Expert Systems With Applications 132 (2019) 28–43

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree

Gaurav Mishra*, Sraban Kumar Mohanty

Department of Computer Science and Engineering, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India

ARTICLE INFO

Article history:
Received 19 December 2018
Revised 19 April 2019
Accepted 19 April 2019
Available online 20 April 2019

Keywords:
Hybrid clustering
Local nearest neighbor
Minimum spanning tree
Dispersion
Merge index
Gene expression dataset

ABSTRACT

With rapid explosion of information, clustering emerged as an active research area for knowledge discovery. Most of the existing clustering algorithms become ineffective when inappropriate parameters are provided or applied on a dataset which consists of clusters of diverse shapes, sizes, and varying densities. To overcome these issues, many graph based hybrid clustering algorithms have been proposed but these algorithms first generate a complete graph of the dataset which takes $O(N^2)$ time where N is the number of data points which limits their application on large datasets. This paper proposes an algorithm namely a fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree to reduce the computational overhead. In the first step, the algorithm partitions the dataset into large number of sub-clusters based on dispersion of data points to capture the geometry of clusters. After partitioning the dataset, a minimum spanning tree based on the centroids of each of the sub-clusters is constructed to identify the adjacent pairs. A novel merge method is proposed to find the genuine clusters by repeatedly merging the adjacent sub-clusters. The cohesion and intra-similarity are introduced to compute the level of dispersion of data points with respect to the centers of an adjacent pair and average edge weight of a sub-clusters respectively. The algorithm takes $O(N^{3/2})$ time which is a $\sqrt{N}$ factor improvement over the popular hybrid clustering algorithms. Experimental analyses on both synthetic as well as gene expression datasets demonstrate that the proposed technique shows significant improvement over competing clustering algorithms in terms of execution time and improved cluster quality. Moreover, the proposed algorithm does not require any user defined parameters and it can estimate the number of clusters more accurately.

1. Introduction

Clustering is an unsupervised learning technique which has been used extensively to discover intrinsic groups in a given dataset in many real life applications. It aims to form groups on a given dataset in such a way that data points within a cluster are more similar to each other than to those in a different clusters based on some similarity measure. Because of its wide application in different areas, many clustering algorithms have been proposed, but most of them become inadequate when improper parameters are given or when applied on the datasets of different shapes, sizes, and varying densities and also due to their high computational complexity. The clustering techniques are broadly classified into partitional, hierarchical and density based methods (Bouguettaya, Yu, Liu, Zhou, & Song, 2015; Chen, 2015; Ester, Kriegel, Sander, Xu et al., 1996; Jain, 2010; Karypis, Han, & Kumar, 1999; Murtagh, 1983; Zhou, Xu, & Liu, 2017). Partitional clustering methods directly split the N data points into k clusters using an objective function, e.g., sum of squared error (Chen, 2015; Jain, 2010). The partitional clustering algorithm is popular because of its linear complexity and ability to detect the spherical shaped clusters. However, they fail to identify the convex shaped and irregular size clusters and number of clusters, k, is required a priori.

Hierarchical clustering method can identify the good clusters, but only at the cost of intensive computation (Bouguettaya et al., 2015; Murtagh, 1983; Zhou et al., 2017). The main advantage of hierarchical approach is to produce a nested sequence of clusters in the from of a binary tree called dendrogram which is more informative than the partitional approach. Hierarchical clustering methods are further classified into agglomerative and divisive approaches. Agglomerative clustering methods begin by considering each data points as a cluster and iteratively combine the two most

* Corresponding author.
E-mail addresses: gaurav.m@iiitdmj.ac.in (G. Mishra), sraban@iiitdmj.ac.in (S.K. Mohanty).

https://doi.org/10.1016/j.eswa.2019.04.048
0957-4174/

similar clusters based on some linkage criteria until exact number of clusters are obtained (Bouguettaya et al., 2015; Karypis et al., 1999; Murtagh, 1983; Zhou et al., 2017) in $O(N^2 \lg(N))$ time. Compared to agglomerative approaches, divisive approaches are more computationally intensive. They begin with all data points in a single cluster and iteratively partition them into two sub-clusters (Guénoche, Hansen, & Jaumard, 1991). For bi-partitioning the N data points into $k-$cluster, a divisive method can produce the global optimal result, if all possible $2^N - 1$ bi-partitions are considered which limits the application of it for large datasets.

Density based clustering techniques were introduced where dense regions of data points are considered as clusters for identifying arbitrary shape clusters (Kriegel, Kröger, Sander, & Zimek, 2011). One of the most popular density based clustering approach is density based spatial clustering of application with noise (DBSCAN) (Ester et al., 1996). It clusters data points with their closed neighbors and data points that lie isolated in low density regions are called outliers. The cost of finding the nearest neighbor of a point is $O(N)$ so for $O(N)$ data points it is $O(N^2)$. The performance of DBSCAN depends on two user defined parameters such as minimum number of nearest neighbors (MinPts) and distance threshold (Eps). DBSCAN is not suitable on datasets having nonuniform density distribution because it is difficult to choose the value of the input parameters (Karypis et al., 1999; Zhong, Miao, & Fränti, 2011).

The traditional clustering approaches fail to detect the clusters of heterogeneous nature of a dataset that consist of diverse shapes, sizes, and varying densities (Karypis et al., 1999; Sutanto & Warwick, 1995; Zhong et al., 2011). To overcome this issues, many graph based clustering methods have been proposed (Grygorash, Zhou, & Jorgensen, 2006; Ren, Li, & Tu, 2015; Schaeffer, 2007; Zahn, 1971). In graph based approaches, the dataset is represented as $G = (V, E)$ in which vertex set (V) represents the data points and edge set E illustrates the similarity between data points. The most popular graph based clustering algorithm is minimum spanning tree(MST) based clustering algorithm (Grygorash et al., 2006; Jothi, Mohanty, & Ojha, 2015; Zhong et al., 2011). Minimum spanning tree is a spanning tree of a graph such that sum of edge weights is minimized. Longest edges in a MST is considered as inconsistent edges. The inconsistent edges are removed from the graph to find out the sub-graphs(clusters).The advantage of MST based clustering algorithm is to detect the clusters of irregular boundaries. The computational complexity of MST based clustering algorithm is $O(N^2)$ which limits the application of MST. As every pair of data points is associated with edge, the cost of finding the nearest neighbor of a point is $O(N)$ and the total cost for $O(N)$ data points is $O(N^2)$. Another drawback of MST based clustering algorithms is to identify the inconsistent edges to partition the MST to obtain the desired clusters. Most of the existing clustering algorithm failed to derive an objective function for identifying the optimal partition (Jothi et al., 2015; Karypis et al., 1999; Zhong et al., 2011).

These traditional approaches are not suitable for clustering a heterogeneous dataset because the difficulty of choosing the input parameters or the model is not adequate to capture the structure of clusters. Most of them also take quadratic time in N(number of data points). Thus, the high computational cost limits the application of theses methods for large size datasets (Karypis et al., 1999; Zhong et al., 2011). To overcome these issues, several hybrid clustering algorithms have been proposed that combine the advantage of these methods. Hybrid clustering algorithm analyzes the dataset in two phases. In the first phase, the dataset is partitioned into a large number of sub-clusters using different partitioning approaches. In the second stage, the produced sub-clusters are merged into the actual clusters based on some merging criteria. Various hybrid clustering algorithms have been proposed in the literature based on the different partition and merging approaches (Guha, Rastogi, & Shim, 2001; Karypis et al., 1999; Lin & Chen, 2005; Liu, Jiang, & Kot, 2009; Zhong et al., 2011).

A hybrid Clustering Using Representatives (CURE) algorithm to handle the large datasets was proposed to take the advantage of both partitional and single link techniques (Guha et al., 2001). In CURE, a cluster is represented by a constant number of representative points and the similarity of two sub-clusters are measured by the closest distance between the representative points in different sub-clusters. Since the selection of representative points of CURE dependents upon shapes and sizes of clusters and outliers in the dataset, therefore the clustering results also depends on the selection of representative points (Huang, Gao, Chiew, Chen, & He, 2014; Karypis et al., 1999). The computational complexity of this algorithm is $O(N^2 \log N)$. A robust and efficient data clustering with cohesion self-merging (CSM) (Lin & Chen, 2005) is an hybrid approach that first uses the k-means to partitions the dataset into k sub-clusters and iteratively merged them into actual clusters based on cohesion. The final clustering results may be unstable because it uses k-means as a partitioning approach which can produce different partitions in different runs (Zhong et al., 2011). Split and merge using multiple prototypes with small separations was proposed in (Liu et al., 2009) to discover the clusters of arbitrary shape and size. However, clustering result of this approach depends on many experimental parameters (Zhong et al., 2011).

CHAMELEON is one of hybrid clustering algorithm (Karypis et al., 1999) which can suitable cluster a heterogenous dataset. It constructs the k-nearest neighborhood graph and applies the graph partition technique to partition the dataset into several sub-clusters. The produced sub-clusters are merged into actual clusters based on the relative inter-connectivity and closeness of the sub-clusters. The computational complexity of it is $O(N^2)$, because it constructs the k-nearest neighborhood graph on a complete graph which limits its application in large datasets. Also preprocessing is performed for choosing the parameters to construct the nearest neighborhood graph (Zhong et al., 2011).

A minimum spanning tree based clustering method using divide-and-conquer approach (DMST) was proposed for constructing an approximate MST (Wang, Wang, & Wilkes, 2009). It detects the longest edges in early stage of clustering which may not participate in the construction of MST, using cut-and-cycle properties of a graph in order to avoid extra computations. However, the worst case complexity of the algorithm still remains $O(N^2)$ though the average case complexity is sub-quadratic. Also, the clustering quality is compromised because many edges are not considered during the construction of approximate MST (Jothi, Mohanty, & Ojha, 2018; Zhong, Malinen, Miao, & Fränti, 2015).

Split-and-merge (SAM) is another example of a hybrid clustering algorithm. It constructs a MST neighborhood graph to represent the dataset (Grygorash et al., 2006; Zhong et al., 2011). It selects the $\sqrt{N}$ highest degree nodes as the cluster center and then applies the k-means algorithm to partition the dataset into large number of sub-clusters. The inter-connectivity and intra-similarity between the sub-clusters are designed to merge the sub-clusters into the actual clusters. The computational complexity of the algorithm is $O(N^2)$, which is dominated by the graph construction phase. SAM can not classify the datasets containing clusters with diverse shaped and different dispersion level (Cheng, Lu, Liu, Huang, & Cheng, 2016).

A parameter free clustering technique called Gaussian density distance (GDD) was proposed using Gaussian kernel and distance considering both data density and shape to determine the clusters (Güngör & Özmen, 2017). The computational complexity of the algorithm is $O(N^2)$. The technique is not suitable for data sets containing touching clusters.

MST is useful graph structure that is used to capture the neighborhood information of data points. Therefore MST is employed for representing the intrinsic structure of heterogeneous datasets. As every pair of data points is associated with an edge, the cost of finding the nearest neighbor of a point is $O(N)$ so the total cost for $O(N)$ data points is $O(N^2)$. If we could represent the structure of the dataset using local neighborhood graph instead of a complete graph, we can reduce the time complexity to some extent.

With this motivation, we propose a graph based hybrid clustering algorithm using local-neighborhood technique to speed up the clustering process. Our clustering algorithm composed of three steps. In the first step, $\sqrt{N}$ dense sub-clusters are extracted from the dataset based on the dispersion of the data points within a sub-cluster. In the second step, the representative point of each-sub-cluster is computed as a sub-centroid and then MST on these sub-centroid is computed to find the adjacent sub-clusters. Finally, the merge index for each adjacent pair is computed to determine the neighboring relationship. We consider the cohesion and intra-similarity to compute the merge index. Cohesion measures how well the sub-clusters are connected in terms of density and distribution of data points located between a adjacent pair. Intra-similarity simply considers the closeness of a sub-clusters based on the average edge weight within that sub-cluster. The intra-similarity of an adjacent sub-clusters becomes high when the similarity of individual sub-clusters are close to each other. The computational complexity of the proposed algorithm is $O(N^{3/2})$ which is a $\sqrt{N}$ factor improvement over most of the hybrid clustering techniques. The proposed algorithm produces the proper partitions of dataset without any user defined parameter.

Experimental analysis on synthetic as well as gene expression datasets is performed to study the efficacy of the proposed technique. We show that the execution time of the proposed algorithm is reduced significantly over the other competing hybrid clustering algorithms. It reveals improved cluster quality in terms of external cluster validity measures as compared to other algorithms on different characteristics of clusters. The experimental analysis also shows that the proposed algorithm can determine the number of clusters in a dataset.

The rest of the paper is organized as follows. The proposed technique is described in Section 2. The results of experimental validation are reported and demonstrated in Section 3, and Section 4 presents the conclusion and future work.

2. Proposed clustering technique

We propose an efficient hybrid algorithm to identify the clusters in heterogeneous datasets. In the traditional clustering techniques, usually similarity between every pair of points are computed. However, in many cases such unnecessary computations can be avoided without compromising the intrinsic property of the data points. For example, to find the nearest neighbor of a data points, it is not necessary to search the whole dataset but a small local portion of it can be sufficient. With this observation, we employ the divide-and-conquer technique to identify the actual clusters with improved efficiency. Following the divide and conquer approach, we illustrate three steps according to Cormen, Leiserson, Rivest, and Stein (2001) to classify fast hybrid clustering algorithm as follows:

Divide step For a given dataset of N data points, the dispersion level of data points is used to partition the dataset into sub-clusters.

Conquer step An exact MST algorithm such as Kruskal's or Prim algorithm is employed to construct the MST for each sub-cluster.

Combine step The produced sub-clusters are merged into actual clusters based on cohesion and intra similarity.

The steps of the proposed algorithm are illustrated in Fig. 2. In the first step, the dataset is partitioned into several sub-clusters based on the local variance in each dimension of the dataset. The local dimension in which the dataset illustrates the maximum variance demonstrates the geometry of the cluster in the dataset to a greater extent. This process is applied in a iterative way to capture the possible local variance in the dataset until we get $\sqrt{N}$ number of sub-clusters. After partitioning, the sub-clusters are merged into actual clusters. In the merging process, it is hard to decide which pair of sub-cluster should be merged. By brute force, all possible pairs to merge. Therefore, the MST neighborhood graph is computed on the sub-centroid of each partition to filtered out the farthest pair of sub-clusters. In MST, only connected sub-clusters are considered as an adajacent pair. A sub-cluster may have more than one adjacent pairs and all of them may not belong to the same cluster. Therefore the actual clusters are obtained by performing an efficient merge approach that aims to maximize the cohesiveness and the intra-cluster similarity between data points of the sub-clusters. The flow chart of proposed approach is depicted in Fig. 1. The details of the proposed algorithm is illustrated in the following subsections.

2.1. Partitioning the dataset

In this step, the dataset is divided into a large number of dense sub-clusters. The proposed partitioning method is a divisive approach in which a cluster C_m is divided into sub-clusters C_{m_1} and C_{m_2} that maximizes the dissimilarity across the sub-clusters. The dataset is partitioned based on the dispersion level of data points. The local dimension in which the data points show the greater variability determines the geometry of the cluster to a greater extent (Rojas-Thomas, Santos, & Mora, 2017). Therefore, we are interested in capturing the geometry of the dataset along the dimension as accurately as possible. To compute the variability of each direction, we compute the variance of each direction to find out the local dimension of maximum variability. Let a set of data points $X = \{x_1, x_2, x_3, \ldots\ldots, x_N\}$, where $x_i = (x_i^1, x_i^2, x_i^3, \ldots\ldots, x_i^d)^T \in \mathbb{R}^d$ is a feature vector. Let $\mu(C_m)$ denote the centroid of the cluster C_m.

Definition 1 (Maximum variance)**.** The maximum variance of a cluster C_m (denoted as $mv(C_m)$) is defined as the dimension in which the data points of C_m show greatest variability and is defined as follows:

$$mv(C_m) = \underset{1\le j\le d}{\operatorname{argmax}} \frac{1}{n}\sum_{i=1}^{n}(x_i^j - \mu(C_m)^j)^2 \tag{1}$$

where $\mu(C_m)^j = \frac{1}{n}\sum_{i=1}^{n} x_i^j$, d is the dimension of the dataset and n is the number of data points in the $m-th$ cluster.

Definition 2 (Furthest point of a cluster along a dimension)**.** The furthest point of a cluster C_m along the dimension $mv(C_m)$ is the furthest data point to the mean in the direction of maximum variance is defined as follows:

$$fp(C_m) = \underset{1\le i\le n}{\operatorname{argmax}} |x_i^{mv(C_m)} - \mu(C_m)^{mv(C_m)}| \tag{2}$$

Definition 3 (Vector of maximum variance)**.** The vector of maximum variance of a cluster C_m denoted as $\vec{v}_{\max}(C_m)$ is the vector whose magnitude is the difference of the centroid and furthest point of C_m along the dimension of maximum variance $mv(C_m)$.

$$\vec{v}_{max}(C_m) = \left(x_{fp(C_m)}^{mv(C_m)} - \mu(C_m)^{mv(C_m)}\right) e_{mv(C_m)} \tag{3}$$

where $e_{mv(C_m)}$ denotes the unit vector along $mv(C_m)$.

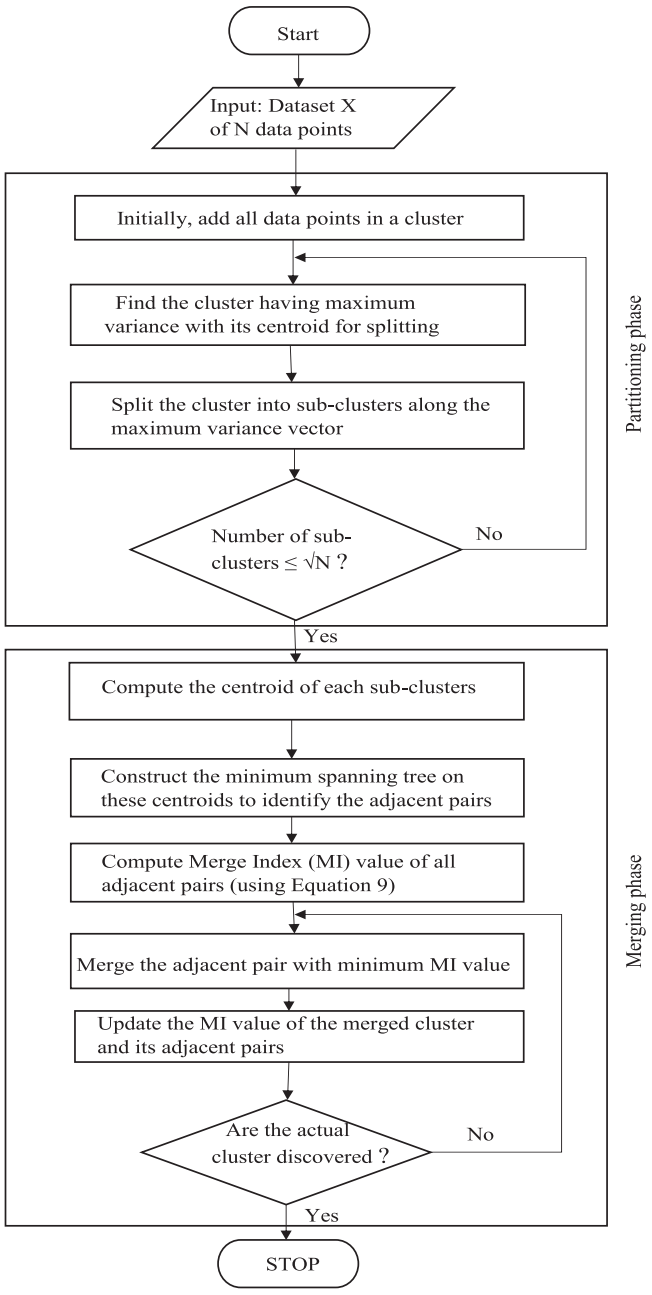

Fig. 1. Flow chart of the proposed algorithm.

Let $\vec{p}_i = \vec{x}_i - \vec{\mu}(C_m)$ denote the position vector of $\vec{x}_i$ with respect to the centroid $\mu(C_m)$ (Rojas-Thomas et al., 2017). Let the position vector of each data points with respect to the centroid of C_m is $P_m = \{\vec{p}_1, \vec{p}_2, \vec{p}_3, \ldots\ldots\ldots, \vec{p}_n\}$.

The hyperplane that goes through the centroid $\mu(C_m)$ of the data points and is perpendicular to the vector of greater variance, $\vec{v}_{\max}(C_m)$, partitions the cluster C_m into two sub-clusters denoted as C_{m_1} and C_{m_2} based on the sign of the dot product of the position vector and the vector of maximum variance (Rojas-Thomas et al., 2017) given in the Definition 4.

Definition 4. Let $\vec{p}_i$ and $\vec{v}_{\max}(C_m)$ be the position vector and vector of maximum variance receptively. The membership of $x_i \in C_m$ is defined as follows:

$$C_{m_1} = \{x_i \mid (x_i \in C_m) \wedge (\vec{p}_i.\vec{v}_{max}(C_m) \geq 0)\}$$
$$C_{m_2} = \{x_i \mid (x_i \in C_m) \wedge (\vec{p}_i.\vec{v}_{max}(C_m) < 0)\} \quad (4)$$

We iteratively choose one partition for splitting into two sub-clusters till the number of sub-clusters is equal to $\sqrt{N}$. The sub-cluster having maximum variance with its centroid is chosen as the cluster to be split (Zhong, Miao, Wang, & Zhou, 2008). If we split every cluster iteratively, though the method is efficient but it will not consider the effect of splitting on quality of the clusters. Again, if we select the largest cluster for splitting, then it will produce balance sub-clusters but it may ignore the scatter property of the data. The cluster having maximum variance with respect to its centroid considers the scatter property well, so we use maximum deviation of sub-cluster as a criteria to select the next sub-cluster for partitioning (Zhong et al., 2008). Suppose there are M sub-clusters at a certain step, say $C_0, C_1, \ldots\ldots, C_M$. One of the sub-clusters will be selected for the further partition based on the following criteria.

Let $CS = \{C_m : 0 \leq m \leq M-1\}$ be the set of sub-clusters.

$$C_{max} = \underset{C_m \in CS}{\operatorname{argmax}} \ (var(C_m)) \quad (5)$$

where $var(C_m) = \frac{1}{|C_m|} \sum_{x \in C_m} ||(x - \mu(C_m))||$, $||.||$ denotes the Euclidian norm. $C_{\max}$ is the sub-cluster to be split in next step. The dataset is partitioned into $\sqrt{N}$ sub-clusters and the above method is described in Algorithm 1.

Fig. 3 illustrates the steps to divide a cluster into two sub-clusters: In the first step we compute the maximum variance vector ($\vec{v}_{\max}(C_m)$) shown in (a); then in (b) the position vector ($\vec{p}_i$) of data points(black circle) with respect to the centroid of cluster(red circle) are calculated; in (c) we compute the perpendicular vector that passes through the centroid of the cluster and this vector partitions the cluster into two sub-clusters; and (d) illustrates that each sub-clusters has its own centroid for further partitioning. Fig. 4 demonstrates the partitioning approach in each iterations on full moon dataset.

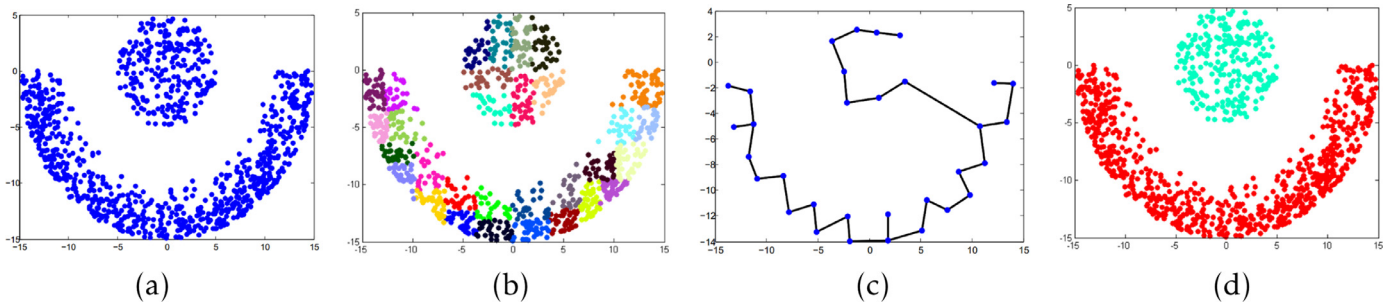

Fig. 2. Overview of proposed algorithm: (a) An example dataset(data points = 1000) with two clusters. (b) In divide stage, dataset partitioned into $\sqrt{N}$ number of sub-clusters. (c) The MST of sub-centroids of sub-clusters to find out the adjacent pairs. (d) In merge stage, the adjacent pairs are merged into actual clusters.

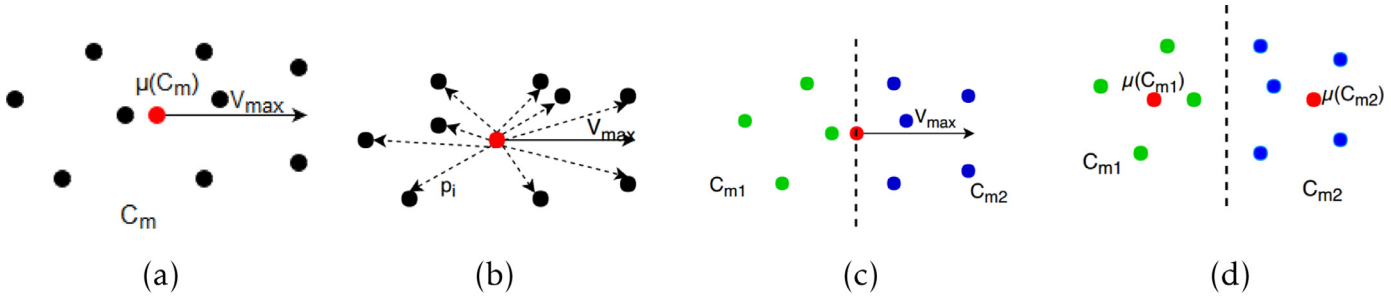

Fig. 3. Process of partitioning a dataset: (a) Computing of vector of maximum variance ($\vec{v}_{\max}(C_m)$). (b) Computing the position vector ($\vec{p}_i$) of data points with respect to the centroid of the cluster. (c) A hyperplane passing through the centroid for partitioning the data points into two parts. (d) Each sub-clusters has its own centroid(red circles) for further partitioning. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

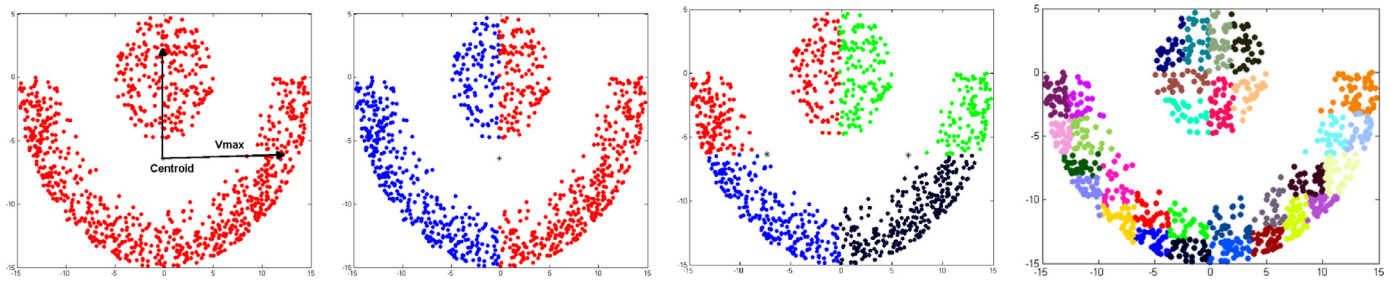

Fig. 4. Illustration of partitions of full moon dataset in each iteration.

Algorithm 1 Partition algorithm.

Input: Dataset X
Output: Set of sub-clusters $C = \{C_1, C_2, C_3, \ldots, C_{k'}\}$

1. Initially consider all data points in a cluster (C_0).
2. Add cluster (C_0) to C; $C = \{C_0\}$.
3. Let $k' = |C|$ // # of elements in C.
4. **while**($k' \leq \sqrt{(N)}$) **do**
   (a) Find the sub-cluster $c_{\max} \in C$ having maximum variance with its centroid for splitting using equation 5.
   (b) Compute the maximum variance vector of *Cmax* using equation 3.
   (c) Compute the position vector of each data points with respect to centroid of $C_{\max}$, i.e., $P_m = \{\vec{p}_1, \vec{p}_2, \vec{p}_3, \ldots\ldots\ldots, \vec{p}_n\}$.
   (d) Compute the dot product between position vector of each data points $\vec{p}_i$ and maximum variance vector $\vec{v}_{\max}$ to partition the sub-cluster $C_{\max}$ into two new sub-clusters C_{m_1} and C_{m_2} using equation 4.
   (e) Delete the sub-cluster $C_{\max}$ and add C_{m_1} and C_{m_2} in C, i.e., $C{=}C - \{C_{\max}\} \cup \{C_{m_1}, C_{m_2}\}$.
   (f) $k' = |C|$.
5. **end while.**
6. **return** C

2.2. Finding the adjacent pair

After dataset has been partitioned into ($k' = \sqrt{N}$) sub-clusters, the merging approach is applied to obtain the final clusters. In the merging approach, it is necessary to determine which pair of sub-clusters should be merged out of ($k'(k'-1)$)/2 possible pairs. In this paper, in order to avoid extra computation, MST of sub-clusters is used to easily determine adjacent pairs to filter out the farthest pairs for merging. A complete graph $G = (V, E)$ is constructed taking centroids of the sub-clusters as vertices and the distance between them as edges. Then the MST of the graph G is computed. Two sub-clusters are called an adjacent pair if their centroids are connected by an MST edge. This step is illustrated in Fig. 5.

Definition 5. Any pair of sub-clusters (C_i, C_j) is called an adjacent pair if there exist an MST edge between them.

2.3. Merging of two sub-clusters

A sub-cluster may have more than one adjacent pairs and all of them may not belong to the same cluster. For selecting the best pair, the merging approach is applied. Many merging approaches have been proposed in the literature to merge the sub-clusters into actual clusters (Guha et al., 2001; Karypis et al., 1999; Zhong et al., 2011). But most of the merging approaches either depend on some user defined parameters or they are not very efficient in terms of quality or computational time. In CURE, number of representative points are used to represent each sub-cluster and closest distance between the representative points in different sub-clusters determines the cluster similarity (Guha et al., 2001). Quality of clustering results is affected by the selection of representative points, since representative points dependent upon shapes and sizes of clusters and outliers in the dataset (Huang et al., 2014; Karypis et al., 1999). In Chameleon, cluster similarity is computed based on how well-connected data points are within a cluster and on the proximity of clusters (Karypis et al., 1999). That is, two clusters are merged if their relative inter connectivity and closeness are high. Clustering results of CHAMELEON is depended on user defined parameters (Cheng et al., 2016; Zhong et al., 2011). Merging process of SAM is based on inter-connectivity and intra-similarity of sub-clusters (Zhong et al., 2011). However, it requires the number of clusters as a input parameter (Cheng et al., 2016). Based on the issues in existing merging approaches, we propose an efficient novel merge index to rank a pair for merging.

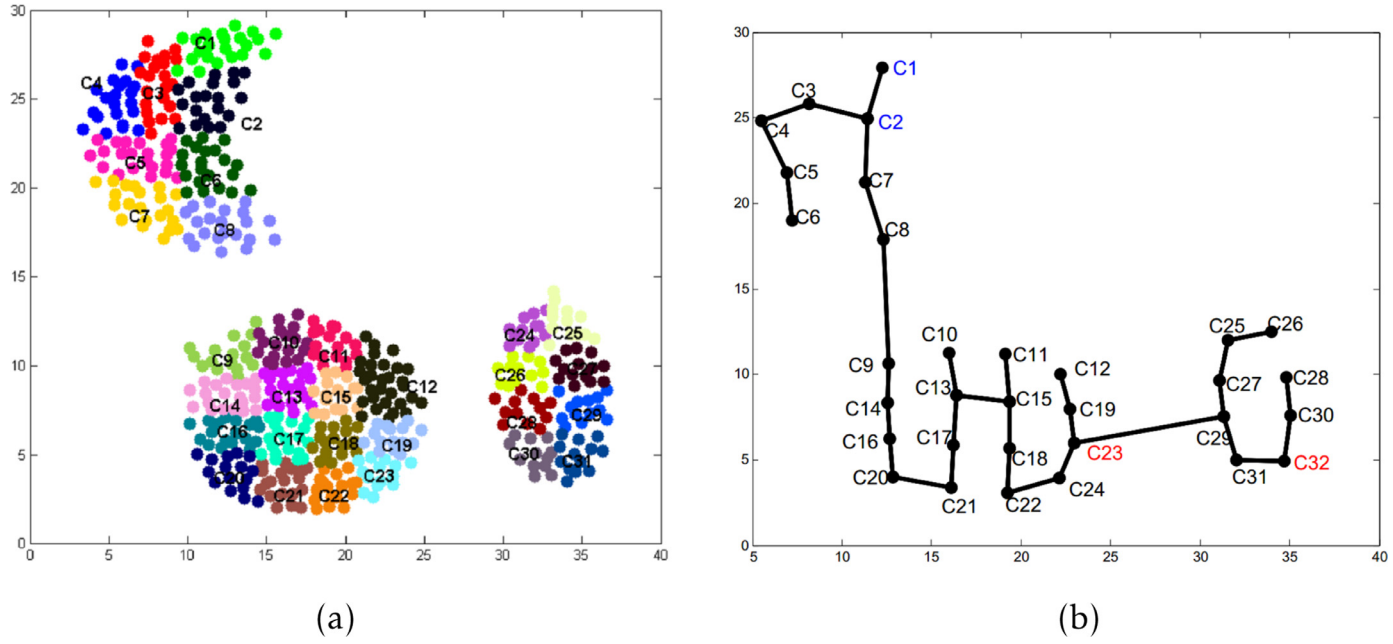

Fig. 5. Process of identifying the adjacent pairs: (a) Computing the centroid of each sub-clusters *C*. (b) Computing the MST graph on these centroids (only connected pair is considered as an adjacent pair).

It consists of two parameters such as cohesion and intra-similarity. These parameters are defined as follows:

2.3.1. Cohesion of a pair of clusters

In this section, we propose a new technique to measure the inter-connectivity of an adjacent pair based on cohesion which is intrinsically different from the existing approaches (Karypis et al., 1999; Lin & Chen, 2002; Rojas-Thomas et al., 2017; Zhong et al., 2011). Cohesion measures the similarity or dissimilarity of two sub-clusters based on the density and distribution of data points in the intermediate region of the two centroids. The more populated the intermediate region is, the more cohesive the sub-clusters will be. The level of cohesion between the sub-clusters is measured by the level of scattering of data points with respect to the middle of the MST edges. The cohesion is defined as follows:

$$\begin{aligned} cohesion(C_i, C_j) = \frac{\sum_x d(x, \mu_{ij})}{n+m} \text{ such that} \\ x \in (C_i \cup C_j) \wedge d(x, \mu_{ij}) < \min(d(\mu_{ij}, \mu(C_i)), \\ d(\mu_{ij}, \mu(C_j))) \text{ and } \mu_{ij} = (\mu(C_i) + \mu(C_j))/2, \\ n = |C_i|,\ m = |C_j| \end{aligned} \tag{6}$$

For example, in Fig. 6, let $\mu(C_1)$ and $\mu(C_2)$ are the centroids of the sub-clusters C_1 and C_2 respectively and μ_{12} is their mid point. The data points (*f, g, h, i, j, k, l, m*) are only considered as the boundary data points because the distance between the mid point and these points is less as compared to the distance of centroids and mid point. If the intermediate region is dense then the adjacent pair has more chance of merging together.

2.3.2. Intra similarity of a pair of clusters

In this section, the intra similarity is defined based on the similarity of data points within a cluster. A few existing approaches compute the similarity index of a cluster based on a single pair of points which makes them less tolerant to outliers and noise (Guha, Rastogi, & Shim, 1998; Zhong et al., 2011). Proposed approach measures the closeness of the two clusters by computing the average similarity of data points in the clusters. The average strength of the MST edges within a cluster can provide a good indication of

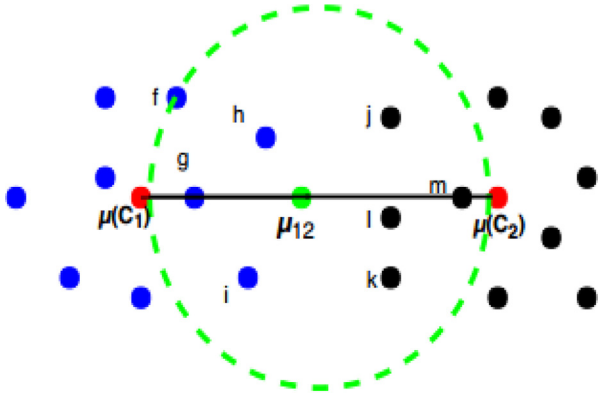

Fig. 6. Illustration of the boundary data points to compute the cohesion.

similarity of data points within a cluster. Given a cluster *C* the MST of all the points of *C* is computed and let us assume that $MST = (V_i, E_i)$. The proposed method computes the similarity index of *C* using Gaussian function on MST edges inspired by existing approaches (Barany & Vu, 2007; Das & Sil, 2007; Guha et al., 1998; Mishra & Mohanty, 2018; Rodriguez & Laio, 2014). The similarity index of a cluster is defined as follows:

Definition 6 (Similarity index)**.** Let *C* is a cluster, D_C is the average weight of MST edges within *C*, then the similarity index of the cluster *C* denoted as *S*(*C*) is defined as follows:

$$\begin{aligned} S(C) = 1/\sqrt{2\pi} \cdot (\exp(-D_C^2/2)) \\ where\ D_C = \frac{1}{|E_i|} \sum_{e_i \in E_i} |e_i| \end{aligned} \tag{7}$$

where $|e_i|$ is weight of the MST edge e_i in *C*.

Similarity index of a sub-cluster is used to measure the quality of the sub-clusters representing how closely the data points are related with each other within the sub-clusters. For computing the intra-similarity of an adjacent pair the similarity indices of both the clusters are used using the technique of Karypis et al. (1999),

Zhong et al. (2011), Das and Sil (2007) and Mishra and Mohanty (2018).

Definition 7 (Intra-Similarity (Zhong et al., 2011)). Let (C_i, C_j) is an adjacent pair, the intra-similarity(IS) of the pair $IS(C_i, C_j)$ is defined as:

$$IS(C_i, C_j) = \frac{1}{(|C_i| \cdot |C_j|)} * \frac{\sqrt{(S(C_i) \cdot S(C_j))}}{(S(C_i) + S(C_j))} \quad (8)$$

Intra-similarity between pair (C_i, C_j) is high when two average edge weights are close to each other. The small sizes sub-clusters have more chance to be merged because size of sub-clusters also contributes in computing the intra-similarity.

Definition 8 (Merge Index). Let (C_i, C_j) is an adjacent pair, the merge index for selecting the best pair is defined as:

$$MI(C_i, C_j) = cohesion(C_i, C_j) \cdot IS(C_i, C_j) \quad (9)$$

Merge index considers both cohesion and intra-similarity. A adjacent pair with minimum *MI* value is merged in each iteration and the next pair is considered based on the new *MI* value. The iteration stops when either k partitions or no improvement in the cluster quality index is achieved. The proposed merging approach is described in Algorithm 2.

Algorithm 2 Merging algorithm.

Input: Sub-clusters $C = \{C_1, C_2, \ldots, C_{k'}\}$, k = number of clusters.
Output: Final k clusters.

1. Compute the set of centroids $\mu(C) = \{\mu(C_i) : 1 \le i \le k'\}$ of each sub-clusters.
2. Construct a complete graph $G = (V, E)$ where $V = \mu(C)$ and $E = \{d(\mu(C_i), \mu(C_j)) | 1 \le i < j \le k'\}$ // V is the set of centroids and E is the distance between each pair of centroids.
3. Compute the minimum spanning tree (MST) of the complete graph G.
4. Compute adjacent pairs (C_i, C_j) if $\mu(C_i)$ and $\mu(C_j)$ are connected by an edge in the MST.
5. Compute the merge index $MI(C_i, C_j)$ of all the adjacent pairs using equation 9.
6. Let $k' = |C|$ // # of elements in C.
7. **while**$(k' > k)$ **do**
   (a) Find the adjacent pair (C_i, C_j) with minimum $MI(C_i, C_j)$ value.
   (b) Merge C_i and C_j into a new sub-cluster, i.e., $C_{new} = C_i \cup C_j$.
   (c) Delete C_i and C_j and add C_{new} into C, i.e., $C{=}C \setminus \{C_i, C_j\} \cup C_{new}$.
   (d) Update the adjacent pairs of C_{new}, i.e, (C_{new}, C_l) becomes an adjacent pair if C_l is an adjacent pair of either C_i or C_j.
   (e) Compute the merge index of all the adjacent pairs of C_{new}, i.e., compute $MI(C_{new}, C_l), \forall C_l$ where C_l is adjacent to C_{new}.
   (f) $k' = |C|$
8. **end while**
9. **return** C

2.4. Computational complexity

The overall computational complexity of the proposed algorithm depends on the amount of time it requires to partition the dataset into sub-clusters, find the adjacent pairs and merge. Time required for partitioning the dataset depends on the number of data points in the cluster. In the worst case, the amount of time required to compute the variance and the membership value takes $O(N)$ time. Finding the maximum variance direction vector and repeatedly splitting the dataset into sub-clusters depend on the number of sub-clusters. The number of iterations required to split the dataset into $O(\sqrt{N})$ sub-clusters is $O(\sqrt{N})$. Therefore, if the partition is balance the algorithm takes less time which is $O(N)$ only, however if the algorithm returns highly imbalance partitions in each step then the worst case complexity for partitioning is $O(N^{3/2})$.

Next, the time required to find out the adjacent pairs depends on time require for computing the MST of $\sqrt{N}$ vertices which is $O(N)$. Finally, the time required for merging the $\sqrt{N}$ sub-clusters depends on the complexity to compute the cohesion and intra-similarity for each pair of sub-clusters and the amount of time needed for selecting the most similar pair of sub-clusters to merge. The complexity of computing the cohesion and intra-similarity of a pair of sub-clusters is $O(N)$. In merging step, the worst case complexity is obtained when the merging algorithm repeatedly chooses the same sub-cluster and merge it with another. Thus, it requires $O(\sqrt{N} - k)$ iteration to find out the k clusters. Therefore, the time required to merge the $(\sqrt{N})$ intermediate sub-clusters is $O(N(\sqrt{N} - k))$. Hence the overall complexity of the proposed clustering technique is $O(N^{3/2} + N + N(\sqrt{N} - k)) = O(N^{3/2})$.

Table 1
Synthetic datasets description: number of data points(N), dimension of dataset(d), and number of clusters(k).

Datasets	N	d	k
DS1: Full moon	1000	2	2
DS2: Cluster inside cluster	713	2	2
DS3: Varying densities	150	2	3
DS4: Well separate	545	2	3
DS5: Touching cluster	83	2	2
DS6: Flame dataset	240	2	2
DS7: Mixed dataset(9 clusters)	10,000	2	9
DS8: Mixed dataset(8 clusters)	8000	2	8

3. Experimental results and discussion

In this section, we discuss the performance of the proposed algorithm on synthetic, gene expression and real datasets with respect to six other competing algorithms. Our implementations are performed using MATLAB R2013a on a Desktop running Windows 64bit with Intel(R) i7 3.4 GHz processors and 8GB of RAM.

3.1. Dataset

3.1.1. Synthetic datasets

Proposed algorithm is evaluated on different synthetic datasets having very specific and different characteristics. The first six datasets are taken from Zhong et al. (2011), Cheng et al. (2016), Pasi and et al. (2015) and the next two (DS7 and DS8) are taken from Karypis et al. (1999) and Hyde and et al. (2014). The datasets correspond to characteristic of clusters that differ in terms of the density, dispersion and geometry as shown in Fig. 7. The parameters of the DBSCAN algorithm for DS1, DS2, DS3, DS4 (three clusters are considered only), DS5, DS6, DS7, and DS8 datasets are set to $(MinPts = 1, Eps = 2)$, $(MinPts = 1, Eps = 2)$, $(MinPts = 1, Eps = 2)$, $(MinPts = 1, Eps = 2)$, $(MinPts = 1, Eps = 2)$, $(MinPts = 7, Eps = 0.8)$, $(MinPts = 4, Eps = 6.2)$ and $(MinPts = 4, Eps = 5.9)$ respectively through experiments. The detailed description about these datasets is given in Table 1.

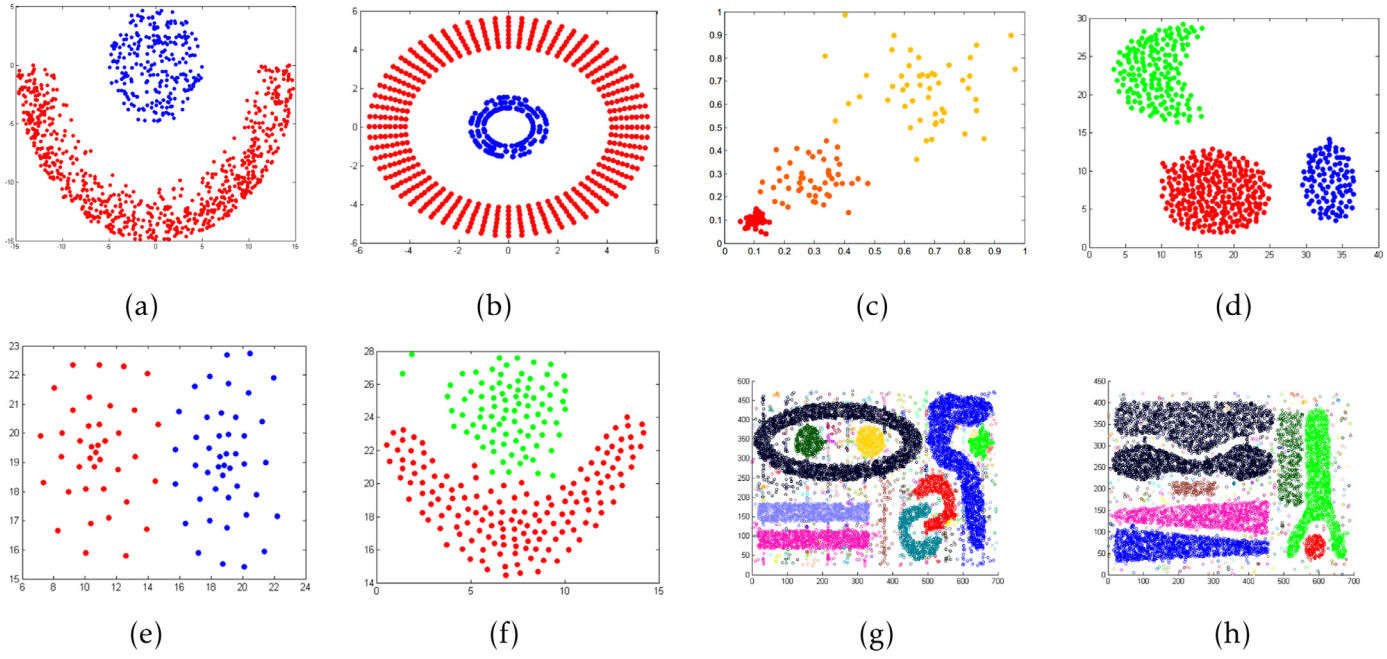

Fig. 7. Datasets description: (a) Full moon (b) Nested ring (c) Varying densities (d)Well separated (e) Neck shaped (f) Flame (g)Mixed dataset with nine clusters (h) Mixed dataset with eight clusters.

Table 2
Gene expression datasets description: number of data points(N), dimension (d), number of clusters(k).

Datasets	N	d	k
DR1: BreastB	49	1213	4
DR2: DLBCLA	141	661	3
DR3: Novartis	103	1000	4
DR4: Lukemia	248	985	6
DR5: LungA	197	1000	4
DR6: Yeast1	384	17	5
DR7: Lung cancer	32	56	3

Table 3
Real datasets description: number of data points(N), dimension (d), number of clusters(k).

Datasets	N	d	k
DUR1: Iris	150	4	3
DUR2: Wine	178	13	3
DUR3: Glass	214	9	7
DUR4: Tumor	339	17	22
DUR5: Bridge	4096	16	256
DUR6: Missa	6480	16	256
DUR7: Libras movement	360	91	15
DUR8: Buddy move	249	7	25

3.1.2. Gene expression datasets

Furthermore, the proposed algorithm is also applied for analyzing the gene expression datasets. The clustering results of the proposed algorithm is evaluated on seven gene expression datasets; the details of these datasets are given in Table 2. The BreastB, DLBCLA, Novartis, Lukemia and LungA are available at Broad Institute (2018). The yeast1 is downloaded from Yeung and et al. (2000) and lung cancer dataset is taken from Blake and Merz (1998).

3.1.3. Real datasets

Furthermore, the proposed algorithm is also applied for analyzing eight real datasets taken from the UCI Machine Learning Repository(UCI) (Blake & Merz, 1998) and Pasi and et al. (2015). The details of these datasets are given in Table 3. The iris, wine, glass, tumor, libras movements, buddy move are given at Blake and Merz (1998) and Bridge and Missa are available at Pasi and et al. (2015).

3.2. Result on synthetic datasets

DS1: This dataset contains a half ring and spherical cluster. The clustering results are illustrated in the Fig. 8. Proposed algorithm, DBSCAN, DMST, SAM and GDD identify the proper clusters. k-means and hierarchical(Average linkage) fail on this. Since, k-means favors spherical clusters, it fails on DS1. Average linkage produces the unsuitable result because it measure the distance between two clusters as the average distance between the pairs of data points in the two clusters. Although DBSCAN produces the proper result, it is challenging to tune the exact parameters to achieve the proper result. DMST and GDD identify proper clusters because it contains well seprated clusters of diverse shape.

DS2: The dataset consists of two ring clusters in which one is inside the other. Fig. 9 demonstrates the clustering results. All algorithms except k-means and hierarchical produce the proper partition because the mean of both the clusters located in the center of the inner ring.

DS3: This dataset contains three clusters with different densities. The clustering result is shown in Fig. 10. The proposed algorithm and SAM identify the proper clusters. k-means, average linkage, DBSCAN, DMST and GDD fail to identify the actual clusters. Since the dataset contains the data points with varying densities, therefore it is difficult to select the right combination of parameters for the DBSCAN. DMST removes the longest edges in early phase of clustering therefore, it does not identify proper clusters with varying densities.

DS4: This dataset contains three clusters and clusters are well separated to each other. The clustering results are illustrated in Fig. 11. All clustering methods produce the proper partitions.

DS5: This dataset contains two clusters close to each other like a neck. Fig. 12 demonstrates the clustering results. Proposed

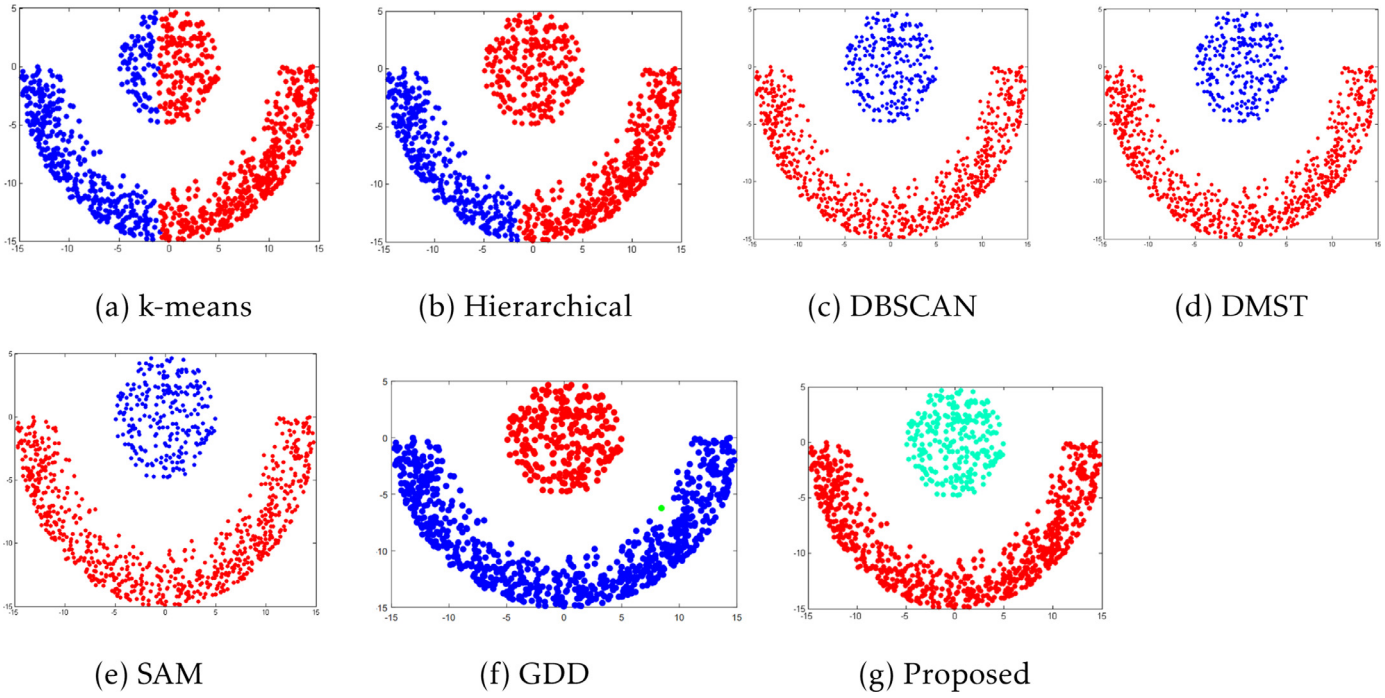

Fig. 8. Illustration of clustering results on Full moon with $k = 2$.

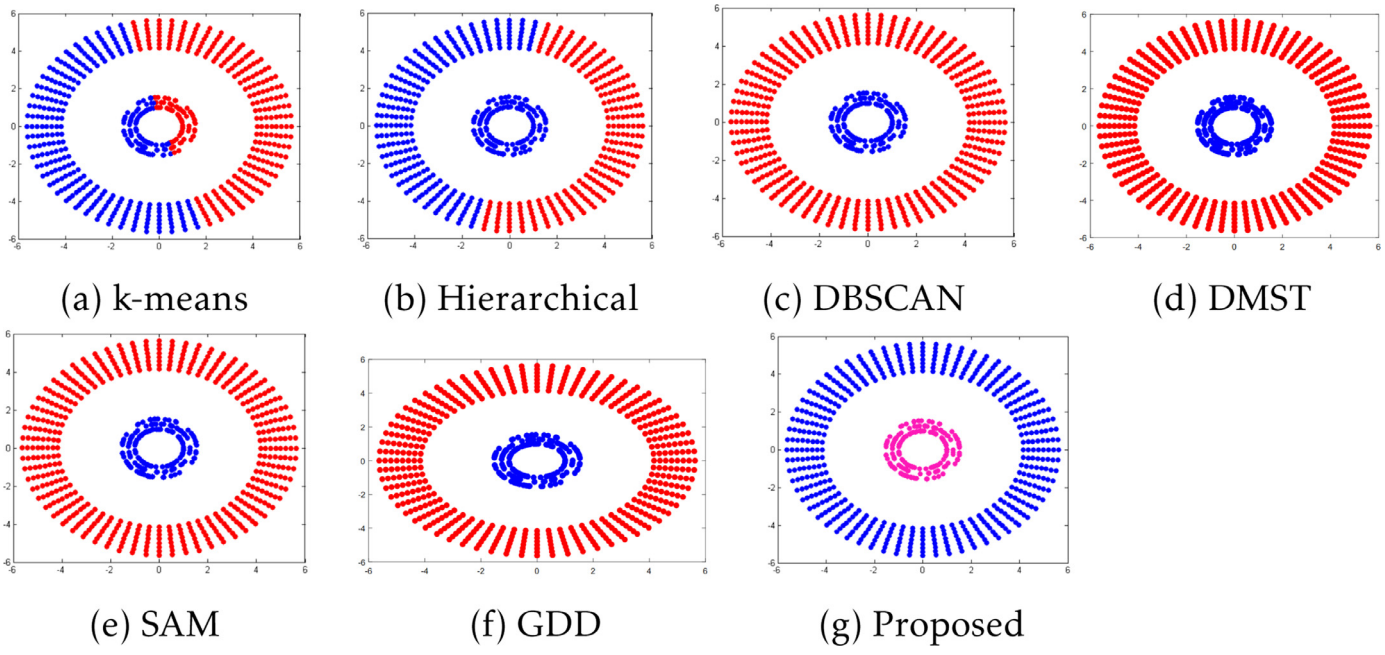

Fig. 9. Illustration of clustering results on cluster inside cluster dataset with $k = 2$.

algorithm as well as k-means identify the clusters properly as it contains spherical shape clusters whereas remaining methods do not identify the clusters.

DS6: The dataset in which one cluster has higher dispersion level than the other cluster. The result of the clustering algorithm is illustrated in Fig. 13. The proposed algorithm and SAM only identify the expected clusters. DBSCAN, k-means, DMST, GDD and hierarchical fail to detect actual clusters because it contains diverse shape clusters having different dispersion level.

DS7: The clustering results of this dataset is illustrated in Fig. 14. Proposed algorithm produces satisfactory clustering results on this dataset, whereas k-means, hierarchical, DBSCAN, DMST, SAM and GDD produce improper results. SAM fails to detect the exact clusters on this dataset because the data points are very close to each other and contain vertical streaks with varying densities which increase the connection span between the clusters. Therefore, SAM keeps such sub-clusters together producing an improper result. DMST and GDD produce improper clusters because the dataset contains clusters having complex pattern with varying densities.

DS8: The dataset DS8 has eight clusters of diverse shapes, sizes, densities and it also contains random noise. The important feature of this data is that the clusters are very close to each other with different densities. The clustering results are demonstrated

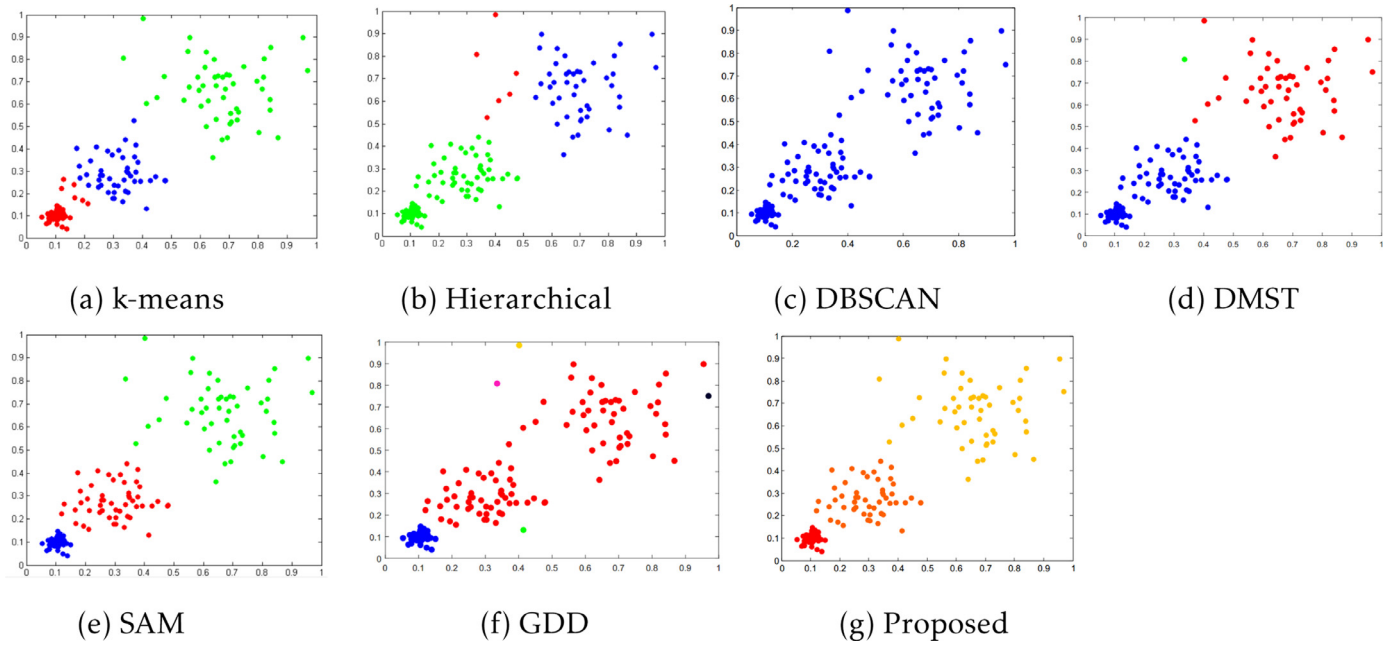

Fig. 10. Illustration of clustering results on varying density dataset with $k = 3$.

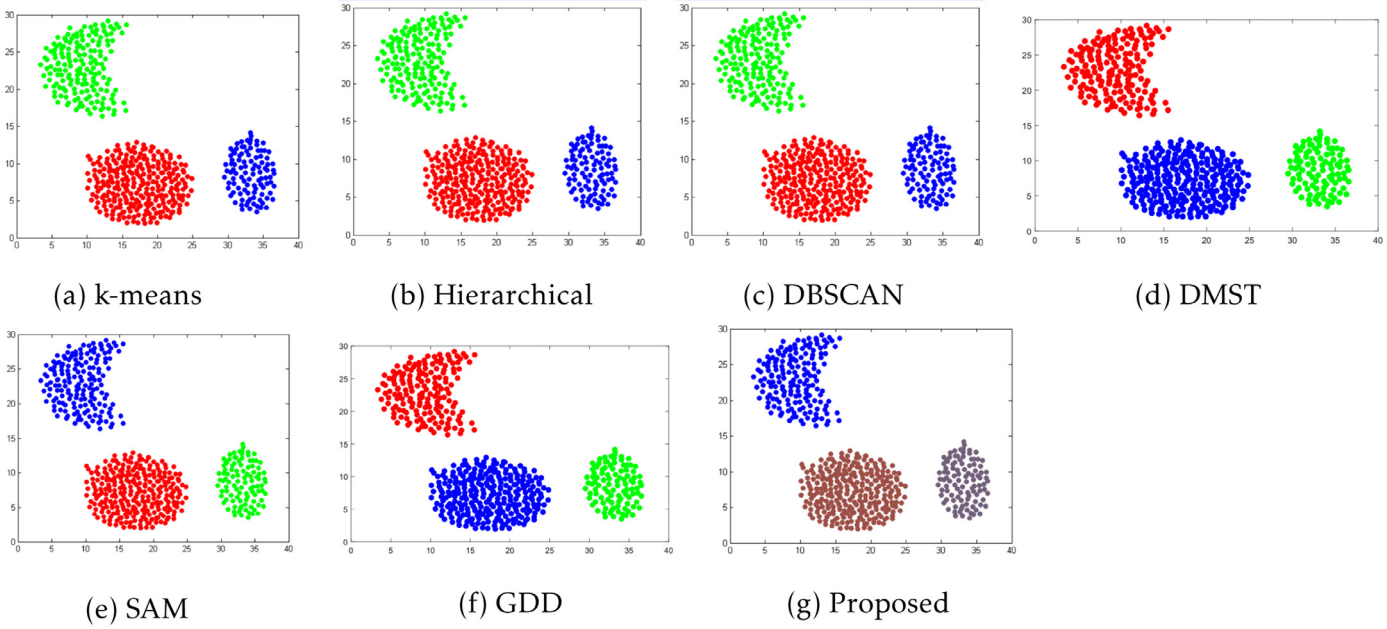

Fig. 11. Illustration of clustering results on well separated dataset with $k = 3$.

in Fig. 15. Proposed algorithm produces satisfactory clustering results on this dataset, whereas remaining competing clustering algorithms produce improper results.

3.3. Cluster quality analysis

We also validate the clustering results using quality indices which measure the degree of agreement between a predicted clustering result and the ground truth partitions. There are two methods for clustering validation: internal and external evaluation measures. Internal validity indices evaluate the clustering performance based on some properties of clusters such as their compactness and degree of separation, whereas, external validity indices evaluate the clustering performance based on the predicted class label by target class label. We evaluate the performance using external quality measures such as Rand Index(RI), Adjusted Rand Index(ARI), Purity and Normalized Mutual Information(NMI) (Bezdek & Pal, 1998; Cui, Xie, Cai, Huang, & Liu, 2014; Kashef & Kamel, 2009; Pluim, Maintz, & Viergever, 2003; Rand, 1971; Rojas-Thomas et al., 2017).

Let $P = \{P_1, P_2, P_3, \ldots, P_L\}$ is the set of target partition and $C = \{C_1, C_2, \ldots\ldots, C_R\}$ is a set of partition produced by any clustering algorithm. Let P_{11} be the number of pairs that are clustered in both P and C. Let P_{00} be the number of pairs that are in different

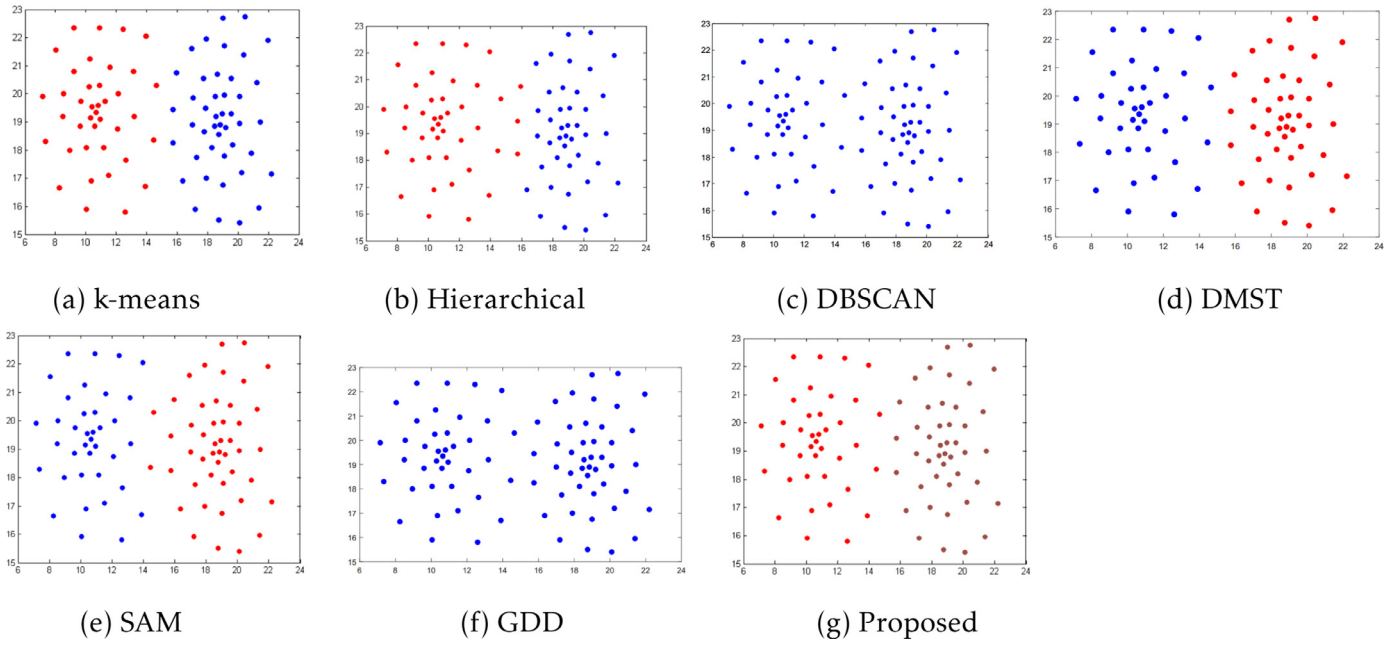

Fig. 12. Illustration of clustering results on neck type dataset with $k = 2$.

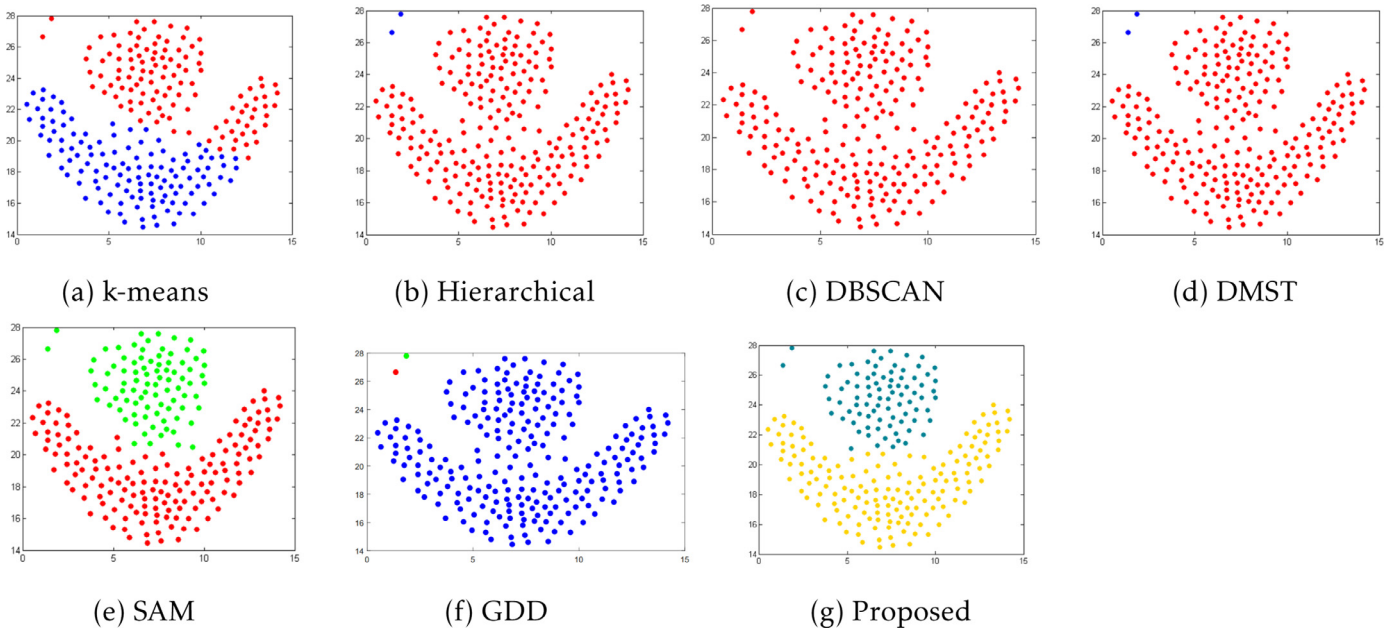

Fig. 13. Illustration of clustering results on flame dataset with $k = 2$.

clusters in both P and C. Let P_{10} be the number of pairs that are same in cluster P and different in clusters C. Let P_{01} be the number of pairs that are different in clusters P and same in cluster C. RI, ARI, Purity, and NMI are defined in Eqs. (10)–(13) respectively.

$$RI = \frac{(P_{00} + P_{11})}{(P_{00} + P_{01} + P_{10} + P_{11})} \tag{10}$$

Rand index measures the percentage of decisions that are correct over the total pairs (Rand, 1971). Rand index value of 1 indicates proper clustering whereas 0 indicates improper clustering.

$$ARI = \frac{2 \cdot (P_{11} \cdot P_{00} - P_{10} \cdot P_{01})}{(P_{11} + P_{10}) \cdot (P_{10} + P_{01}) + (P_{11} + P_{01}) \cdot (P_{01} P_{00})} \tag{11}$$

ARI is the corrected for a chance version of the Rand Index (Rand, 1971). The Adjusted Rand index takes a value between -1 and 1, where -1 indicates completely disagree on any pair of points, the value 1 indicates that ground truth partition and actual clusters are exactly same.

$$Purity(P, C) = \frac{1}{N} \sum_{r \in R} \underset{l \in L}{\operatorname{argmax}} |C_r \cap P_l| \tag{12}$$

Purity is the percentage of total number of truly classified data points over the sample size (Kashef & Kamel, 2009). Purity takes value between 0 and 1, where 0 indicates improper and 1 indicates proper clustering respectively.

Normalized mutual information (NMI) is defined as the measure of diversity among different clusters which is given as follows

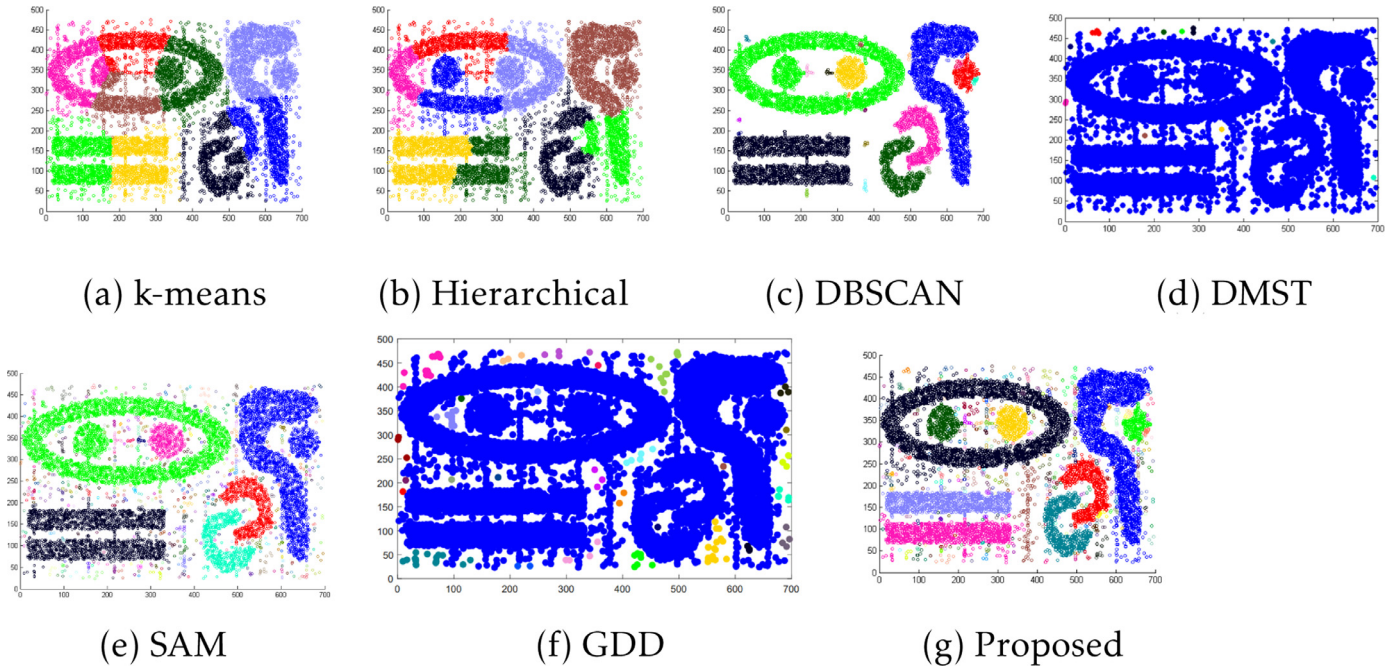

Fig. 14. Illustration of clustering results on mixed dataset with $k = 9$.

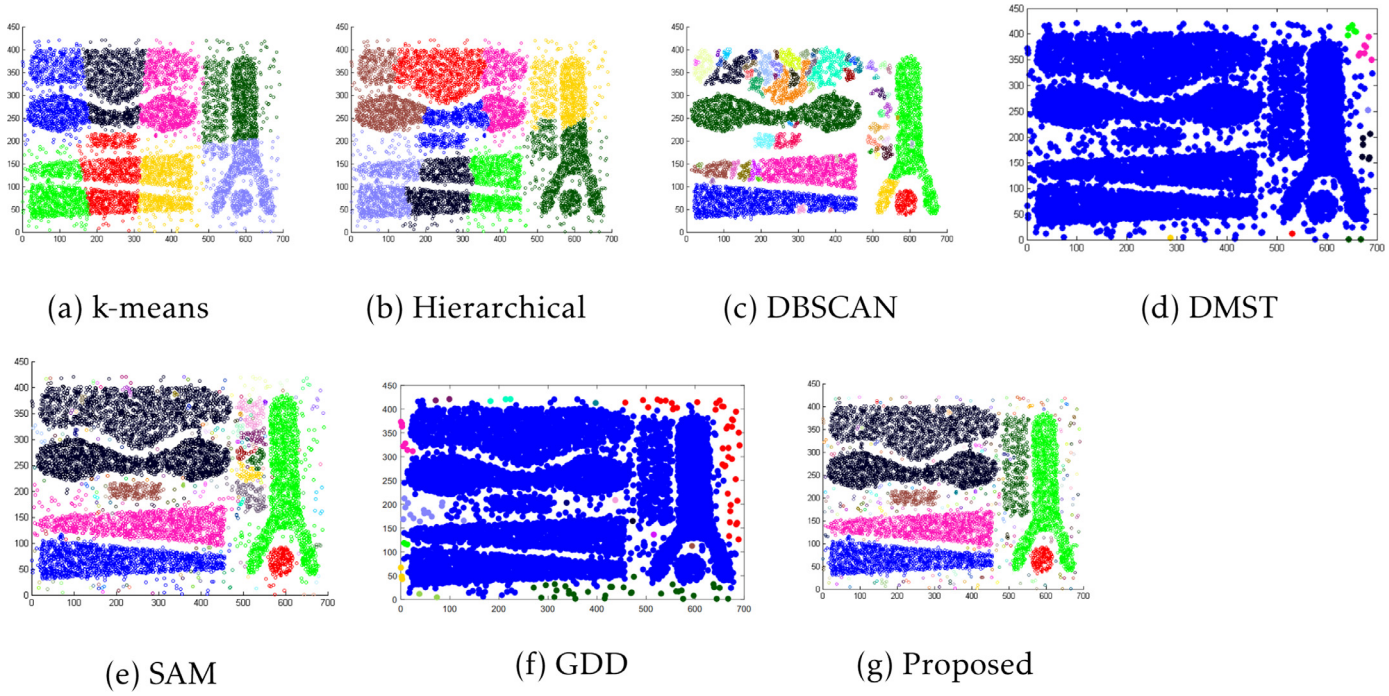

Fig. 15. Illustration of clustering results of mixed dataset with $k = 8$.

(Kashef & Kamel, 2009):

$$NMI(P, C) = \frac{2 * I(P; C)}{[H(P) + H(C)]} \tag{13}$$

Here $I(P; C)$ denotes the mutual information between partition P and C, $H(P)$ and $H(C)$ are the entropy of target class P and partition C produced by clustering algorithm respectively. A perfect clustering has a NMI of 1 and bad clusterings have values close to 0. The ARI, NMI, and Purity values of the various algorithms on synthetic datasets are shown in Tables 4–6 respectively. It can be observed from the results that proposed algorithm outperforms other competing clustering methods on all these datasets.

3.4. Result on gene expression datasets

The parameters of the DBSCAN algorithm for BreastB, DLBCLA, Novartis, Lukemia, LungA, Yeast1 and Lung cancer datasets are set to $(MinPts = 5, Eps = 0.5230)$, $(MinPts = 4, Eps = 0.3135)$, $(MinPts = 5, Eps = 0.1810)$, $(MinPts = 3, Eps = 0.2522)$, $(MinPts = 5, Eps = 0.3961)$, $(MinPts = 4, Eps = 0.3302)$, and $(MinPts = 4, Eps = 5)$ respectively through experiments.

Table 4
Comparison of algorithms according to ARI on synthetic datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DS1	−0.0022	0.0217	**1**	**1**	**1**	0.9968	**1**
DS2	−0.0016	0.0643	**1**	**1**	**1**	**1**	**1**
DS3	0.8671	0.5149	0	0.4674	**1**	0.5586	**1**
DS4	**1**	**1**	**1**	**1**	**1**	**1**	**1**
DS5	1	0.8143	0	**1**	0.9471	0	**1**
DS6	0.4997	0.0128	0	0.0128	0.9092	0.0127	**0.9176**
DS7	0.3586	0.3426	0.7240	−0.000014	0.7335	−0.00048	**0.9063**

Table 5
Comparison of algorithms according to NMI on synthetic datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DS1	0.0115	0.2170	**1**	**1**	**1**	0.9933	**1**
DS2	0.00006	0.2489	**1**	**1**	**1**	**1**	**1**
DS3	0.8568	0.6971	0	0.6583	**1**	0.7053	**1**
DS4	**1**	**1**	**1**	**1**	**1**	**1**	**1**
DS5	**0.9176**	0.7673	0	**0.9176**	0.8346	0	**0.9176**
DS6	0.4622	0.0479	0	0.0497	0.8122	0.0453	**0.8519**
DS7	0.4708	0.7494	0.6936	0.0136	0.7162	0.0196	**0.9153**

Table 6
Comparison of algorithms according to purity on synthetic datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DS1	0.7500	0.7500	**1**	**1**	**1**	**1**	**1**
DS2	0.7181	0.7181	**1**	**1**	**1**	**1**	**1**
DS3	0.9533	0.6667	0.3333	0.6667	**1**	0.6867	**1**
DS4	**1**	**1**	**1**	**1**	**1**	**1**	**1**
DS5	**0.9880**	0.9518	0.5422	0.9880	0.9633	0.5422	**0.9880**
DS6	0.8642	0.6458	0.6375	0.6458	0.9215	0.6458	**0.9792**
DS7	0.4640	0.6820	0.6040	0.3045	0.6348	0.3056	**0.9568**

Table 7
Comparison of algorithms according RI on gene expression datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DR1	0.5901	0.3605	0.5816	0.5340	0.3605	0.6338	**0.7015**
DR2	0.5982	0.3424	0.6794	0.6184	0.3424	0.6695	**0.7442**
DR3	0.7441	0.6383	0.7434	0.6637	0.6244	0.7563	**0.8091**
DR4	**0.8752**	0.5075	0.3305	0.8522	0.2408	0.7830	0.8302
DR5	0.6312	0.6958	0.5482	**0.6995**	0.5365	0.4754	0.5409
DR6	0.5827	0.2492	0.5021	0.5861	0.2419	0.7710	**0.8219**
DR7	0.6492	0.5887	0.5060	0.4194	0.3589	0.6794	**0.7944**

Table 8
Comparison of algorithms according ARI on gene expression datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	Proposed
DR1	0.0892	0.0565	0.0333	0.0270	0.0565	**0.1392**
DR2	0.1279	0.0034	0.0962	0.2455	0.0034	**0.2860**
DR3	0.3509	0.3290	0.3192	0.3689	0.3184	**0.3689**
DR4	**0.6215**	0.1922	0.0325	0.5658	0.0015	0.3538
DR5	0.1360	0.3967	0.1270	**0.4014**	0.0282	0.1189
DR6	0.0294	−0.0008	0.0451	0.1630	−0.0015	**0.3166**
DR7	0.1920	0.1690	0.0234	0.0261	−0.0092	**0.4471**

The performance of seven gene expression datasets is evaluated using the clustering validity indices: Rand index, Adjusted Rand index. In Tables 7 and 8, the results based on RI and ARI measures in gene expression datasets illustrate that the proposed algorithm outperforms the k-means, hierarchical, SAM except Lukemia and LungA datasets. k-means and DMST get the best result in Lukemia and LungA datasets respectively. By repeatedly setting the parameters, DBSCAN achieves better result than the hierarchical and SAM, especially in BreastB, DLBCLA, Novartis and Yeast1 datasets. But it is very hard to set the parameters manually. The performance of hierarchical and SAM are relatively stable but the score of RI and ARI are generally low. Similarly, the ARI values of GDD is very low as compared to all other methods on all the datasets. That's why we prefer to not include the values in the table. In all these datasets the proposed algorithm has a satisfying performance over other competing clustering algorithms.

3.5. Result on real datasets

The parameters of the DBSCAN algorithm for iris, wine, glass, tumor, bridge, missa, libras, and buddy move datasets are set to $(MinPts = 10, Eps = 0.1)$, $(MinPts = 30, Eps = 0.1)$, $(MinPts = 6, Eps = 0.1)$, $(MinPts = 3, Eps = 0.2)$, $(MinPts = 40, Eps = 0.1)$, $(MinPts = 2, Eps = 0.1)$, $(MinPts = 20, Eps = 0.1)$,

Table 9
Comparison of algorithms according to RI on real datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DUR1	0.6667	**0.9067**	0.3289	0.8018	0.7766	0.7719	0.8247
DUR2	0.7187	0.6262	0.6504	0.5414	0.7062	0.6776	**0.7495**
DUR3	0.8278	**0.8485**	0.7319	0.7857	0.7486	0.2598	0.7492
DUR4	0.8511	0.6451	0.1680	0.8262	0.2230	0.8916	**0.9129**
DUR5	0.9866	0.9072	0.4646	0.9152	0.6053	0.2803	**0.9868**
DUR6	0.9146	0.4158	0.1284	0.8406	0.7154	0.9201	**0.9274**
DUR7	0.8090	0.8860	0.0641	0.8616	0.1867	0.9364	**0.9522**
DUR8	0.8129	0.8901	0.7739	0.8318	0.6362	0.6868	**0.9288**

Table 10
Comparison of algorithms according to ARI on real datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DUR1	0.4197	**0.7592**	0	0.5435	0.5638	0.5384	0.6129
DUR2	0.3711	0.2926	0.2743	0.1493	0.4001	0.3431	**0.4675**
DUR3	0.4730	**0.5389**	0.4077	0.4725	0.4181	0	0.4650
DUR4	0.0932	0.0136	0	0.2176	-0.0156	0.0192	**0.5098**
DUR5	0.0164	0.0175	0.0002	0.0638	0.0861	0.00016	**0.4161**
DUR6	0.0065	0.0360	0.0049	0.1367	0.1227	0	**0.1687**
DUR7	0.3051	0.2877	0	0.2898	0.0023	0.0139	**0.5356**
DUR8	0.1524	0.1824	0.1322	0.1963	0.0576	0.0451	**0.2603**

Table 11
Comparison of algorithms according to NMI on real datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DUR1	0.5921	**0.8058**	0	0.6300	0.7355	0.7424	0.6772
DUR2	0.4288	0.4158	0.3621	0.3286	0.3968	0.4051	**0.6165**
DUR3	0.7027	**0.7645**	0.6039	0.6797	0.5981	0	0.6684
DUR4	0.3631	0.3122	0	0.5761	0.1483	0.6251	**0.6511**
DUR5	0.4620	0.3907	0.0471	0.7098	0.4803	0.0442	**0.8404**
DUR6	0.1698	0.2156	0.0484	0.5608	0.1629	0.5629	**0.7298**
DUR7	0.5893	0.5998	0	0.6516	0.1711	0.6853	**0.7948**
DUR8	0.4922	0.4942	0.4862	0.5016	0.3219	0.2262	**0.5555**

Table 12
Comparison of algorithms according to Purity on real datasets.

Datasets	k-means	Hierarchical	DBSCAN	DMST	SAM	GDD	Proposed
DUR1	0.6667	**0.9067**	0.3333	0.8200	0.6800	0.6667	0.8333
DUR2	0.6685	0.6461	0.7022	0.6236	0.6825	0.6798	**0.7022**
DUR3	0.8598	**0.8879**	0.7383	0.7383	0.6729	0.3551	0.7430
DUR4	0.4336	0.3599	0.2478	0.5732	0.2832	0.5850	**0.6401**
DUR5	0.1157	0.0930	0.0168	0.3321	0.1730	0.0205	**0.3855**
DUR6	0.1366	0.1903	0.1380	0.4601	0.1813	**1**	0.9240
DUR7	0.4611	0.4639	0.0667	0.5389	0.1250	**1**	0.6306
DUR8	0.4506	0.1968	0.4900	0.4538	0.2972	0.2490	**0.4900**

and $(MinPts = 20, Eps = 0.1)$ respectively through experiments. The performance of eight real datasets is evaluated using the clustering validity indices: Rand index, Adjusted Rand index, Normalized Mutual Information and Purity. InTables 9–12, the results based on RI, ARI, NMI and Purity measures in real datasets, illustrate that the proposed algorithm outperforms all other competing clustering algorithms except iris and glass datasets. Hierarchical gets the best result in iris and glass dataset. Though the Purity value of GDD on Missa and Libras movements datasets is one, but that is due to the limitations of Purity that when each data point is considered as a cluster. In all these datasets the proposed algorithm has a satisfying performance over other competing clustering algorithms.

3.6. Estimation of number of clusters

We also compare the performance of the algorithms on the basis of estimating the optimal number of clusters present in the dataset based on external validity index. The following parameters are used to measure the performance (Rojas-Thomas et al., 2017):

(a) Number of hits: The number of times an algorithm correctly predicts the optimal number of clusters in the datasets.
(b) Average error: The average difference between the number of clusters predicted by the algorithm and the correct number of clusters in a given dataset. Let $|D|$ denotes the number of datasets used for evaluation. The average error is defined as:

$$avg\_err = \frac{\sum_{i=1}^{|D|} |actual - predicted|}{|D|} \tag{14}$$

The optimal number of clusters given by each of the algorithms is the basis for the comparison of the performance. We use the actual partitions to compare the algorithm with the predicted partitions. We compute the hits using Adjusted Rand Index to compare the number of clusters predicted by clustering algorithm with the target one. This is applied on both synthetic

Table 13
The number of cluster predicted by each algorithms (columns) on synthetic and gene expression datasets(rows) using adjusted rand index. The last two rows shows the number of hits and average error.

Datasets	Actual clusters	Hierarchical	DBSCAN	GDD	Proposed
DS1	2	**2**	**2**	3	**2**
DS2	2	**2**	**2**	**2**	**2**
DS3	3	4	1	6	**3**
DS4	3	**3**	**3**	**3**	**3**
DS5	2	3	1	1	**2**
DS6	2	4	1	3	3
DR1	4	6	7	–	5
DR2	3	6	8	–	**3**
DR3	4	**4**	10	–	**4**
DR4	6	8	5	–	7
DR5	4	**4**	6	–	6
DR6	5	8	4	–	**5**
DR7	3	7	1	–	**3**
Number of hits	–	5	3	–	**9**
Average error	–	1.38	1.84	–	**0.38**

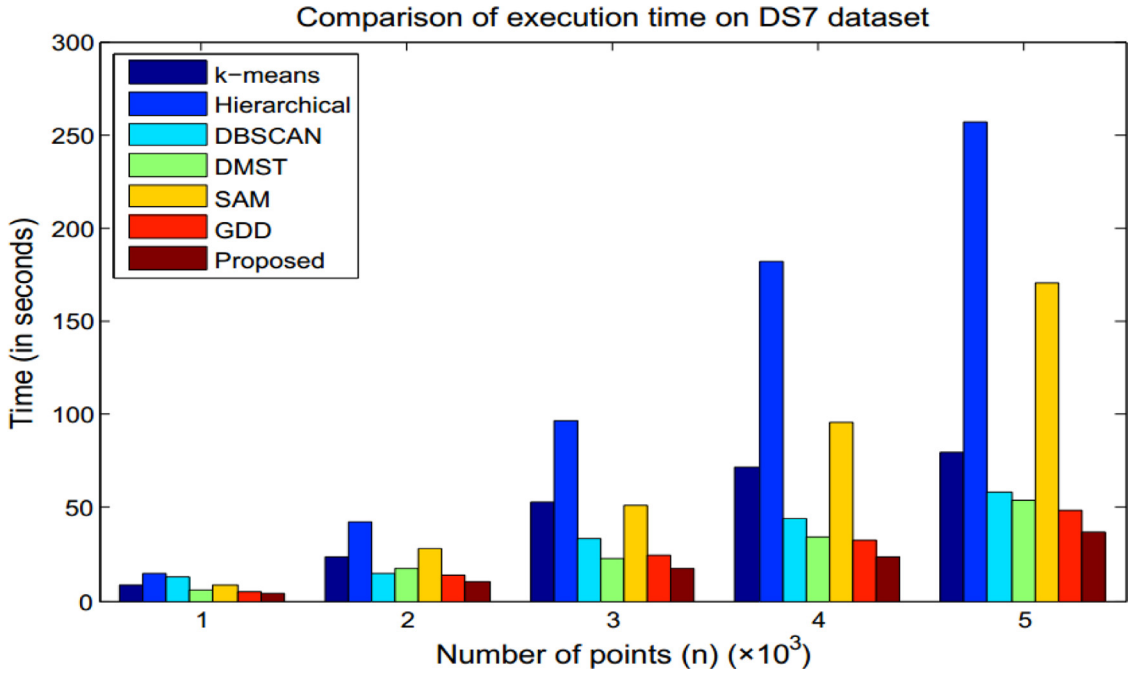

Fig. 16. Comparison of execution time(in seconds) from different algorithms on DS7 dataset.

and gene expression datasets, where the target partition is given in Broad Institute (2018), Yeung and et al. (2000), Blake and Merz (1998) and Pasi and et al. (2015). Table 13 illustrates the summary of the number of hits and average error of the different algorithms using ARI. The number of predicted clusters of GDD is much greater than the number of actual clusters on Gene expression datasets. We prefer to not include these results in the table. It can be seen that the hits of the proposed algorithm is much larger and average error rate is much lower than other competing algorithms, in terms of the number of clusters recognized.

3.7. Speed analysis

The performance of the proposed algorithm is compared against other competing clustering algorithms based on execution time. The sample size varies from 1000 to 5000 with a increase of 1000 in each run. Fig. 16 illustrates the time in seconds of competing clustering algorithms by varying the size of DS7 dataset. Due to repeated average similarity computation, hierarchical (average linkage) performs much slower as compared to other algorithms. SAM takes second highest time because it constructs the neighborhood graph using multiple round MST of a complete graph. DBSCAN also takes more time due to finding the nearest neighbor of each data points. GDD also incur more time due to computation of many statistical parameters. The proposed algorithm partitions the dataset into sub-clusters and construct MST on each sub-clusters only. Therefore, the overall running time of the proposed algorithm is less as compared to other clustering algorithms.

4. Conclusion and future work

In this paper, we proposed a new hybrid clustering technique based on local nearest neighbor using a minimum spanning tree which captures the geometry of clusters with respect to different dispersion level along dimensions in a heterogeneous dataset. The time complexity of the proposed algorithm is $O(N^{3/2})$ which is $O(\sqrt{N})$ factor improvement over popular hybrid clustering algorithms. The proposed approach is also free from any user-defined parameters. The dispersion level of data points is used for partitioning the data points into sub-clusters and MST of centroids

of these sub-clusters is constructed to identify the adjacent pairs. Finally, only adjacent sub-clusters are merged to identify the final clusters using cohesion and intra similarity. We demonstrated the performance through experiments on both synthetic as well as gene expression datasets that the proposed algorithm outperforms other competing clustering algorithms in terms of less computation time, improved cluster quality and nondependent on input parameters. An important direction of our future work is to devise a technique by using similar local nearest neighbor properties in identifying the local outliers in density-based clustering without any input parameters.

References

Barany, I., & Vu, V. (2007). Central limit theorems for gaussian polytopes. *The Annals of Probability, 35*(4), 1593–1621.

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

Blake, C., & Merz, C. (1998). Uci repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html department of information and computer science University of California, Irvine, CA, 55.

Bouguettaya, A., Yu, Q., Liu, X., Zhou, X., & Song, A. (2015). Efficient agglomerative hierarchical clustering. *Expert Systems with Applications, 42*(5), 2785–2797.

Chen, X. (2015). A new clustering algorithm based on near neighbor influence. *Expert Systems with Applications, 42*(21), 7746–7758.

Cheng, Q., Lu, X., Liu, Z., Huang, J., & Cheng, G. (2016). Spatial clustering with density-ordered tree. *Physica A: Statistical Mechanics and its Applications, 460*, 188–200.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms second edition*. The MIT Press.

Cui, H., Xie, M., Cai, Y., Huang, X., & Liu, Y. (2014). Cluster validity index for adaptive clustering algorithms. *IET Communications, 8*(13), 2256–2263.

Das, A. K., & Sil, J. (2007). Cluster validation using splitting and merging technique. In *International conference on computational intelligence and multimedia applications: 2* (pp. 56–60). IEEE.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD: 96* (pp. 226–231). ACM.

Grygorash, O., Zhou, Y., & Jorgensen, Z. (2006). Minimum spanning tree based clustering algorithms. In *18th international conference on tools with artificial intelligence ICTAI* (pp. 73–81). IEEE.

Guénoche, A., Hansen, P., & Jaumard, B. (1991). Efficient algorithms for divisive hierarchical clustering with the diameter criterion. *Journal of Classification, 8*(1), 5–30.

Guha, S., Rastogi, R., & Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. In *ACM Sigmod record: 27* (pp. 73–84). ACM.

Guha, S., Rastogi, R., & Shim, K. (2001). Cure: An efficient clustering algorithm for large databases. *Information Systems, 26*(1), 35–58.

Güngör, E., & Özmen, A. (2017). Distance and density based clustering algorithm using gaussian kernel. *Expert Systems with Applications, 69*, 10–20.

Huang, H., Gao, Y., Chiew, K., Chen, L., & He, Q. (2014). Towards effective and efficient mining of arbitrary shaped clusters. In *30th international conference on data engineering (ICDE)* (pp. 28–39). IEEE.

Hyde, R., & et al. (2014). Lancaster university clustering datasets. http://www.lancaster.ac.uk/pg/hyder/Downloads/downloads.html.

Broad Institute (2018). Broad institute cancer program datasets. http://broadinstitute.org/cgi-bin/cancer.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters, 31*(8), 651–666.

Jothi, R., Mohanty, S. K., & Ojha, A. (2015). Fast minimum spanning tree based clustering algorithms on local neighborhood graph. In *International workshop on graph-based representations in pattern recognition* (pp. 292–301). Springer.

Jothi, R., Mohanty, S. K., & Ojha, A. (2018). Fast approximate minimum spanning tree based clustering algorithm. *Neurocomputing, 272*, 542–557.

Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer, 32*(8), 68–75.

Kashef, R., & Kamel, M. S. (2009). Enhanced bisecting k-means clustering using intermediate cooperation. *Pattern Recognition, 42*(11), 2557–2569.

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1*(3), 231–240.

Lin, C.-R., & Chen, M.-S. (2002). A robust and efficient clustering algorithm based on cohesion self-merging. In *Proceedings of the eighth international conference on knowledge discovery and data mining (SIGKDD)* (pp. 582–587). ACM.

Lin, C.-R., & Chen, M.-S. (2005). Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 145–159.

Liu, M., Jiang, X., & Kot, A. C. (2009). A multi-prototype clustering algorithm. *Pattern Recognition, 42*(5), 689–698.

Mishra, G., & Mohanty, S. K. (2018). A minimum spanning tree based non-parametric, partitioning and merging clustering technique for heterogeneous data sets. Manuscript submitted for publication.

Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal, 26*(4), 354–359.

Pasi, F., & et al. (2015). Clustering datasets. http://cs.uef.fi/sipu/datasets/.

Pluim, J. P., Maintz, J. A., & Viergever, M. A. (2003). Mutual-information-based registration of medical images: A survey. *IEEE Transactions on Medical Imaging, 22*(8), 986–1004.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association, 66*(336), 846–850.

Ren, W., Li, G., & Tu, D. (2015). Graph clustering by congruency approximation. *IET Computer Vision, 9*(6), 841–849.

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science, 344*(6191), 1492–1496.

Rojas-Thomas, J., Santos, M., & Mora, M. (2017). New internal index for clustering validation based on graphs. *Expert Systems with Applications, 86*, 334–349.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review, 1*(1), 27–64.

Sutanto, E., & Warwick, K. (1995). Multivariable cluster analysis for high-speed industrial machinery. *IEE Proceedings-Science, Measurement and Technology, 142*(5), 417–423.

Wang, X., Wang, X., & Wilkes, D. M. (2009). A divide-and-conquer approach for minimum spanning tree-based clustering. *IEEE Transactions on Knowledge and Data Engineering, 21*(7), 945–958.

Yeung, K. Y., & et al. (2000). Validating clustering for gene expression data. http://faculty.washington.edu/kayee/cluster.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers, 100*(1), 68–86.

Zhong, C., Malinen, M., Miao, D., & Fränti, P. (2015). A fast minimum spanning tree algorithm based on k-means. *Information Sciences, 295*, 1–17.

Zhong, C., Miao, D., & Fränti, P. (2011). Minimum spanning tree based split-and-merge: A hierarchical clustering method. *Information Sciences, 181*(16), 3397–3410.

Zhong, C., Miao, D., Wang, R., & Zhou, X. (2008). Divfrp: An automatic divisive hierarchical clustering method based on the furthest reference points. *Pattern Recognition Letters, 29*(16), 2067–2077.

Zhou, S., Xu, Z., & Liu, F. (2017). Method for determining the optimal number of clusters based on agglomerative hierarchical clustering. *IEEE Transactions on Neural Networks and Learning Systems, 28*(12), 3007–3017.