Project in Graphs Algorithms - The use of Minimum Spanning tree in microarray expression data

Gkirtzou Katherine

December 12, 2005

1 Introduction

With the advent of microarray chip technology, large data sets are available and need to be analyzed and useful biological information to be borned. Clustering technology is a common method for analyzing gene expression data. There are a large number of algorithms for clustering gene expression patterns, such as hierarchical clustering, K-Means clustering and self-organizing maps. All these algorithms have some basic problems, despite they are staple. Non of them guarantee to bring forth a globally optimal clustering and some of them, like K-means clustering, depends on the geometric shape of cluster boundaries. In this report three different algorithms are being presenting that are based on the concept of the minimum spanning tree(MST), either as a clustering by creating an MST by the the original data or as a criterion to test the inter-cluster property. Firstly some useful definitions are given, then tree algorithms the expression data clustering analysis, the iterative algorithm and the dynaimcall growing seif-organizing tree are discribing.

2 Definitions

Here are some useful definitions:

Microarray: is a glass slide onto which single-stranded DNA molecules are attached at fixed locations(spots). There may be ten of thousands of spots on an array, each related to a single gene. [1] The DNA microarray technology enables the massive parallel measurement of gene expression of thousands genes simultaneously. The usefulness of this technology is the ability to compare the activity of many genes in diseased and healthy cells, categorize a disease into subgroups and discover new drug and toxicology studies.

Clustering: a common technique for data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis, bioinformatics and search engines. Clustering consists of partitioning a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often similarity or proximity for some defined distance measure.

There are essentially two types of clustering methods: hierarchical algorithms and non- hierarchical algorithms. The hierarchical algorithms can be divided into agglomerative and splitting procedures. The first type of hierarchical clustering starts from the finest partition possible (each observation forms a cluster) and groups them. The second type starts with the coarsest partition possible: one cluster contains all of the observations. It proceeds by splitting the single cluster up into smaller sized clusters. The non-hierarchical algorithms start from a given group definition and proceed by exchanging elements between groups until a certain score is optimized. The main difference between the two clustering techniques is that in hierarchical clustering once groups are found and elements are assigned to the groups, this assignment cannot be changed. In non-hierarchical techniques, on the other hand, the assignment of objects into groups may change during the algorithm application.

Minimum spanning tree(MST): a minimum weighted tree in a weighted graph which contains all of the graph's vertices. An MST can be efficiently computed in $O(N^2)$ time using either Prim's or Kruskal's algorithm.

Voronoi Diagram: are diagrams that have the property that for each site every point in the region around that site is closer to that site than to any of the other sites.

3 Expression Data Clustering Analysis and Visualization Resource

Xu at al. suggest a new algorithm based on minimum spanning tree(MST) representation of the data. With the MST representation the multi-dimensional clustering problem is transformed into a tree partitioning problem. It must be pointed out that no essential information is lost for the aim of clustering. A strictly proof that each cluster corresponds to one subtree, which does not overlap with any other representing subtree is given below.

3.1 Representing the Data Set as a Spanning Tree

But first let's define the MST representation. Let $D = d_i$ be a set of expression data with each $d_i = (e_i^1, \ldots, e_i^t)$ representing the expression levels at time 1 through time t of gene i. We define a weighted, undirected graph $G(D) = (V, E)$ as follows. The vertex set $V = \{d_i | d_i \in D\}$ and the edge set $E = \{(d_i, d_j) | for d_i, d_j \in D \text{ and } i \neq j\}$ [3]. Therefore the $G(D)$ is a complete graph. The weight of each edge is $(u, v) \in E$ is the distance $\rho(u, v)$ between u and v. The distance between two vertices can be defined as the Euclidean distance, the correlation coefficient or some other distance measure. Xu et al. have noticed that the data points of the same cluster are connected with short, in distance, tree edges, while long tree edges connect the clusters together(Like figure 1). This is an intuition for what a cluster consists of. But what is the separability condition of a cluster? Let D be a data set and ρ represent the distance between two data points of D. Then $C \subseteq D$ forms a

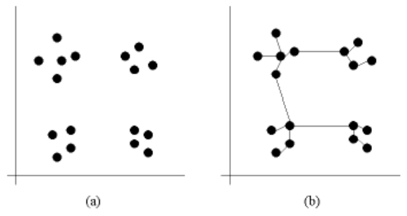


Figure 1: An MST representation of a set of data points.(a) A set of 2D points (b) An MST connecting all the data points, using the Euclidean distance.These data points form four natural clusters, based on their relative distances.

cluster in D only if for any arbitrary partition $C = C_1 \cup C_2$, the closest data point d to C_1, $d \in D - C1$, is from C_2. Formally, this can be written as

$$arg \min_{d \in D - C_1} \{\min\{\rho(d, c) | c \in C_2\}\}$$

where $D - C$ stands for the subset of D by removing all points of C. Now the proof that each cluster is exactly one subtree of the MST representation is apposed. In other words:

> If c_1 and c_2 are two points of a cluster C, then all data points in the tree path, P, connecting c_1 and c_2 in the MST, must be from C.

Proof: Let's assume that the statement is incorrect. Hence there exists a point α in path P, which does not belong to C. Without lost of generality, we assume that α is right next to c_1 on P so that (c_1, α) is an edge in P. We define a data set A as follows $A = \{c_1\}$. Then repeatedly A is expanded using the following operation until A converges: *select the data point x from D-A, which is closest to A;if $x \in C$ add x to A*. Apparently when A converges, $A = C$, based on the separability condition of C being a cluster. This means that there exists a path, P', from c_1 to c_2 that consists of only data points of C and all its edges have smaller distances (ρ) than $\rho(c_1, \alpha)$ (see Figure 2).It is known that at least one edge of P' is not in the current MST. For the simplicity of discussion it is assumed that exactly one edge, e, of P' is not in the current MST. So $P \cup P'$ contains a cycle with another spanning tree with smaller total distance. This contradicts the fact that a minimum spanning tree has the the minimum total distance among all spanning trees. By having this contradiction the statement is proved. [3]

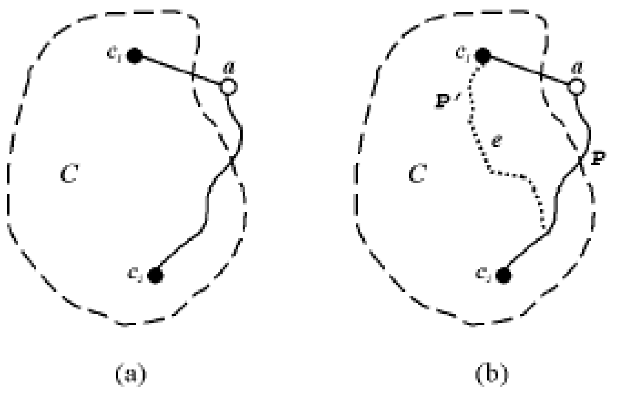


Figure 2: (a)A path connecting two vertices c_1 and c_2 of the same cluster C with one vertex α from a different cluster.(b)A schematic of the result of the expansion operation.

3.2 Clustering Algorithms based on MST representation

Xu et al. propose three different algorithms for clustering. All these algorithms are partitioning the minimum spanning tree into K subtrees, for a given integer $K > 0$.

The first algorithm uses the intuition that said above about the clusters, meaning that two data points with short edge-distance should belong to the same cluster, therefore to the same subtree, while long edge-distance are inter-cluster edges. It tries to cut off the $K - 1$ longest edges and create K subtree (clusters). Sometimes the given K can be to big and therefore the cluster will not be optimal. In order to upturn this problem, the algorithm examines the K-clustering for all $K = 1, 2, \ldots$ until K reaches the optimal number of clusters. Unfortunately this algorithm has one drawback. It works very well only if the intra-cluster edges are smaller than the inter-cluster edges.

The second algorithms tries to minimize the distance between the center of the cluster(subtree) and the data points. This is accomplished with the use of a more general function, which is:

$$\sum_{i=1}^{K} \sum_{d \in T_i} \rho(d, center(T_i)) \tag{1}$$

The algorithm starts with an arbitrary K-partitioning of the MST and for each pair of adjacent clusters, go through all tree edges within the merged cluster of the two to find the edge to cut, which globally optimizes the 2-partitioning of the merged cluster, measured by the objective function (1), until the process converges.

The last algorithm tries to optimize the K subtrees of the MST representation by selecting "best" representatives from the data set, in order to group around them the rest data points. The objective function of this is:

$$\sum_{i=1}^{K} \sum_{d \in T_i} \rho(d, d_i) \tag{2}$$

where $\rho()$ is the distance function, $T_1, \ldots, T_K$ are the K subtrees and $d_1, \ldots, d_K \in D$ a set of data points to be found so as the function (2) is minimized. The algorithm begins by converting the MST into a rooted by picking up an arbitrary vertex as root. Then for each vertex is defined the minimum value of the objective function (2) on the subtree rooted at vertex u, as $S(u, k, d)$, under the constraint that the subtree is partitioned into k subtrees and the representative of the subtree rooted at u is d. By definition the following gives the global minimum of the objective function:

$$\min_{d \in D} S(root, K, d) \tag{3}$$

The algorithm uses a dynamic approach to calculate the S() values at each tree vertex u, based on the S() values of its children. This means that for every vertex u with children S() is given from:

$$S(u,k,d) = \min_{x \subseteq C_u} \min_{\sum_{i=1}^{\|C_u\|} k_i = k + \|X\| - 1, k_i > 0} \Big(\sum_{u_j \in C_u - X} S(u_j, k_j, \overline{d}) + \sum_{u_j \in X} S(u_j, k_j, d) + \rho(u,d) \Big) \tag{4}$$

where

$$S(u_j, k_j, \overline{d}) = \min_{x \in D, x \neq d} S(u_j, k_j, x)$$

and C_u represents the set of all children of vertex u. For the vertex with no children the S() value is calculated by:

$$S(u,k,d) = \begin{cases} +\infty & \text{for k¿1} \\ \rho(u,d) & \text{for k=1} \end{cases} \tag{5}$$

4 An iterative clustering algorithm

Varma et al. propose an interative algorithm for clustering genes. The algorithm bases on the concept of minimum spanning tree and as a clustering measure the Fukuyama-Sugeno measure. Due to the concept of the MST if an edge is deleted from the tree it returns two disconnected trees. Let's assume that the length of the deleted edge is δ and the two subtrees we get is denoted as V_1 and V_2, we have that there are no pairs of $(x_1, x_2), x_1 \in V_1, x_2 \in V_2$ such that $d(x_1, x_2) < \delta$. As a result the separetion between the two subtrees V_1 and V_2 is at least δ. So if it is interested in finding out all possible binary partitions, they can be obtained by deleteing single edges from the MST.

4.1 Clustering Measure

The clustering measure that used in this algorithm for the comparison between the partitions, which are obtained by the different deletions of the MST, is the Fukuyama-Sugeno. It is defined as:

$$FS(S) = \sum_{k=1}^{2}\sum_{j=1}^{N_k}[\|x_j^k - \mu_k\|^2 - \|\mu_k - \mu\|^2] \tag{6}$$

where S_1, S_2 are the two partion of the set S, with each S_k contains N_k samples, denote by μ_k the mean of the samples in S_k and μ the global mean of all samples. Also denote by x_j^k the j-th sample in the cluster S_k.

4.2 Feature selection

With the feature selection it is measured the gene's support for a partition or the gene's relevant to a partition. By gene's support of gene's relevant to a partiton it means how differently the gene is expressed in samples beloning to different clusters. In Varma et al. algorithm the feature selection is measured by the two sample t-statistic with pooled variance. The t-statistic is calculated for each gene and the genes with absolute t-statistic greater that a threshold T_{thresh} are selected. In order to calculate the threshold $T_{threash}$ is needed the percentile threshold parameter $P_{thresh} \in (0, 100)$.

4.3 The algorithm

The algorithm needs as input parameters the gene expression martrix $\{x_{s,g}\}$, the maximum number of partitions to be found $MaxN_p$ and the percentile threashold P_{thresh}. The outer loop runs until the the number of the discoved clusters are smaller that the given $MaxN_p$. Initially the set of selected genes is all the given set of genes and the cutoff t is initialized as $T_{thresh}/2$. In the inner loop the MST is created by the set of selected genes, the genes from F, and for all the binary partitions obtained by deleting an edge from the tree the F-S measure is calculated. For the partition P^* with the lowest F-S measure, the genes are selected based on the t-statistic. These genes form the new gene set F_{new}. The next iteration the clustering is done with these selected genes and the cutoff t is increased, until the the selected gene subset remains the same betwwen two iterations. Then the current partition along with the gene set F is output and the number of discoved partitions is increased and an other interation of the outer loop is performed.

5 A dynamically growing self-organizing tree

In the previous algorithms the minimum spanning tree is constructed on the original set of data and used to test the intra-cluster quantity, while here the MST is used as a criterion to test the inter-cluster property. But before that let's examine the algorithm of the dynamically growing self-organizing tree(DGSOT).

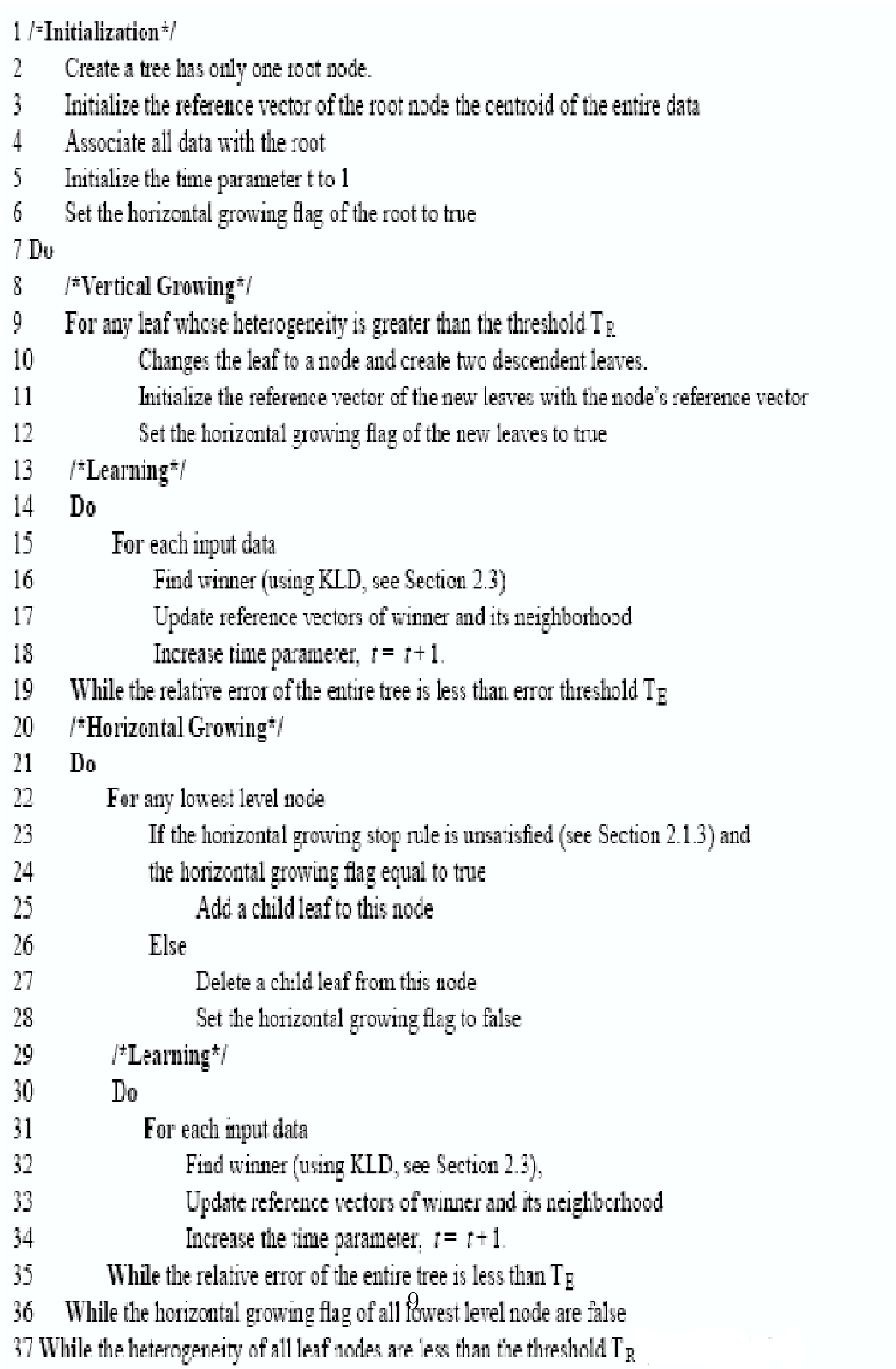
5.1 The algorithm

The DGSOT is a tree structure self-organizing neural network, which discovers suitable hierarchical structure of the fundamental data. The DGSOT grows vertically and horizontally. Each vertically growth step is followed by a horizontal growth step and this process continues until the heterogeneity of all leaves is less than the threshold T_R.

The algorithm begins with the construction of one leaf-node, the root of the tree, and the initialization of root's reference vector with the entire data. Then the vertical growing of the tree follows. In the vertical growing for every node that is a leaf and has heterogeneity greater than a threshold T_R, will change the leaf to a node and create two descendent leaves. The reference vectors of the two new leaves are initialized by their parent's vectors. After that the learning process, that is described below, takes place. The heterogeneity can be defined with two ways, as the variability, which is defined as the maximum distance among the input data associated with the leaf node, or as the average distortion d of a leaf, which is defined as:

$$d_i = \sum_{j=1}^{D} \frac{d(x_j n_i)}{|D|}$$

where D is the total number of input data assigned to the leaf i, $d(x_j, n_i)$ is the distance between data j and the leaf i and n_i is the reference vector of the leaf i.



```
/*Initialization*/
    Create a tree has only one root node.
    Initialize the reference vector of the root node the centroid of the entire data
    Associate all data with the root
    Initialize the time parameter t to 1
    Set the horizontal growing flag of the root to true
Do
    /*Vertical Growing*/
    For any leaf whose heterogeneity is greater than the threshold $T_R$
        Changes the leaf to a node and create two descendent leaves.
        Initialize the reference vector of the new leaves with the node's reference vector
        Set the horizontal growing flag of the new leaves to true
    /*Learning*/
    Do
        For each input data
            Find winner (using KLD, see Section 2.3)
            Update reference vectors of winner and its neighborhood
            Increase time parameter, $t = t + 1$.
    While the relative error of the entire tree is less than error threshold $T_E$
    /*Horizontal Growing*/
    Do
        For any lowest level node
            If the horizontal growing stop rule is unsatisfied (see Section 2.1.3) and
            the horizontal growing flag equal to true
                Add a child leaf to this node
            Else
                Delete a child leaf from this node
                Set the horizontal growing flag to false
        /*Learning*/
        Do
            For each input data
                Find winner (using KLD, see Section 2.3),
                Update reference vectors of winner and its neighborhood
                Increase the time parameter, $t = t + 1$.
        While the relative error of the entire tree is less than $T_E$
    While the horizontal growing flag of all lowest level node are false
While the heterogeneity of all leaf nodes are less than the threshold $T_R$
```

Figure 3: The DGSOT algorithm.

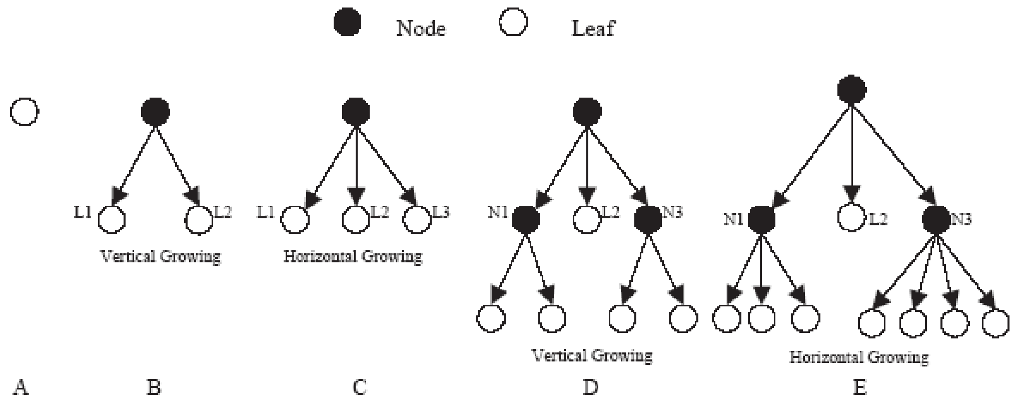


Figure 4: Illustration of DGSOT algorithm.

After that the horizontal growing follows. In the horizontal growing the DSGOT tries to find the optimal number of leaf nodes for every node. This optimal number is calculated by adding one leaf at the time to the lowest non-leaf nodes until a certain cluster validation criterion is satisfied(see section 5.2). Then the learning process takes place, like the vertical growing. In the learning process the data are distributed to the leaves. The best way to do this is to find the best matching node for the input data, with the use of K-level up distribution. This mean that the best matching node has the minimum distance to the input data. After the "correct" node is picked up the reference vector of this node and its neighbor will be updated. By neighborhood of the node we mean the node itself, the parent node and the siblings that are leaf (For an example of the DGSOT algorithm see figure 4).

5.2 The cluster validation criterion

The cluster validation criterion is used to find the optimal number of clusters for the input data.In DGSOT the validation criterion is calculated without human intervention and is based on geometric characteristics of the clusters. This is accomplished by creating the Voronoi diagram of the input data. The Voronoi diagram divides a set S of D data into n regions (clusters) V(p), which are called the Voronoi cell. Each Voronoi cell stands for a centroid reference vector. Let's assume that p represents a centroid of a region, then all data inside that region are nearer to the centroid p rather than any other centroid q

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

So the problem of finding the best number of clusters is transformed into finding the suitable Voronoi diagram.

The proper Voronoi diagram can be found by studying its geometric properties through a graph. Let'e define a weighted undirected graph $G(V, E)$. The vertices of the graph G is the set of the centroids of the Voronoi cell V(p), while the edge set E is defined as $E = \{(p_i, p_j)|p_i, p_j \in V(p)\&i \neq j\}$. The weight for each edge (p_i, p_j) is the distance between two centroids. The distance between two centroids can be defined as the Euclidea distance, as the correlation coefficient or some other distance measure.

The DGSOT uses the minimum spanning tree of the graph G as a measure to decide the proper partition of the data. This new cluster validation criterion is named as cluster separation (CS) and is defined as

$$CS = \frac{E_{min}}{E_{max}}$$

where E_{max} is the maximum length edge in the MST of the graph $G(V, E)$, while E_{min} is the minimum length edge in the MST of the graph $G(V, E)$. These means that E_{max} represents the two centroids that are at the maximum separation and E_{min} represents the two centroids that are at the minimum separation respectively. So the CS represents the relative separation of the centroids and its value ranges from 0 to 1. A low value of the CS means that the two centroids are to close to each other and the Voronoi partition is not valid, while a high CS value means that the Voronoi partition is valid. The threshold that the CS is described as low of high value is given by the user of the DGSOT algorithm. The way the DGSOT works is for every lowest non-leaf node the CS is evaluated, check if it is less than then threshold. If that is true, then it increases the number of clusters (child leaves nodes) by 1 and test the CS again, until the CS is less than the threshold.

6 Conclusions

The tree algorithms presented in this report have provided comparable result to those obtained by classic clustering algorithms, without their drawbacks, and superior to those obtained by standard hierachical clustering. For example the DGSOT algorithm provides considerably higher in biological significances results that the K-means clustering algorithm.

References

[1] Brazma A, Vilo J. *Gene expression data analysis*Microbes Infect (2001)

[2] Luo F, Khan L, Bastani F, Yen IL, Zhou J. *A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles.* Bioinformatics Vol. 20, No. 16 (2004)

[3] Xu Y, Olman V, Xu D. *Minimum spanning trees for gene expression data clustering.* Genome Informatics Vol. 12 (2001)

[4] Varma S, Simon R. *Iterative class discovery and feature selection using Minimal Spanning Trees.* BMC Bioinformatics (2004)