Constraint-free Optimal Dual Similarity Validity Clusters Using Dynamic Minimum Spanning Tree

S. John Peter[1], S.P. Victor[2]
1. Assistant Professor, 2. Associate Professor
Department of Computer Science and Research Center St. Xavier's College,
Alayamkottai Tamil Nadu, India.

Summary

Clustering is a process of discovering groups of objects such that the objects of the same group are similar, and objects belonging to different groups are dissimilar. A number of clustering algorithms exist that can solve the problem of clustering, but most of them are very sensitive to their input parameters. Therefore it is very important to evaluate the result of them. The minimum spanning tree clustering algorithm is capable of detecting clusters with irregular boundaries. In this paper we propose a constraint-free minimum spanning tree based clustering algorithm. The algorithm constructs hierarchy from top to bottom. At each hierarchical level, it optimizes the number of cluster, from which the proper hierarchical structure of underlying dataset can be found. The algorithm uses a new cluster validation criterion based on the geometric property of data partition of the data set in order to find the proper number of clusters at each level. The radius and diameter of the clusters are computed to find the tightness of the individual clusters. The variance of the clusters is also computed to find the compactness of the individual clusters. In this paper we compute tightness and compactness of clusters, which reflects good measure of the efficacy of clustering. The algorithm works in two phases. The first phase of the algorithm produces subtrees. The second phase converts the subtrees into dendrogram. The key feature of the algorithm is it uses both divisive and agglomerative approaches to find optimal Dual similarity clusters.

Key Words:
Euclidean minimum spanning tree, Clustering, Eccentricity, Center, Hierarchical clustering, Dendrogram, Subtree, Cluster validity, Cluster Separation.

1. Introduction

The problem of determining the correct number of clusters in a data set is perhaps the most difficult and ambiguous part of cluster analysis. The "true" number of clusters depends on the "level" on is viewing the data. Another problem is due to the methods that may yield the "correct" number of clusters for a "bad" classification [10]. Furthermore, it has been emphasized that mechanical methods for determining the optimal number of clusters should not ignore that the fact that the overall clustering process has an unsupervised nature and its fundamental objective is to uncover the unknown structure of a data set, not to impose one. For these reasons, one should be well aware about the explicit and implicit assumptions underlying the actual clustering procedure before the number of clusters can be reliably estimated or, otherwise the initial objective of the process may be lost. As a solution for this, Hardy [10] recommends that the determination of optimal number of clusters should be made by using several different clustering methods that together produce more information about the data. By forcing a structure to a data set, the important and surprising facts about the data will likely remain uncovered.

In some applications the number of clusters is not a problem, because it is predetermined by the context [11]. Then the goal is to obtain a mechanical partition for a particular data using a fixed number of clusters. Such a process is not intended for inspecting new and unexpected facts arising from the data. Hence, splitting up a homogeneous data set in a "fair" way is much more straightforward problem when compared to the analysis of hidden structures from heterogeneous data set. The clustering algorithms [15, 21] partitioning the data set in to *k* clusters without knowing the homogeneity of groups. Hence the principal goal of these clustering problems is not to uncover novel or interesting facts about data.

Numerical methods can usually provide only guidance about the true number of clusters and the final decision is often an ad hoc decision that is based on prior assumptions and domain knowledge. Therefore, the choice between the different numbers of clusters is often made by comparing several alternatives, and the final decision is a subjective problem that can be solved in practice only by humans. Nevertheless, a number of methods for objective assessment of cluster validity have been developed and proposed. Because the recognition of cluster structures is difficult especially in high-dimensional spaces, various visualization technique can also be of valuable help to the cluster analyst.

Given a connected, undirected graph $G = (V, E)$, where V is the set of nodes, E is the set of edges between pairs of nodes, and a weight $w(u, v)$ specifying weight of the edge (u, v) for each edge $(u, v) \in E$. A spanning tree is an acyclic subgraph of a graph G, which contains all

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

vertices from *G*. The Minimum Spanning Tree (**MST**) of a weighted graph is minimum weight spanning tree of that graph. Several well established **MST** algorithms exist to solve minimum spanning tree problem [24, 19, 20]. The cost of constructing a minimum spanning tree is $O\ (m \log n)$, where *m* is the number of edges in the graph and *n* is the number of vertices. More efficient algorithm for constructing **MST**s have also been extensively researched [18, 5, 13]. These algorithms promise close to linear time complexity under different assumptions. A Euclidean minimum spanning tree (**EMST**) is a spanning tree of a set of *n* points in a metric space ($\mathbf{E^n}$), where the length of an edge is the Euclidean distance between a pair of points in the point set.

The hierarchical clustering approaches are related to graph theoretic clustering. Clustering algorithms using minimal spanning tree takes the advantage of **MST**. The **MST** ignores many possible connections between the data patterns, so the cost of clustering can be decreased. The **MST** based clustering algorithm is known to be capable of detecting clusters with various shapes and size [34]. Unlike traditional clustering algorithms, the **MST** clustering algorithm does not assume a spherical shapes structure of the underlying data. The **EMST** clustering algorithm [23, 24] uses the Euclidean minimum spanning tree of a graph to produce the structure of point clusters in the *n*-dimensional Euclidean space. Clusters are detected to achieve some measures of optimality, such as minimum intra-cluster distance or maximum inter-cluster distance [2]. The **EMST** algorithm has been widely used in practice.

Clustering by minimal spanning tree can be viewed as a hierarchical clustering algorithm which follows a divisive approach. Using this method firstly **MST** is constructed for a given input. There are different methods to produce group of clusters. If the number of clusters *k* is given in advance, the simplest way to obtain *k* clusters is to sort the edges of minimum spanning tree in descending order of their weights and remove edges with first *k*-1 heaviest weights [2, 33].

Geometric notion of centrality are closely linked to facility location problem. The distance matrix *D* can computed rather efficiently using Dijkstra's algorithm with time complexity $O(\mid V\mid^2 \ln \mid V \mid)$ [29].

The *eccentricity* of a vertex *x* in *G* and radius ρ (G), respectively are defined as

$$e(x) = \max_{y \in V} d(x\ ,\ y) \quad and \quad \rho(G) = \min_{x \in V} e(x)$$

The *center* of *G* is the set

$$\mathbf{C}\ (G) = \{x \in V \mid e(x) = \rho\ (G)\}$$

C (G) is the center to the "*emergency facility location problem*" which is always contain single block of *G*. The length of the longest path in the graph is called *diameter* of the graph *G*. we can define diameter D (G) as

$$D\ (G) = \max_{x \in V} e(x)$$

The *diameter* set of *G* is

$$Dia\ (G) = \{x \in V \mid e(x) = D\ (G)\}$$

All existing clustering Algorithm require a number of parameters as their inputs and these parameters can significantly affect the cluster quality. Our algorithm does not require a predefined cluster number. In this paper we want to avoid experimental methods and advocate the idea of need-specific as opposed to care-specific because users always know the needs of their applications. We believe it is a good idea to allow users to define their desired similarity within a cluster and allow them to have some flexibility to adjust the similarity if the adjustment is needed. Our Algorithm produces clusters of *n*-dimensional points with a naturally approximate intra-cluster distance.

Hierarchical clustering is a sequence of partitions in which each partition is nested into the next in sequence. An Agglomerative algorithm for hierarchical clustering starts with disjoint clustering, which places each of the *n* objects in an individual cluster [1]. The hierarchical clustering algorithm being employed dictates how the proximity matrix or proximity graph should be interpreted to merge two or more of these trivial clusters, thus nesting the trivial clusters into second partition. The process is repeated to form a sequence of nested clustering in which the number of clusters decreases as a sequence progress until single cluster containing all *n* objects, called the *conjoint clustering*, remains[1].

Nearly all hierarchical clustering techniques that include the tree structure have two short comings: (1) they do not properly represent hierarchical relationship and (2) once the data are assigned improperly to a given cluster it cannot later reevaluate and placed in another cluster.

In this paper, we propose a new clustering algorithm: *Dynamically Growing Euclidean Minimum Spanning Tree* (**DGEMST**) algorithm, which can overcome these shortcomings. The **DGEMST** algorithm optimizes the number of clusters at each hierarchical level with the cluster validation criteria during the minimum spanning tree construction process. Then the hierarchy constructed by the algorithm can properly represent the hierarchical structure of the underlying dataset, which improves the accuracy of the final clustering result.

Our **DGEMST** clustering algorithm addresses the issues of undesired clustering structure and unnecessary large number of clusters. Our algorithm does not require a predefined cluster number. The algorithm constructs an **EMST** of a point set and removes the inconsistent edges that satisfy the inconsistence measure. The process is repeated to create a hierarchy of clusters until optimal numbers of clusters (regions) are obtained. Hence the title! In section 2 we review some of the existing works on cluster validity and graph based clustering algorithms. In Section 3 we propose **DGEMST** algorithm which produces optimal number of clusters with Dendrogram. Hence we named this new cluster as *Optimal Dual similarity clusters.* Finally in conclusion we summarize the strength of our methods and possible improvements.

2. Related Work

Determining the true number of clusters, also known as the cluster validation problem, is a fundamental problem in cluster analysis. Many approaches to this problem have been proposed [25, 32, 10]. Two kinds of indexes have been used to validate the clustering [6, 7]: one based on relative criteria and other based on external and internal criteria. The first approach is to choose the best result from set of clustering result according to a prespecified criterion. Although the computational cost of the approach is light, human intervention is required to find the best number of clusters. The **DGEMST** algorithm tries to find the proper number of clusters automatically which makes the first approach unsuitable for clustering validation in the **DGEMST** algorithm.

The second approach is based on statistical tests and involves computations of both inter-cluster and intra-cluster quality to determine the proper best number of clusters. The evaluation of the criteria can be completed automatically. However the computational cost of this type of cluster validation is very high. The second type of this kind of approach is also not suitable for **DGEMST** algorithm when it is used to cluster a large dataset. A successful and practical cluster validation criteria used in the **DGEMST** algorithm for large dataset must have modest computational cost and can be easily evaluated automatically.

Clustering by minimal spanning tree can be viewed as a hierarchical clustering algorithm which follows the divisive approach. Clustering Algorithm based on minimum and maximum spanning tree were extensively studied. In the mid of 80's, Avis [3] found an $O(n^2 \log^2 n)$ algorithm for the min-max diameter-2 clustering problem. Asano, Bhattacharya, Keil and Yao [2] later gave optimal $O(n \log n)$ algorithm using maximum spanning trees for minimizing the maximum diameter of a bipartition. The problem becomes NP-complete when the number of partitions is beyond two [17]. Asano, Bhattacharya, Keil and Yao also considered the clustering problem in which the goal to maximize the minimum inter-cluster distance. They gave a *k*-partition of point set removing the *k*-1 longest edges from the minimum spanning tree constructed from that point set [2]. The identification of inconsistent edges causes problem in the **MST** clustering algorithm. There exist numerous ways to divide clusters successively, but there is not a suitable choice for all cases.

Zahn [34] proposes to construct **MST** of point set and delete inconsistent edges – the edges, whose weights are significantly larger than the average weight of the nearby edges in the tree. Zahn's inconsistent measure is defined as follows. Let *e* denote an edge in the **MST** of the point set, v_1 and v_2 be the end nodes of *e, w* be the weight of *e.* A *depth neighborhood* ***N*** of an end node *v* of an edge *e* defined as a set of all edges that belong to all the path of length *d* originating from *v*, excluding the path that include the edge *e*. Let N_1 and N_2 be the depth *d* neighborhood of the node v_1 and v_2. Let $\hat{W}_{N1}$ be the average weight of edges in N_1 and σN_1 be its standard deviation. Similarly, let $\hat{W}_{N2}$ be the average weight of edges in N_2 and σN_2 be its standard deviation. The inconsistency measure requires one of the three conditions hold:

1. $w > \hat{W}N_1 + c \text{ x } \sigma N_1$ *or* $w > \hat{W}N_2 + c \text{ x } \sigma N_2$

2. $w > \max(\hat{W}N_1 + c \text{ x } \sigma N_1, \hat{W}N_2 + c \text{ x } \sigma N_2)$

3. $\dfrac{w}{\max(c \text{ x } \sigma N_1, c \text{ x } \sigma N_2)} > f$

where *c* and *f* are preset constants. All the edges of a tree that satisfy the inconsistency measure are considered inconsistent and are removed from the tree. This result in set of disjoint subtrees each represents a separate cluster.

Paivinen [22] proposed a Scale Free Minimum Spanning Tree **(SFMST)** clustering algorithm which constructs scale free networks and outputs clusters containing highly connected vertices and those connected to them.

The **MST** clustering algorithm has been widely used in practice. Xu (Ying), Olman and Xu (Dong) [33] use MST as multidimensional gene expression data. They point out that **MST**- based clustering algorithm does not assume that data points are grouped around centers or separated by regular geometric curve. Thus the shape of the cluster boundary has little impact on the performance of the algorithm. They described three objective functions and the corresponding cluster algorithm for computing *k*-partition of spanning tree for predefined *k* > 0. The algorithm simply removes *k*-1 longest edges so that the weight of the subtrees is minimized. The second objective function is defined to minimize the total distance between the center and each data point in the cluster. The algorithm removes first *k*-1 edges from the tree, which creates a *k*-partitions.

Hierarchical clustering algorithm proposed by S.C.Johnson [16] uses proximity matrix as input data. The algorithm is an agglomerative scheme that erases rows and columns in the proximity matrix as old clusters are merged into new ones. The algorithm is simplified by assuming no ties in the proximity matrix. Graph based Hierarchical Algorithm was proposed by Hubert [12] using single link and complete link methods. He used threshold graph for formation of hierarchical clustering. An algorithm for single-link hierarchical clustering begins with the minimum spanning tree (MST) for G (∞), which is a proximity graph containing *n(n-1)/2* edge was proposed by Gower and Ross [8]. Later Hansen and DeLattre [9] proposed another hierarchical algorithm from graph coloring.

The procedure of evaluating the results of a clustering algorithm is known under the term cluster validity. In general terms, there are three approaches to investigate cluster validity [31]. The first is based on *external criteria*. This implies that we evaluate the results of a clustering algorithm based on a pre-specified structure, which is imposed on a data set and reflects our intuition about the clustering structure of the data set. The second structure is based on *internal criteria*. In this case the clustering results are evaluated in terms of quantities that involve the vectors of the data set themselves (e.g. proximity matrix). The third approach of clustering validity is based on *relative criteria*. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different input parameter values.

Given *n* d-dimensional data objects or points in a cluster, we can define the centroid x_0, radius *R*, diameter *D* and variance of the cluster as

$$X_0 = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$R = \left(\frac{\sum_{i=1}^{n} (X_i - X_0)^2}{n}\right)^{1/2}$$

$$D = \left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (X_i - X_j)^2}{n(n-1)}\right)^{1/2}$$

where *R* is the average distance from member objects to the centroid, and *D* is the average pair wise distance within a cluster. Both *R* and *D* reflect the tightness of the cluster around centroid[35].

The Cluster compactness measure is based on the variance of the data points distributed in the subtrees (clusters). The variance of cluster *T* is computed as

$$v(T) = \left(\frac{1}{n}\sum_{i=1}^{n} d^2(x_i, x_0)\right)^{1/2}$$

Where $d(x_i, x_j)$ is distance metric between two points(objects) x_i and x_j, where n is the number of objects in the subtree T_i and x_0 is the mean of the subtree *T.* A smaller the variance value indicates, a higher homogeneity of the objects in the data set, in terms of the distance measure *d ( ).* Since *d ( )* is the Euclidean distance, $v\ (T_i)$ becomes the statistical variance of data set $\sigma\ (T_i)$.

Many different methods for determining the number of clusters have been developed. Hierarchical clustering methods provide direct information about the number of clusters by clustering objects on a number of different hierarchical levels, which are then presented by a graphical tree structure known as *dendrogram.* One may apply some external criteria to validate the solutions on different levels or use the dendrogram visualization for determining the best cluster structure.

In order to measure the efficacy of clustering, a measure based upon the radius and diameter of each subtree (cluster) is devised. The radius and diameter values of each cluster are expected low value for good cluster. If the values are large that the points (objects) are spread widely and may overlap. The cluster tightness measure is a within – cluster estimate of clustering effectiveness , however it is possible to devise inter- cluster measure also, to better measure the separation between the various clusters.

The selection of the correct number of clusters is actually a kind of validation problem. A large number of clusters provides a more complex “model” where as a small number may approximate data too much. Hence, several methods and indices have been developed for the problem of cluster validation and selection of the number of clusters [27, 8, 26, 28, 30]. Many of them based on the within and between-group distance.

3. Our Clustering Algorithm

A tree is a simple structure for representing binary relationship, and any connected components of tree is called *subtree*. Through this **MST** representation, we can convert a multi-dimensional clustering problem to a tree partitioning problem, ie., finding particular set of tree edges and then cutting them. Representing a set of multi-dimensional data points as simple tree structure will clearly lose some of the inter data relationship. However many clustering algorithm proved that no essential information is lost for the purpose of clustering. This is achieved through rigorous proof that each cluster corresponds to one subtree, which does not overlap the representing subtree of any other cluster. Clustering problem is equivalent to a problem of identifying these subtrees through solving a tree partitioning problem. The inherent cluster structure of a point set in a metric space is closely related to how objects or concepts are embedded in the point set. In practice, the approximate number of embedded objects can sometimes be acquired with the help of domain experts. Other times this information is hidden and unavailable to the clustering algorithm. In this section we preset **DGEMST** clustering algorithm which produce optimal number of clusters, with dendrogram for each of them.

3.1 DGEMST Clustering Algorithm

Given a point set S in $\mathbf{E}^{\mathbf{n}}$, the hierarchical method starts by constructing a Minimum Spanning Tree (**MST)** from the points in S. The weight of the edge in the tree is Euclidean distance between the two end points. So we named this **MST** as **EMST1.** Next the average weight Ŵ of the edges in the entire **EMST1** and its standard deviation σ are computed; any edge with $W > \hat{W} + \sigma$ or *current longest edge* is removed from the tree. This leads to a set of disjoint subtrees $S_T = \{T_1, T_2 \ldots\}$ *(divisive approach)*. Each of these subtrees T_i is treated as cluster. Oleksandr Grygorash et al proposed algorithm [21] which generates k clusters. Our previous algorithm [15] generates k clusters with centers, which used to produce Dual similarity clusters. Both of the minimum spanning tree based algorithms assumed the desired number of clusters in advance. In practice, determining the number of clusters is often coupled with discovering cluster structure. Hence we propose a new algorithm named, *Dynamically Growing Euclidean Minimum Spanning Tree algorithm (***DGEMST**), which does not require a predefined cluster number. The algorithm works in two phases. The first phase of the algorithm partitioned the **EMST1** into sub trees (clusters/regions). The centers of clusters or regions are identified using eccentricity of points. These points are a representative point for the each subtree $S_{T.}$ A point c_i is assigned to a cluster i if $c_i \in T_i$. The group of center points is represented as $C = \{c_1, c_2 \ldots \ldots c_k\}$. These center points c_1, c_2 ....c_k are connected and again minimum spanning tree **EMST2** is constructed is shown in the Figure 4. This **EMST2** is used for finding optimal number clusters. A Euclidean distance between pair of clusters can be represented by a corresponding weighted edge. Our algorithm is also based on the minimum spanning tree but not limited to two-dimensional points. There were two kinds of clustering problem; one that minimizes the maximum intra-cluster distance and the other maximizes the minimum inter-cluster distances. Our Algorithm produces clusters with intra-cluster similarity. The Second phase of the algorithm converts the subtree/cluster into dendrogram (*agglomerative approach*). This algorithm use both divisive as well as agglomerative approach to find Dual similarity clusters. Since the subtrees are themselves are clusters, are further, classified in order to get more informative similarity clusters.

Here, in this algorithm we use a cluster validation criterion based on the geometric characteristics of the clusters, in which only the inter-cluster metric is used. The **DGMST** algorithm is a nearest centroid-based clustering algorithm, which creates region or subtrees (clusters/regions) of the data space. The algorithm partitions a set S of data of data D in data space in to n regions (clusters). Each region is represented by a centroid reference vector. If we let p be the centroid representing a region (cluster), all data within the region (cluster) are closer to the centroid p of the region than to any other centroid q:

$$R(p) = \{x \in D \mid dist(x, p) \leq dist(x, q) \ \forall q\}$$

Thus, the problem of finding the proper number of clusters of a dataset can be transformed into problem of finding the proper region (clusters) of the dataset. Here, we use the **MST** as a criterion to test the inter-cluster property. Based on this observation, we use a cluster validation criterion, called Cluster Separation (CS) in **DGMST** algorithm [4].

Cluster separation (CS) is defined as the ratio between minimum and maximum edge of MST. ie

$$CS = E_{min} / E_{max},$$

where E_{max} is the maximum length edge of **MST**, which represents two centroids that are at maximum separation, and E_{min} is the minimum length edge in the MST, which represents two centroids that are nearest to each other. Then, the CS represents the relative separation of centroids. The value of CS ranges from 0 to 1. A low value of CS means that the two centroids are too close to each other and the corresponding partition is not valid. A high CS value means the partitions of the data is even and valid. In practice, we predefine a threshold to test the CS. If the CS is greater than the threshold, the partition of the dataset is valid. Then again partitions the data set by creating subtree (cluster/region). This process continues until the CS is smaller than the threshold. At that point, the proper number of clusters will be the number of cluster minus one. The CS criterion finds the proper binary relationship among clusters in the data space. The value setting of the threshold for the CS will be practical and is dependent on the dataset. The higher the value of the threshold the smaller the number of clusters would be. Generally, the value of the threshold will be > 0.8[4]. Figure 3 shows the CS value versus the number of clusters in hierarchical clustering. The CS value < 0.8 when the number of clusters is 5. Thus, the proper number of clusters for the data set is 4. Furthermore, the computational cost of CS is much lighter because the number of subclusters is small. This makes the CS criterion practical for the **DGEMST** algorithm when it is used for clustering large dataset.

```
Algorithm: DGEMST ( )
Input    : S the point set
Output   : Optimal number of clusters with dendrograms

Let e1 be an edge in the EMST1 constructed from S
Let e2 be an edge in the EMST2 constructed from C
Let $W_e$ be the weight of e1
Let $\sigma$ be the standard deviation of the edge weights
 in EMST1
Let $S_T$ be the set of disjoint subtrees of the EMST1
Let $n_c$ be the number of clusters

 1. Construct an EMST1 from S
 2. Compute the average weight of $\hat{W}$ of all the
    Edges from EMST1
 3. Compute standard deviation $\sigma$ of the edges
 4. $S_T = ø$; $n_c = 1$
 5. Repeat
 6.   For each $e1 \in$ EMST1
 7.     If ($W_e > \hat{W} + \sigma$) or (current longest edge e1)
 8.        Remove e1 from EMST1 which result T’, a
           is new disjoint subtree
 9.        $S_T = S_T$ U {T’} // T’ is new disjoint
           subtree
10.        $n_c = n_c + 1$
11.        Compute the center $C_i$ of $T_i$ using
           eccentricity of points
12.        Compute the diameter of $T_i$ using
           eccentricity of points
13.        Compute the variance of $T_i$
14.        $C = U_{Ti} \in S_T \{C_i\}$
15.        Construct an EMST2 T from C
16.        $E_{min}$ = get-min-edge (T)
17.        $E_{max}$ = get-max-edge (T)
18.        $CS = E_{min} / E_{max}$
19.        Begin with T’, disjoint clusters with level
           $L_{nc}(0) = 0$ and sequence number $m = 0$
20.      While (T’ has some edge)
21.           e2 = get-min-edge(T’) // for least
              dissimilar pair of clusters
22.           (i, j) = get-vertices (e2)
23.           Increment the sequence number
              $m = m + 1$, merge the clusters (i) and (j),
              into single cluster to form next
              clustering m and set the level of this
              cluster to $L_{nc}(m) =$ e2;
24.           Update T’ by forming new vertex by
              combining the vertices i, j;
25. Until CS < 0.8
26. Return optimal clusters with dendrogram
```

Figure 1 shows a typical example of **EMST1** constructed from point set S, in which inconsistent edges are removed to create subtree (clusters/regions). Our algorithm finds the center of the each cluster, which will be useful in many applications. Figure 2 shows the possible distribution of the points in the two cluster structures with their center vertex as 5 and 3.

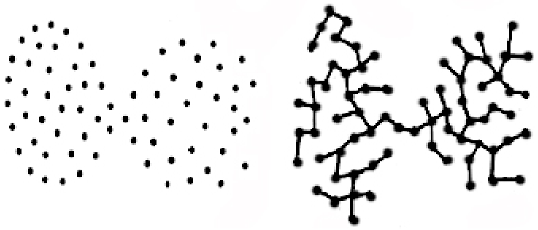
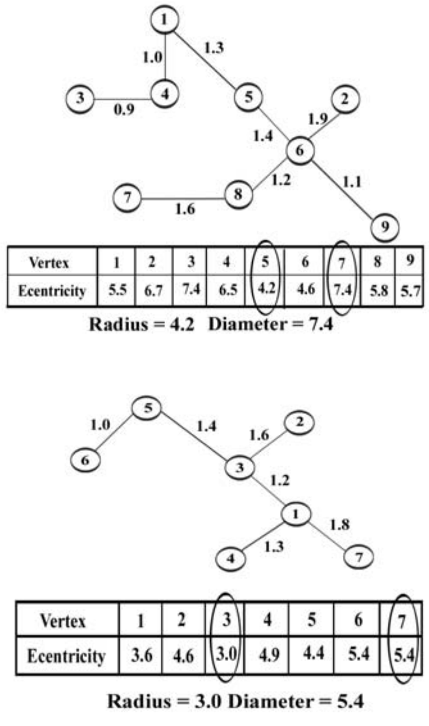


Figure 1. EMST1 - Clusters connected through a point



Vertex	1	2	3	4	5	6	7	8	9
Ecentricity	5.5	6.7	7.4	6.5	4.2	4.6	7.4	5.8	5.7

Radius = 4.2 Diameter = 7.4

Vertex	1	2	3	4	5	6	7
Ecentricity	3.6	4.6	3.0	4.9	4.4	5.4	5.4

Radius = 3.0 Diameter = 5.4

Figure 2. Two Clusters with Center vertices 5 and 3

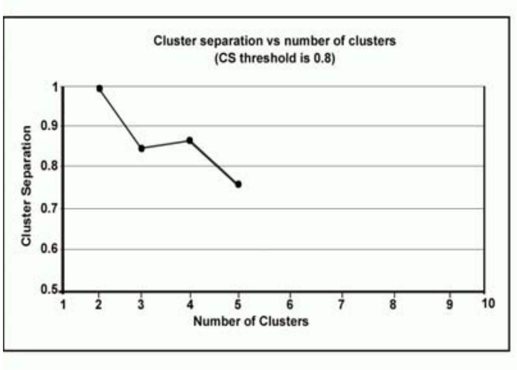


Figure 3. Number of Clusters vs. Cluster Separation

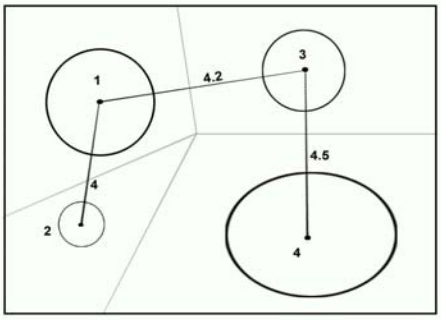


Figure 4. EMST2 From 4 region/cluster center points

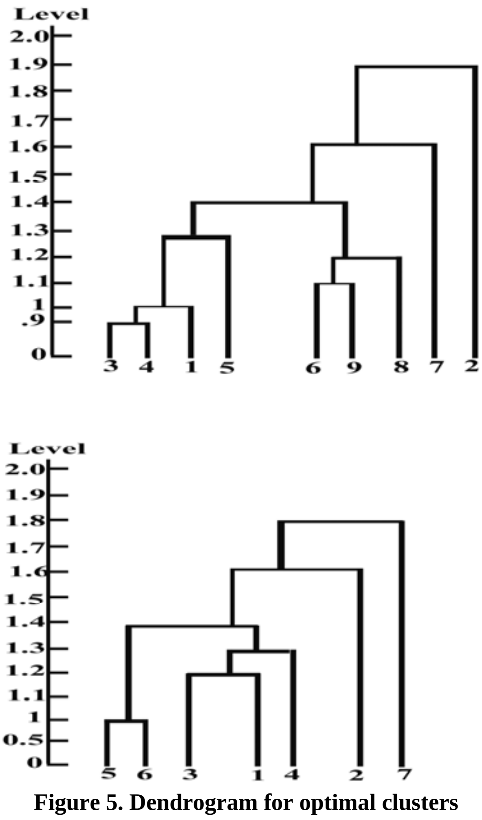


Figure 5. Dendrogram for optimal clusters

Our **DGEMST** algorithm works in two phases. The first phase of the algorithm (lines 1-18) uses *divisive approach* of hierarchical clustering. Euclidean minimum spanning tree **EMST1** is constructed in line 1. The average and standard deviation of the weighted edges of the Euclidean minimum spanning tree are computed to find inconsistent edges are specified in the lines 2-3. The inconsistent edges are identified and removed from Euclidean minimum spanning tree **EMST1** in order to generate subtree *T'* is specified in the lines 7-9. The center of each subtree is computed. The radius, diameter and variance of subtree (cluster) are computed (Lines (11-13). Lines 15-18 in the algorithm are used find the value of cluster separation (CS). This value is useful to find optimal number of clusters.

The second phase of the algorithm converts the subtrees *T'* into dendrograms is shown in the figure 5 (only two dendrograms are shown). For the newly created subtree *T'* again further hierarchical clustering is performed (lines 20-24). It places the entire disjoint cluster at level 0 (line 19). It then checks to see if *T'* still contains some edge (line 20). If so, it finds minimum edge *e2* (line 21). It then finds the vertices i, j of an edge *e2* (line 22). It then merges the vertices and forms a new vertex (*agglomerative approach*). At the same time the sequence number is increased by one and the level of the new cluster is set to the edge weight (line 23). Finally, updation of Euclidean minimum spanning tree is performed at line 24. The lines 20-24 in the algorithm are repeated until optimal number of clusters are obtained, which can be determined using CS value (line 18). Our algorithm uses both divisive as well as agglomerative approach in the **DGEMST** algorithm to find optimal Dual similarity clusters.

In order to measure the efficacy of clustering, a measure based upon the radius and diameter of each subtree (cluster) is devised. The radius and diameter values of each cluster are expected low value for good cluster is shown in Figure 6. If the values are large that the points (objects) are spread widely and may overlap. The cluster tightness measure is a within – cluster estimate of clustering effectiveness. The radius and diameter are good measure to find the tightness of clusters. The radius and diameter values of each cluster are expected low value for good cluster. If the values are large that the points (objects) are spread widely.

The variance for each subtree (cluster) is computed to find the compactness of clusters is shown in Figure 7. A smaller the variance value indicates, a higher homogeneity of the objects in the data set. The cluster compactness measure evaluates how well the subtrees (clusters) of the input is redistributed in the clustering process, compared with the whole input set, in terms of data homogeneity reflected by Euclidean distance metric used by the clustering process. Smaller the cluster compactness value indicates a higher average compactness in the out put clusters.

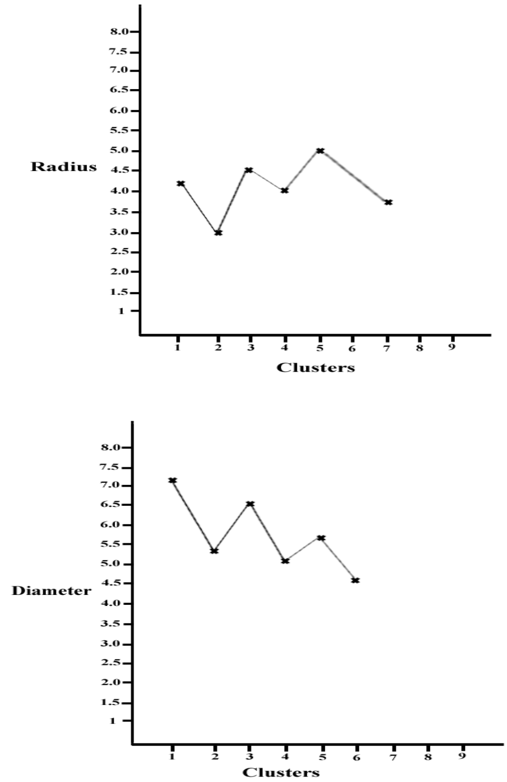


Figure 6. Tightness of individual clusters

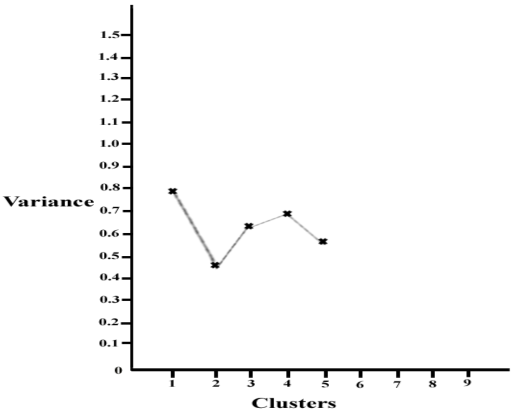


Figure 7. Compactness of individual clusters

4. Conclusion

Our **DGEMST** clustering algorithm does not assumes any predefined cluster number. The algorithm gradually finds clusters with center for each cluster. These clusters ensure guaranteed intra-cluster similarity. Our algorithm does not require the users to select and try various parameters combinations in order to get the desired output. Our **DGEMST** clustering algorithm uses a new cluster validation criterion based on the geometric property of partitioned regions/clusters to produce optimal number of "true" clusters with center for each of them. The inter-cluster distances between centers of clusters/regions are used to find optimal number of clusters. The **DGEMST** clustering algorithm generates dendrogram for optimal clusters, which is used to find the relationship between objects with in a cluster. The algorithm also finds radius, diameter and variance of individual clusters using eccentricity of points in a cluster. The radius and diameter values give the information about tightness of individual clusters. The variance value of the cluster is useful in finding the compactness of individual cluster. This information will be very useful in many applications. The validity assessment approaches proposed in the **DGEMST** algorithm will works better in various domains. All of these look nice from theoretical point of view. However from practical point of view, there is still some room for improvement for running time of the clustering algorithm. This could perhaps be accomplished by using some appropriate data structure. In the future we will explore and test our proposed clustering algorithm in various domains. The **DGEMST** algorithm uses both divisive as well agglomerative approaches. In this paper we used both the approaches to find optimal Dual similarity clusters. We will further study the rich properties of **EMST**-based clustering methods in solving different clustering problems.

References:

[1] Anil K. Jain, Richard C. Dubes "Algorithm for Clustering Data", Michigan State University, Prentice Hall, Englewood Cliffs, New Jersey 07632.1988.

[2] T. Asano, B. Bhattacharya, M.Keil and F.Yao. "Clustering Algorithms based on minimum and maximum spanning trees". In Proceedings of the 4th Annual Symposium on Computational Geometry,Pages 252-257, 1988.

[3] D. Avis "Diameter partitioning." Discrete and Computational Geometr, 1:265-276, 1986.

[4] Feng Luo,Latifur Kahn, Farokh Bastani, I-Ling Yen, and Jizhong Zhou, "A dynamically growing self-organizing tree(DGOST) for hierarchical gene expression profile",Bioinformatics,Vol 20,no 16, pp 2605-2617, 2004.

[5] M. Fredman and D. Willard. "Trans-dichotomous algorithms for minimum spanning trees and shortest paths". In Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science,pages 719-725, 1990.

[6] M. Halkidi, Y.Batistakis and M. Vazirgiannis "On clustering validation techniques", J.Intel. Inform. System., 17, 107-145, 2001

[7] M. Halkidi, Y.Batistakis and M. Vazirgiannis, "Clustering validity checking methods:part II" SIGMOD record., 31, 19-27, 2002

[8] G. Hamerly and C. Elkan, "Learning the k in k-means, in Advances in Neural Information Processing Systems" 16, S. Thrun, L. Saul, and B. Schölkopf, eds., MIT Press, Cambridge, MA, 2004.

[9] P. Hansen and M. Delattre, "Complete-link cluster analysis by graph coloring" Journal of the American Statistical Association 73, 397-403, 1978.

[10] A. Hardy, "On the number of clusters", Computational Statistics and Data Analysis, 23, pp. 83–96, 1996.

[11] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference and prediction", Springer-Verlag, 2001.

[12] Hubert L. J "Min and max hierarchical clustering using asymmetric similarity measures" Psychometrika 38, 63-72, 1973.

[13] H.Gabow, T.Spencer and R.Rarjan. "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs", Combinatorica, 6(2):109-122, 1986.

[14] J.C. Gower and G.J.S. Ross "Minimum Spanning trees and single-linkage cluster analysis" Applied Statistics 18, 54-64, 1969.

[15] S. John Peter, S.P. Victor, "A Novel Algorithm for Dual similarity clusters using Minimum spanning tree". Journal of Theoretical and Applied Information technology, Vol.14. No.1 pp 60-66, 2010.

[16] S. C. Johnson, "Hierarchical clustering schemes" Psychometrika 32, 241-254, 1967.

[17] D. Johnson, "The np-completeness column: An ongoing guide". Journal of Algorithms,3:182-195, 1982.

[18] D. Karger, P. Klein and R. Tarjan, "A randomized linear-time algorithm to find minimum spanning trees". Journal of the ACM, 42(2):321-328, 1995.

[19] J. Kruskal, "On the shortest spanning subtree and the travelling salesman problem", In Proceedings of the American Mathematical Society, pp 48-50, 1956.

[20] J. Nesetril, E.Milkova and H.Nesetrilova. Otakar boruvka on "Minimum spanning tree problem": Translation of both the 1926 papers, comments, history. DMATH: Discrete Mathematics, 233, 2001.

[21] Oleksandr Grygorash, Yan Zhou, Zach Jorgensen. "Minimum spanning Tree Based Clustering Algorithms". Proceedings of the 18th IEEE International conference on tools with Artificial Intelligence (ICTAI'06) 2006.

[22] N. Paivinen, "Clustering with a minimum spanning of scale-free-like structure".Pattern Recogn. Lett.,26(7): 921-930, 2005.

[23] F. Preparata and M.Shamos. "Computational Geometry": An Introduction. Springer-Verlag, Newyr, NY ,USA, 1985

[24] R. Prim. "Shortest connection networks and some generalization". Bell systems Technical Journal,36:1389-1401, 1957.

[25] R. Rezaee, B.P.F. Lelie and J.H.C. Reiber, "A new cluster validity index for the fuzzy C-mean", Pattern Recog. Lett., 19,237-246, 1998.

[26] D. M. Rocke and J. J. Dai, "Sampling and subsampling for cluster analysis in data mining: With applications to sky survey data", Data Mining and Knowledge Discovery, 7, pp. 215–232, 2003.

[27] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms", in Proceedings Sixteenth IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004, Los Alamitos, CA, USA, IEEE Computer Society, pp. 576–584 , 2004.

[28] S. Still and W. Bialek, "How many clusters?" , An information-theoretic perspective, Neural Computation, 16, pp. 2483–2506, 2004.

[29] Stefan Wuchty and Peter F. Stadler. "Centers of Complex Networks". 2006

[30] C. Sugar and G. James, "Finding the number of clusters in a data set ", An information theoretic approach, Journal of the American Statistical Association, 98 pp. 750–763, 2003.

[31] S. Theodoridis, K. Koutroubas, "Pattern recognition" Academic Press, 1999

[32] R. Tibshirani, G. Walther and T.Hastie "Estimating the number of clusters in a dataset via the gap statistic". J.R. Stat. Soc.Ser.B,63.411-423, 2001.

[33] Y.Xu, V.Olman and D.Xu. "Minimum spanning trees for gene expression data clustering". Genome Informatics,12:24-33, 2001.

[34] C. Zahn. "Graph-theoretical methods for detecting and describing gestalt clusters". IEEE Transactions on Computers, C-20:68-86, 1971.

[35] T.Zhang, R.Ramakrishnan and M.Livny. "BIRCH: an efficient data clustering method for very large databases". In Proc.1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96), pages 103-114, Montreal, Canada, June 1996.

S. John Peter is working as Assistant professor in Computer Science, St.Xavier's college (Autonomous), Palayamkottai, Tirunelveli. He earned his M.Sc degree from Bharadhidasan University, Trichirappalli. He also earned his M.Phil from Bharadhidasan University, Trichirappalli. Now he is doing Ph.D in Computer Science at Manonmaniam Sundranar University, Tirunelveli. He has published research papers on clustering algorithm in various national and international Journals.

Dr. S. P. Victor earned his M.C.A. degree from Bharathidasan University, Tiruchirappalli. The M. S. University, Tirunelveli, awarded him Ph.D. degree in Computer Science for his research in Parallel Algorithms. He is the Head of the department of computer science, and the Director of the computer science research centre, St. Xavier's college (Autonomous), Palayamkottai, Tirunelveli. The M.S. University, Tirunelveli and Bharathiar University, Coimbatore has recognized him as a research guide. He has published research papers in international, national journals and conference proceedings. He has organized Conferences and Seminars at national and state level.