VOL. 2, NO. 11, October 2011 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

Detection of Outlier-Communities using Minimum Spanning Tree

S. Chidambaranathan[1], S. John Peter[2]

[1]*Department of MCA*
[2]*Department of Computer Science and Research Center*
St. Xavier's College, Palayamkottai, Tamil Nadu, India.

ABSTRACT

Community (also known as clusters) is a group of nodes with dense connection. Detecting outlier-communities from database is a big desire. In this paper we propose a novel Minimum Spanning Tree based algorithm for detecting outlier-communities from complex networks. The algorithm uses a new community validation criterion based on the geometric property of data partition of the data set in order to find the proper number of communities. The algorithm works in two phases. The first phase of the algorithm creates optimal number of communities, whereas the second phase of the algorithm finds outlier-communities.

Keywords: *Euclidean minimum spanning tree, Clustering, Eccentricity, Center, Community validity, Community Separation, Outliers*

1. INTRODUCTION

A community is typically thought of as group of nodes with dense connections within groups and sparse connections between groups as well. Finding communities in complex networks is a nontrivial task, since the number of communities in the network is typically unknown and the communities often have arbitrary size and shape. There are some nodes in a special role like outliers. Outlier is an observation so much that it arouses suspicious that was generated by a different mechanism from the most part of data [1]. In clustering, outliers are considered as noise observations that should be removed in order to make more reasonable clustering [2]. Outliers can often be individual or groups of clients exhibiting behavior outside the range of what is considered normal. Outliers can be removed or considered separately in *regression modeling* to improve accuracy which can be considered as benefit of outliers. Identifying them prior to modeling and analysis is important [3]. In clustering–based methods, outlier is defined as observation that does not fit to the overall clustering pattern [34].

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 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 [5, 6, 7]. 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 [8, 9, 10]. 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.

Finding communities (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 [11, 12].

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 $O(|V|^2 \ln |V|)$ [13].

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

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

The *center* of G is the set

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

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

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

In some applications the number of clusters is not a problem, because it is predetermined by the context [14]. Then the goal is to obtain a mechanical partition for a

particular data using a fixed number of communities (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, 16] partitioning the data set in to *k* clusters without knowing the homogeneity of groups.

In this paper, we propose a new algorithm *Outlier-Community Detection using Minimum Spanning Tree* (**OCDMST**) which can overcome these shortcomings. The **OCDMST** algorithm optimizes the number of communities 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 **OCDMST** clustering algorithm addresses the issues of undesired community structure and unnecessary large number of communities. Our algorithm does not require a predefined community (cluster) number as input. 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 communities until best numbers communities are obtained. In section 2 we review some of the existing works on outliers' detection algorithms and graph based clustering. In Section 3 we propose **OCDMST** algorithm which produces best number of communities (clusters) with outlier-communities. Finally in conclusion we summarize the strength of our methods and possible improvements.

2. RELATED WORK

There is no single universally applicable or generic outlier detection approach [17, 18]. Therefore there is many approaches have been proposed to deduct outliers. These approaches are classified into four major categories as *distribution-based, distance-based, density-based and clustering-based* [19]. Here we discuss about *density-based* and *clustering-based* outlier detection approaches.

In *Density-based* methods outlier is defined from local density of observation. These methods used different density estimation strategies. A low local density on the observation is an indication of a possible outlier. Brito et al [20] proposed a *Mutual k-Nearest-Neighbor* (**MkNN**) graph based approach. **MkNN** graph is a graph where an edge exits between vertices v_i and v_j if they both belong to each others k-neighborhood. **MkNN** graph is undirected and is special case of *k-Nearest-Neighbor* (**kNN**) graph, in which every node has pointers to its *k* nearest neighbors. Each connected component is considered as cluster, if it contains more than one vector and an outlier when connected component contains only one vector. Connected component with just one vertex is defined as an outlier. The problem with this approach is that outlier too close to inliers, can be misclassified. *Density-based* approaches [21, 22] compute the density of the regions in the data and declare the objects in low dense regions as outliers.

Clustering-based approaches [17, 23, 24, 25], consider clusters of small sizes as outliers. In these approaches, small clusters (clusters containing significantly less points than other clusters) are considered as outliers. The advantage of *clustering- based* approaches is that they do not have to be supervised.

Jiang et al., [25] proposed a two-phase method to detect outliers. In the first phase, clusters are produced using modified K-means algorithm, and then in the second phase, an Outlier-Finding Process **(OFP)** is proposed. The *small clusters* are selected and regarded as outliers. *Small cluster* is defined as a cluster with fewer points than half the average number of points in the *k* number of clusters. Loureio [17] proposed a method for detecting outlier. Hierarchical clustering technique is used for detecting outliers. The key idea is to use the size of the resulting clusters as indicators of the presence of outliers. Almedia [26] is also used similar approach for detecting outliers. Using the K-means clustering algorithm Yoon [27] proposed a method to detect outliers.

Zahn [28] 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. The **MST** clustering algorithm has been widely used in practice. Xu (Ying), Olman and Xu (Dong) [12] 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.

The selection of the correct number of communities is actually a kind of validation problem. A large number of communities 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 community or cluster

validation and selection of the number of clusters [29, 30, 31, 32, and 33]. Many of them based on the within-and between-group distance.

3. OUR OCDMST ALGORITHM

Given a point set S in $\mathbf{E^n}$, 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 $\hat{W}$ 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 community (cluster). Oleksandr Grygorash et al proposed minimum spanning tree based clustering algorithm [16] which generates k clusters. Our previous algorithm [15] generates k clusters with centers, which used to produce Meta similarity clusters. Both of the minimum spanning tree based algorithms assumed predefined cluster number as input data. In practice, determining the number of communities or clusters is often coupled with discovering community structure. Hence we propose a new algorithm named, *Outlier-Community Detection Using Dynamic Minimum Spanning Tree* algorithm *(***OCDMST**), which does not require a predefined community or cluster number as input.

The algorithm works in two phases. The first phase of the algorithm creates communities by partitioned the **EMST1** into sub trees (communities/clusters). The centers of communities or clusters 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 $c_1, c_2 \ldots . c_k$ are connected and again minimum spanning tree **EMST2** is constructed is shown in the Figure 4. A Euclidean distance between pair of communities (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. Our algorithm produces communities with both intra-cluster and inter-cluster similarity. The Second phase of the algorithm detects outlier-communities from the communities (clusters), which are from the first phase of the OCDMST algorithm.

To detect the outliers from the communities, we assign an *outlyingness factor* for each object in the subtree (communities). Factor depends on its distance from cluster centroid. The method finds the community with maximum distance to the partitioned centroid d_{max} is defined as:

$$d_{max} = max\{ ||dist(x_i - c)|| \} \quad (1)$$

Outlyingness factor for each point (community) in the communities (subtrees/MST) are then computed. We define the *outlyingness* using Minimum Spanning Tree as:

$$O_i = dist(x_i - c) / d_{max} \quad (2)$$

We see that all *outlyingness factor* of the data set are normalized to the scale [0, 1]. The greater the value, more likely the objects is an outlier. As an example of the data set clustered in to two communities (shown in Figure 2) and computed *outlyingness factor* for the communities are shown in Figure 4. The point (community) for which $O_i > THR$ are defined as outlier-community. While scanning the communities (EMST2 T **),** the edges are ordered from smaller to larger lengths. Then we define the threshold as:

$$THR = min (L_i - L_{i-1}) * t \quad (3)$$

Where L_i is smallest in the order and $t \in [0, 1]$ is a user defined parameter.

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

$$R(p)=\{x \in D \,|\, dist(x,p) \leq dist(x,q)\ \forall q\} \quad (4)$$

Thus, the problem of finding the proper number of communities 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-community property. Based on this observation, we use a cluster validation criterion, called Community Separation (CS) in **OCDMST** algorithm [34].

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

$$CS=E_{min}/E_{max} \quad (5)$$

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 communities (clusters) will be the number of Community minus one. The CS criterion finds the proper binary relationship among communities (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 communities (clusters) would be. Generally, the value of the threshold will be > 0.8[34]. Figure 3 shows the CS value versus the number of communities (clusters) in hierarchical clustering. The CS value < 0.8 when the number of community (cluster) is 5. Thus, the proper number of community (cluster) for the data set is 4. Further more, the computational cost of CS is much lighter because the number of subclusters is small. This makes the CS criterion practical for the **OCDMST** algorithm when it is used for finding communities in large dataset.

Algorithm: OCDMST ( )

Input : S the point set, THR
Output : communities with outlier-community

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 $e1$ and σ be the standard deviation of the edge weights in **EMST1**
Let S_T be the set of disjoint subtrees of **EMST1**
Let OC be set of outlier-communities

1. Construct an **EMST1** from S
2. Compute the average weight of Ŵ of all the Edges from **EMST1**
3. Compute standard deviation σ of the edges from **EMST1**
4. $S_T = ø$; $n_c = 1$; $C = ø$; $O = ø$; $OC = ∅$
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**
9. $S_T = S_T \cup \{ T' \}$ // 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. $C = U_{Ti} \in S_T \{C_i\}$
13. Construct an **EMST2** T from C
14. E_{min} = get-min-edge (T)
15. E_{max} = get-max-edge (T)
16. CS = E_{min} / E_{max}
17. **Until** CS < 0.8
18. Compute the d_{max} using equation (1)
19. **For** each point (community) x_j *in* T
20. Compute outlyingness O_i using equation (3)
21. **IF** $O_i > THR$ **then** $OC = OC \cup \{ x_j \}$
22. **Return** communities with outlier-communities

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

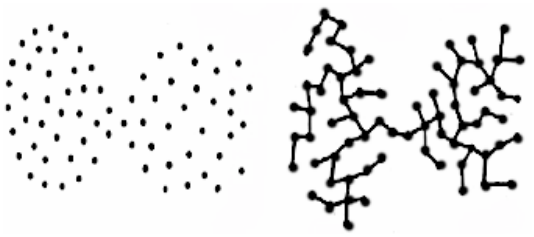
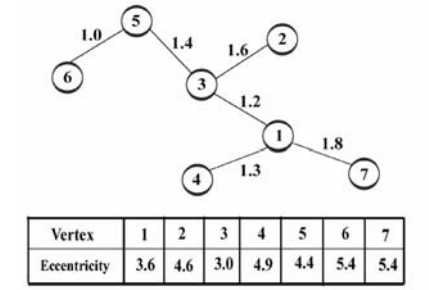
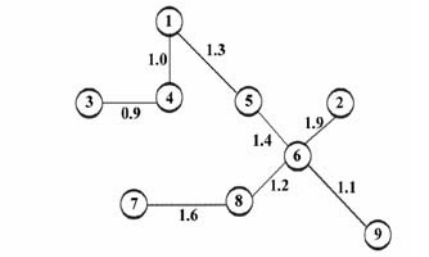


Figure 1. Clusters connected through points -EMST1



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



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

Figure 2. Two Communities with center points 5 and 3

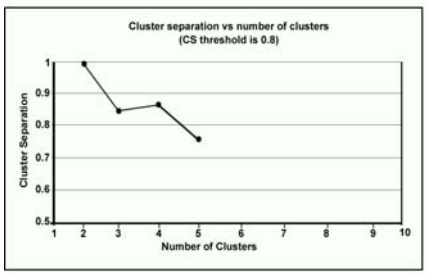
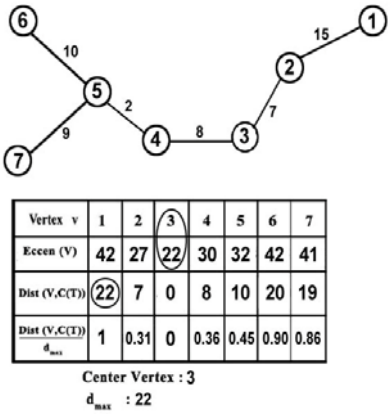


Figure 3. Number of Clusters vs. Cluster Separation



Vertex v	1	2	3	4	5	6	7
Eccen (V)	42	27	22	30	32	42	41
Dist (V,C(T))	22	7	0	8	10	20	19
Dist (V,C(T)) / d_{max}	1	0.31	0	0.36	0.45	0.90	0.86

Center Vertex : 3
d_{max} : 22

Figure 4. Outlier-communities are 1, 6 and 7

Our **OCDMST** algorithm works in three phases. The first phase (lines 1-16) of the **OCDMST** algorithm creates communities (clusters) with their center. It first constructs **EMST1** form set of point S (line 1). Average weight of edges and standard deviation are computed (lines 2-3). Inconsistent edges are identified and removed from **EMST1** to generate subtree *T'* (lines 7-9). The center for each subtree (cluster/community) is computed at line 11. Using the communities/clusters center point again another minimum spanning tree **EMST2** is constructed (line 13). Using the new evaluation criteria, best number of clusters/communities is identified (lines 14-16).

We use figure 4 to illustrate the second phase of the **OCDMST** algorithm. Here seven communities are represented in the form of MST which is generated from the first phase of the **OCDMST** algorithm. From the seven communities the second phase the **OCDMST** algorithm (lines 18-22) detects outlier-communities as shown in figure 4. The nodes having *outlyingness factor* 1, 0.90 and 0.86 are considered as outliers.

4. CONCLUSION

Our **OCDMST** clustering algorithm does not assumes any predefined community number. The algorithm gradually finds communities (clusters) with center for each community. These communities ensure guaranteed intra-community similarity. Our algorithm does not require the users to select and try various parameters combinations in order to get the desired output. Outlier-communities are detected using outlyingness factor. Our **OCDMST** algorithm uses a new community validation criterion based on the geometric property of partitioned community to produce best number of "true" communities with center for each of them. This will be very useful in many applications. 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 **OCDMST** algorithm in various domains.

REFERENCES

[1] B. Ghosh-Dastidar and J.L. Schafer, "Outlier Detection and Editing Procedures for Continuous Multivariate Data". *ORP Working Papers,* September 2003.

[2] S. Guha, R. Rastogi, and K. Shim. "CURE an efficient clustering algorithm for large databases". *In Proceeding of the 1998 ACM SIGMOD Int. Conf. on Management of Data , pp 73-84, Seattle, Washington,* 1998.

[3] G. Williams, R. Baxter, H. He, S. Hawkins and L. Gu, "A Comparative Study for RNN for Outlier Detection in Data Mining", *In Proceedings of the 2nd IEEE International Conference on Data Mining*, page 709, Maebashi City, Japan, December 2002.

[4] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: A new data clustering algorithm and its applications". *Data Mining and Knowledge Discovery*, 1(2):141-182, 1997.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] S. John Peter, S.P. Victor, "A Novel Algorithm for Meta similarity clusters using Minimum spanning tree". *International* Journal *of Computer Science and Network Security,* Vol.10 No.2 pp. 254 – 259, 2010.

[16] 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.

[17] A.Loureiro, L.Torgo and C.Soares, "Outlier detection using Clustering methods: A data cleaning Application", *in Proceedings of KDNet Symposium on Knowledge-based systems for the Public Sector.* Bonn, Germany, 2004.

[18] K.Niu, C.Huang, S.Zhang and J.Chen, "ODDC: Outlier Detection Using Distance Distribution Clustering", *T.Washio et al. (Eds.): PAKDD 2007 Workshops, Lecture Notes in Artificial Intelligence (LNAI)* 4819,pp.332-343,*Springer-Verlag,* 2007.

[19] J. Zhang and N. Wang, "Detecting outlying subspaces for high-dimensional data: the new task, Algorithms and Performance", *Knowledge and Information Systems,* 10(3):333-555, 2006.

[20] M. R. Brito, E. L. Chavez, A. J. Quiroz, and J. E. Yukich. "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection". *Statistics & Probability Letters,* 35(1):33-42, 1997.

[21] M. Breunig, H.Kriegel, R.Ng and J.Sander, Lof: "Identifying density-based local outliers". *In Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. ACM Press,* pp 93-104, 2000.

[22] S.Papadimitriou, H.Kitawaga, P.Gibbons and C. Faloutsos, "LOCI: Fast outlier detection using the local correlation integral" *Proc. Of the International Conference on Data Engineering* ,pp.315-326, 2003.

[23] Gath and A.Geva, "Fuzzy Clustering for the estimation of the Parameters of the components of Mixtures of Normal distribution", *Pattern Recognition letters*, 9, pp.77-86, 1989.

[24] Z. He, X. Xu and S. Deng, "Discovering cluster-based Local Outliers", *Pattern Recognition Letters,* Volume 24, Issue 9-10, pp 1641 – 1650, June 2003.

[25] M. Jaing, S. Tseng and C. Su, "Two-phase Clustering Process for Outlier Detection", *Pattern Recognition Letters,* Volume 22, Issue 6 – 7, pp 691 – 700, May 2001.

[26] J.Almeida, L.Barbosa, A.Pais and S.Formosinho, "Improving Hierarchical Cluster Analysis: A new method with OutlierDetection and Automatic Clustering," *Chemometrics and Intelligent Laboratory Systems* 87:208-217, 2007.

[27] K.Yoon, O.Kwon and D.Bae, "An approach to outlier Detection of Software Measurement Data using the K-means Clustering Method", *First International Symposium on Empirical Software Engineering and Measurement(ESEM 2007), Madrid.,*pp:443-445, 2007.

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

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.