GAPS: A clustering method using a new point symmetry-based distance measure

Sanghamitra Bandyopadhyay, Sriparna Saha*

Abstract

In this paper, an evolutionary clustering technique is described that uses a new point symmetry-based distance measure. The algorithm is therefore able to detect both convex and non-convex clusters. Kd-tree based nearest neighbor search is used to reduce the complexity of finding the closest symmetric point. Adaptive mutation and crossover probabilities are used. The proposed GA with point symmetry (GAPS) distance based clustering algorithm is able to detect any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristic of symmetry. GAPS is compared with existing symmetry-based clustering technique SBKM, its modified version, and the well-known K-means algorithm. Sixteen data sets with widely varying characteristics are used to demonstrate its superiority. For real-life data sets, ANOVA and MANOVA statistical analyses are performed.

Keywords: Unsupervised classification; Genetic algorithm; Symmetry; Point symmetry-based distance; Kd-tree

1. Introduction

Clustering is a core problem in data-mining with innumerable applications spanning many fields. There are several applications of cluster analysis. When the data set available is unlabelled the classification problem is known as *unsupervised classification*. Clustering is an important unsupervised classification technique where a set of patterns, usually vectors in a multi-dimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. Cluster analysis is a difficult problem due to a variety of ways of measuring the similarity and dissimilarity concepts, which do not have a universal definition. Therefore, cluster seeking is experiment-oriented in the sense that clustering algorithms that can deal with all situations equally well are not yet available. A good review of clustering can be found in Ref. [1].

Many types of clustering algorithms have been developed, the prominent among them being the partitional methods, hierarchical and graph theoretic methods. Typical examples of the three methods are the well-known K-means algorithm, single linkage and the minimal spanning tree (MST)-based algorithms, respectively [2–6]. In order to improve the performance of the K-means algorithm, several improved K-means algorithms have been developed in the past several years. In Ref. [7], Kövesi et al. presented a stochastic K-means algorithm to improve the clustering result of K-means. Based on the Kd-tree data structure [8], Kanungo et al. [9] presented an improved K-means algorithm which can speed up the time performance while preserving the same clustering result as in the K-means algorithm. In Ref. [10], the global K-means algorithm is presented which is an incremental approach to clustering that dynamically adds one cluster center at a time through a deterministic global search procedure consisting of N (N is the size of the data set) executions of the K-means algorithm. In Ref. [11], an algorithm based on K-means, namely, SMCK-means, is proposed for circular invariant clustering of vectors. Fuzzy C-shells clustering (FCS) methods have been proposed in Refs. [12,13] which are well established for detecting and representing hyperspherical (specially ellipsoidal) clusters. Some more clustering algorithms have been found in Refs. [14–16]. Other algorithms have been developed that are appropriate for large data sets [17,18]. In Ref. [19], a fuzzy clustering algorithm is proposed for the extraction of a smooth curve from the unordered noisy data. An algorithm for extracting high quality polygonal contours from connected objects in binary images is described in Ref. [20]. In Ref. [21], some level set methods are introduced to identify density peaks and valleys in density landscape for data clustering. The method relies on advancing contours to form cluster cores. A generic iterative clustering scheme is proposed in Ref. [22] that, coupled with some particular reweighting scheme, may indeed bring improvements over "classical"clustering from the theoretical standpoint. A new overlapping clustering algorithm, which partitions n objects into k non-exhaustive clusters that may overlap with each other is introduced in Ref. [23]. A modification of fuzzy c-means algorithm [24] is proposed in Ref. [25], in which distances between cluster centers and the data are determined by the density of the data itself. A few typical examples of non-convex clusters are shown to be correctly identified by this algorithm. However, this method is unable to identify highly overlapping clusters.

Neural networks, for example, competitive learning networks [26], self-organizing feature maps (SOFM) [27] and adaptive resonance theory (ART) networks [28,29] have also often been used to cluster data. Both SOFM and ART are suitable for detecting only hyper-spherical clusters [30]. A two-layer network that employs regularized Mahalanobis distance to extract hyperellipsoidal clusters was proposed in Ref. [31]. In Ref. [32], a clustering method is proposed based on SOFM with quadratic neurons. The network is updated in an unsupervised manner to cluster data into hyperellipsoidal-shaped or hyperspherical-shaped clusters based on the underlying structure of the data set. But it will fail for straight line or shell type of clusters. In Ref. [33], ART based clustering algorithm is used to cluster well-separated and non-overlapping clusters with arbitrary shapes. But it will fail for data sets where clusters are not at all well separated. All the algorithms mentioned above have their own merits and disadvantages.

Several clustering algorithms with different distance measures have been developed for clustering data sets with different geometric shapes [34–38]. These algorithms were used to detect compact clusters [34], straight lines [34,36], ring-shaped clusters [37], or contours with polygonal boundaries [12,38]. However, the performance of these algorithms were poor when the data had clusters of other shapes. In Ref. [39] a clustering technique is proposed which can automatically detect any number of well-separated clusters which may be of any shape, convex and/or non-convex. But it fails for overlapping clusters.

In order to mathematically identify clusters in a data set, it is usually necessary to first define a measure of similarity or proximity which will establish a rule for assigning patterns to the domain of a particular cluster centroid. The measure of similarity is usually data dependent. One commonly used measure of similarity is the Euclidean distance D between two patterns $\overline{x}$ and $\overline{z}$ defined by $D=\|\overline{x}-\overline{z}\|$. Smaller Euclidean distance means better similarity and vice versa. This measure has been used in the K-means clustering algorithm [6]. By this algorithm hyperspherical clusters of almost equal size can be easily identified. This measure fails when clusters tend to develop along principal axes. To take care of hyperellipsoidal-shaped clusters the Mahalanobis distance from $\overline{x}$ to $\overline{m}$, $D(\overline{x},\overline{m})=(\overline{x}-\overline{m})^{\mathrm{T}}\sum^{-1}(\overline{x}-\overline{m})$, is one of the popular choices. Here $\overline{x}$ represents an input pattern, the matrix $\sum$ is the covariance matrix of a pattern population constituting a particular cluster, $\overline{m}$ is the mean vector of the vectors which are in the same cluster. One of the difficulties associated with Mahalanobis distance as a similarity measure is that one has to recompute the inverse of the sample covariance matrix every time a pattern changes its cluster domain; this is computationally expensive. The Mahalanobis distance was used in Ref. [31] to extract hyperellipsoidal clusters.

It may be noted that one of the basic feature of shapes and objects is symmetry. Symmetry is considered a pre-attentive feature which enhances recognition and reconstruction of shapes and objects [40]. Almost every interesting area around us consists of some generalized form of symmetry. As symmetry is so common in the natural world, it can be assumed that some kind of symmetry exists in the clusters also. Based on this, Su and Chou have proposed a symmetry-based clustering technique [41]. Here they have assigned points to a particular cluster if they present a symmetrical structure with respect to the cluster center. A new type of non-metric distance, based on point symmetry, is proposed which is used in a K-means based clustering algorithm, referred to as symmetry-based K-means (SBKM) algorithm. SBKM is found to provide good performance on different types of data sets where the clusters have internal symmetry. However, it can be shown that SBKM will fail for some data sets where the clusters themselves are symmetrical with respect to some intermediate point since the point symmetry distance ignores the Euclidean distance in its computation. Though this has been mentioned in a subsequent paper by Chou et al. [42] where they have suggested a modification, the modified measure has the same limitation of the previous one [41]. No experimental results have been provided in Ref. [42].

Based on the above observations, a new point symmetry-based distance (PS-distance) is proposed in this article which incorporates both the Euclidean distance as well as a measure of symmetry. For reducing the complexity of computing the PS-distance, use of Kd-tree [8] is proposed in this article. As already mentioned, K-means is a widely used clustering algorithm that has also been used in conjunction with the point-symmetry-based distance measure in Ref. [41]. However, K-means is known to get stuck at sub-optimal solutions depending on the choice of the initial cluster centers. In order to overcome this limitation, genetic algorithms (GAs) have been used for solving the undulying optimization algorithm [43]. GAs [44] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and having a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In view of the advantages of the GA-based clustering method [43] over the standard K-means, the former has been used in this article. In the proposed GA with point symmetry (GAPS) distance clustering technique, the assignment of points to different clusters are done based on the newly proposed point symmetry distance rather the Euclidean distance. This enables the proposed algorithm to detect both convex and non-convex clusters of any shape and sizes as long as the clusters do have some symmetry property.

Experimental results comparing the performance of the GAPS clustering algorithm, SBKM algorithm, SBKM with modified PS distance as suggested in Ref. [42] (which is subsequently referred to as Mod-SBKM) and K-means algorithm are provided for several artificial and real-life data sets. For the purpose of comparison, analysis of variance (ANOVA) [45] and multivariate analysis of variance (MANOVA) [46] techniques have been used. Experimental results demonstrate the effectiveness of GAPS clustering for different types of data sets, both where K-means performs well while SBKM fails and vice versa. Additionally, GAPS is also able to detect the proper clustering from the data sets having clusters of significantly different sizes where all the other three algorithms fail to do this. The superiority of the proposed GAPS clustering technique is also demonstrated on seven higher dimensional real-life data sets of varying characteristics. Results on 16 artificially generated and real-life data sets establish the fact that GAPS is well-suited to detect clusters with symmetrical shapes irrespective of any geometric structure, size and overlapping.

This paper is organized as follows. Section 2 describes in detail the existing point symmetry distance and the SBKM clustering algorithm. Section 3 describes in detail the proposed point symmetry distance. Section 4 describes the use of Kd-tree based approximate nearest neighbor (ANN) search. A description of the newly proposed GA with point symmetry distance based (GAPS) clustering technique is given in Section 5. In Section 6 complexity analysis of both the existing SBKM and the newly proposed GAPS clustering technique is given. Sections 7 and 8 provide the experimental results. Section 9 concludes the paper.

2. Symmetry-based distance measures

As already discussed, a point symmetry distance was proposed by Su and Chou in Ref. [41]. This is defined as follows: given N patterns, $\overline{x}_j$, $j = 1, \ldots, N$, and a reference vector $\overline{c}$ (e.g., a cluster centroid), the point symmetry distance (PS-distance) between a pattern $\overline{x}_j$ and the reference vector $\overline{c}$ is defined as

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

where the denominator term is used to normalize the distance so as to make it insensible to the Euclidean distances $\|\overline{x}_j - \overline{c}\|$ and $\|\overline{x}_i - \overline{c}\|$. It may be noted that the numerator of Eq. (1) is actually the distance between the mirror image point of $\overline{x}_j$ with respect to $\overline{c}$ and its nearest neighbor in the data set. If the right hand term of the above equation is minimized when $\overline{x}_i = \overline{x}_{j^*}$, then the pattern $\overline{x}_{j^*}$ is denoted as the symmetrical pattern relative to $\overline{x}_j$ with respect to $\overline{c}$. Here it can be easily seen that the above equation is minimized when the pattern $\overline{x}_i = (2\overline{c} - \overline{x}_j)$ exists in the data set (i.e., $d_s(\overline{x}_j, \overline{c}) = 0$). This idea of point symmetry is very simple and intuitive. Based on this point symmetry-based distance, Su and Chou have proposed a clustering algorithm which mimics the K-means algorithm but assigns the patterns to a particular cluster depending on the symmetry-based distance d_s rather than the Euclidean distance [41], only when d_s is greater than some user specified threshold θ. Otherwise, assignment is done according to the Euclidean distance, as in normal K-means. The algorithm is discussed in detail in Fig. 1.

It is evident from Eq. (1) that this similarity measure can be useful to detect clusters which have symmetrical shapes. But this clustering algorithm will fail for data sets where clusters themselves are symmetrical with respect to some intermediate point. Note that minimization of $d_s(\overline{x}_j, \overline{c})$ means minimization of its numerator and maximization of its denominator. In effect, if a point $\overline{x}_j$ is almost equally symmetrical with respect to two centroids $\overline{c}_1$ and $\overline{c}_2$, it will be assigned to that cluster that is the farthest. This is intuitively unappealing. In the example shown in Fig. 2, there are three clusters which are well separated. The centers of the clusters are denoted by $\overline{c}_1$, $\overline{c}_2$ and $\overline{c}_3$, respectively. Let us take the point $\overline{x}$. After application of K-means algorithm point $\overline{x}$ is being assigned to the cluster 1. But when SBKM is applied on the result given by K-means algorithm, the following will happen. The symmetrical point of $\overline{x}$ with respect to $\overline{c}_1$ is $\overline{x}_1$. Since it is the first nearest neighbor of the point $\overline{x}_1^* = (2 \times \overline{c}_1 - \overline{x})$. Let the Euclidean distance between $\overline{x}_1^*$ and $\overline{x}_1$ be d_1. Therefore, the symmetrical distance of $\overline{x}$ with respect to $\overline{c}_1$ is

$$d_s(\overline{x}, \overline{c}_1) = \frac{d_1}{d_e(\overline{x}, \overline{c}_1) + d_e(\overline{x}_1, \overline{c}_1)}, \tag{2}$$

where $d_e(\overline{x}, \overline{c_1})$ and $d_e(\overline{x}_1, \overline{c}_1)$ are the Euclidean distances of $\overline{x}$ and $\overline{x}_1$ from $\overline{c}_1$, respectively. Similarly symmetrical point of $\overline{x}$ with respect to $\overline{c}_2$ is $\overline{x}_2$. And the symmetrical distance of $\overline{x}$ with respect to $\overline{c}_2$ becomes

$$d_s(\overline{x}, \overline{c}_2) = \frac{d_2}{d_e(\overline{x}, \overline{c}_2) + d_e(\overline{x}_2, \overline{c}_2)}. \tag{3}$$

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

Chou et al. have noted the above-mentioned limitation of the measure proposed in Ref. [41], and have suggested a modified measure d_c in Ref. [42] that is defined as follows:

$$d_c(\overline{x}_j, \overline{c}) = d_s(\overline{x}_j, \overline{c}) d_e(\overline{x}_j, \overline{c}), \tag{4}$$

where $d_s(\overline{x}_j, \overline{c})$ is the PS-distance of $\overline{x}_j$ with respect to $\overline{c}$, and $d_e(\overline{x}_j, \overline{c})$ denotes the Euclidean distance between $\overline{x}_j$ and $\overline{c}$. No experimental results are provided in Ref. [42] corresponding to this new measure. A little thought will show that even this modification will not work for the situation shown in Fig. 2. Moreover, if the term $d_s(\overline{x}_j, \overline{c})$ becomes 0 then there will be no effect of the Euclidean distance. We refer to the clustering algorithm based on this modified measure as Mod-SBKM algorithm. Let $\overline{x}_j^*$ be the symmetrical point of $\overline{x}_j$ with respect to $\overline{c}$. Therefore from Eq. (4) we obtain

$$d_c(\overline{x}_j, \overline{c}) = \frac{d_{symm}(\overline{x}_j, \overline{c})}{d_e(\overline{x}_j, \overline{c}) + d_e(\overline{x}_j^*, \overline{c})} d_e(\overline{x}_j, \overline{c}), \tag{5}$$

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

Fig. 1. Steps of SBKM algorithm.

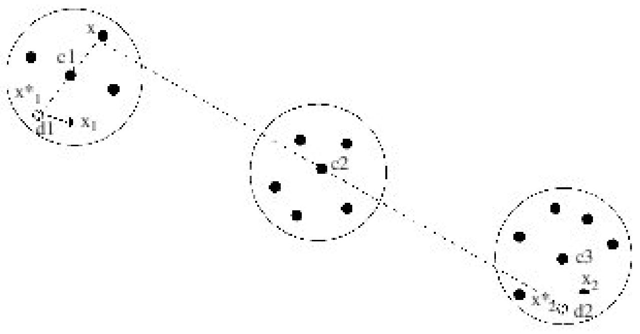

Fig. 2. Example where point symmetry distance proposed by Su and Chou fails.

where $d_{symm}(\overline{x}_j, \overline{c}) = \|(\overline{x}_j - \overline{c}) + (\overline{x}_j^* - \overline{c})\|$. It can be also noted that $d_e(\overline{x}_j, \overline{c}) \approx d_e(\overline{x}_j^*, \overline{c})$. Therefore from Eq. (5), we obtain

$$d_c(\overline{x}_j, \overline{c}) \propto d_{symm}(\overline{x}_j, \overline{c}). \tag{6}$$

As a result there is no impact of Euclidean distance, only symmetrical distance plays an important role in assignment of points to different clusters. It has been shown experimentally that Mod-SBKM with this measure will also fail for several data sets considered in Section 7.

The most limiting aspect of the measures suggested in Refs. [41,42] is that in cases where K-means provides reasonably good clusters, application of the fine-tuning phase (see algorithm in Fig. 1) will destroy this structure. Another limitation of the SBKM is that it requires a prior specification of a parameter θ, based on which assignment of points to clusters is done either on the basis of the PS distance or the Euclidean distance. Su and Chou have chosen θ equals to 0.18. However, we have observed that clustering performance is significantly affected by the choice of θ, and its best value is dependent on the data characteristics. No guidelines for the choice of θ is provided in Ref. [41].

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

3. A new definition of the point symmetry distance

As discussed in Section 2, both the PS-based distances, d_s and d_c, will fail when the clusters themselves are symmetrical with respect to some intermediate point. It has been shown, in such cases the points are assigned to the farthest cluster. In order to overcome this limitation, we propose a new PS distance in this article which is called $d_{ps}(\overline{x}, \overline{c})$ associated with point $\overline{x}$ with respect to a center $\overline{c}$. The proposed point symmetry distance is defined as follows: let a point be $\overline{x}$. The symmetrical (reflected) point of $\overline{x}$ with respect to a particular center $\overline{c}$ is $2 * \overline{c} - \overline{x}$. Let us denote this by $\overline{x}^*$. Let the first and the second unique nearest neighbors of $\overline{x}^*$ be at Euclidean distances of d_1 and d_2, respectively. Then

$$d_{ps}(\overline{x}, \overline{c}) = \frac{(d_1 + d_2)}{2} \times d_e(\overline{x}, \overline{c}), \tag{7}$$

where $d_e(\bar{x}, \bar{c})$ is the Euclidean distance between the point $\bar{x}$ and $\bar{c}$.

The basic differences between the PS-based distances in Refs. [41,42], and the proposed point symmetry distance, $d_{ps}(\bar{x}, \bar{c})$, are as follows:

(1) Instead of computing Euclidean distance between the original reflected point $\bar{x}^* = 2 \times \bar{c} - \bar{x}$ and its first nearest neighbor as in Refs. [41,42], here the average distance between $\bar{x}^*$ and its first and the second unique nearest neighbors have been taken. Consequently the term, $(d_1 + d_2)/2$ will never be equal to 0, and the effect of $d_e(\bar{x}, \bar{c})$, the Euclidean distance, will always be considered. This will reduce the problems discussed in Fig. 2.
(2) Considering both d_1 and d_2 in the computation of d_{ps} makes the PS-distance more robust and noise resistant. From an intuitive point of view, if both d_1 and d_2 of $\bar{x}$ with respect to $\bar{c}$ is less, then the likelihood that $\bar{x}$ is symmetrical with respect to $\bar{c}$ increases. This is not the case when only the first nearest neighbor is considered which could mislead the method in noisy situations.
(3) We also provide a rough guideline of the choice of θ, the threshold value on the PS-distance. It is to be noted that if a point is indeed symmetric with respect to some cluster center then the symmetrical distance computed in the above way will be small, and can be bounded as follows. Let d_{NN}^{max} be the maximum nearest neighbor distance in the data set. That is

$$d_{NN}^{max} = \max_{i=1,\dots,N} d_{NN}(\bar{x}_i), \tag{8}$$

where $d_{NN}(\bar{x}_i)$ is the nearest neighbor distance of $\bar{x}_i$. Assuming that $\bar{x}^*$ lies within the data space, it may be noted that

$$d_1 \leqslant \frac{d_{NN}^{max}}{2}. \tag{9}$$

Ideally, a point $\bar{x}$ is exactly symmetrical with respect to some $\bar{c}$ if $d_1 = 0$. However, considering the uncertainty of the location of a point as the sphere of radius d_{NN}^{max} around $\bar{x}$, we have kept the threshold θ equals to d_{NN}^{max}.

It is evident that the symmetrical distance computation is very time consuming. Computation of $d_{ps}(\bar{x}_i, \bar{c})$ is of complexity $O(N)$. Hence for N points and K clusters, the complexity of assigning the points to the different clusters is $O(N^2K)$. In order to reduce the computational complexity, an ANN search using the Kd-tree approach is adopted in this article.

4. Kd-tree based nearest neighbor computation

A K-dimensional tree or Kd-tree is a space-partitioning data structure for organizing points in a K-dimensional space. A Kd-tree uses only those splitting planes that are perpendicular to one of coordinate axes. In the nearest neighbor problem a set of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of p to q can be reported efficiently. ANN is a library written in $C{+}{+}$ [47], which supports data structures and algorithms for both exact and ANN searching in arbitrarily high dimensions. In this article ANN is used to find d_1 and d_2 in Eq. (7) efficiently. The ANN library implements a number of different data structures, based on Kd-trees and box-decomposition trees, and employs a couple of different search strategies. The Kd-tree data structure has been used in this article. ANN allows the user to specify a maximum approximation error bound, thus allowing the user to control the tradeoff between accuracy and running time.

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

For computing $d_{ps}(\bar{x}, \bar{c})$ in Eq. (7), d_1 and d_2 need to be computed. This is a computation intensive task that can be speeded up by using the Kd-tree based nearest neighbor search. This approach is used in the proposed algorithm, GAPS. For the purpose of this article, the exact nearest neighbor is computed; so the ε is set equal to 0. In order to find symmetric distance of a particular point $\bar{x}$ with respect to the center $\bar{c}$, we have to find the first two nearest neighbors of $\bar{x}^*$ (where $\bar{x}^* = 2 * \bar{c} - \bar{x}$). Therefore the query point q is set equal to $\bar{x}^*$ and k is set to 2. After getting the k-nearest neighbors of $\bar{x}^*$, the symmetrical distance of $\bar{x}$ with respect to a centre $\bar{c}$ is calculated using Eq. (7).

5. GAPS: the genetic clustering scheme with the proposed PS distance

As mentioned earlier, the GA-based clustering algorithm [43] is used in this article since it is known to provide good clusters when K is known. However, instead of the Euclidean distance, now $d_{ps}(\bar{x}, \bar{c})$ is used as the distance measure for computing the *clustering_metric* (M) defined in Fig. 4. The task of the GA is to search for the appropriate cluster centers $z_1, z_2 \dots z_K$ such that M is maximized.

5.1. String representation and population initialization

The basic steps of GAPS, that closely follow those of the conventional GA, are shown in Fig. 3. Here centre based encoding of the chromosome is used. Each string is a sequence of real numbers representing the K cluster centers. The K cluster centers encoded in each chromosome are initialized to K randomly chosen points from the data set. This process is repeated for each of the *Popsize* chromosomes in the population, where

Popsize is the size of the population. Thereafter five iterations of the K-means algorithm is executed with the set of centers encoded in each chromosome. The resultant centers are used to replace the centers in the corresponding chromosomes. This makes the centers separated initially.

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

Fig. 3. Basic steps of GA based clustering.

5.2. Fitness computation

In order to compute the fitness of the chromosomes, the clustering procedure clustering-PS() (as shown in Fig. 4) is called. Here a point $\overline{x}_i$, $1 \leqslant i \leqslant n$, is assigned to cluster k iff $d_{ps}(\overline{x}_i, \overline{c}_k) \leqslant d_{ps}(\overline{x}_i, \overline{c}_j)$, $j = 1, \ldots, K$, $j \neq k$ and $(d_{ps}(\overline{x}_i, \overline{c}_k)/d_e(\overline{x}_i, \overline{c}_k)) \leqslant \theta$. For $(d_{ps}(\overline{x}_i, \overline{c}_k)/d_e(\overline{x}_i, \overline{c}_k)) > \theta$, point $\overline{x}_i$ is assigned to some cluster m iff $d_e(\overline{x}_i, \overline{c}_m) \leqslant d_e(\overline{x}_i, \overline{c}_j)$, $j = 1, 2, \ldots, K$, $j \neq m$. In other words, point $\overline{x}_i$ is assigned to that cluster with respect to whose centers its PS-distance is the minimum, provided the total "symmetricity" with respect to it is less than some threshold θ. Otherwise assignment is done based on the minimum Euclidean distance criterion as normally used in Ref. [43] or the K-means algorithm. The value of θ is kept equal to the maximum nearest neighbor distance among all the points in the data set. Thus the computation of θ is automatic and does not require user intervention. After the assignments are done, the cluster centers encoded in the chromosome are replaced by the mean points of the respective clusters. Subsequently for each chromosome *clustering_metric, M*, is calculated as defined below:

$M = 0$
For $k = 1$ to K do
  For all data points $\overline{x}_i$, $i = 1$ to n and $\overline{x}_i \in k$th cluster, do

$$M = M + d_{ps}(\overline{x}_i, \overline{c}_k). \tag{10}$$

```
Procedure : clustering_PS()
•   Assignment of data points :
    1. For all data points $\overline{x}_i$, $1 \le i \le n$, compute
    $k^* = Argmin_{k=1\ldots K} d_{ps}(\overline{x}_i, \overline{c}_k)$
    2. If $(d_{ps}(\overline{x}_i, \overline{c}_{k^*})/d_e(\overline{x}_i, \overline{c}_{k^*}) < \theta)$
    /* $d_e(\overline{x}_i, \overline{c}_{k^*})$ is the Euclidean distance between the point $\overline{x}_i$ and cluster centroid $\overline{c}_{k^*}$ */
    assign the data point $\overline{x}_i$ to the $k^*th$ cluster.
    3. Otherwise, the data point is assigned to the $k^*$ cluster where
      $k^* = Argmin_{k=1\ldots K} d_e(\overline{x}, \overline{c}_k)$
•   Clustering metric calculation :
    1. Clustering_metric = 0
    2. For $k = 1$ to $K$ do
        For all data points $\overline{x}_i$, $i=1$ to $n$ and $\overline{x}_i \in k$th cluster, do
                    $Clustering\_metric += d_{ps}(\overline{x}_i, \overline{c}_k)$        (12)
•   Updation of centres : Compute the new centroids of the K clusters as follows:
                    $\overline{c}_k(t+1) = \frac{\sum_{i \in S_k(t)} \overline{x}_i}{N_k}$        (13)
    where $k = 1, \ldots K$ and $S_k(t)$ is the set of elements that are assigned to the $k$th cluster at
    generation $t$ and $N_k = |S_k|$.
```

Fig. 4. Main steps of clustering_PS() procedure

Then the fitness function of that chromosome, *fit*, is defined as the inverse of M, i.e.,

$$fit = \frac{1}{M}. \tag{11}$$

This fitness function, *fit*, will be maximized by using GA. (Note that there could be other ways of defining the fitness function.)

5.3. Selection

Roulette wheel selection is used to implement the proportional selection strategy.

5.4. Crossover

Here, we have used the normal single point crossover [44]. Crossover probability is selected adaptively as in Ref. [48]. The expressions for crossover probabilities are computed as follows. Let f_{max} be the maximum fitness value of the current population, $\overline{f}$ be the average fitness value of the population and f' be the larger of the fitness values of the solutions to be crossed. Then the probability of crossover, μ_c, is calculated as

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

$$\mu_c = k_3, \quad \text{if } f' \leqslant \overline{f}.$$

Here, as in Ref. [48], the values of k_1 and k_3 are kept equal to 1.0. Note that, when $f_{max} = \overline{f}$, then $f' = f_{max}$ and μ_c will be equal to k_3. The aim behind this adaptation is to achieve a trade-off between exploration and exploitation in a different manner. The value of μ_c is increased when the better of the two chromosomes to be crossed is itself quite poor. In contrast when it is a good solution, μ_c is low so as to reduce the likelihood of disrupting a good solution by crossover.

5.5. Mutation

Each chromosome undergoes mutation with a probability μ_m. The mutation probability is also selected adaptively for each chromosome as in Ref. [48]. The expression for mutation probability, μ_m, is given below:

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

$$\mu_m = k_4 \quad \text{if } f \leqslant \overline{f}.$$

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

We have used the mutation operation similar to that used in GA based clustering [43]. In GAPS, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the clustering problem. Elitism has been implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus on termination, this location contains the centers of the final clusters.

6. Complexity analysis

This section contains a complexity analysis of SBKM and GAPS clustering methods.

6.1. Complexity analysis of SBKM

The complexity of the SBKM clustering algorithm is as follows (refer to Fig. 1):

(1) Step 1 of the algorithm needs constant time.
(2) Each iteration of the K-means algorithm needs $(NK + K)$ time. So if K-means algorithm needs maximum *maxiter_kmeans* number of iterations then total complexity due to K-means algorithm is $((NK + K) \times maxiter\_kmeans)$.
(3) In order to find the symmetrical distance of one point in step 3, the time needed is (NK). So for a total of N points, the complexity is (N^2K). If this fine-tuning procedure is repeated for $maxiter\_symmetry$ number of times then total complexity due to this procedure is $(N^2K \times maxiter\_symmetry)$.
(4) Step 4 of the algorithm i.e., updating of centers, needs $O(K)$ time.

So, in general total time complexity becomes

$$((NK + K) * maxiter\_kmeans + N^2K \times maxiter\_symmetry + K). \tag{14}$$

As $N \gg maxiter\_kmeans$, so, the complexity of SBKM becomes $O(N^2K \times maxiter\_symmetry)$.

6.2. Complexity analysis of GAPS

Below we have analyzed the complexity of the proposed GAPS clustering.

(1) As discussed above Kd-tree data structure has been used in order to find the nearest neighbor of a particular point. The construction of Kd-tree requires $O(N \log N)$ time and $O(N)$ space [8].
(2) Initialization of GA needs $Popsize \times stringlength$ time where *Popsize* and *stringlength* indicate the population size and the length of each chromosome in the GA, respectively. Note that for K clusters in d-dimensional space, *stringlength* will become $K \times d$.
(3) Fitness is computed by calling the *clustering_PS* procedure.
    (a) In order to assign each point to a cluster we have to calculate the minimum symmetrical distance of that point with respect to all clusters. For this purpose the Kd-tree based nearest neighbor search is used. If the points are roughly uniformly distributed, then the expected case complexity is $O(c^d + \log N)$, where c is a constant depending on dimension and the point distribution. This is $O(\log N)$ if the dimension d is a constant [49]. Friedman et al. [50] also reported $O(\log N)$ expected time for finding the nearest neighbor. So in order to find minimal symmetrical distance of a particular point, $O(K \log N)$ time is needed.
    For N points total complexity becomes $O(KN \log N)$.
    (b) For updating the centers total complexity is $O(K)$.
    So the fitness evaluation has total complexity $= O(Popsize \times KN \log N)$.
(4) Selection step of the GA requires $O(Popsize \times stringlength)$ time.
(5) Mutation and crossover require $O(Popsize \times stringlength)$ time each.

So in general total time complexity becomes $O(KN \log N \times Popsize)$ per generation. For maximum *maxgen* number of generations total complexity becomes $O(KN \log N \times Popsize \times maxgen)$. As *Popsize* is constant, total complexity of GAPS clustering is $O(NK \log N \times maxgen)$. So it is evident that the use of Kd-tree makes GAPS clustering method more efficient as compared to SBKM.

7. Implementation results

The experimental results comparing the performance of GAPS, SBKM [41], Mod-SBKM [42] and K-means algorithm are provided for nine artificial data sets and seven real-life data sets. For SBKM algorithm, θ is set equal to 0.18 as suggested in Ref. [41]. For Mod-SBKM, θ is chosen as 0.5. For the newly developed GAPS-clustering, value of θ is determined from the data set as discussed in Section 3.

Initially experiments were carried out with fixed mutation probability, μ_m and fixed crossover probability, μ_c. Good results were obtained with $\mu_c = 0.8$ and $\mu_m = 0.02$, keeping

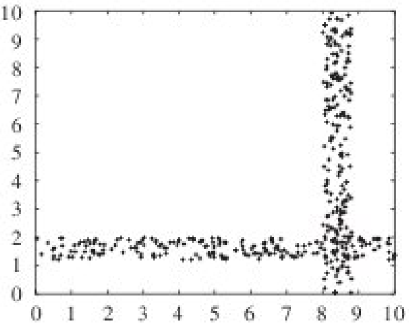

Fig. 5. *Data1*.

number of generations $= 30$. Keeping the value of $\mu_c = 0.8$ fixed and changing μ_m to 0.05, GAPS got stuck at a local optimum for *Data7*, but for all the other data sets its performance was good. Again, keeping the value of μ_c fixed and decreasing the value of μ_m to 0.008, GAPS got stuck at a local optimum for *Data1* but for the other data sets its performance was good. For $\mu_m = 0.02$ and $\mu_c = 0.99$, GAPS was not able to find out the proper clustering for *Data7* and *Data6*. For $\mu_c = 0.7$ and $\mu_m = 0.02$, again GAPS could not find out the proper clustering for *Data5*, *Data6* and *Data7*. The experimental results show that good performance of GAPS depends on the initial choice of μ_c and μ_m. Hence the crossover probability, μ_c, and mutation probability, μ_m, are determined adaptively as described in Sections 5.4 and 5.5. The population size, P is set equal to 100. The total number of generations is kept equal to 20. Executing it further did not improve the performance.

The 16 data sets used for comparison are divided into four groups.

(1) Group 1: The three data sets in this group (*Data1*, *Data2* and *Data3*) are similar to those used in Refs. [41,42] (these were generated by us, but by keeping the structure of the clusters as close as possible to that described in Refs. [41,42]). The clusters present in these data sets are internally symmetrical but clusters themselves are not symmetrical with respect to any intermediate point. For these data sets, SBKM and GAPS can easily find the true clustering but K-means fails to find this.
    (a) *Data1*: This data set consists of two bands as shown in Fig. 5, where each band consists of 200 data points. The final clustering results obtained by K-means, SBKM, Mod-SBKM and GAPS are given in Figs. 6(a), (b), (c) and (d), respectively. All the above-mentioned algorithms except the K-means are able to find out the proper clustering for this data. As expected K-means shows poor performance for this data since the clusters are not hyperspherical in nature.
    (b) *Data2*: This data set is a combination of ring-shaped, compact and linear clusters shown in Fig. 7. The total number of points in it is 400. The final results obtained after application of K-means, SBKM, Mod-SBKM and GAPS are shown in Figs. 8(a), (b), (c) and (d), respectively, where K-means is found to fail in providing the

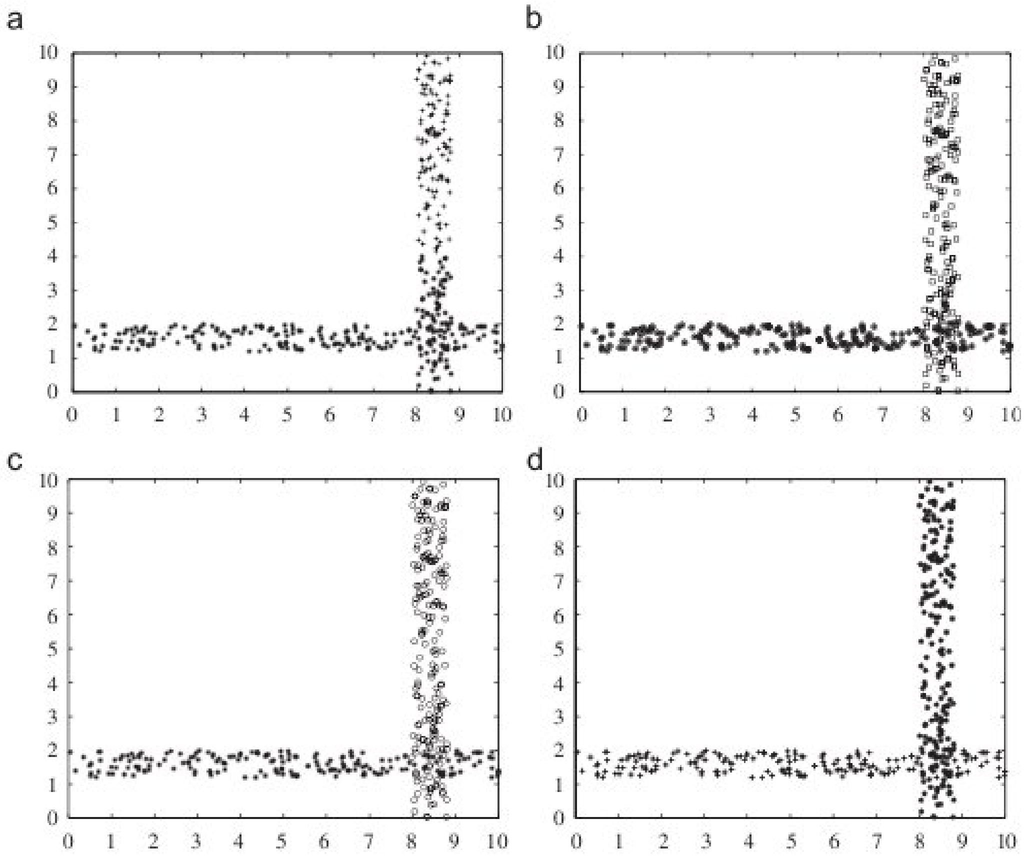
Fig. 6. Clustering of *Data1* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

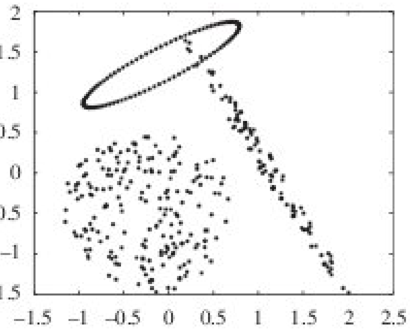
Fig. 7. *Data2*.

proper clusters. For SBKM (Fig. 8(b)) portions of the upper ring have gone into the elongated elliptic cluster (denoted by '*'). But Mod-SBKM is able to find out the proper clustering from the data set. In contrast, the proposed GAPS (Fig. 8(d)) groups the points inside the ring, but part of the elongated, elliptic cluster, with those of the ring. In absence of any class information about the points, such a grouping is not really surprising.

(c) *Data3*: This data set is a combination of a ring-shaped cluster, a rectangular cluster and a linear cluster as shown in Fig. 9, with total number of points equal to 400 and three clusters. The final results corresponding to K-means algorithm, SBKM, Mod-SBKM and GAPS are shown in Figs. 10(a), (b), (c) and (d), respectively. As evident from Fig. 10(a), K-means fails in correctly detecting the linear cluster; it includes points from the rectangular cluster in the linear cluster. As expected, the ring is properly detected. The other three methods yield the correct partitioning for this data.

It is to be noted that for each of the above three data sets, the K-means is unable to provide the correct clustering. However, the PS-distance based cluster assignments, performed in the fine-tuning phase of SBKM, rectifies the wrong cluster assignments provided by the K-means.

(2) Group 2: The four data sets in this category are those used in Ref. [51]. The clusters present in these data sets are internally symmetrical and clusters are also symmetrical with respect to some intermediate point.

(a) *Data4*: This data set consists of 250 data points distributed over five spherically shaped clusters as shown in Fig. 11. The clusters present here are highly overlapping, each consisting of 50 data points. Figs. 12(a), (b), (c) and (d) show the results for K-means, SBKM, Mod-SBKM and GAPS, respectively. As is evident, K-means performs the best for this data. Although GAPS is also able to detect the clusters reasonably well, it

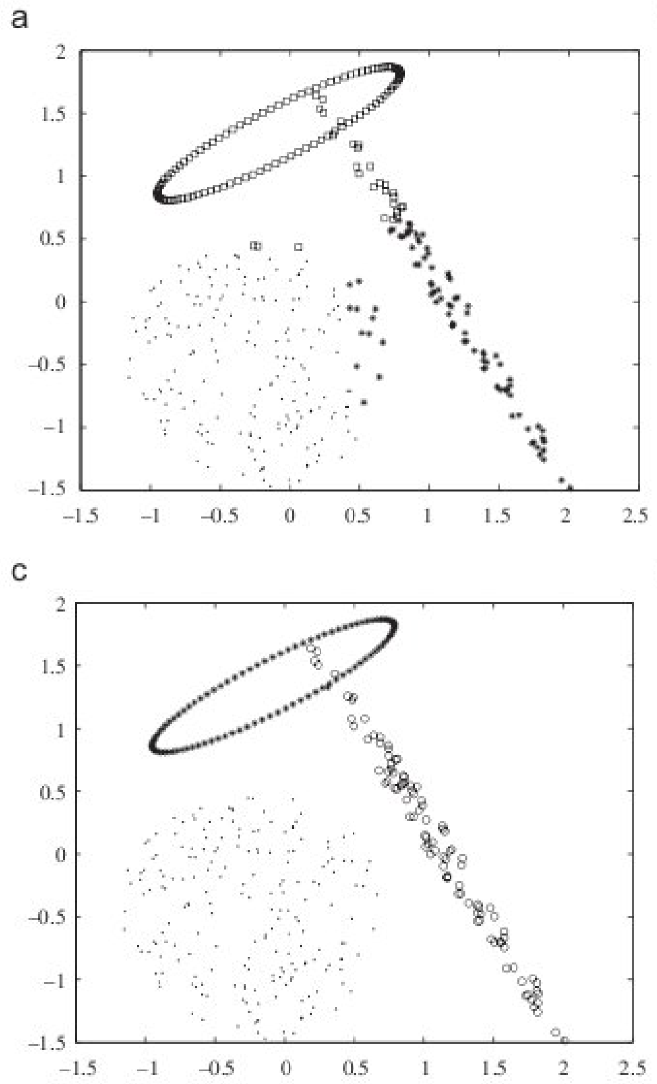

Fig. 8. Clustering of *Data2* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

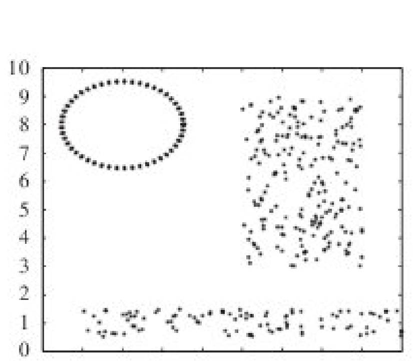

Fig. 9. *Data3*.

is found to somewhat over-approximate the central cluster (which extends to the left). The reason is as follows. Let us take a point p which actually belongs to left cluster but after application of GAPS it is included in the central cluster (it is shown in Fig. 12(d)). It can be seen from the figure that the point p is more symmetrical with respect to the central cluster, $c1$. Here even though $d_e(p, c1)$ is slightly greater than $d_e(p, c2)$ but due to the absence of symmetry with respect to $c2$, p is assigned to the central cluster.

Both SBKM and Mod-SBKM fail in detecting the proper clustering here because data points are more symmetrical with respect to some other cluster center than the actual cluster center (because of the limitations in the definitions of d_s and d_c). The point to be noted here is that the SBKM method destroys the proper clustering achieved by the K-means method. This is demonstrated in Figs. 13(a)–(d) that show how the cluster centres move with iterations during the application of SBKM. Evidently SBKM is trying to bring all the cluster centres at the center of the whole data set thereby providing very poor performance.

(b) *Data5*: This data set consists of 76 data points distributed over three clusters. Some points of one

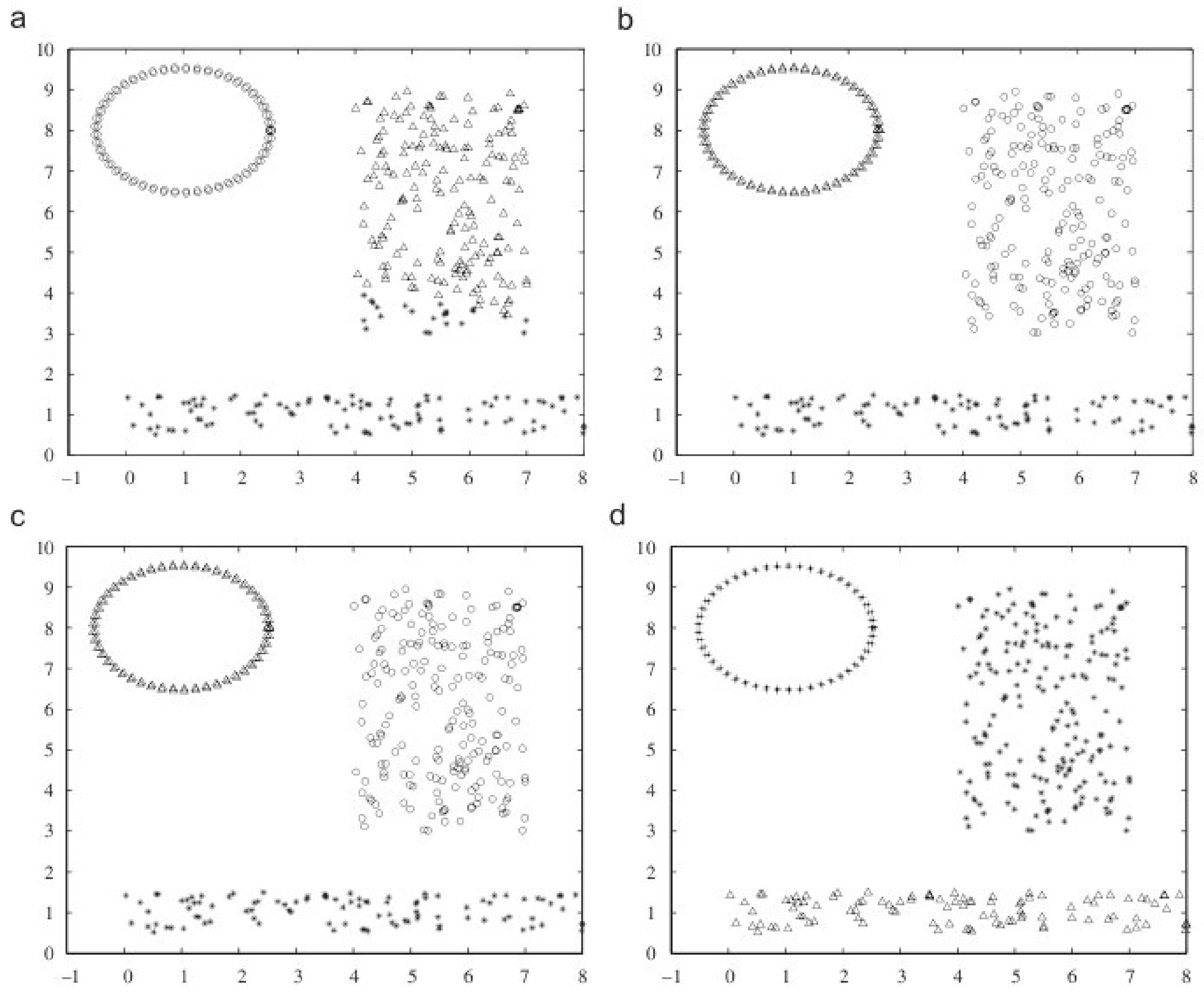
Fig. 10. Clustering of *Data3* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

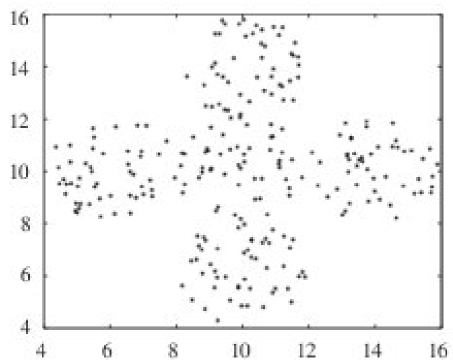
Fig. 11. *Data4*.

cluster are symmetrical with respect to the other clustercentre. This data set is shown in Fig. 14. Figs.15 (a), (b), (c) and (d) show the clusters obtained by K-means, SBKM, Mod-SBKM and GAPS, respectively. As is evident, both K-means and GAPS succeed in providing the proper partitioning while SBKM and Mod-SBKM fail in doing so.

(c) *Data6*: This data set consists of 400 data points in three-dimensional space distributed over four hyperspherical disjoint clusters where each cluster contains 100 data points. This data set is shown in Fig. 16. Figs. 17(a), (b), (c) and (d) show the clusters obtained by K-means, SBKM, Mod-SBKM and GAPS, respectively. As is evident, both K-means and GAPS again succeed in providing the proper clusters while SBKM and Mod-SBKM fail miserably in doing so. As is evident from Fig. 17(b), SBKM provides just three clusters for this data, which are denoted by '$*$', '$\triangle$', 'o'. Note that the cluster shown by 'o' has just two points in it, which appears among the points in the lowermost group. Although Mod-SBKM is able to identify four clusters (Fig. 17(c)), the clusters themselves are improper.

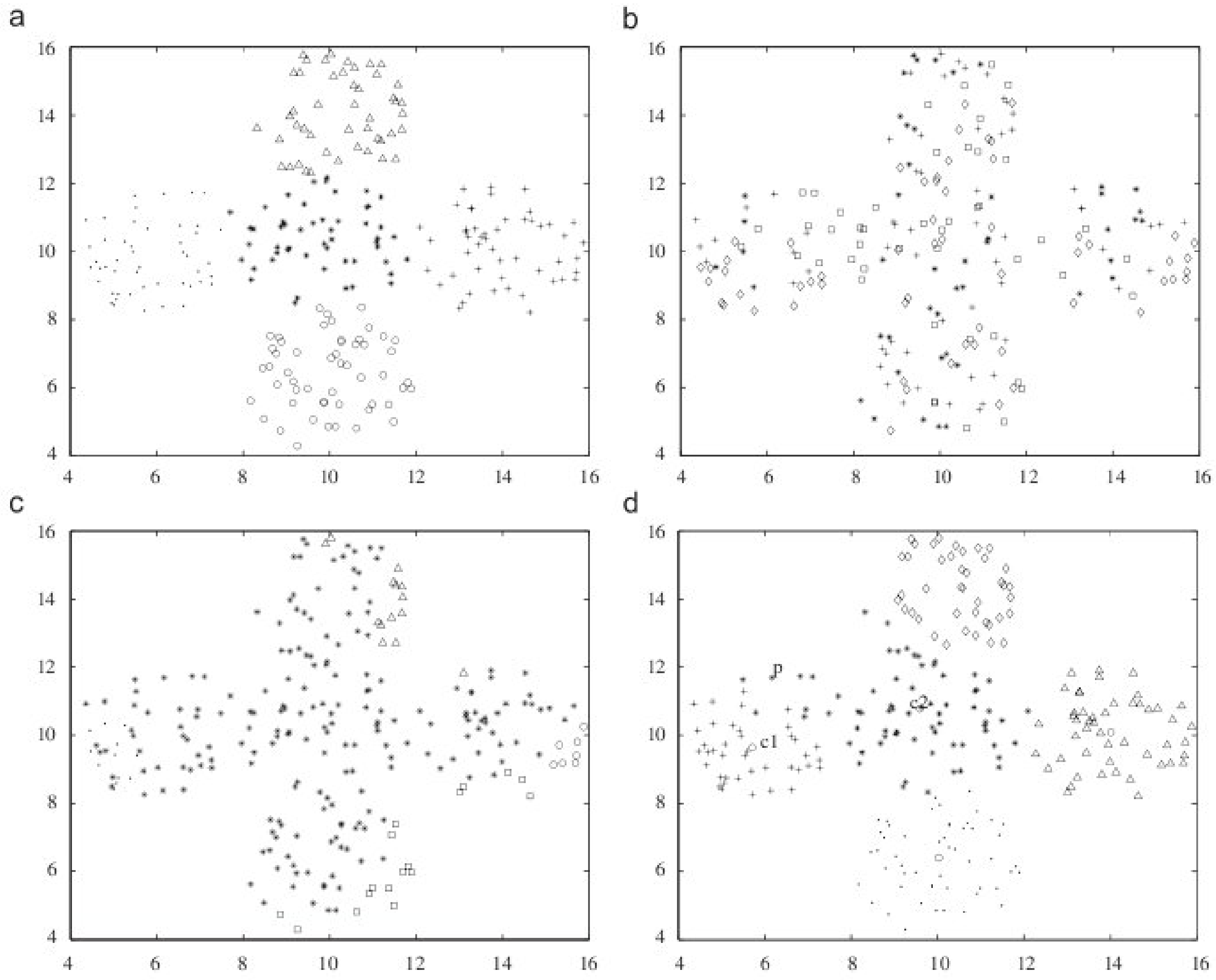

Fig. 12. Clustering of *Data4* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

(d) *Data7*: This data set consists of 300 data points that are distributed over six different clusters with 50 points in each cluster. This data set is shown in Fig. 18. Figs. 19(a), (b), (c) and (d) show the clusters obtained by K-means, SBKM, Mod-SBKM and GAPS, respectively. As in the previous cases, both K-means and GAPS succeed in providing the proper clusters while SBKM and Mod-SBKM provide only four clusters for this data (see Figs. 19(b) and (c)) by merging two pairs of clusters. This observation is not surprising given the problem that exists in the definitions of the PS-distances as proposed in Refs. [41,42].

(3) Group 3: This group consists of two data sets which are used in Ref. [52]. These two data sets are taken in order to show that the proposed GAPS-clustering algorithm is able to find the proper clustering from a data set where clusters are of significantly different sizes. (Note that clusters of widely varying sizes may be present in several real-life domains, e.g., medical images, satellite images, fraud detection.)

(a) *Data8*: This data set consists of 43 points that are distributed over 2 different clusters of significantly different sizes as shown in Fig. 20. Figs. 21(a), (b), (c) and (d) show the partitionings obtained by K-means, SBKM, Mod-SBKM and GAPS, respectively. As is evident, only GAPS is able to find out the proper partitioning. Since K-means is able to detect clusters of almost equal sizes, it merges several points from the larger cluster to the smaller one. SBKM and Mod-SBKM are also misled in clustering because of the aforementioned problems in the definitions of the PS-distance. Only the proposed GAPS is successful for this data since it not only tries to minimize the symmetry distance but also the Euclidean distance.

(b) *Data9*: This is a two-dimensional data set consisting of 49 points distributed in three clusters as shown in Fig. 22. This data set consists of two small clusters (one has six elements and the other has three) separated by a large (40 element) cluster. Figs. 23(a), (b), (c) and (d) show the partitionings obtained by K-means, SBKM,

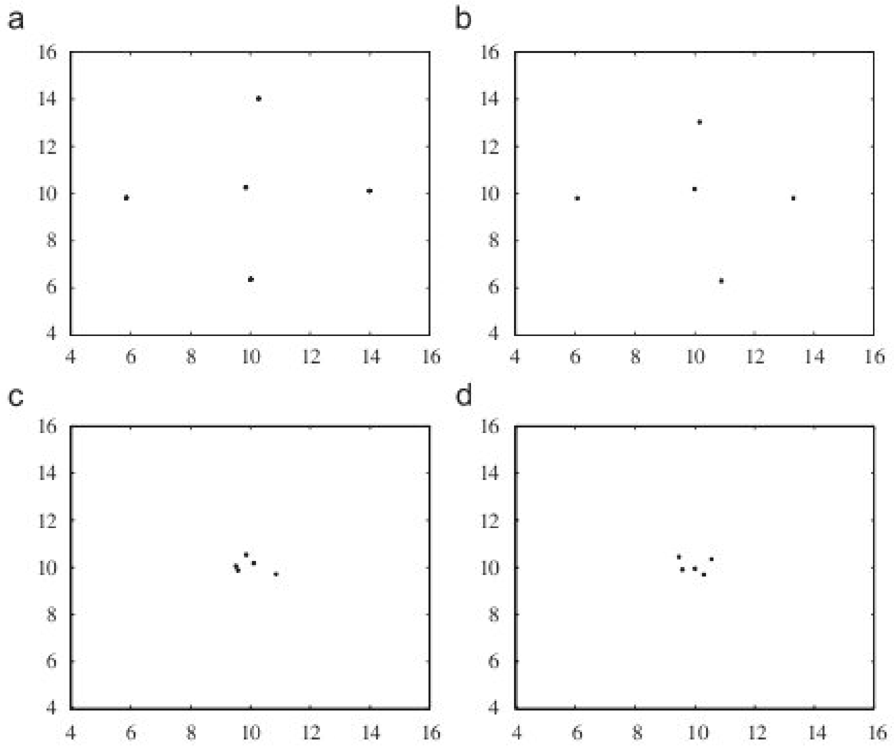

Fig. 13. Change of the cluster centers obtained by SBKM on *Data4* after (a) application of the K-means algorithm (b) 1 iteration (c) 10 iterations (d) 20 iterations.

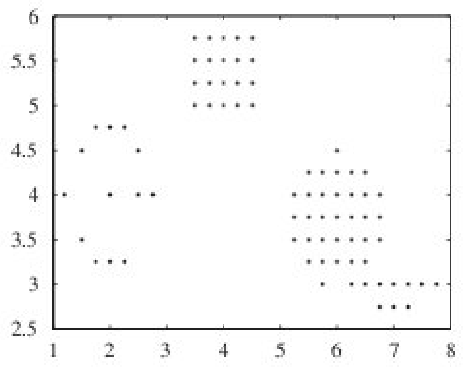

Fig. 14. *Data5*.

Mod-SBKM and GAPS, respectively. As earlier, only GAPS is able to find out the proper partitioning, while the other three algorithms fail.

(4) Group 4: This category consists of seven real life data sets: *Iris*, *Cancer*, *Newthyroid*, *Wine*, *Glass*, *LungCancer* and *Liverdisorder* data sets obtained from Ref. [53]. For these data sets, the *Minkowski Scores* [54] are reported for each algorithm. This is a measure of the quality of a solution given the true clustering. Let T be the "true" solution and S the solution we wish to measure. Denote by n_{11} the number of pairs of elements that are in the same cluster in both S and T. Denote by n_{01} the number of pairs that are in the same cluster only in S, and by n_{10} the number of pairs that are in the same cluster in T. *Minkowski Score* (MS) is then defined as

$$\mathrm{MS}(\mathrm{T},\mathrm{S})=\sqrt{\frac{n_{01}+n_{10}}{n_{11}+n_{10}}}. \tag{15}$$

For MS, the optimum score is 0, with lower scores being "better". The MS scores and their variances are reported in Table 1 for all the data sets. Statistical ANOVA [45] is performed for the real-life data sets on the combined MS values of the four algorithms when each is executed ten times. ANOVA results are reported in detail for *Iris* (Table 2), *Newthyroid* (Table 3) and *Glass* (Table 4) only to restrict the size of the article.

(a) *Iris*: Iris data set consists of 150 data points distributed over three clusters. Each cluster has 50 points. This data set represents different categories of irises characterized by four feature values [55]. 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.

As seen from Table 1, the MS-scores of Mod-SBKM is the best for *Iris*, while the performance of GAPS is second. However, it can be seen from Table 2 that the difference in the means of the MS scores of GAPS and K-means is not significant indicating their similar performance. The performance of SBKM algorithm is found to be poor.

(c) *Cancer*: Here we use the Wisconsin Breast cancer data set consisting of 683 sample points. Each pattern

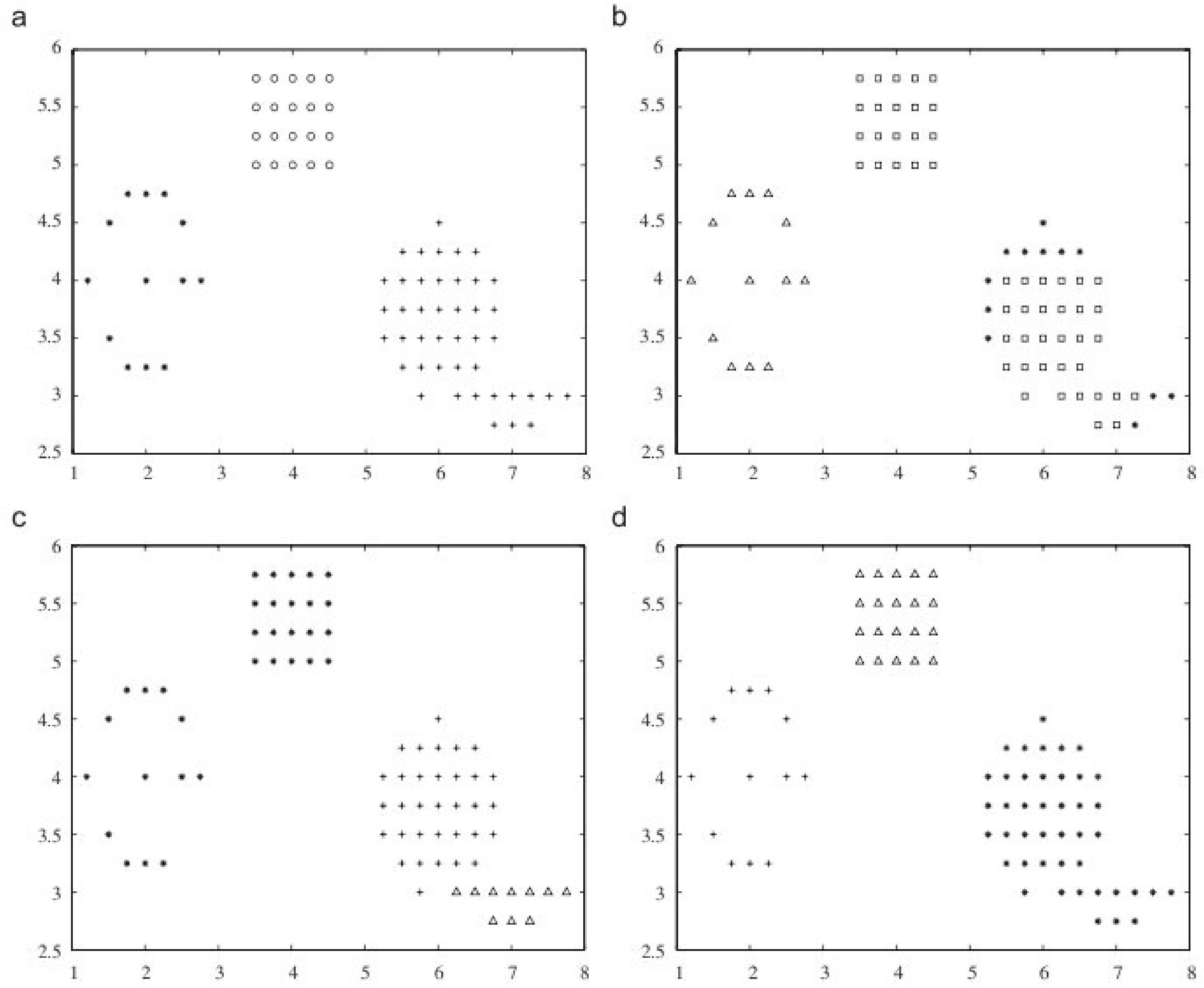

Fig. 15. Clustering of *Data5* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

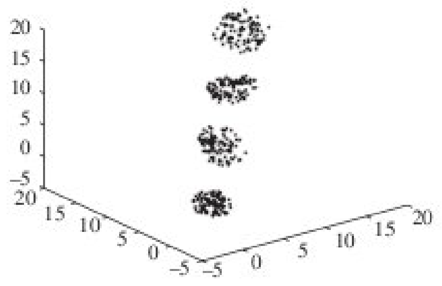

Fig. 16. *Data6*.

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. As can be seen from Table 1, the performance of K-means, SBKM and Mod-SBKM are similar, while that of GAPS is better than them. ANOVA tests show that the differences in mean MS scores of GAPS with respect to each of the other three algorithms are significant. The results indicate that the two clusters are convex as well as highly symmetrical.

Newthyroid (or, Thyroid gland data): 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. The total number of attributes is five. From Tables 1 and 3, it is evident that GAPS performs the best (providing the lowest MS score), while K-means performs the worst. The improvement in performance obtained by GAPS as compared to the other three techniques is statistically significant. SBKM is also found to provide improved performance over K-means and Mod-SBKM, and this improvement is also significant.

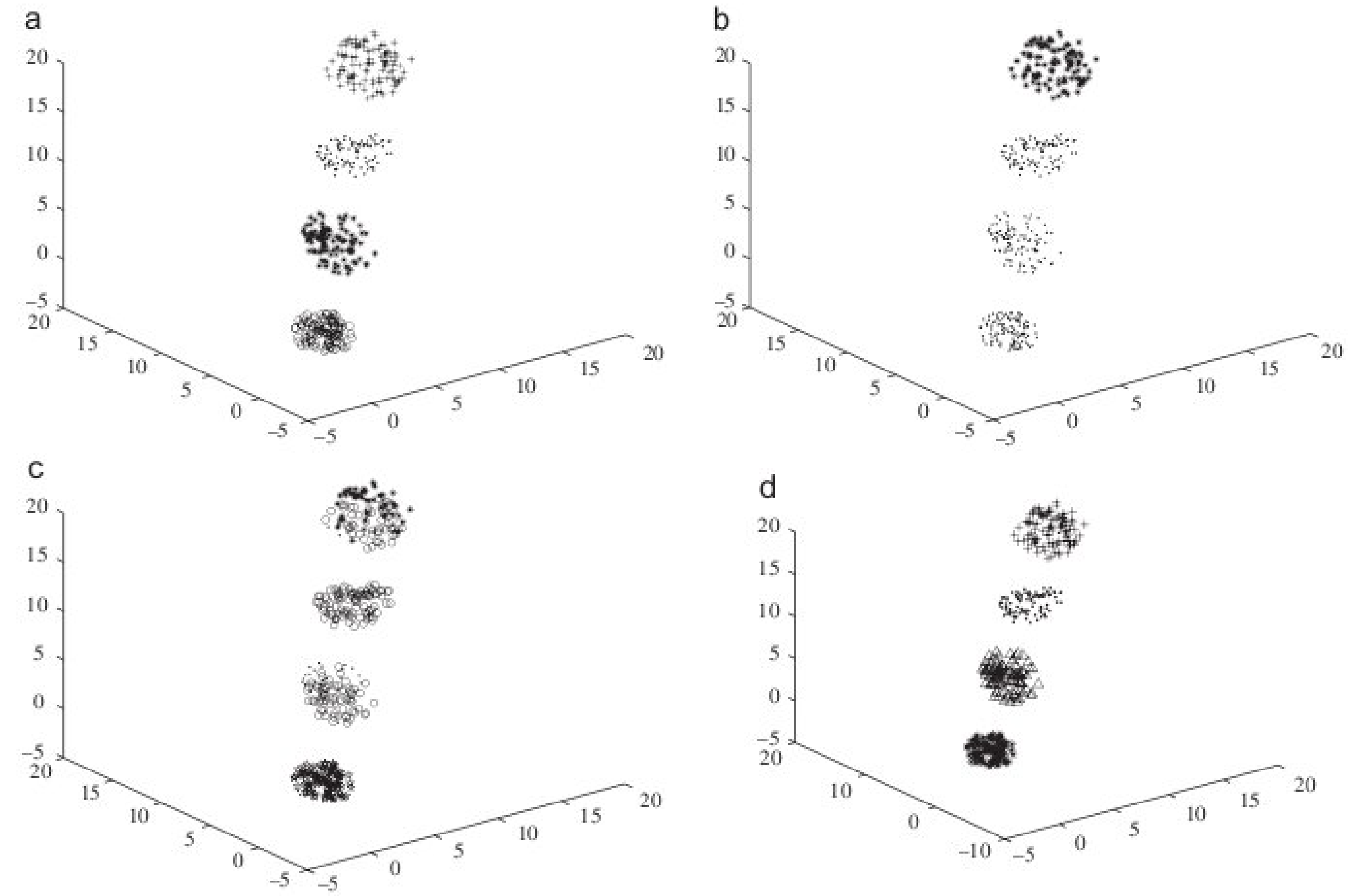

Fig. 17. Clustering of *Data6* obtained by (a) *K*-means (b) SBKM (c) Mod-SBKM (d) GAPS.

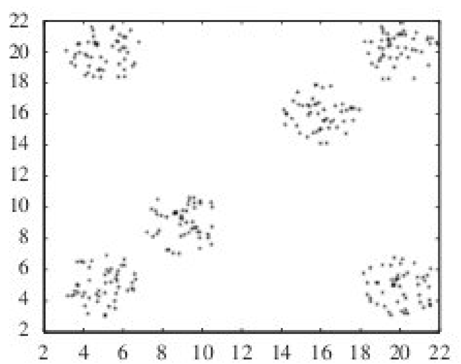

Fig. 18. *Data7*.

(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. From Table 1, it is evident that GAPS performs the best (providing the lowest MS score) for this data set. ANOVA statistical analysis is also done here. The analysis shows that the mean MS differences of all the algorithms are statistically significant.

(e) *Glass*: This is the glass identification data consisting of 214 instances having nine 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 six categories present in this data set. From Tables 1 and 4 it is again evident that GAPS performs much better than the other three algorithms (providing the lowest MS score). ANOVA results, provided in Table 4, shows that this improvement is statistically significant.

(f) *LungCancer*: This data consists of 32 instances having 56 features each. The data describes three types of pathological lung cancers. The MS scores, reported in Table 1, demonstrate the superior performance of GAPS. ANOVA statistical analysis is also done here. The analysis shows that the mean MS differences of all the algorithms are statistically significant.

(g) *Liverdisorder*: This is the Liver Disorder data consisting of 345 instances having six features each. The data has two categories. The MS scores of the partitions,

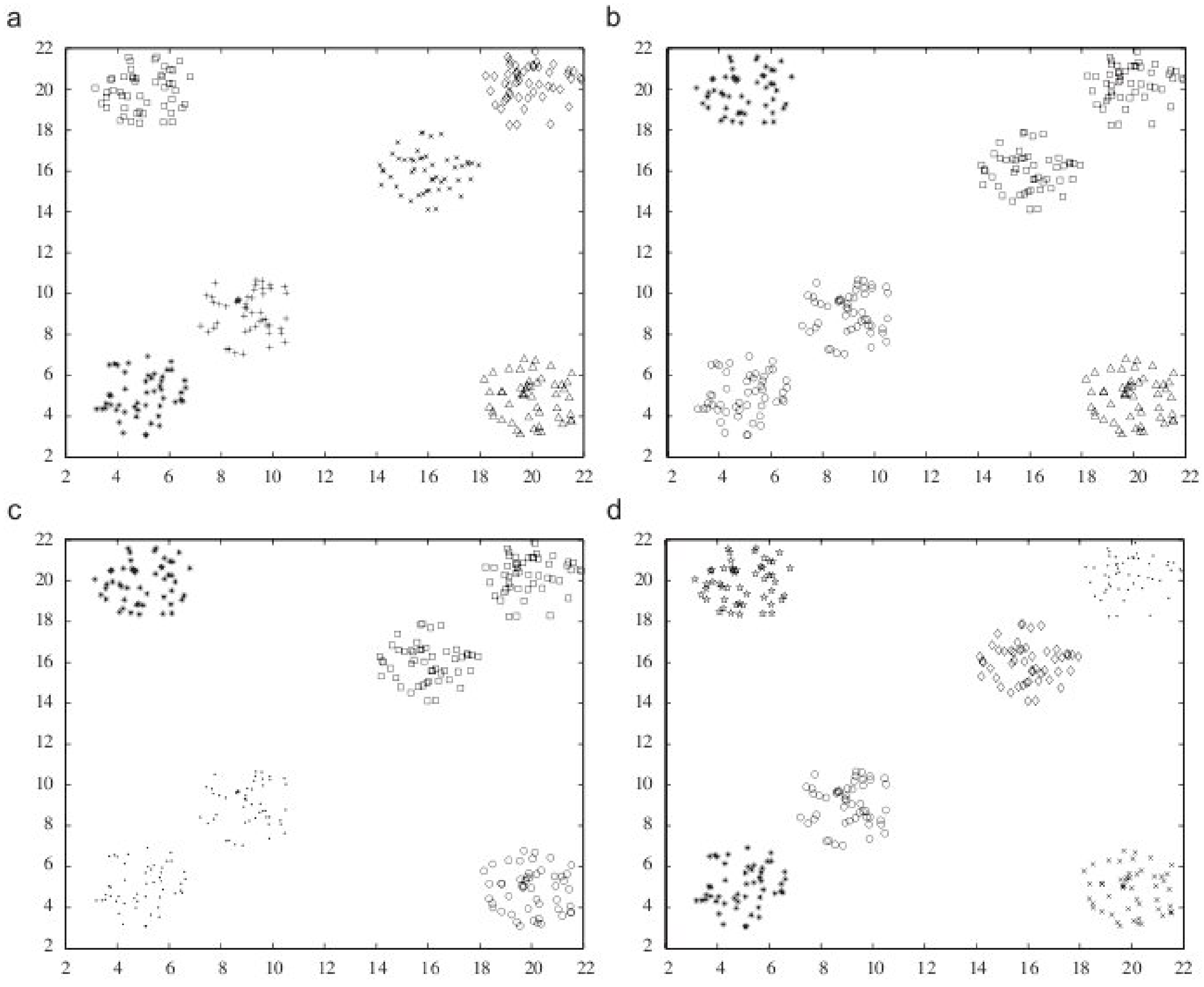

Fig. 19. Clustering of *Data7* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

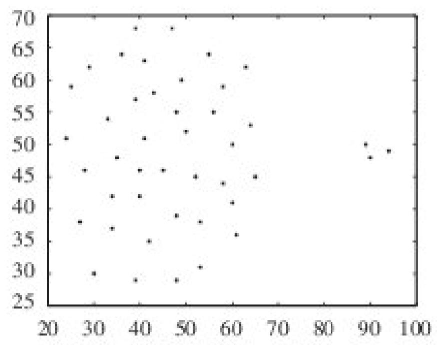

Fig. 20. *Data8*.

reported in Table 1, again shows the superior performance of GAPS. ANOVA statistical analysis again reveals that the mean MS differences of all the algorithms are statistically significant.

Note that the above implementation of GAPS used a Kd-tree structure to reduce computational time of identifying the nearest neighbors. In order to actually demonstrate this, GAPS was also executed without using Kd-tree data structure for the seven artificial data sets. The computational times (GAPS is implemented in C and was executed in a machine having linux platform, PIV processor, 1.6 GHz speed) are mentioned in Table 5. As is evident, incorporation of Kd-tree significantly reduces the computational burden of the process.

8. Comparing the individual clusters

MANOVA technique [46] is used for assessing group differences between the actual clusters and those obtained using GAPS, SBKM, Mod-SBKM and K-means. Here results are shown only for the first three data sets in Group 4 just for an illustration. MANOVA is a powerful statistical tool used in actual or quasi-experimental situations. It provides information on the nature and predictive power of the independent

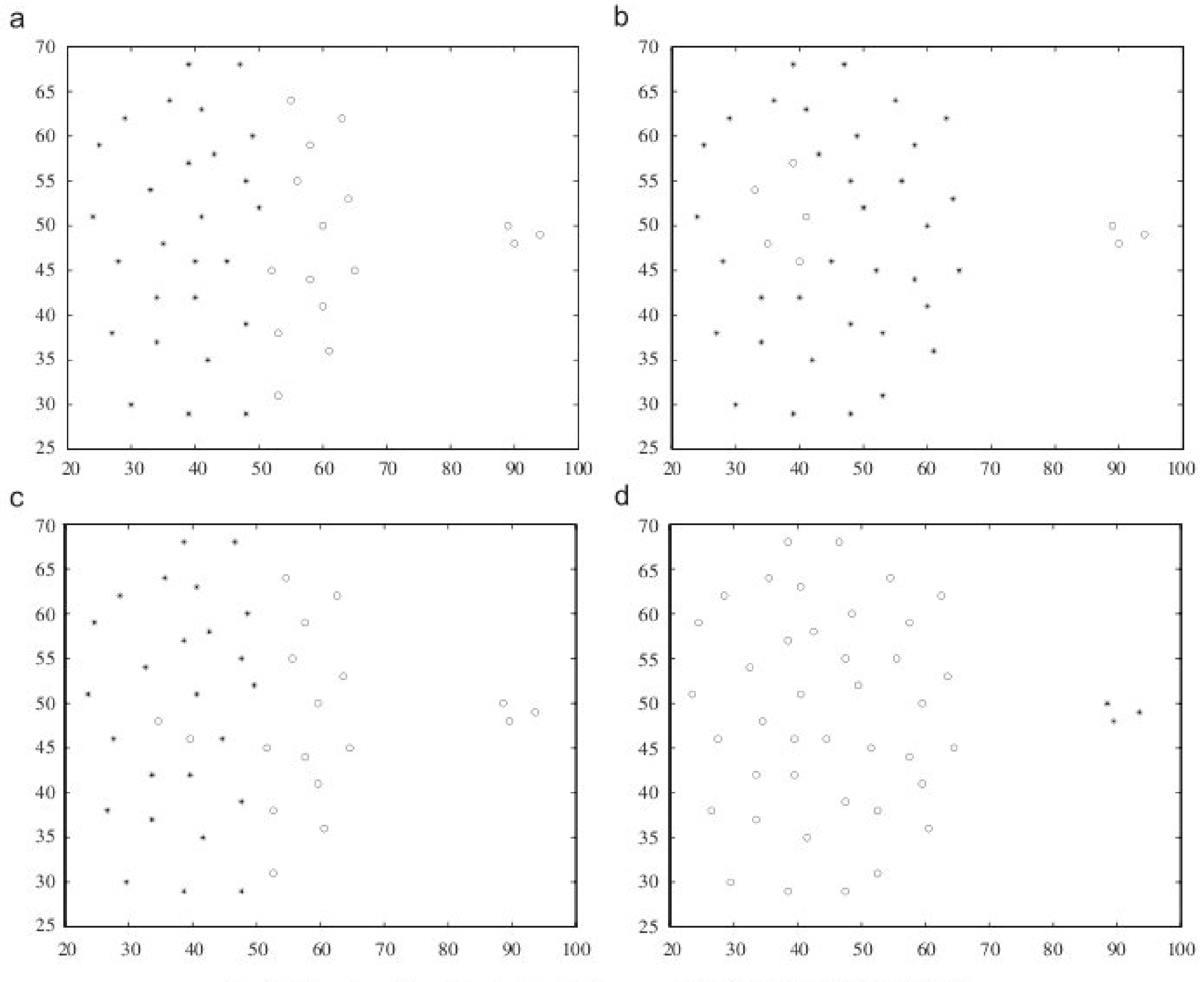
Fig. 21. Clustering of *Data8* obtained by (a) K-means (b) SBKM (c) Mod-SBKM (d) GAPS.

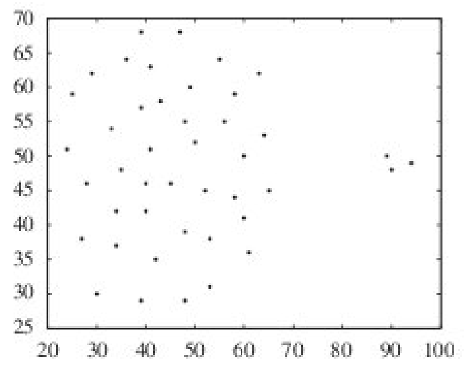
Fig. 22. *Data9*.

measures, as well as the relationships and differences seen in the dependent measures. It is an extension of ANOVA [45] used to accommodate more than one dependent variable. MANOVA measures the group differences between two or more metric dependent variables simultaneously, using a set of categorical non-metric variables (the cluster labels in this case). The matlab statistical toolbox, MANOVA1 is used for this purpose. The results are reported in Table 6. Here if $d = 0$ we cannot reject the null hypothesis that the means are the same, while if $d = 1$, we reject the hypothesis that the means are the same but we cannot reject the hypothesis that they lie on a line. A larger p value means the null hypothesis is more significant and vice versa. If $p < 0.05$, then we will reject the null hypothesis. The column *gm* in Table 6 represents the Mahalanobis distance between each pair of group means.

It can be seen from Table 6 that for *Iris* data set, GAPS is able to find the first cluster correctly ($d = 0$, $p = 1$ and $gm = 0$). This signifies that means of data items forming cluster 1 after application of GAPS and that of the actual cluster are the same. On the other hand GAPS could not find the other two clusters accurately. It is reflected from the combined result of d and p values where although the d values are 0 but this result is not significant as the p values are quite small. This is

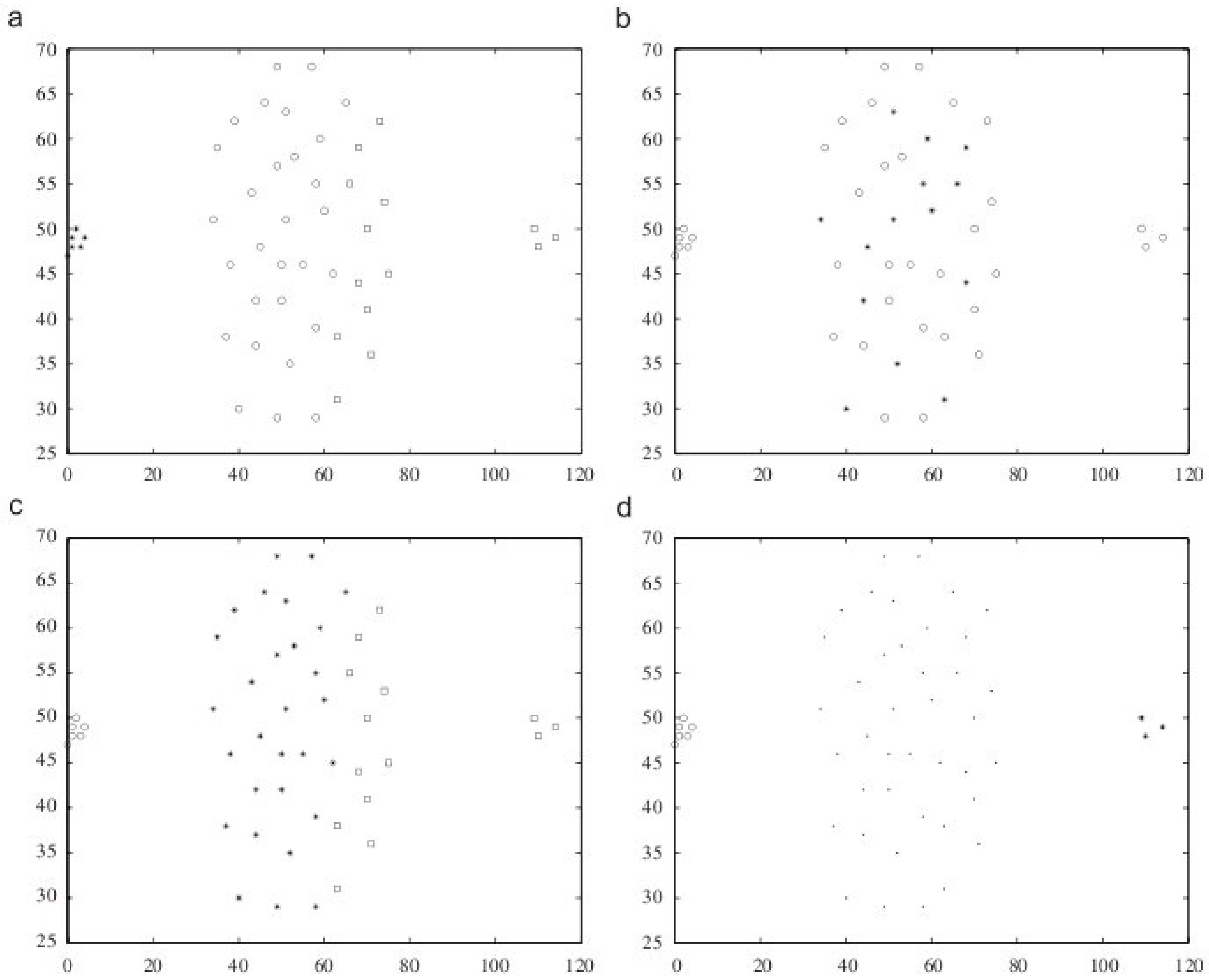

Fig. 23. Clustering of *Data9* obtained by (a) *K*-means (b) SBKM (c) Mod-SBKM (d) GAPS.

Table 1
Mean and variance of Minkowski score obtained by the four algorithms on seven real-life data sets

Data set	*K*-means	SBKM	Mod-SBKM	GAPS
Iris	0.68266 ± 0.002	0.9613 ± 0.0105	0.6481 ± 0.0149	0.671884 ± 0.001
Cancer	0.367056 ± 0.002	0.367056 ± 0.00	0.367056 ± 0.00	0.346018 ± 0.00
Newthyroid	0.940682 ± 0.00	0.6272 ± 0.0014	0.7640 ± 0.0129	0.591140 ± 0.001
Wine	1.3996 ± 0.001	1.2505 ± 0.002	0.975854 ± 0.00	0.912391 ± 0.002
Glass	1.6880 ± 0.01	1.0877 ± 0.000766	1.1174 ± 0.00041154	0.8223 ± 0.04
LungCancer	1.4558 ± 0.002	1.0904 ± 0.0025	1.0904 ± 0.0025	0.892647 ± 0.02
LiverDisorder	0.9777 ± 0.0001	0.9894 ± 0.0021	0.9852 ± 0.0	0.9639 ± 0.01

also evident from the high *gm* values as shown in Table 6. For SBKM, $d=1$ and p values are very low for all the three clusters indicating its extremely poor performance. Mod-SBKM and K-means perform similarly overall.

For *Cancer* data set, GAPS provides higher correspondence to the actual clusters, for both cluster1 and cluster2, as compared to the other algorithms. This is evident from the p and *gm* values in Table 6. For all the algorithms we get $d = 0$, but p value of GAPS is much better than p values of all the other algorithms. This is also evident from the *gm* value. *gm* values obtained by GAPS is smaller than the *gm* values obtained by all the other algorithms.

In the case of *Newthyroid* data, GAPS and SBKM perform well for cluster1 (in both the cases $d = 0$, $p > 0.05$ and *gm* is also small). For cluster 2, though GAPS and SBKM provide $d = 0$ but p is very close to 0.05. This is also evident from

Table 2
Estimated marginal means and pairwise comparisons of different algorithms on Minkowski score obtained by ANOVA testing for *Iris* data (GAPS:GP, *K*-means:KM, Mod-SBKM:MSB, SBKM:SB)

Algo. name (I)	Comparing algorithm (J)	Mean difference (I-J)	Significance value
GP	KM	$-1E-03 \pm 0.00163$	0.784
	MSB	$2.3E-02 \pm 0.004$	0.00
	SB	-0.2894 ± 0.023	0.00
KM	GP	$1E-03 \pm 0.00163$	0.78
	MSB	$3.73E-02 \pm 0.004$	0.00
	SB	-0.27864 ± 0.01	0.00
MSB	GP	$-2.3E-02 \pm 0.004$	0.00
	KM	$-3.73E-02 \pm 0.004$	0.00
	SB	-0.3132 ± 0.0012	0.00
SB	GP	0.2894 ± 0.023	0.00
	KM	0.27864 ± 0.01	0.00
	MSB	0.3132 ± 0.0012	0.00

Table 3
Estimated marginal means and pairwise comparisons of different algorithms on Minkowski score obtained by ANOVA testing for *Newthyroid* data (GAPS:GP, *K*-means:KM, Mod-SBKM:MSB, SBKM:SB)

Algo. name (I)	Comparing algorithm (J)	Mean difference (I-J)	Significance value
GP	KM	-0.359 ± 0.0012	0.00
	MSB	-0.16 ± 0.02	0.00
	SB	$-3.6E-02 \pm 0.0014$	0.00
KM	GP	0.359 ± 0.0012	0.00
	MSB	0.196 ± 0.01	0.00
	SB	0.34 ± 0.0014	0.00
MSB	GP	0.16 ± 0.02	0.00
	KM	-0.196 ± 0.01	0.00
	SB	0.136 ± 0.0129	0.00
SB	GP	$3.6E-02 \pm 0.0014$	0.00
	KM	-0.34 ± 0.0014	0.00
	MSB	-0.136 ± 0.0129	0.00

Table 4
Estimated marginal means and pairwise comparisons of different algorithms on Minkowski score obtained by ANOVA testing for *Glass* data (GAPS:GP, *K*-means:KM, Mod-SBKM:MSB, SBKM:SB)

Algo. name (I)	Comparing algorithm (J)	Mean difference (I-J)	Significance value
GP	KM	-0.865 ± 0.01	0.00
	MSB	-0.2951 ± 0.05	0.0
	SB	-0.27 ± 0.03	0.0
KM	GP	0.865 ± 0.01	0.00
	MSB	0.57 ± 0.02	0.0
	SB	0.6 ± 0.04	0.0
MSB	GP	0.2951 ± 0.05	0.0
	KM	-0.57 ± 0.02	0.0
	SB	$2.9E-02 \pm 0.004$	0.0
SB	GP	0.27 ± 0.03	0.0
	KM	-0.6 ± 0.04	0.0
	MSB	$-2.9E-02 \pm 0.004$	0.0

Table 5
Execution time (GAPS is implemented in C and executed in Linux platform, PIV processor, 1.66 GHz speed) in seconds by GAPS with and without Kd-tree based search

Data set	GAPS with Kd-tree	GAPS with out Kd-tree
Data1	30	1714
Data2	77	5280
Data3	61	5599
Data4	62	2268
Data5	10	136
Data6	128	6112
Data7	90	3870

the larger *gm* values. The other two algorithms yield very poor approximations of cluster1 and cluster2 since here $d = 1$. For cluster3, K-means and Mod-SBKM perform better than GAPS and SBKM as evident from the *gm* values in Table 6. Since 150 points (out of 215) of this data belong to cluster1, which GAPS is able to approximate well, the overall MS score is found to be the best for it (Table 3).

9. Summary of results

It can be seen from the above results that proposed GAPS is able to find out the proper clustering where SBKM and Mod-SBKM succeed while K-means fails (data sets from Group 1) as well as where K-means succeeds while SBKM and Mod-SBKM fail (data sets of Group 2). The results on data sets of Group 3 show that GAPS is able to detect symmetric clusters irrespective of their sizes where SBKM, Mod-SBKM and K-means all fail. The superiority of GAPS is also established on seven real-life data sets. These real-life data sets are of different characteristics with the number of dimensions varying from 4 to 56. Results on 16 artificial and real-life data sets establish the fact that GAPS is well-suited to detect clusters of widely varying characteristics. The improved performance of GAPS can be attributed to the fact that in the newly proposed point symmetry-based distance, d_{ps}, there is an impact of both the symmetrical distance as well as the Euclidean distance. This was lacking in the earlier definitions of the point symmetry-based distances [41,42], which resulted in some serious problems as discussed in Section 2 and displayed pictorially in Fig. 2.

The K-means, SBKM and Mod-SBKM are based on local search and hence may often get stuck at local optima depending on the choice of the initial cluster centers. The use of GA in GAPS in order to minimize the total symmetrical distance overcomes this problem. Moreover, the use of adaptive mutation and crossover probabilities also helps GAPS to converge faster. The experimental results on a wide variety of data sets show that GAPS is able to detect any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristic of symmetry. Based on this observation, and the fact that the property of symmetry is widely evident in real-life situations, application of GAPS in

Table 6
Result of MANOVA testing by different algorithms on real-life data sets (here *gm* stands for Mahalanobis distance)

Data set	GAPS			SBKM			Mod-SBKM			*K*-means		
	d	*p*	*gm*	*d*	*p*	*gm*	*d*	*p*	*gm*	*d*	*p*	*gm*
Iris (cluster1)	0	1	0	1	0	16.45	0	0.20	0.22	0	1	0
Iris (cluster2)	0	0.06	0.33	1	3.27 e − 009	1.65	0	0.33	0.23	0	0.26	0.26
Iris (cluster3)	0	0.11	0.41	1	2.36 e − 009	1.48	0	0.20	0.22	0	0.09	0.31
Cancer (cluster1)	0	0.99	0.015	0	0.97	0.013	0	0.97	0.023	0	0.98	0.01
Cancer (cluster2)	0	0.98	0.0104	0	0.98	0.018	0	0.98	0.0114	0	0.98	0.02
Newthyroid (cluster1)	0	0.54	0.051	0	0.56	0.052	1	0.038	0.17	1	0.04	0.17
Newthyroid (cluster2)	0	0.073	1.0338	0	0.093 e − 005	0.54	1	4.47	1.54	1	4.47	1.54
Newthyroid (cluster3)	0	0.37	0.44	0	0.55	0.38	0	0.57	0.33	0	0.57	0.33

most clustering tasks seems justified and is therefore recommended.

10. Discussion and conclusions

In this paper a new point symmetry-based distance is proposed which incorporates both the Euclidean distance as well as a measure of symmetry with respect to a point in its computation. Kd-tree based nearest neighbor search is used to reduce the complexity of symmetry-based distance computation. A genetic clustering technique (GAPS) is also proposed here that incorporates the new point symmetry distance while performing cluster assignments of the points and in the fitness computation. The major advantages of GAPS are as follows. In contrast to *K*-means, use of GA enables the algorithm to come out of local optima, making it less sensitive to the choice of the initial cluster centers. Again, the proposed GAPS is able to detect clusters that may be of widely varying sizes, where *K*-means, SBKM and Mod-SBKM fail. Such situations may arise in several real-life domains, e.g., medical images, satellite images, fraud detection. Incorporation of the proposed PS-distance enables GAPS to detect symmetric clusters, both convex and non-convex, even if the clusters are symmetrical with respect to some intermediate point. This is in contrast to SBKM and Mod-SBKM, which fail in such situations. Moreover, use of Kd-tree makes the computation of the point symmetry distance significantly faster than both SBKM and Mod-SBKM.

Experimental results on different data sets demonstrate the superiority of GAPS as compared to SBKM [41], its modified version Mod-SBKM [42] and the *K*-means algorithm. GAPS is found to provide satisfactory performance both where *K*-means fails but SBKM succeeds as well as SBKM fails but *K*-means succeeds. Results on data sets of Group 3 demonstrate that GAPS is able to detect symmetric clusters of any size where all the other algorithms fail. The superiority of GAPS is also established on seven real-life data sets. These real-life data sets are of different characteristics, with the number of dimensions varying from 4 to 56. Results on 16 artificial and real-life data sets establish the fact that GAPS is well-suited to detect clusters of widely varying characteristics. Comparison of the obtained results by different algorithms are performed by various statistical tests such as ANOVA and MANOVA. The present clustering algorithm based on the proposed PS-distance does not require point and its symmetrical point (or its nearest neighbor) to belong to the same cluster. This may prove to be a disadvantage as was evident for *Data4* where the central cluster got overestimated. Further research needs to be carried out to rectify this limitation. The authors are currently working in this direction. Another area of future research is the development of new cluster validity indices as well as automatic clustering methods based on the proposed point symmetry distance.

Acknowledgement

The authors gratefully acknowledge the comments of the Editor and the anonymous reviewers which helped them in improving the quality of the paper.

References

[1] A.K. Jain, M. Murthy, P. Flynn, Data clustering: a review, ACM Comput. Rev. November (1999).
[2] M.D. Berg, M.V. Kreveld, M. Overmars, O. Schwarzkopf, Cluster Analysis for Application, Academic Press, New York, 1973.
[3] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[4] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.
[5] B.S. Everitt, S. Landau, M. Leese, Cluster Analysis, Arnold, London, 2001.
[6] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[7] B. Ko'vesi, J. Boucher, S. Saoodi, Stochastic k-means algorithm for vector quantization, Pattern Recognition Lett. 22 (2001) 603–610.
[8] M.R. Anderberg, Computational Geometry: Algorithms and Applications, Springer, Berlin, 2000.
[9] T. Kanungo, D. Mount, N.S. Netanyahu, C. Piatko, R. Silverman, A. Wu, An efficient k-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892.
[10] A. Likas, N. Vlassis, J.J. Verbeek, The global k-means clustering algorithm, Pattern Recognition 36 (2003) 451–461.
[11] D. Charalampidis, A modified k-means algorithm for circular invariant clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (12) (2005) 1856–1865.
[12] R.N. Dave, K. Bhaswan, Adaptive fuzzy c-shells clustering and detection of ellipses, IEEE Trans. Neural Networks 3 (5) (1992) 643–662.
[13] R. Krishnapuram, O. Nasraoui, H. Frigui, The fuzzy c-spherical shells algorithm: A new approach, IEEE Trans. Neural Networks 3 (5) (1992) 663–671.

[14] E. Hartuv, R. Shamir, A clustering algorithm based on graph connectivity, Inform. Process. Lett. (2000) 75–181.

[15] P. Hansen, N. Mladenovic, J-means: a new local search heuristic for minimum sum of squares clustering, Pattern Recognition 34 (2001) 405–413.

[16] C. Wong, C. Chen, M. Su, A novel algorithm for data clustering, Pattern Recognition 34 (2001) 425–442.

[17] P. Bradley, U. Fayyad, C. Reina, Scaling clustering algorithms to large databases, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, August (1998) 9–15.

[18] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.

[19] H. Yan, Fuzzy curve-tracing algorithm, IEEE Trans. Syst. Man Cybern. Part B 31 (5) (2001) 768–780.

[20] A. Mikheev, L. Vincent, V. Faber, High-quality polygonal contour approximation based on relaxation, in: Proceedings of Sixth International Conference on Document Analysis and Recognition (ICDAR'01), IEEE Computer Society, Los Alamitos, CA, USA, 2001, p. 0361.

[21] A.M. Yip, C. Ding, T.F. Chan, Dynamic cluster formation using level set methods, IEEE Trans. Pattern Anal. Mach. Intell. 28 (6) (2006) 877–889.

[22] R. Nock, F. Nielsen, On weighting clustering, IEEE Trans. Pattern Anal. Mach. Intell. 28 (8) (2006) 1223–1235.

[23] Y.-L. Chen, H.-L. Hu, An overlapping cluster algorithm to provide non-exhaustive clustering, Eur. J. Oper. Res. 173 (2006) 762–780.

[24] J.C. Bezdek, Fuzzy mathematics in pattern classification, Ph.D. Thesis, 1973.

[25] G. Beliakov, M. King, Density based fuzzy c-means clustering of non-convex patterns, Eur. J. Oper. Res. 173 (2006) 717–728.

[26] T. Kohonen, The 'neural' phonetic typewriter, IEEE Computer 27 (3) (1988) 11–12.

[27] T. Kohonen, Self-organization and associative memory, third ed., Springer, New York, Berlin, 1989.

[28] G. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Comput. Vision, Graphics, Image Proc. 37 (3) (1987) 54–115.

[29] G. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns, Appl. Opt. 26 (23) (1987) 4919–4930.

[30] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA, 1991.

[31] J. Mao, A.K. Jain, A self-organizing network for hyperellipsoidal clustering, IEEE Trans. Neural Networks 7 (1996) 16–29.

[32] M.-C. Su, N. DeClaris, T. Kang Liu, Application of neural networks in cluster analysis, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, vol. 1, Orlando, FL, USA, 1997, pp. 1–6.

[33] M.-C. Su, Y.-C. Liu, A new approach to clustering data with arbitrary shapes, Pattern Recognition 38 (2005) 1887–1901.

[34] I. Gath, A. Geva, Unsupervised optimal fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 11 (1989) 773–781.

[35] D. Gustafson, W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, Proceedings of the IEEE Conference Decision Control, 1979, pp. 761–766.

[36] R.N. Dave, Use of the adaptive fuzzy clustering algorithm to detect lines in digital images, Intell. Robots Comput. Vision VIII 1192 (1989) 600–611.

[37] Y. Man, I. Gath, Detection and separation of ring-shaped clusters using fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 16 (8) (1994) 855–861.

[38] F. Höppner, Fuzzy shell clustering algorithms in image processing: fuzzy c-rectangular and 2-rectangular shells, IEEE Trans. Fuzzy syst. 5 (1997) 599–613.

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

[40] F. Attneave, Symmetry information and memory for pattern, Am. J. Psychol. 68 (1995) 209–222.

[41] M.-C. Su, C.-H. Chou, A modified version of the k-means algorithm with a distance based on cluster symmetry, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 674–680.

[42] C.H. Chou, M.C. Su, E. Lai, Symmetry as a new measure for cluster validity, in: Second WSEAS International Conference on Scientific Computation and Soft Computing, 2002, pp. 209–213.

[43] U. Maulik, S. Bandyopadhyay, Genetic algorithm based clustering technique, Pattern Recognition 33 (2000) 1455–1465.

[44] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, AnnArbor, 1975.

[45] T.W. Anderson, S. Scolve, Introduction to the Statistical Analysis of Data, Houghton Mifflin, Boston, MA, 1978.

[46] T.W. Anderson, An Introduction to Multivariate Statistical Analysis, Wiley, New York, 1984.

[47] D.M. Mount, S. Arya, ANN: a library for approximate nearest neighbor searching, 2005. ⟨http://www.cs.umd.edu/~mount/ANN⟩.

[48] M. Srinivas, L. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Trans. Syst. Man Cybern. 24 (4) (1994) 656–667.

[49] J.L. Bentley, B.W. Weide, A.C. Yao, Optimal expected-time algorithms for closest point problems, ACM Trans. Math. Software 6 (4) (1980) 563–580.

[50] J.H. Friedman, J.L. Bently, R.A. Finkel, n algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Software 3 (3) (1977) 209–226.

[51] S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition 2 (2002) 1197–1208.

[52] A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, R.F. Murtagh, Validity-guided reclustering with applications to image segmentation, IEEE Trans. Fuzzy Syst. 4 (2) (1996) 112–123.

[53] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[54] A. Ben-Hur, I. Guyon, Detecting Stable Clusters using Principal Component Analysis in Methods in Molecular Biology, Humana press, Clifton, UK, 2003.

[55] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 3 (1936) 179–188.

About the Author—SANGHAMITRA BANDYOPADHYAY (SM'05) did her Bachelors in Physics and Computer Science in 1988 and 1991 respectively. Subsequently, she did her Masters in Computer Science from Indian Institute of Technology (IIT), Kharagpur in 1993 and Ph.D. in Computer Science from Indian Statistical Institute, Calcutta in 1998. Currently she is an Associate Professor, Indian Statistical Institute, Kolkata, India. Dr. Bandyopadhyay is the first recipient of Dr. Shanker Dayal Sharma Gold Medal and Institute Silver Medal for being adjudged the best all round postgraduate performer in IIT, Kharagpur in 1994. She has worked in Los Alamos National Laboratory, Los Alamos, USA, in 1997, University of New South Wales, Sydney, Australia, in 1999, Department of Computer Science and Engineering, University of Texas at Arlington, USA, in 2001, University of Maryland, Baltimore County, USA, in 2004, Fraunhofer Institute AiS, St. Augustin, Germany, in 2005 and Tsinghua University, China, in 2006. She has delivered lectures at Imperial College, London, UK, Monash University, Australia, University of Aizu, Japan, University of Nice, France, University Kebangsaan Malaysia, Kuala Lumpur, Malaysia, and also made academic visits to many more Institutes/Universities around the world. She is a co-author of 3 books and more than 100 research publications. Dr. Bandyopadhyay received the Indian National Science Academy (INSA) and the Indian Science Congress Association (ISCA) Young Scientist Awards in 2000, as well as the Indian National Academy of Engineering (INAE) Young Engineers' Award in 2002. She has guest edited several journal special issues including IEEE Transactions on Systems, Man and Cybernetics, Part B. Dr. Bandyopadhyay has been the Program Chair, Tutorial Chair, Associate Track Chair and a Member of the program committee of many international conferences. Her research interests include Pattern Recognition, Data Mining, Evolutionary and Soft Computation, Bioinformatics and Parallel and Distributed Systems.