for automatic evolution of clusters, IEEE Transactions on Knowledge and Data Engineering 20 (November (11)) (2008) 1–17.
[14] S. Bandyopadhyay, S. Saha, GAPS: a clustering method using a new point symmetry based distance measure, Pattern Recognition 40 (2007) 3430–3451.
[15] S. Saha, S. Bandyopadhyay, Application of a new symmetry based cluster validity index for satellite image segmentation, IEEE Geoscience and Remote Sensing Letters 5 (April (2)) (2008) 166–170.
[16] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 56–76.
[17] Y.J. Park, M.S. Song, A genetic algorithm for clustering problems, in: Proceeding of the 3rd Annual Conference on Genetic Programming, Paris, France, 1998, pp. 568–575.
[18] S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing based multi-objective optimization algorithm: AMOSA, IEEE Transactions on Evolutionary Computation 12 (June (3)) (2008) 269–283.
[19] S. Kirkpatrik, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[20] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE-Transactions on Pattern Analysis and Machine Intelligence 6 (6) (1984) 721–741.
[21] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[22] S. Saha, S. Bandyopadhyay, Some connectivity based cluster validity indices, Applied Soft Computing 12 (May (5)) (2012) 1555–1565, http://dx.doi.org/10.1016/j.asoc.2011.12.013.
[23] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognition 12 (1980) 261–268.
[24] S. Bandyopadhyay, An automatic shape independent clustering technique, Pattern Recognition 37 (2004) 33–45.
[25] B.S. Everitt, Cluster Analysis, third ed., Halsted Press, NY, 1993.
[26] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms, John Wiley and Sons, Ltd., England, 2001.
[27] D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene-expression data: a survey, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1370–1386.
[28] S. Bandyopadhyay, S.K. Pal, Classification and Learning Using Genetic Algorithms Applications in Bioinformatics and Web Intelligence, Springer-Verlag, Hiedelberg, Germany, 2007.
[29] S.K. Pal, S. Mitra, Fuzzy versions of Kohonen's net and MLP-based classification: performance evaluation for certain nonconvex decision regions, Information Sciences 76 (1994) 297–337.
[30] S. Mitra, S.K. Pal, Fuzzy multi-layer perceptron, inferencing and rule generation, IEEE Transactions on Neural Networks 6 (1995) 51–63.
[31] D.N.A. Asuncion, UCI Machine Learning Repository, 2007, http://www.ics.uci.edu/ mlearn/MLRepository.html
[32] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 3 (1936) 179–188.
[33] J. Handl, J. Knowles, Multiobjective Clustering with Automatic Determination of the Number of Clusters, 2007, http://dbkgroup.org/handl/mock/
[34] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters via the gap statistics, Journal of Royal Statistical Society 16 (2004) 1299–1323.
[35] U. Maulik, S. Bandyopadhyay, Genetic algorithm based clustering technique, Pattern Recognition 33 (2000) 1455–1465.
[36] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E.Y. Chang, PSC: Parallel Spectral Clustering, 2008, Software Available at http://www.cs.ucsb.edu/∼wychen/sc
[37] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, 1996, p. 226.
[38] M. Daszykowski, B. Walczak, D.L. Massart, Looking for natural patterns in data. Part 1: density based approach, Chemometrics and Intelligent Laboratory Systems 56 (2001) 83–92.
[39] http://www.chemometria.us.edu.pl/download/DBSCAN.M.
[40] P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1) (1987) 53–65, http://dx.doi.org/10.1016/0377-0427(87)90125-7.
[41] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30, URL http://dl.acm.org/citation.cfm?id=1248547.1248548
[42] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694, URL http://www.jmlr.org/papers/volume9/garcia08a/garcia08a.pdf
[43] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701, URL http://www.jstor.org/stable/2279372
[44] P. Nemenyi, Distribution-free Multiple Comparisons, Ph.D. thesis, 1963.