An automatic shape independent clustering technique

Sanghamitra Bandyopadhyay*

Machine Intelligence Unit, Indian Statistical Institute, 203, B.T. Road, Kolkata 700 108, India

Received 9 May 2002; received in revised form 23 January 2003; accepted 19 May 2003

Abstract

This article describes a clustering technique that can automatically detect any number of well-separated clusters which may be of any shape, convex and/or non-convex. This is in contrast to most other techniques which assume a value for the number of clusters and/or a particular cluster structure. The proposed technique is based on an iterative partitioning of the relative neighborhood graph, coupled with a post-processing step for merging small clusters. Techniques for improving the efficiency of the proposed scheme are implemented. The clustering scheme is able to detect outliers in data. It is also able to indicate the inherent hierarchical nature of the clusters present in a data set. Moreover, the proposed technique is also able to identify the situation when the data do not have any natural clusters at all. Results demonstrating the effectiveness of the clustering scheme are provided for several data sets.

Keywords: Graph partitioning; Hierarchical clusters; Non-convex clusters; Relative neighborhood; Unsupervised pattern classification; Variable number of clusters

1. Introduction

Clustering [1–4] is an exploratory data analysis tool that deals with the task of grouping objects that are similar to each other. It has applications in a wide domain viz., pattern recognition, biological sciences, social sciences, psychology, data mining, etc. Clustering in N-dimensional Euclidean space $\mathbb{R}^N$ is the process of partitioning a given set of n points into a number, say K, of groups (or, clusters) based on some similarity/dissimilarity metric. Let the set of n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be represented by the set X and the K clusters be represented by $\mathscr{C}_K = \{C_1, C_2, \ldots, C_K\}$. Then, in general,

$$C_i \neq \emptyset \quad \text{for } i = 1, \ldots, K,$$

$$C_i \cap C_j = \emptyset \quad \text{for } i = 1, \ldots, K,\ j = 1, \ldots, K \text{ and } i \neq j$$

and

$$\bigcup_{i=1}^{K} C_i = X.$$

Many types of clustering algorithms have been developed, the prominent among them being the partitional methods, hierarchical methods and graph theoretic methods. Typical examples of the three methods are the well-known K-means algorithm, single linkage and the minimal spanning tree-based algorithms, respectively [1,5]. Some more recent clustering algorithms may be found in Refs. [6–9]. Other algorithms are currently being developed that are appropriate for large data sets [10,11].

The first task prior to clustering data involves identifying whether or not there is any cluster structure. For example, a data set where all the values are drawn from a uniform distribution or from any unimodal distribution will not exhibit natural groupings. The task of assessing whether a data set has inherent structure in it is called estimating its clustering tendency. Once it is determined that clustering needs to be performed, identifying a good clustering method becomes a challenge. Most of the clustering algorithms often require the number of clusters to be specified a priori. For example, the K-means algorithm and the single linkage method require the a priori specification of the number of clusters, which may not be feasible to do in many real-life situations. The minimum spanning tree (MST) method uses the

* Tel.: +91-33-2577-8085; fax: +91-33-2578-3357.
E-mail address: sanghami@isical.ac.in (S. Bandyopadhyay).

concept of inconsistent edges to get around this problem. However, the definition of inconsistency has been found to be problem specific, and may often require the knowledge about the shape of the clusters [12]. Several other methods also assume the geometry of the data; typically the clusters are assumed to be convex in nature. For example, the algorithms based on minimization of squared error criterion assume hyperspherical clusters of approximately equal sizes.

As an extension to the MST-based methods, and in order to overcome its limitations [13], the concept of relative neighborhood of a finite planar set [14] has been proposed. This has been used for designing graph theoretic clustering algorithm in Refs. [15,16]. Although the above concept is intuitively appealing, as it is based on the visual perceptual model, it has not received much attention. The algorithm in Ref. [15] is based on the relative neighborhood graph, and is able to detect well-separated clusters. However, it requires the specification of a parameter σ, and the performance of the algorithm depends heavily on the proper choice of σ. This is explained in the next section, and experimentally demonstrated in Ref. [16]. In this paper, we also use the concept of relative neighborhood for designing a clustering algorithm, which we call CLUSTER, that is able to detect non-overlapping clusters of any shape without requiring the number of clusters to be known a priori. The algorithm works in an iterative manner, successively partitioning the data set until a termination criterion is met. It is able to provide several levels of clusters in an hierarchy, one nested within the other, up to the level that is natural to the given data set. This is in contrast to the method in Ref. [16], which works in a single pass by finding the connected subgraphs in the region of influence graph, and is unable to provide the cluster hierarchy. Moreover, this is also in contrast to the hierarchical clustering schemes like the single linkage and complete linkage algorithms which generate all possible levels of clusterings, irrespective of whether the data naturally exhibits them or not.

2. Graph theoretical clustering based on relative neighborhood

In this section, we first describe the concept of relative neighborhood graph (RNG) [14]. This is followed by a detailed description of the clustering algorithm proposed by Urquhart [15] that is derived from the RNG.

2.1. Relative neighborhood graph

Let X be a set of points represented as $X = \{x_1, x_2, \ldots, x_n\}$. Two points x_i and x_j are said to be relative neighbors if the following condition holds:

$$d(x_i,x_j) \leqslant \max[d(x_i,x_k), d(x_j,x_k)], \quad \forall x_k \in X,\ k \neq i,j. \tag{1}$$

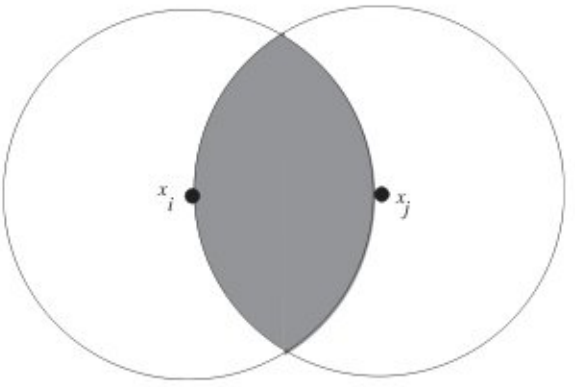

Fig. 1. Relative neighbors and the lune. x_i and x_j are relative neighbors if there are no points that lie within the line.

Intuitively this means that two points are said to be relative neighbors if they are at least as close to each other as they are to any other point. Or, in other words, there is no other point that is closer to x_i (or, x_j) than x_j (or, x_i). This is depicted in Fig. 1. Here, x_i, x_j are relative neighbors if there is no point that lies in the lune formed by these two points (the shaded region). A graph formed by joining each pair of points that are relative neighbors is called the RNG.

2.2. Clustering based on limited neighborhood set

The region of influence of two points x_i and x_j in the RNG, denoted by $R_{RNG}(x_i,x_j)$, formed by applying the definition in Eq. (1) is given as

$$R_{RNG}(x_i,x_j) = \{x : \max[d(x_i,x), d(x_j,x)]) < d(x_i,x_j),\ i \neq j\}. \tag{2}$$

In Ref. [15], an additional parameter σ has been considered, and the region of influence R is enhanced as follows:

$$R(x_i,x_j,\sigma) = R_{RNG}(x_i,x_j) \cup \{x : \sigma \min[d(x,x_i), d(x_j,x)] < d(x_i,x_j),\ i \neq j\}. \tag{3}$$

Here σ is called the relative edge consistency. Fig. 2 shows the shape of a typical region of influence. The clustering algorithm based on this and other definitions of region of influence (for example in Ref. [16]) is given as follows [13].

```
For i=1 to n
    For j=i + 1 to n
        Determine the region of influence R(x_i, x_j)
        If R(x_i, x_j) ∩ {X − {x_i, x_j}} = ∅
            Add the edge connecting x_i, x_j
        EndIf
    EndFor
EndFor
```

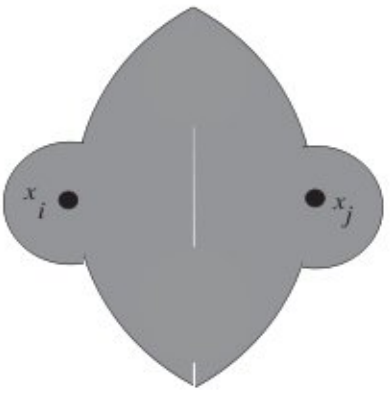

Fig. 2. A typical region of influence.

Determine the connected components of the connected graph. These are identified as clusters.

In Ref. [15], results have been provided showing the ability of the above algorithm in correctly clustering several data sets. However, it can be easily observed that this algorithm is going to fail to yield proper clusters even in many obvious cases. For example, let x and y be two points which obviously lie in different clusters. However, this will be not detected as such if there happens to be no point lying in the region of influence of this pair for some value of σ. This can obviously be the case for very sparse clusters that are reasonably close to each other. It may be argued that increasing σ should be able to resolve this problem. However, this may, in turn, lead to splitting of natural clusters. The failure of the method in Ref. [15] has also been demonstrated in Ref. [16] for several data sets which have obvious clusters. As an extension of the method described in Ref. [15], Osbourn and Martinez [16] have defined some empirically fixed regions of influence (in contrast to the method of Ref. [15] where the region of influence is a factor of the distance between a pair of points) for clustering a data set. It is shown to perform reasonably well for several data sets. However, with the coordinates of the regions fixed a priori, this is also expected to fail under conditions similar to the one mentioned above. Moreover, the method in Ref. [16] is not able to provide hierarchical clustering, which can often be a desired feature.

In the following section, we describe the clustering algorithm proposed in this paper. Analogous to the algorithms described in Refs. [15,16], this method uses the concept of RNG, and is not able to detect touching or overlapping clusters. It is able to produce hierarchical clustering as in Ref. [15], while being more robust.

3. The proposed clustering method

The proposed clustering scheme is based on the successive thresholding of the RNG until a termination criterion is attained. The algorithm called *CLUSTER* is provided below. Here RNG $= (X, E)$ denotes the RNG of the data set X, where the vertices of the graph are the points in X, and E is the set of edges in the RNG. Let the weight of an edge e_{ij}, connecting points x_i and x_j be equal to $d(x_i, x_j)$, the distance between them, and let m be the cardinality of E.

CLUSTER(RNG $= (X,E)$)
Begin
SortedE = Sort E in ascending order of edge weights /*size of *SortedE* $= m$ */
SortedE = Eliminate duplicates in *SortedE* /*size of *SortedE* $= m'$ */
$Min = SortedE[1]$, $Max = SortedE[m']$
if ($Max < 2Min$) return(X) /* cannot cluster X further */
FirstOrder = First Order Difference of *SortedE*
/*i.e., $FirstOrder[i] = SortedE[i+1] - SortedE[i]$ */
/*size of *FirstOrder* $= m' - 1$ */
FirstOrder=Sort *FirstOrder* in ascending order
$t = \frac{FirstOrder[1] + FirstOrder[m'-1]}{2.0}$
thresh $= SortedE[j]$ such that $SortedE[j+1] - SortedE[j] \geqslant t$, $j = 1, 2, \ldots, (m'-1)$
and $SortedE[j] \geqslant 2Min$
if (*thresh* not found) return(X) /* cannot cluster X further */
$RNG' = RNG - \{e_{ij} | d_{ij} > thresh\}$ /*thresh is called the splitting threshold */
Component = connected components in RNG' /*Each element of *Component* is
itself an *RNG* */
if ($|Component| = 1$) return(X) /* cannot cluster X further */
if ($|Component| > \sqrt{|X|}$) return(X) /* Overfragmentation of X. $|X|$ denotes cardinality of X */
else CLUSTER($Component_i$) for $i = 1$ to $|Component|$.
End

CLUSTER is based on the assumption that if x_i and x_j are relative neighbors, and they belong to two distinct clusters, then $d(x_i, x_j) > d(x_i, x_k)$ for all k, such that x_k is in

Fig. 3. An example showing that although the distance ab is greater than *thresh*, the cluster will not be split.

the same cluster as x_i. Therefore, the purpose is to identify an appropriate threshold, such that relative neighbors whose separation exceeds the threshold are put in separate clusters, while those whose separation is smaller than the threshold are put in the same cluster. Note that if this is done in a single pass, then a situation may arise when two distinct clusters are erroneously merged together because of the presence of another cluster that is significantly separated from the other two, and which increases the threshold value. In order to avoid this, CLUSTER is applied iteratively, while keeping the threshold value adaptive, which is computed using the *Component* under consideration. Note that, although the methods in Refs. [15,16] also implicitly use such a threshold, the differences are: (i) the technique in Ref. [16] operates in a single pass, and is, therefore, unable to indicate the nested, or hierarchical, clusters, and (ii) the parameters are fixed a priori in Ref. [15], which may not work equally well for all the data sets.

Below we outline some characteristics of the CLUSTER algorithm as well as suggest some improvements to its basic structure:

- The above algorithm terminates under the following conditions:
  - When the inter-cluster relative neighbors are close to each other (the condition Max $<$ 2 Min in CLUSTER).
  - An appropriate *thresh* ($\geqslant$ 2 Min) is not found.
  - When only one cluster is formed by the algorithm, i.e., $|Component| = 1$. This can happen in spite of finding a suitable *thresh*. For example, consider Fig. 3 that shows a part of the RNG corresponding to a set of points X. Here points a and b are connected directly, as also through the set of remaining points. Assume that *thresh* is so chosen as to break only the direct link between a and b. However, these two points will still remain connected, and X will not be split further.
- The above algorithm is guaranteed to terminate. This will surely happen when the clusters comprise two points. However, such an overfragmented condition will not arise in practice, since the algorithm will terminate well before that.

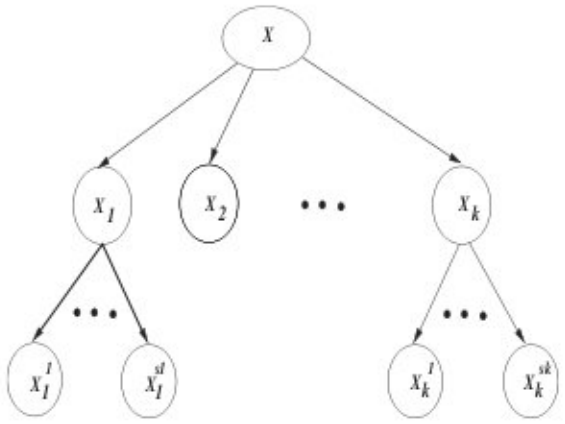

Fig. 4. Clusters obtained in an hierarchy.

- The above algorithm gives rise to hierarchical clusters because of the recursive call to CLUSTER. This is demonstrated in Fig. 4. Here X denotes the entire data set. Let us assume that in the first pass, k connected components (denoted by $X_1, X_2, \ldots, X_k$) are found. In the second pass, each of these X_i's, $i = 1, \ldots, k$, may be further divided into *si* components. Note that it is not necessary that all the components can be partitioned further. For example, it may not be possible to partition X_2 any further, while this may be possible for the other components. This process continues until none of the components can be partitioned any further. The user may also desire to stop at some given level of the hierarchy.
- The number of clusters is determined automatically by the method, and this is equal to the number of *Components* formed on termination of the algorithm. It may be mentioned in this context that for a data set of size n, the rule of the thumb states that the maximum number of clusters that may be present is approximately $\sqrt{n}$. Hence, a stopping criterion is kept which checks for this condition, and terminates accordingly.
- Very small clusters may be formed as a consequence of the above algorithm. A condition may be included which checks the size of a component, and if it is smaller than some threshold (which may be prespecified or kept adaptive), then it is merged with the nearest component cluster $Component_j$, unless the threshold value at which the small cluster got partitioned out is larger than λ times the Max value of $Component_j$ where $\lambda \geqslant 2$. In our experiments, we have kept $\lambda = 3$, although we have tested with several other values as well. This can effectively inhibit the formation of very small clusters because of some minor variation in the distances.
- The above scheme will not merge small clusters which are at a significant distance from the other points in the data set. These points may effectively be considered as outliers or noise, which should be removed from the data

set before further processing. Thus, the said scheme is able to detect such noisy points as well.

- It may be seen that the complexity of one pass of CLUSTER is $O(m \log m)$, where m is the number of edges in the RNG. This is due to the observation that the complexity is guided mainly by the two sorting routines. The initial sorting on the E array is of complexity $O(m \log m)$. The next sorting on *FirstOrder* is of complexity $O(m' \log m')$, where $m' \leqslant m$. Hence, the overall complexity is $O(m \log m)$. Although, the sorting routine for *SortedE* is invoked for every call of CLUSTER, this can be easily eliminated as follows. All subsequent calls of CLUSTER can effectively use the sorted array of the first pass after suitably retaining only those distances that are relevant to the $Component_i$ for the particular invocation of CLUSTER.
- The complexity of the method may be further reduced if the task of sorting the edges in E can be eliminated, and the size of *FirstOrder* can be reduced. In this context it may be mentioned that the edge weights that are very close to each other can be effectively treated as same, and can be clubbed together. However, this clubbing should ensure that edges between clusters are retained separately. A suitably small discretization of the values between Min and Max, defined as a fraction of Max − Min, may be done for this purpose. Let us denote this fraction by δ. Then the maximum number of possible discretizations (or, the maximum number of values in *SortedE*) is $1/\delta$. Obviously, for improved performance $1/\delta \ll m$. Using this discretization, for each edge weight, the interval in which it lies can be directly computed, hence eliminating the need for sorting the weights. Again, since this reduces the size of *SortedE*, the size of *FirstOrder* is also likely to be reduced, thereby reducing the time required to sort it.
- The RNG of a set of points may be computed in $O(n^2)$ complexity, where n is the size of the data set, by the method proposed in Ref. [16]. Hence, the overall complexity of the clustering algorithm is $O(n^2 + m \log m) \approx O(n^2)$.

4. Experimental results

The effectiveness of the clustering algorithm is demonstrated on eight data sets of different characteristics. The data sets are first described followed by a detailed description of the results and their analysis. For the experiments, clusters of size less than 5% of the size of the data set are merged. The value of δ is kept equal to 0.001, i.e., the range [Min, Max] is discretized into 1000 intervals. We have also experimented with other values of δ viz., 0.005 and 0.01 with similar results. As mentioned earlier, the value of λ (used for detecting outliers while merging small clusters) is kept equal to 3 for the experiments. However, we have tried different values of λ in the range of 2–10 without having any change in the results.

Table 1
Description of the data sets

Data	Number of points	Number of dimensions	Number of clusters
Normal	300	2	3
RC1	42	2	2
ADS 1	557	2	3
ADS 2	417	2	2
Encircle	200	3	2
Concentric	3000	6	2
Concentric_noisy	3002	6	2 + 2 outlier points
Uniform	1000	3	none

4.1. Data sets

This section describes the data sets used for the experiments. Table 1 summarizes the number of points, dimensions and the number of clusters in each data set.

Normal: This is a two-dimensional, three class data generated using a mixture of three Gaussian distributions with equal a priori probabilities of each class ($=\frac{1}{3}$). The data set has 300 points (see Fig. 5). The parameters of the three normal distributions are

$$\mu_1 = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 1.5 \\ 0.0 \end{pmatrix} \quad \text{and} \quad \mu_3 = \begin{pmatrix} 0.0 \\ 3.5 \end{pmatrix},$$

$$\Sigma_1 = \begin{bmatrix} 0.01 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 0.01 & 0.0 \\ 0.0 & 1.0 \end{bmatrix} \quad \text{and}$$

$$\Sigma_3 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.01 \end{bmatrix}.$$

RC1: This data set is taken from Ref. [17]. It consists of two clusters which have significantly different sizes. Fig. 6 demonstrates the data set.

ADS1: This data set is shown in Fig. 7. As can be seen from the figure, it has one cluster totally surrounding two smaller clusters. There are 557 data points here. The cluster boundaries are seen to be highly non-linear, although the classes are separable.

ADS2: This data set is shown in Fig. 8. As can be seen from the figure, it has two horseshoe-shaped clusters. There are 417 points in this data set. The cluster boundaries are again seen to be highly non-linear, although the classes are separable.

Encircle: This is a three-dimensional data set having two clusters. The data set is shown in Fig. 9. There are 200 points, with 100 in each class. One class is elongated while the other class encircles the first around its middle.

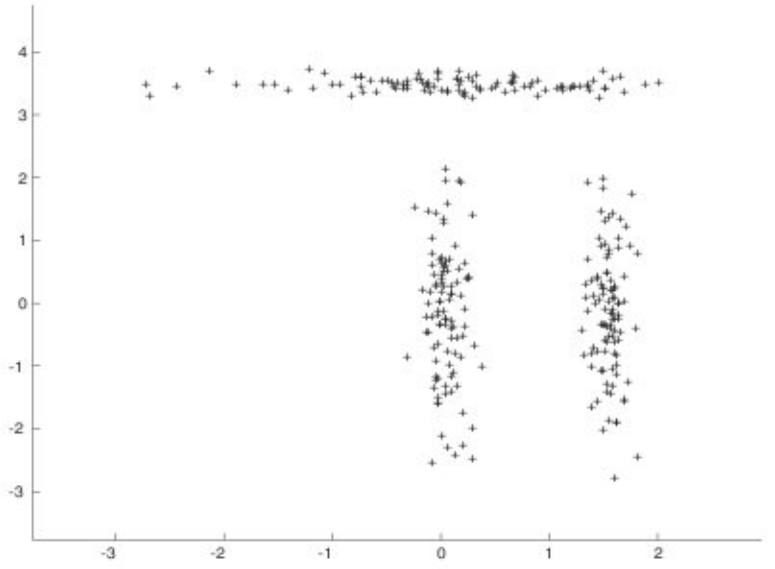

Fig. 5. *Normal* data set.

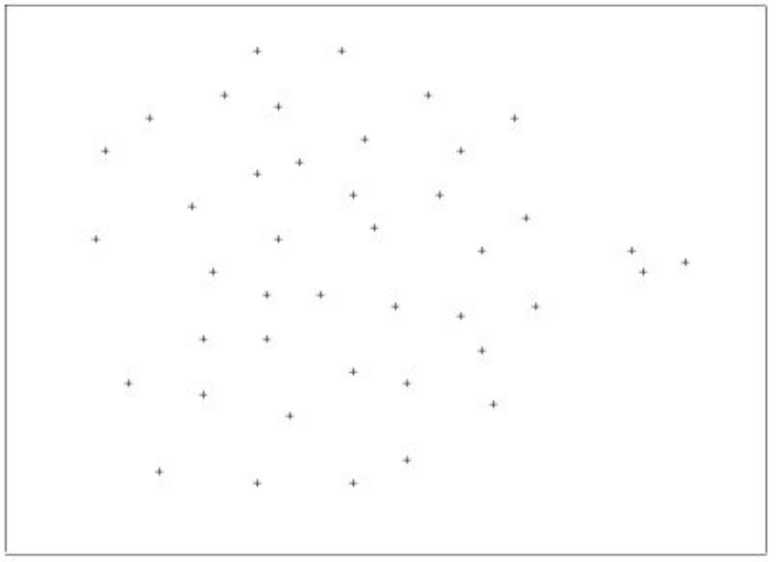

Fig. 6. *RCI* data set.

Concentric: This is a six-dimensional data set which consists of a hypersphere surrounded by a concentric hyper-ring. Fig. 10 shows the conceptual structure of the data set in two dimensions. It has 3000 points, with 500 points in the inner class, and 2500 points in the outer class. It is used for demonstrating the effectiveness of the clustering scheme for high-dimensional data sets, where the number of points in the classes vary significantly. Moreover, it also demonstrates that estimating the number of clusters for methods like single linkage algorithms can be extremely difficult especially for high-dimensional data sets, since a projection of such data on a two-dimensional plane can hardly indicate its nature. Fig. 11, which is a two-dimensional projection of the six-dimensional data, underlines this fact. In order to demonstrate the effectiveness of the technique in detecting outliers, a few points were added to this data set which were significantly separated from the other points. We call this data set *Concentric_noisy*.

Uniform: This is a three-dimensional data set that is uniformly distributed in a unit cube. It has 1000 points. This

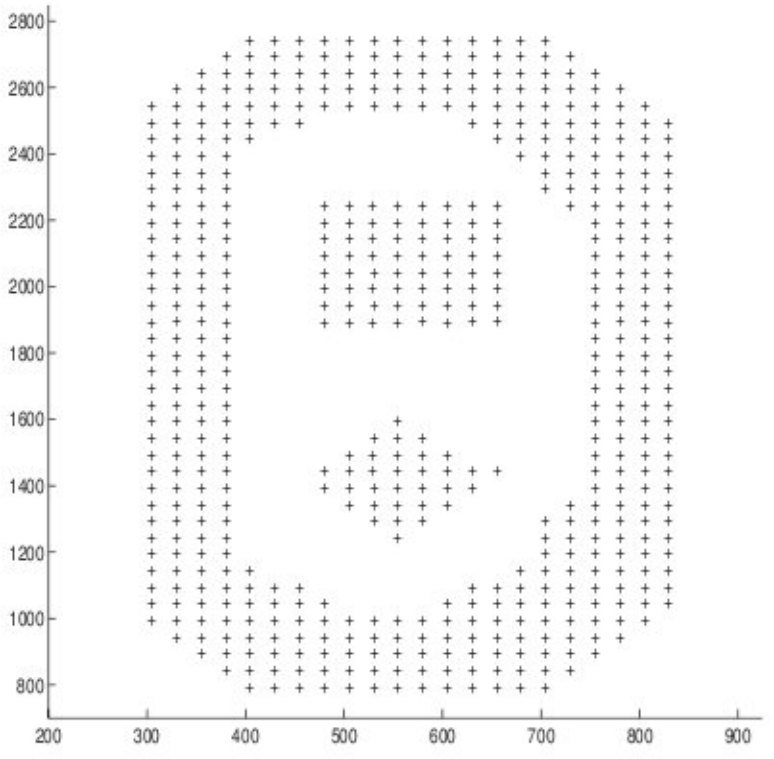

Fig. 7. *ADS 1* data set.

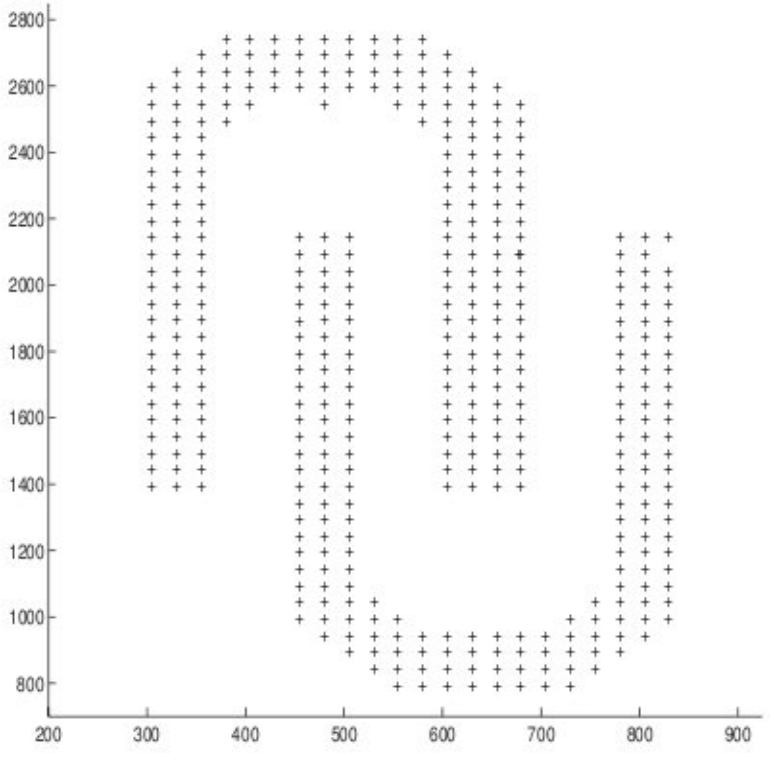

Fig. 8. *ADS 2* data set.

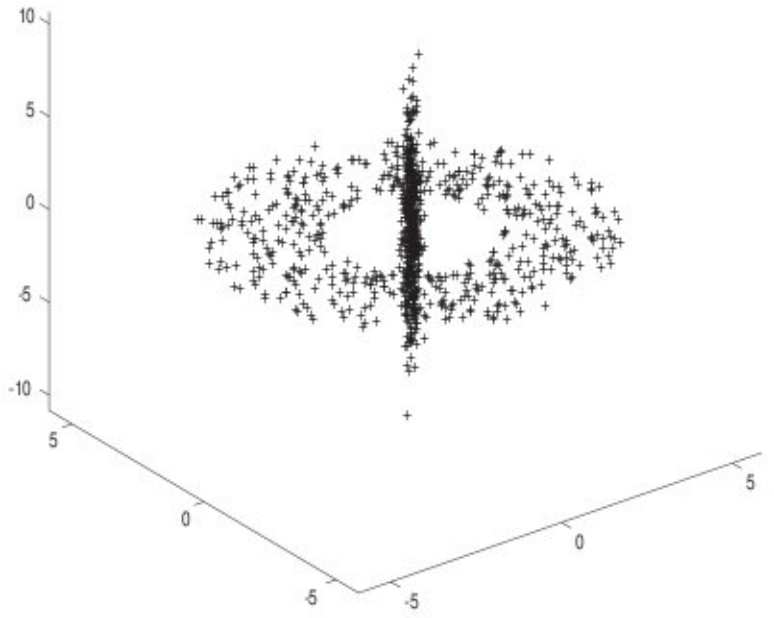

Fig. 9. *Encircle* data set.

Fig. 10. A conceptual view of *Concentric*.

data set has no natural clusters, and has been chosen because of this particular characteristic.

4.2. Partitions obtained by the clustering method

The partitioning obtained for *Normal* is shown in Fig. 12. The values of Min, Max and *thresh* are 0.003761, 1.281093 and 0.939181, respectively. On the first pass, three clusters are formed. However, after subsequent calls to CLUSTER, some small clusters are obtained (the maximum number of points in these clusters is 4). While merging, it is found that the splitting threshold for these clusters (e.g., it is 0.224714 in a particular case) is close to the Max value of the closest component (e.g., it is 0.197570 for the corresponding closest component). Since $0.224714 < 3 \times 0.197570$ ($\lambda = 3$), hence these points are merged with the nearest cluster. For *RC1*, the values of Min, Max and *thresh* are 2.236068, 24.515301 and 24.504151, respectively. The final clustered data is shown in Fig. 13. Note that the final partitioning is again obtained after several recursive calls to CLUSTER and subsequent merging of small clusters.

The clustered *ADS 1* and *ADS 2* are shown in Figs. 14 and 15. The first data set is partitioned into three clusters of sizes 33, 64 and 460. The three clusters thus formed could not be split further in the next pass of CLUSTER. Similarly *ADS 2* is split into two clusters of sizes 194 and 223, and these could not be split further in the subsequent pass of the algorithm.

The three-dimensional *Encircle* data set is split into three clusters in the first pass, of which two correspond to the actual two clusters, and the third is a singleton cluster (the point lying at the bottom of cluster 1 in Fig. 16). The values of Min, Max and *thresh* are 0.007605, 2.488116 and 1.794119, respectively. Some small clusters split out of cluster 2 in subsequent calls to CLUSTER, while cluster 1 was not split further. After merging of small clusters, the resultant partitioning of the data set is shown in Fig. 16.

For the six-dimensional *Concentric* data, two clusters (which correspond exactly to the natural clusters in the data set) are found automatically by the method. The values of Min and Max are, respectively, 0.004019 and 1.226477, and the threshold value is computed as 1.219746. Although the

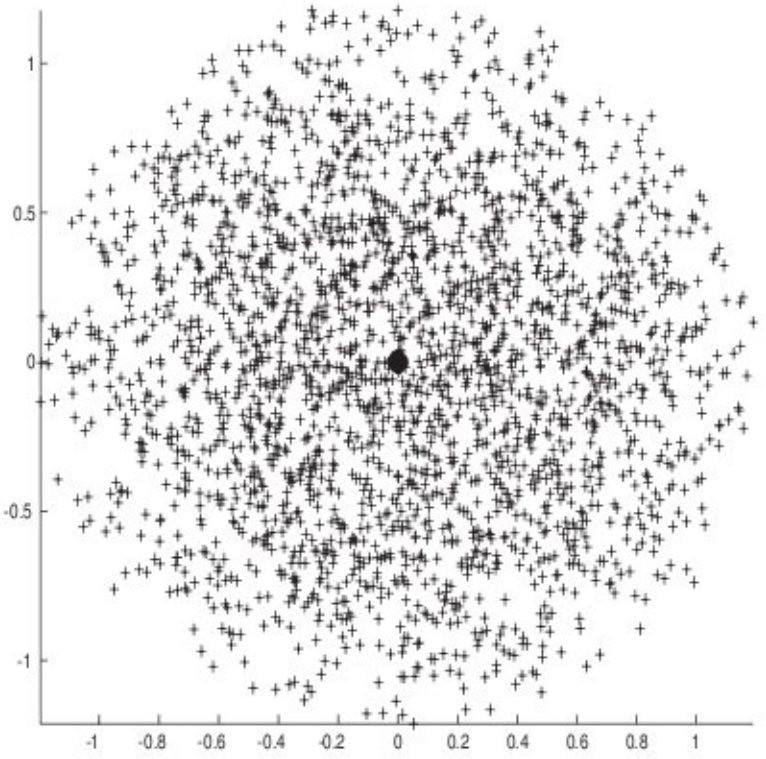

Fig. 11. Two-dimensional projection of *Concentric* data set.

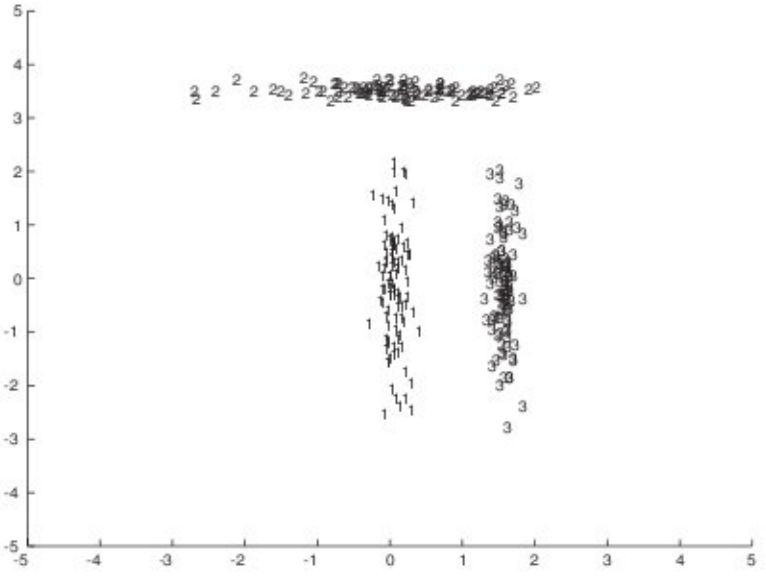

Fig. 12. Clustered *Normal*.

range (Max − Min) is divided into 1000 intervals, the size of the *SortedE* and *FirstOrder* arrays were, respectively, 481 and 480. This indicates that in practice, the number of elements to be sorted is reduced significantly. (The same was observed for the other data sets as well.) On recursive calls to CLUSTER with each of the two clusters formed,

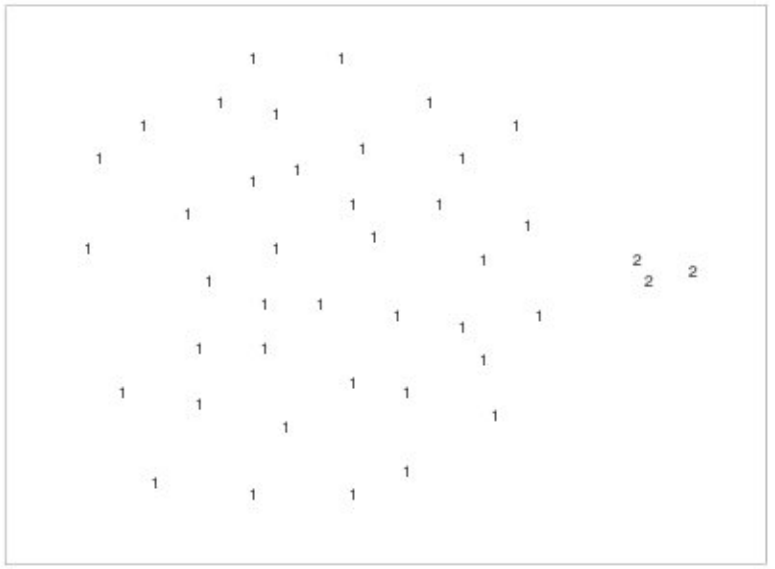

Fig. 13. Clustered *RC1*.

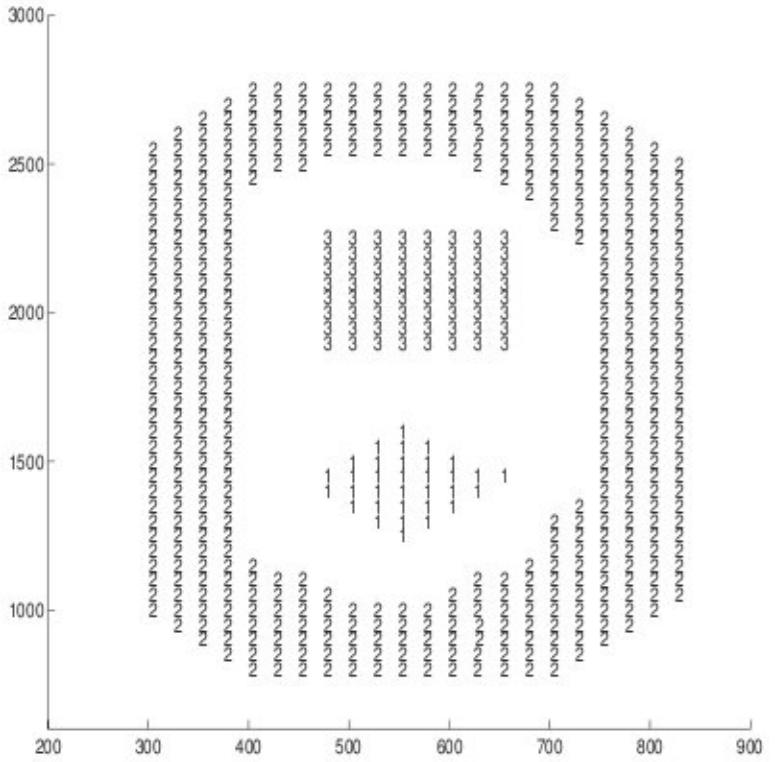

Fig. 14. Clustered *ADS 1*.

it is found that although both are candidates to be clustered (i.e., for both of them the condition Max $\geqslant 2$ Min was met), the number of clusters formed is one in each case. For example, for the inner class, although Max = 0.017861 and Min = 0.003606, and *thresh* = 0.017226 was found, the cluster was not spilt. This effectively indicated that the

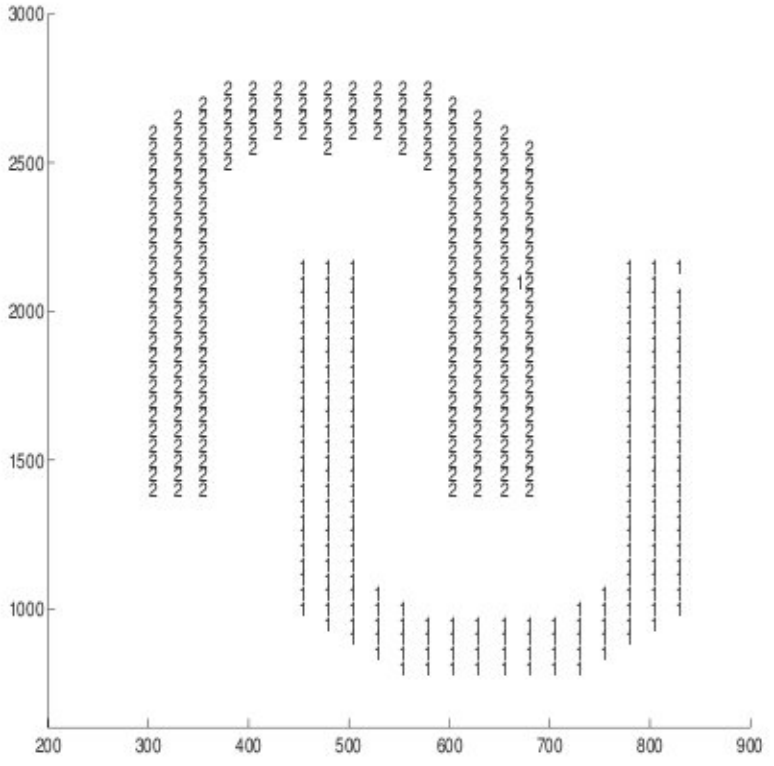

Fig. 15. Clustered *ADS 2*.

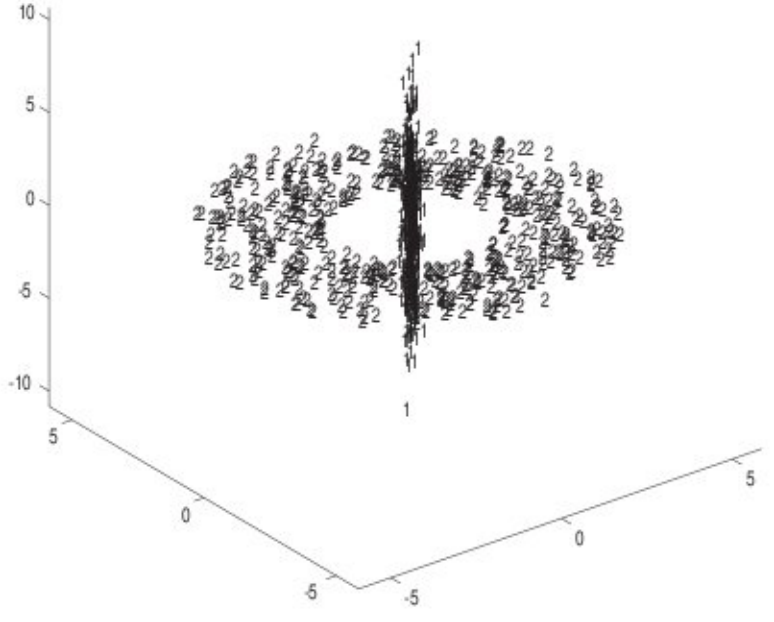

Fig. 16. Clustered *Encircle*.

clusters formed after the fist pass could not be further split. For the case of *Concentric_noisy*, the outlier points are first partitioned out into two groups with a threshold value of 11.161583. The remaining points are thereafter clustered into two groups at a threshold of 1.219746, which could not be partitioned further. When attempting to merge the

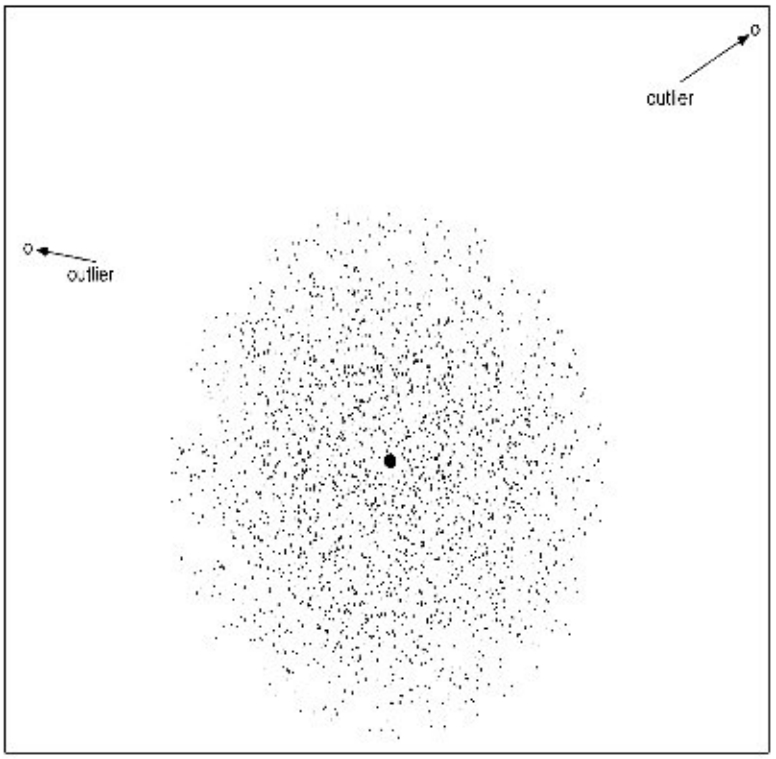

Fig. 17. Clustered *Concentric_noisy* data set projected in the first two dimensions. The outlier points are marked. The inner black region formed from '+' and the dots ('.') correspond to the two clusters identified by CLUSTER.

small clusters with the nearest cluster (which in fact is the outer ring), it is found that the corresponding Max value is 0.821710, while the splitting threshold as mentioned earlier is 11.161583. Since $11.161583 > 3 \times 0.821710$, these points are considered to be outliers, and not merged with the nearest cluster. The clustered *Concentric_noisy* data set, projected on the first two dimensions, is shown in Fig. 17. The inner black regions corresponds to one cluster, and the dots correspond to the other cluster. The outlier points are also marked.

In case of the *Uniform* data, the values of Min, Max and *thresh* are 0.004329, 0.196130 and 0.171075, respectively. In spite of a call to CLUSTER, and computation of the threshold value mentioned above, the algorithm yields only one cluster for this data set, indicating in effect that it has no clustering tendency. This is expected since the points are distributed uniformly in a unit cube. It may be mentioned that even if the method were asked to find some number, say K, of clusters in this data, it would terminate with a message that the data cannot be clustered. This is certainly an intuitively appealing and a desirable characteristic of the clustering algorithm. Note that most other clustering techniques, e.g., the partitional methods like the K-Means, will perforce partition the data set into the said number of clusters.

5. Discussion and conclusions

A clustering technique, CLUSTER, based on an iterative partitioning of the relative neighborhood graph has been described in this article. It is able to automatically cluster the data into an appropriate number of partitions. The clusters need not be of any predefined shape, and may be both convex and non-convex in nature. CLUSTER involves the identification of an appropriate threshold value, which is kept adaptive, and computed based on the current partition being considered. Although the proposed clustering scheme involves the setting of some parameters, it has been found to be robust with respect to the values of these parameters. As with other techniques based on relative neighborhood graphs, CLUSTER is able to detect clusters that are separated. The technique employs a post-processing step of merging small clusters. However, points that are outliers of the data are not merged, even if they form very small clusters of their own.

A characteristic feature of the method is that it is able to provide an hierarchy of clusters, one nested within the other, till it obtains compact clusters. No further division of the clusters will subsequently take place. This is in contrast to many other hierarchical clustering schemes which provide all levels of the hierarchy, from one to n

clusters, where n is the number of points in the data set. These schemes generate all the levels of clustering, although the data may not naturally exhibit all these levels of clustering.

The effectiveness of the clustering technique has been extensively demonstrated for eight data sets having different characteristics. The number of dimensions ranges from two to six. One particular data set, *Uniform*, is so chosen that it does not have any inherent cluster structure. The scheme CLUSTER is automatically able to identify this fact and yields no clusters for this data set. Another data set, *Concentric_noisy*, is chosen in order to demonstrate the robustness of the scheme in the presence of noisy, outlier points. The proposed scheme is able to identify that the small clusters formed from these outlier points should not be merged with the other clusters. Such small clusters may be ignored or handled specially during further high-level processing of the data.

As a scope for future work, we aim to extend the method so that it can be equally effective in case of touching or overlapping clusters. This may prove to be a difficult problem to handle, and therefore extremely challenging. A sensitivity analysis of the method, as well as extensive comparison with some other schemes may constitute another important direction of further research.

Acknowledgements

The author is grateful to the reviewer for the constructive and helpful comments.

References

[1] M.R. Anderberg, Cluster Analysis for Application, Academic Press, New York, 1973.
[2] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[3] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.
[4] B.S. Everitt, Cluster Analysis, 3rd Edition, Edward Arnold, London, 1993.
[5] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[6] K. Alsabti, S. Ranka, V. Singh, An efficient k-means clustering algorithm, in: IPPS: 11th International Parallel Processing Symposium, IEEE Computer Society Press, Silver Spring, MD, 1998.
[7] E. Hartuv, R. Shamir, A clustering algorithm based on graph connectivity, Inf. Process. Lett. (2000) 175–181.
[8] P. Hansen, N. Mladenovic, J-Means: a new local search heuristic for minimum sum of squares clustering, Pattern Recognition 34 (2001) 405–413.
[9] C. Wong, C. Chen, M. Su, A novel algorithm for data clustering, Pattern Recognition 34 (2001) 425–442.
[10] P.S. Bradley, U.M. Fayyad, C. Reina, Scaling clustering algorithms to large databases, in: Fourth International Conference on Knowledge Discovery and Data Mining, August 1998, New York City, USA, AAAI Press, pp. 9–15.
[11] 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, Montreal, Canada, 1996, pp. 103–114.
[12] C. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. C-20 (1972) 68–86.
[13] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, New York, 1999.
[14] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognition 12 (1980) 261–268.
[15] R. Urquhart, Graph theoretical clustering based on limited neighborhood sets, Pattern Recognition 15 (1982) 173–187.
[16] G.C. Osbourn, R.F. Martinez, Empirically defined regions of influence for cluster analysis, Pattern Recognition 28 (1995) 1793–1806.
[17] A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, R.F. Murtagh, Validity guided (re)clustering with applications to image segmentation, IEEE Trans. Fuzzy Syst. 4 (1996) 112–123.

About the Author—SANGHAMITRA BANDYOPADHYAY 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 a faculty member at Indian Statistical Institute, Calcutta, 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 post graduate performer in IIT, Kharagpur in 1994. She has worked in Los Alamos National Laboratory, Los Alamos, USA, in 1997, as a graduate research assistant, in the University of New South Wales, Sydney, Australia, as a post doctoral fellow, and in the Department of Computer Science and Engineering, University of Texas at Arlington, USA, as a faculty and researcher. 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. Her research interests include Pattern Recognition, Data Mining, Evolutionary and Soft Computation, Image Processing and Parallel & Distributed Systems.