BIOINFORMATICS

Vol. 20 no. 16 2004, pages 2605–2617
doi:10.1093/bioinformatics/bth292

A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles

Feng Luo[1], Latifur Khan[1,*], Farokh Bastani[1], I-Ling Yen[1] and Jizhong Zhou[2]

[1]Department of Computer Science, University of Texas at Dallas, Richardson, TX 75252, USA and [2]Environmental Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

Received on January 8, 2004; revised on April 23, 2004; accepted on April 24, 2004
Advance Access publication May 6, 2004

ABSTRACT

Motivation: The increasing use of microarray technologies is generating large amounts of data that must be processed in order to extract useful and rational fundamental patterns of gene expression. Hierarchical clustering technology is one method used to analyze gene expression data, but traditional hierarchical clustering algorithms suffer from several drawbacks (e.g. fixed topology structure; mis-clustered data which cannot be reevaluated). In this paper, we introduce a new hierarchical clustering algorithm that overcomes some of these drawbacks.

Result: We propose a new tree-structure self-organizing neural network, called dynamically growing self-organizing tree (DGSOT) algorithm for hierarchical clustering. The DGSOT constructs a hierarchy from top to bottom by division. At each hierarchical level, the DGSOT optimizes the number of clusters, from which the proper hierarchical structure of the underlying dataset can be found. In addition, we propose a new cluster validation criterion based on the geometric property of the Voronoi partition of the dataset in order to find the proper number of clusters at each hierarchical level. This criterion uses the Minimum Spanning Tree (MST) concept of graph theory and is computationally inexpensive for large datasets. A K-level up distribution (KLD) mechanism, which increases the scope of data distribution in the hierarchy construction, was used to improve the clustering accuracy. The KLD mechanism allows the data misclustered in the early stages to be reevaluated at a later stage and increases the accuracy of the final clustering result. The clustering result of the DGSOT is easily displayed as a dendrogram for visualization. Based on a yeast cell cycle microarray expression dataset, we found that our algorithm extracts gene expression patterns at different levels. Furthermore, the biological functionality enrichment in the clusters is considerably high and the hierarchical structure of the clusters is more reasonable.

Availability: DGSOT is available upon request from the authors.

Contact: lkhan@utdallas.edu

1 INTRODUCTION

In every living organism, the subsets of its gene expression differ across types, stages and conditions. Given a specific condition and stage, there are particular genes expressed. Measuring these gene expression levels across different stages in different tissues or cells, or under different conditions, is very important and useful to understand and interpret the biological process. For a long time, biologists dreamed of getting information about all genes in a genome and the ability to study the complex interplay of all genes simultaneously. The emergence of the microarray technique provides for the fulfillment of this dream. The microarray technique enables the massively parallel measurement of gene expression of thousands genes simultaneously. There are many potential applications for the microarray technique, such as identification of genetic diseases, discovery of new drugs and toxicology studies. The wide application of microarray technologies now generates very large amounts of data. As a result, there is an increasing need for technology that can extract useful and rational fundamental patterns of gene expression from the data. Clustering technology is one of the most useful and popular methods for identifying these patterns.

Generally, there are two classes of cluster algorithms, hierarchical and non-hierarchical. Both have been successfully used in the analysis of gene expression data. Hierarchical agglomerative clustering (HAC) has been used to cluster two spotted DNA microarray datasets (Eisen *et al*., 1998) and to cluster central nervous system gene expression data

*To whom correspondence should be addressed.

from rats (Wen *et al*., 1998). Dopazo *et al*. (1997) proposed a self-organizing tree algorithm for clustering gene expression patterns. With regard to non-hierarchical clustering, many algorithms have been applied. Examples of this approach include using (1) a self-organizing map (SOM) to analyze the expression patterns of 6000 human genes (Tamayo *et al*., 1999), (2) K-means to cluster 3000 yeast gene microarray data (Tavazoie *et al*., 1999), (3) a graph-theoretic algorithm to extract high probability gene structures from gene expression data (Ben-Dor and Yakini, 1999), (4) a heuristic two-step adaptive quality-based algorithm (Smet *et al*., 2002) and (5) a clustering algorithm based on probability models (Yeung *et al*., 2001). Both the hierarchical and non-hierarchical algorithms attempt to find groups of genes that exhibit highly similar expression. Compared with non-hierarchical clustering, hierarchical clustering can find different levels of similarity in the gene expression data, and then detect trends and generalization information.

Traditional HAC algorithm has several drawbacks (Tamayo *et al*., 1999; Luo *et al*., 2003). Recently, a new self-organizing neural network-based hierarchical clustering algorithm called the self-organizing tree algorithm (SOTA) has been proposed (Dopazo *et al*., 1997; Herrero *et al*., 2001). SOTA is based on the Kohonen's SOM (Kohonen, 1997) and Fritzke's growing cell structures (Fritzke, 1994). The SOTA output is a binary tree topological neural network. Compared with HAC, SOTA uses a neural network mechanism and it is robust to noise data. However, as in HAC, its topological structure also has a fixed hierarchical topology, which can affect the final clustering result.

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

In this paper, we propose a new hierarchical clustering algorithm: the dynamically growing self-organizing tree (DGSOT) algorithm, which can overcome these shortcomings. The DGSOT dynamically optimizes the number of clusters at each hierarchical level with the cluster validation criteria during the tree construction process. Then the hierarchy constructed by the DGSOT algorithm can properly represent the hierarchical structure of the underlying dataset, which improves the accuracy of final clustering result. In addition, we also propose a K-level up distribution (KLD) mechanism. The data improperly clustered in early hierarchical clustering stages will have a chance to be reevaluated during the later hierarchical growing stages. Then the final cluster result will be more accurate. The DGSOT algorithm combines with KLD mechanism constitutes demonstrable, qualitative improvement over traditional solutions to the hierarchical clustering problem.

2 ALGORITHMS

2.1 Dynamically growing self-organizing tree algorithm

The DGSOT is a tree structure self-organizing neural network designed to discover the proper hierarchical structure of the underlying data. The DGSOT grows vertically and horizontally. In each vertical growth, the DGSOT adds two children to the leaf whose heterogeneity is greater than a threshold and turns it to a node. In each horizontal growth, the DGSOT dynamically finds the proper number of children (subclusters) of the lowest level nodes. Each vertical growth step is followed by a horizontal growth step. This process continues until the heterogeneity of all leaves is less than a threshold T_R. During vertical and horizontal growth, a learning process similar to the SOTA is adopted. The pseudocode of DGSOT is shown in Table 1. Figure 1 shows an example of the DGSOT algorithm working. Initially there is only one root node (Fig. 1A). All the input data are associated with the root and the reference vector of the root node is initialized with the centroid of the data (Table 1, lines 2–6). When vertical growth is invoked (Table 1, lines 9–19), two children are added to the root node. All input data associated with the root node are distributed between these children by employing a learning process (Fig. 1B). Following vertical growth, horizontal growth is invoked (Table 1, lines 21–36) to determine the proper number of children for the root. In this example, three leaves are proper (Fig. 1C). After this, the heterogeneities of the leaves are checked to determine whether to expand to another level or not. The answer is yes in this example, and a new vertical growth step is invoked. Two children are added to the leaves (L1, L3) whose heterogeneity are greater than the threshold (Fig. 1D) and turn them to nodes (N1, N3). All input data are distributed again with the learning process and a new horizontal growth begins (Fig. 1E). This process continues until the heterogeneity of all the leaves (indicated by empty cycle in Fig. 1) is less than the threshold.

In DGSOT, each leaf represents a cluster that includes all data associated with it. The reference vector of a leaf is the centroid of all data associated with it. Therefore, all reference vectors of the leaves form a Voronoi set of the original dataset and each internal node represents a cluster that includes all data associated with its leaf descendants. The reference vector of an internal node is the centroid of all data associated with its leaf descendents. The internal nodes of each hierarchical level also form a Voronoi set of the dataset with a different resolution.

The time complexity algorithm of DGSOT is similar to SOTA. Let d be the branch factor of the DGSOT. Then, the height of the DGSOT will be $\log_d M$, where M is the number of nodes in the hierarchy. For a full DGSOT tree (each leaf node associated with one data), the M is $O(N)$ where N is the number of data. Let J be the average number of learning iterations for each learning process.

Table 1. The DGSOT algorithm

```
/*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 (see Section 2.1.2)
```

Thus, the time complexity factor for building a full DGSOT will be $O[\log_d N * (J * N + d * J * N)]$, where $J * N$ and $d * J * N$ are due to the time complexity of vertical growing and horizontal growing, respectively. Note that both J and d are usually a small value as compared to N. Hence, we can treat J and d as constants, and the complexity will be $O(cN * \log_d) \approx O(N * \log_d N)$. The height of the DGSOT tree is controlled by the vertical growing threshold, namely the heterogeneity threshold of the leaf node. If only upper level patterns are needed the heterogeneity threshold can be set to larger value and the DGSOT can stop early stages. Then the height of the DGSOT can also be a constant and the time complexity of DGSOT can be reduced to $O(N)$, namely approximately linear.

2.1.1 Learning process Similar to SOTA (Dopazo *et al.*, 1997), the learning process consists of a series of procedures to distribute all the data to the leaves and update the reference vectors in DGSOT. Each procedure is called a cycle. Each cycle contains a series of epochs. Each epoch consists of a presentation of all the input data and each presentation has two steps, namely, finding the best matching node and updating the reference vector. The input data are only compared with the leaf nodes determined by the KLD mechanism to be the best matching node, which is known as the winner. The leaf

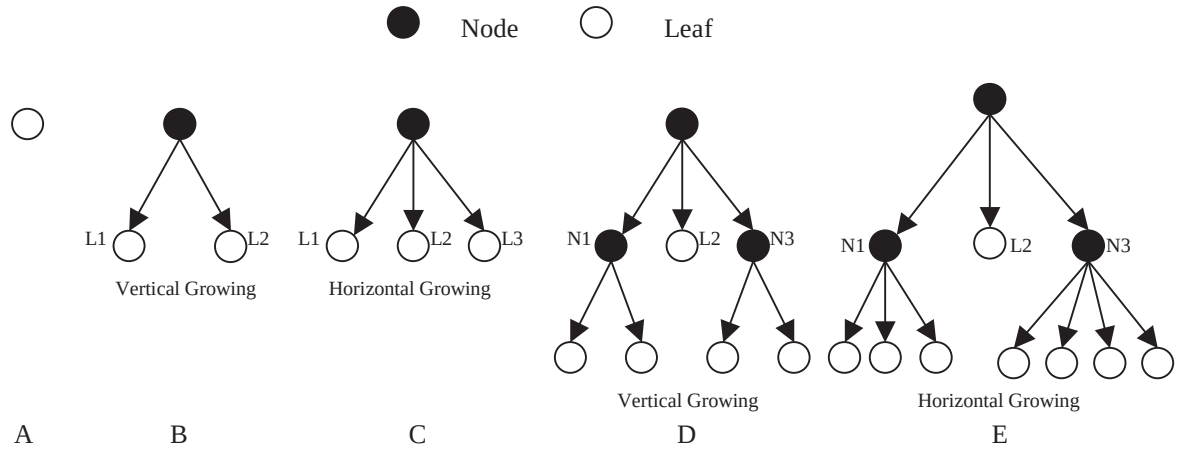

Fig. 1. Illustration of DGSOT algorithm.

node c that has the minimum distance to the input data x is the best match node (the winner).

$$c : \|x - n_c\| = \min_t\{\|x - n_i\|\}. \tag{1}$$

After a winner c is found, the reference vectors of the winner and its neighborhood will be updated using the following function:

$$\Delta w_i = \varphi(t) \times (x - w_i), \tag{2}$$

where $\varphi(t)$ is the learning function:

$$\varphi(t) = \alpha \times \eta(t), \tag{3}$$

and $\eta(t)$ is the learning rate function, α is the learning constant and t is the time parameter. The convergence of the algorithm depends on a proper choice of α and $\eta(t)$. At the beginning of the learning function, $\eta(t)$ should be close to 1; thereafter, it decreases monotonically. One choice can be $\eta(t) = 1/t$.

The neighborhood of the winner includes the winner, the parent node of the winner, and the siblings of the winner that are leaf (Fig. 2). Updating the reference vector of the siblings is important so that data that are dissimilar to each other are brought into a different subtree. The α of the winner, the parent node and the sibling nodes will have different values. Parameters α_w, α_m and α_s are used for the winner, the parent node and the siblings, respectively. The order of the parameters is set as $\alpha_w > \alpha_s > \alpha_m$. This is different from the SOTA setting, which is $\alpha_w > \alpha_m > \alpha_s$. In SOTA, an in-order traversal strategy is used to determine the topological relationship in the neighborhood. While in DGSOT, a post-order traversal strategy is used to determine the topological relationship in the neighborhood. In our opinion, only the non-equal α_w and α_s are critical to partitioning the input dataset into different leaves. The goal of updating the parent node's reference vector is to make it more precisely represent all the data associated with its children.

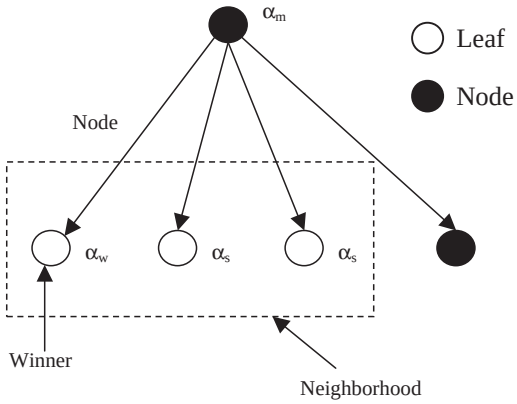

Fig. 2. Neighborhood of the winner in DGSOT.

The Error of the tree, which is defined as the summation of the distance of each input data to the corresponding winner, is used to monitor the convergence of the learning process. A learning process has converged when the relative increase of Error of the tree falls below a given threshold (Dopazo *et al*., 1997),

$$\left|\frac{\text{Error}_{t+1} - \text{Error}_t}{\text{Error}_t}\right| < \text{ErrorThreshold}, T_E. \tag{4}$$

2.1.2 Vertical growth In DGSOT, a non-greedy vertical growing strategy is used. During vertical growth, any leaf whose heterogeneity is greater than a threshold will change itself to a node and create two descendent leaves. The reference vectors of these leaves will be initialized by their corresponding parent's vectors. The vertical growth process will continue until the heterogeneities of all leaves are less than the threshold.

There are several ways to determine the heterogeneity of a leaf. One is the variability (Herrero *et al*., 2001), which

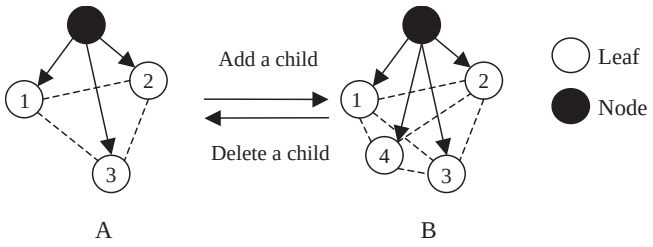

Fig. 3. Illustration of horizontal growing in DGSOT.

is defined as the maximum distance among the input data associated with the leaf node. The threshold of the variability can be determined by sampling the original dataset. Another simple way is to use the average distortion d of a leaf as its heterogeneity. The average distortion d of a leaf is defined as the average distance between the leaf node and the input data associated with the leaf:

$$d_i = \sum_{j=1}^{D} \frac{d(x_j, n_i)}{|D|}, \tag{5}$$

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. For this approach, the average error of each leaf will be similar. However, the number of data associated with the leaves may vary a lot. For all these approaches, the DGSOT can easily stop at higher hierarchical levels, which will save the computational cost.

2.1.3 Horizontal growth In each stage of horizontal growth, the DGSOT tries to find an optimal number of leaf nodes (subclusters) of a node to represent the clusters in each hierarchical level. Therefore, DGSOT adopts a dynamically growing scheme in each horizontal growth stage. For each of the lowest non-leaf nodes, a new child is added each time. This process continues until a certain cluster validation criterion (see Section 2.2) is satisfied. Once the cluster validation criterion is satisfied, the number of children is considered to be optimized. For example, in Figure 3A, the non-leaf node has three children. If the cluster validation criterion does not match, a new child is added (Fig. 3B). Similarly, if the addition of a new child satisfies the cluster validation criterion, the child is deleted and horizontal growth is stopped. After each addition/deletion of a node, a learning process is performed.

The DGSOT hierarchical tree will be a multifurcating tree due to the horizontal growth. We demonstrate that the hierarchical structure of the neural network can affect the final clustering results by comparing DGSOT with SOTA. The clustering validation criteria control the number of clusters in each hierarchical level. The validity of multifurcating in DGSOT is dependent on the validity of the clustering validation criteria. A proper cluster validation criterion will not let the horizontal growth continue forever unless there is no hierarchical structure in the dataset.

2.2 Cluster validation criterion for DGSOT

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 recently (Tibshirani *et al*., 2001; Pal *et al*., 2001; Hardy, 1996; Cuevas *et al*., 2000; Rezaee *et al*., 1998). Two kinds of indexes have been used to validate the clustering (Halkidi *et al*., 2001, 2002): one based on relative criteria and the other based on external and internal criteria. The first approach is to choose the best result from a set of clustering results 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. In addition, most of these indexes need to be constructed off-line, and require manual help to evaluate the result. The DGSOT algorithm tries to find the proper number of clusters in each hierarchical level automatically, which makes the first type of cluster validation measures unsuitable for clustering validation in the DGSOT.

The second approach is based on statistical tests and involves computation of both intercluster and intracluster quality to determine the proper 'true' number of clusters. The evaluation of the criteria can be completed automatically. Thus, this kind of clustering validation criteria, e.g. average silhouette (Rousseeuw, 1987; Vilo *et al*., 2000), can be used in DGSOT to control the horizontal growth. The value of average silhouette lies between -1 and 1. If the average silhouette is equal to 1 the cluster is valid. In addition, if it is less than 0 the data in this cluster is on average closer to members of some other clusters and this cluster is invalid. During horizontal growth process, we can calculate the average silhouette for each subcluster. If adding a new cluster leads the average silhouette of some clusters to be less than 0, then the new added cluster is invalid. Moreover, the proper number of clusters will be the number of clusters minus one. However, these types of cluster validation criteria always have a high computational cost. As the cluster validation criterion is used very heavily in the DGSOT algorithm, the second type of cluster validation criteria are not suitable for DGSOT when it is used to cluster a large dataset. A successful and practical cluster validation criteria used in DGSOT for large dataset must have modest computational cost and can be easily evaluated automatically.

Here, we propose a new cluster validation criterion based on the geometric characteristics of the clusters, in which only the intercluster metric is used. The DGSOT is a nearest centroid-based clustering algorithm, which creates a Voronoi diagram of the data space. The Voronoi diagram partitions a set S of data D in data space into n regions (clusters) $V(p)$, called the Voronoi cell (Fig. 4). Each Voronoi cell 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

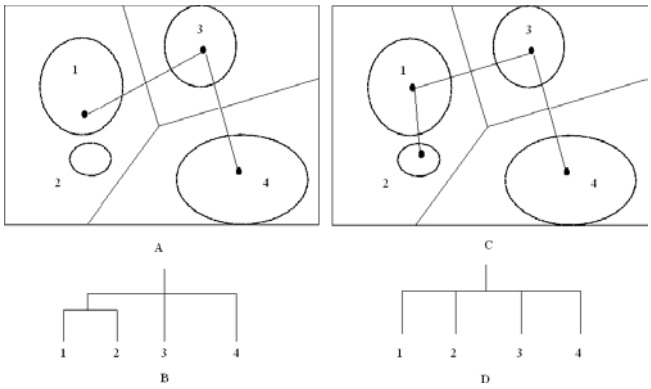

Fig. 4. Voronoi diagram of a 2D sample dataset. The black points represent centroids. The lines represent the MST among the centroids. In A, the dataset is partitioned into three Voronoi cells. The MST of the centroid is 'even'. B is the corresponding hierarchical structure of A. In C, the dataset is partitioned into four Voronoi cells. The MST of the reference vector is not 'even'. One of the MST edges is short. D is the corresponding hierarchical structure of C.

centroid q:

$$V(p) = \{x \in D \mid \text{dist}(x, p) \leq \text{dist}(x, q)\, \forall q\}. \qquad (6)$$

Thus, the problem of finding the proper number of clusters of a dataset at a certain hierarchical level can be transformed into the problem of finding the proper Voronoi partition of the dataset in the hierarchical level. Figure 4 shows two partitions of a two-dimensional (2D) sample dataset and their corresponding hierarchical structure. This 2D dataset contains four clusters. Cluster 1 is very close to cluster 2. In Figure 4A, the data set is partitioned into three clusters—cluster 3, cluster 4 and a big cluster including clusters 2 and 1. The corresponding hierarchical structure is shown in Figure 4B. In Figure 4C, the dataset is partitioned into four clusters—clusters 1, 2, 3 and 4. The corresponding hierarchical structure is shown in Figure 4D. In Figure 4A, two close clusters have been placed at a low hierarchical level. From the hierarchical viewpoint, the hierarchical structure shown in Figure 4B is more reasonable than that shown in Figure 4D. At the first level, the proper number of clusters of the dataset is 3.

By investigating the geometric properties of the Voronoi cell in Figure 4 we can find some clue about the proper Voronoi partition of the dataset at a certain hierarchical level. Let us define a weighted undirected graph $G = (V, E)$. The vertices of the graph G are the centroids of Voronoi cell $V(p)$, and the edge set $E = \{(p_i, p_j) \mid p_i, p_j \in V(p)\ \&\ i \neq j\}$. Note that the weight of each edge (p_i, p_j) represents the distance between two centroids. It can be the Euclidean distance, the correlation coefficient or some other distance measure.

We use the Minimum Spanning Tree (MST) of the graph G as a criterion to determine the proper partition of the data. A spanning tree, T, of a weighted undirected graph G is a minimal subgraph of G, which connects all vertices. A MST is the spanning tree with the smallest total weight. The MST can represent the binary relationship of vertices in the graph. The

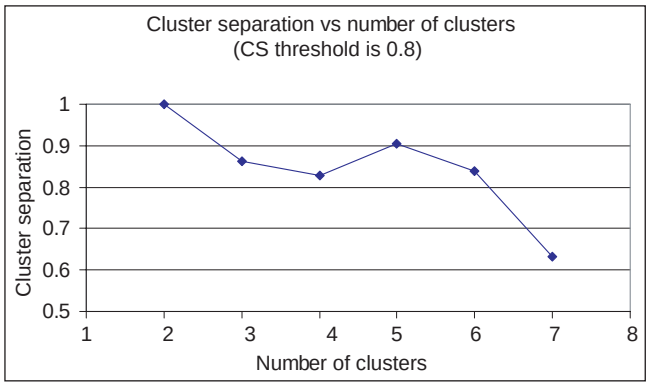

Fig. 5. The cluster separation versus the number of clusters for the 3000 yeast cell cycle gene microarray expression data.

MST has been applied to clustering (Xu *et al*., 2002; Gower and Ross, 1969) and cluster validation (Pal *et al*., 2001), but in all these applications, the MST is constructed on the original dataset and used to test the intracluster quantity. Here, we use the MST as a criterion to test the inter-cluster property. For example, we construct the MST among the centroids in Figure 5A and 5C. In the MST of Figure 4A, the lengths of the edges are similar, and in the MST of Figure 4C, the lengths of the edges are not 'even'. The edge between the centroids of Clusters 1 and 2 is much shorter than other edges. Based on this observation, we propose a new cluster validation criterion, called Cluster separation (CS).

2.2.1 Cluster separation (CS) The definition of CS between clusters is given by the following:

$$\text{CS} = \frac{E_{\min}}{E_{\max}}, \qquad (7)$$

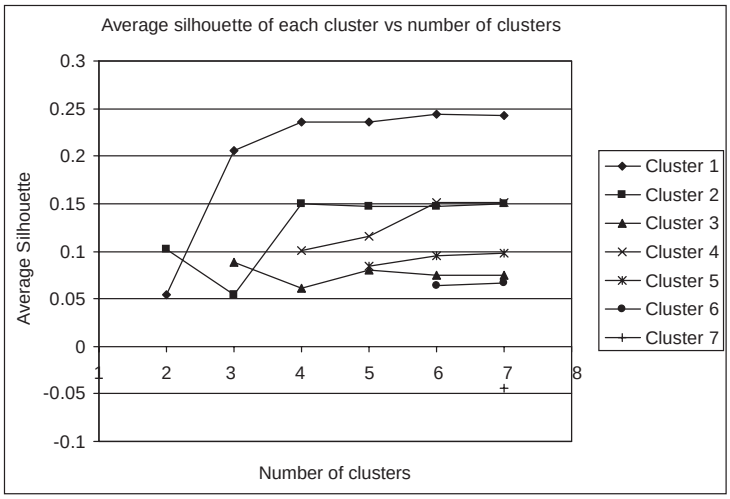

Fig. 6. The average silhouette of each cluster versus the number of clusters for the 3000 yeast cell cycle gene microarray expression data.

where $E_{\max}$ is the maximum length edge in the MST of graph $G(V, E)$, which represents two centroids that are at maximum separation, and $E_{\min}$ is the minimum length edge in the MST of graph $G(V, E)$, which represents two centroids that are nearest to each other. Then, the CS represents the relative separation of the 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 Voronoi partition is not valid. A high CS value means the Voronoi partition 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 Voronoi partition of the dataset is valid. Then, we increase the number of clusters by 1 and test the CS again. This process continues until the CS is smaller than the threshold. At that point, the proper number of clusters will be the number of clusters minus one. The CS criterion finds the proper binary relationship among clusters in the data space and indicates a proper Voronoi partition of the data space, namely, the proper clusters in certain hierarchical level of the DGSOT. 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.

We tested the CS criterion on a 3000 yeast gene cell cycle expression profiles dataset. For this dataset, we chose a CS threshold of 0.8. A horizontal growth process was employed to add a cluster each time. Figure 5 shows the CS value versus the number of clusters in the first level hierarchy. The CS is <0.8 when the number of clusters is 7. Thus, the proper number of clusters for the first hierarchical level of this dataset is 6. To evaluate the CS criterion, we also calculated the average silhouette of each cluster as the number of clusters increases. Figure 6 shows the average silhouette of each cluster versus the number of clusters. The average silhouette of each cluster is kept positive before the number of cluster 7 is added. After the 7th cluster is added its average silhouette is <0. This means the 7th cluster is not valid and only the first six clusters are valid. The proper number of the clusters in the first hierarchical level of this dataset is therefore 6, which is the same number found using the CS criterion. This demonstrates that the CS is consistent with the classical average silhouette measure when a proper threshold for CS criterion is chosen. Furthermore, the computational cost of CS is much lighter because the number of subclusters of each node is small. This makes the CS criterion practical for the DGSOT algorithm when it is used for clustering large dataset.

2.3 *K*-level up distribution

Self-organizing neural network based clustering is a type of distortion-based competitive learning. The nearest neighbor rule is used to make the clustering decision. In original SOTA, data associated with the parent node are distributed between its two children. If data are improperly clustered in the early stages, these errors cannot be corrected in the later learning process. For example, in Figure 7 there are two kinds of data, namely, cluster x and cluster y. In the early clustering stage (Fig. 7A), all data have been partitioned into two clusters, and each one is represented by a centroid. One Y is closer to the upper centroid and is assigned to the cluster that contains all x. When the self-organizing neural network expands to more levels, this improperly clustered Y will not be reevaluated, thus, the lower level nodes inherit the misclustered Y from the early clustering stage (Fig. 7B), and the final clustering result is inaccurate (Fig. 7D).

To improve the cluster result, we propose a new distribution approach called KLD (Khan and Luo, 2002). Recently, Herrero *et al.* (2003) also employed similar mechanism in

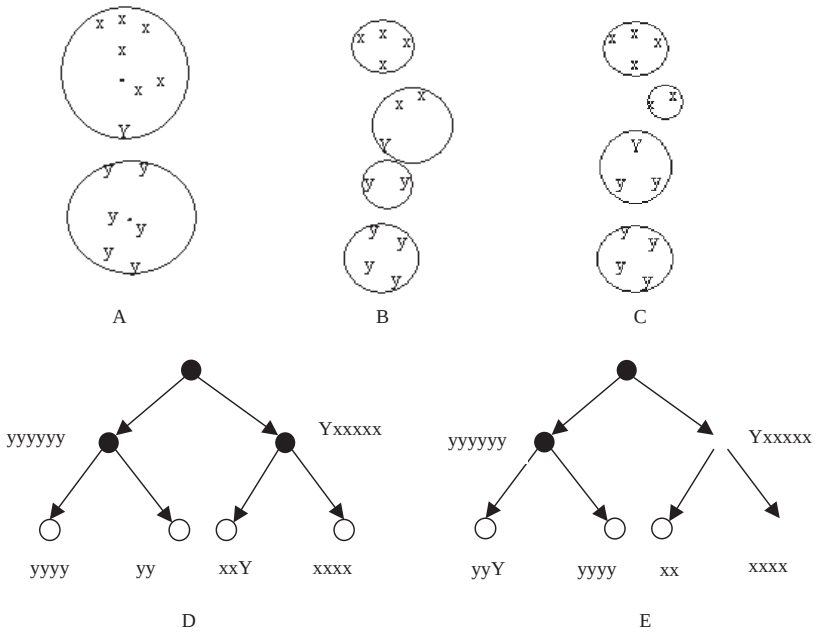

Fig. 7. Comparison of conventional distribution and KLD distribution on sample data. B is the conventional distribution and D is the corresponding clustering result. C is the KLD distribution and E is the corresponding clustering result.

their publicly available version of SOTA. For the KLD, data associated with a parent node will be distributed among its children leaves and also among its neighboring leaves. The following is the KLD strategy:

- For a selected node, its K-level up ancestor node is determined.
- The subtree rooted by the ancestor node is determined.
- Data associated with the selected node is distributed among all leaves of the subtree.

For example, Figure 8 shows the scope of $K = 1$. First, the data associated with node M are distributed among the newly created leaves. For $K = 1$, the immediate ancestor of M is determined to be A. The data associated with node M are distributed among leaves B, C, D and E of the subtree rooted by A. For each data, the winning leaf will be determined among B, C, D and E using Equation (1). Note that if $K = 0$, data of M will be distributed between leaves B and C. This latter approach is identical to the conventional approach. Fig. 7C shows that with the KLD scheme, the misclustered *y* will have the opportunity to be reevaluated when the self-organizing neural network expands to the next level (Fig. 7C), which makes the final clustering result (leaf level) accurate (Fig. 7E). By applying to three benchmark datasets, we show that the KLD mechanism improves the clustering result of DGSOT greatly (Luo *et al*., 2003). The value of K is dependent on each dataset. The KLD does not change the time complexity

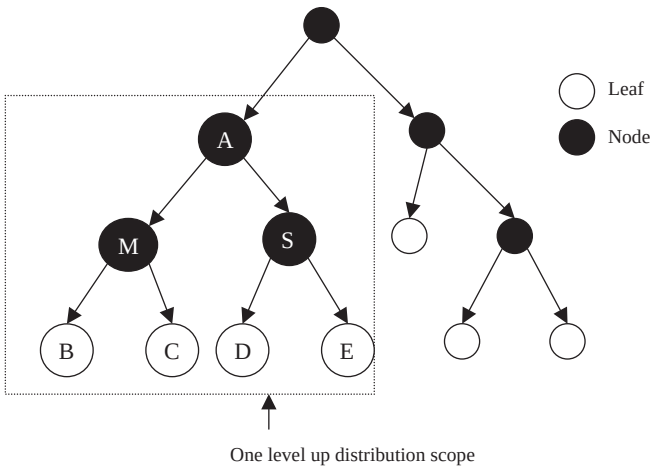

Fig. 8. One level up distribution scope ($K = 1$).

of the DGSOT because it only multiplies a constant, namely 2^K, to the original time.

3 RESULTS

3.1 Yeast cell cycle data

We applied the DGSOT to the yeast gene cell cycle gene expression profiles published by Cho *et al*. (1998). They used Affymetrix oligonucleotide microarrays to test the expression of more than 6000 genes of yeast *Saccharomyces cerevisiae*. The gene expressions of yeast cell were examined in 17 time

points at 10 min intervals, which covered over two cell cycles. In our analysis, we used 15 time points; the 90 and 100 min points were excluded due to the less efficient labeling of the mRNA during the original chip hybridizations (Tavazoie *et al*., 1999). The data were preprocessed as described by Tavazoie *et al*. (1999). First, we normalized the data to have unit variance and zero mean (Z-score normalization). Second, we selected 3000 genes with the most relative variation (standard variance over the mean) in expression level across 15 time points for our clustering. The parameters of DGSOT are shown in our Supplement Table 1. The Euclidean distance was used to calculate the similarity in DGSOT.

3.2 Clustering results

The results of clustering are shown in Figure 9. Figure 9B is the full DGSOT tree. The 3000 gene expression profiles were first clustered into six high-level clusters (Supplement Figure 1A). Figure 9C is an enlarged picture of part of the tree that is indicated by a red bar in Figure 9B. At the tree leaf of Figure 9C, the name and function categories of a gene associated with the leaf of the DGSOT tree are shown. Supplement Figure 1B–G shows 25, second-level subclusters of the 6 highest-level clusters, respectively. The number of the subcluster is the number of its upper level cluster plus a digit to indicate the sequence. For example, cluster 1 has two subclusters denoted by 11 and 12, respectively. The subclusters (Supplement Figures 1B, C and E) of clusters 1, 2 and 4 show the same patterns as their upper level clusters. However, the subclusters (Supplement Figs 1D, F and G) of cluster 4 show different patterns.

3.2.1 Enrichment of functionality categories in subclusters We tested the functional significance in 25, second level subclusters. The functional categories of yeast open reading frames (ORFs) were downloaded from the yeast genome database of MIPS (Mewes *et al*., 2000). For each subcluster we calculated a P-value, which is the probability of observing the frequency of genes in a particular functional category in a certain cluster, using the cumulative hypergeometric probability distribution (Tavazoie *et al*., 1999). We compared the functionality enrichment of these 25 clusters with the results obtained by Tavazoie *et al*. (1999). As shown in the Supplement Table 2, all seven clusters found by Tavazoie *et al*. matched a corresponding cluster in our analysis. The degree of enrichment in the clusters identified by DGSOT is higher than or equal to the result using the K-means algorithm and there were more functionality enrichment clusters identified by the DGSOT. For example, cluster 21 is enriched in 'cell rescue, defense and virulence' function genes; cluster 63 is enriched in 'transcription' function genes. In the supplement Figure 2, we show 43 of the most enriched functionality categories in these 25 subclusters. Some functional categories dominate in a few specific clusters. For example, categories 'ribosome biogenesis' and 'cytoplasm' genes possess significant enrichment in clusters 11 and 12. Note that many genes belong to both categories. The category 'DNA synthesis and replication' dominates in cluster 41.

3.3 Comparison with SOTA

The DGSOT was compared with the SOTA algorithm by submitting the normalized 3000 yeast gene cell cycle microarray expression profiles to the SOTA website (Herrero *et al*., 2003). The parameters of the SOTA were set the same as for the DGSOT with the exception that the learning rate of the parent node was set to 0.05. The normal Euclidean distance was chosen for SOTA. Figure 9A shows the whole hierarchical tree redrawn from the SOTA result (because the output tree of SOTA is too large to show). Figure 10 shows the hierarchical structure when both DGSOT and SOTA have 25 leaf nodes. In Figure 10, the number in the SOTA tree is the number of nodes assigned by SOTA in the order of tree growth. Each 25 leaf nodes represents a Voronoi partition of the original dataset by DGOST and SOTA, respectively. Table 2 summarizes the comparison of SOTA and DGSOT in terms of the functionality enrichment in the 25 clusters. The clustering result of DGSOT is statistically more significant (higher P-value) than the clustering result of SOTA. Furthermore, the proper hierarchical structure of the DGSOT makes the clustering result more reasonable for large clusters. For example, the 'ribosomal proteins' and 'organization of cytoplasm' functional clusters enrich in nodes 39 and 26 using SOTA. However, nodes 39 and 26 belong to two different first level branches (Fig. 10). In contrast, the same functionality enriches in the nodes 11 and 12 using DGSOT, which have the same parent in the first level (Fig. 10).

4 DISCUSSION AND CONCLUSIONS

The rapid development of microarray technology provides us with a processing and global view of the gene expression beyond that of an individual gene. Clustering algorithms enable us to detect a group of genes expressed in different tissues or cells over different developmental stages. The fixed topology of the traditional tree-structure based self-organizing neural network will affect the accuracy of the final clustering result. In this paper, we introduced a new tree-structure self-organizing neural network, called DGSOT, for hierarchical clustering. The DGSOT algorithm combines the horizontal growth and vertical growth to construct a mutlifurcating hierarchical tree from top to bottom to cluster the data. In each vertical growth, the hierarchical tree expand one more hierarchical level. In addition, in each hierarchical level the DGSOT employs a horizontal growth step to optimize the number of the subclusters. If the number or the size of the clusters and subclusters of the dataset are not even, e.g. there is a very large cluster in dataset such as the 'ribosomal proteins' and 'organization of cytoplasm' functional clusters of yeast gene,

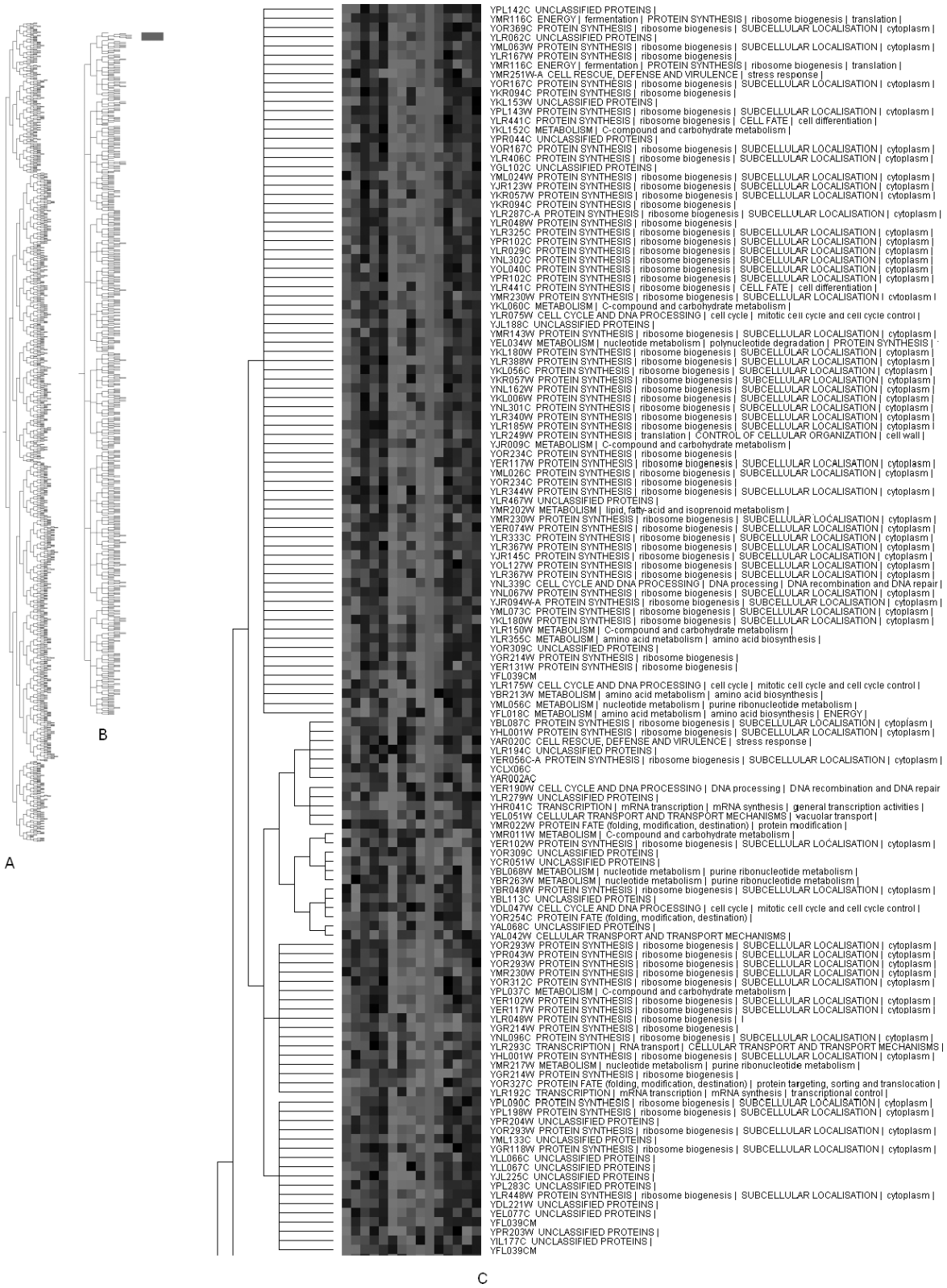

Fig. 9. Hierarchical clustering tree of 3000 yeast cell cycle gene microarray expression data constructed by SOTA and DGSOT. (**A**) The whole tree of SOTA. (**B**) The whole tree of DGSOT. (**C**) An enlarged picture represents part of the DGSOT tree that is indicated by the red bar.

Table 2. Biological validation of the 25 nodes corresponding in DGSOT and SOTA

Cluster number		Number of ORFs		MIPS functional category	ORFs within functional category		P-value ($-\log_{10}$)	
DGSOT	SOTA	DGSOT	SOTA		DGSOT	SOTA	DGSOT	SOTA
11	39	250	184	Ribosomal proteins	122	50	124	32
				Organization of cytoplasm	141	67	87	25
				Translation	10	6	4	2
12	26	182	142	Ribosomal proteins	29	49	12	37
				Organization of cytoplasm	43	64	9	30
21		162		Cell rescue, defense and virulence	18		4	
				Stress response	13		3	
23	44	162	276	C-compound and carbohydrate metabolism	25	36	4	4
				C-compound and carbohydrate utilization	18	25	4	4
				Stress response	12	15	3	2
25		72		C-compound and carbohydrate metabolism	15		4	
31	25	110	128	Cell cycle and DNA processing	22	26	3	4
				Cell cycle	18	22	3	4
				Mitotic cell cycle and cell cycle control	14	15	3	2
				Fungal cell differentiation	11	17	2	3
				Budding, cell polarity and filament formation	8	8	2	2
32	47	110	93	Amino acid metabolism	18	12	8	4
				Amino acid biosynthesis	11	8	5	4
				Nitrogen and sulfur metabolism	9	6	6	3
				Nitrogen and sulfur utilization	6	4	4	3
				mRNA transcription	15	18	2	3
				mRNA synthesis	14	17	2	4
				Transcriptional control	14	17	3	5
41	38	208	205	Deoxyribonucleotide metabolism	5	2	5	1
				Cell cycle and DNA processing	70	51	21	10
				DNA processing	37	25	14	6
				DNA synthesis and replication	25	17	16	8
				DNA recombination and DNA repair	18	13	6	3
				Cell cycle	41	33	9	5
				Mitotic cell cycle and Cell cycle control	36	28	9	5
				Budding, cell polarity and filament formation	15	10	3	1.5
				Fungal cell differentiation	21	21	2	2
				Nucleus	48	39	5	3
				Centrosome	6	5	3	3
43	45	132	148	Cell cycle and DNA processing	22	34	2	6
				Mitotic cell cycle and cell cycle control	18	21	4	4
				Cell cycle checkpoints	6	7	5	6
				Centrosome	6	5	4	3
51	29	127	218	Mitochondrion	24	32	7	6
				Energy	15	23	4	5
52		133		Amino acid metabolism	14		4	
55	30	110	124	Respiration	9	7	5	3
				Mitochondrion	26	23	9	6
63	35	94	108	Transcription	26	24	5	3
				rRNA transcription	10	9	5	4
				tRNA transcription	8	5	5	2
				Nucleus	23	24	3	3
				Centrosome		7		5
64	40	70	88	Splicing	6	4	3	1

The genes in each cluster have been mapped to the functional category of MIPS database. Only significantly enriched functional categories are shown.

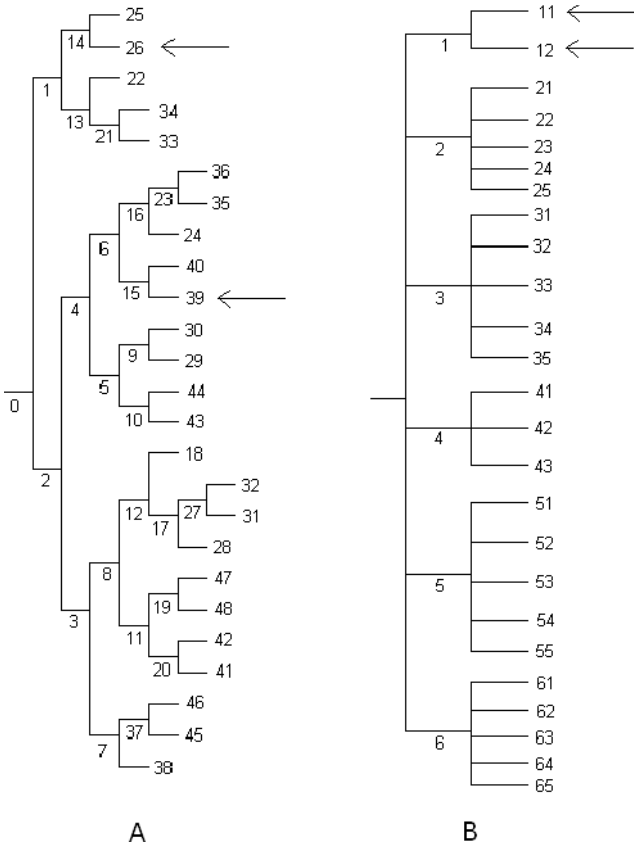

Fig. 10. The hierarchical structure of SOTA (**A**) and DGSOT (**B**) when each has 25 leaves. (The number in the SOTA tree is the number of the node assigned by SOTA in the order of tree growth. The number in the DGSOT tree is the label of its upper level cluster plus a digit to indicate the sequence.)

the combination of horizontal growth and vertical growth let the DGSOT algorithm to find the proper hierarchical structure of the underlying dataset, then get more reasonable final clustering result.

In DOSOT, each node in the hierarchy represents the centroid of all data associated with its leaf descendants. The pattern kept by a node can be treated as the integration of patterns kept by lower-level nodes. The upper-level pattern may not be the same as the lower-level patterns if the data associated with the subtree are diverse. The difference between the upper-level pattern and the lower-level patterns may also be used to determine the growth of the hierarchy.

In addition, based on the observation that the edges of MST among the centroids of clusters are even for a validated clustering in a certain hierarchical level, we developed a new cluster validation criterion for finding the proper number of clusters during the hierarchical clustering of DGSOT algorithm. The new cluster validation criterion is based on the geometric property of the Voronoi partition of the dataset. This criterion uses the MST concept from graph theory to find a proper Voronoi partition of the dataset. The new criterion uses the centroid of each cluster to calculate the intercluster properties, which is computationally inexpensive for large datasets.

The harmonization of the vertical growth and the horizontal growth is important in the DGSOT algorithm to find the proper structure of the underlying dataset. The balance of the vertical growth and horizontal growth is controlled by the clustering validation criterion, which determines the number of horizontal growth. Therefore, the cluster validation criterion is critical in the DGSOT algorithm. In DGSOT, the cluster validation criterion is used to determine the proper number of clusters in each hierarchical level rather than in the whole dataset. For dataset containing even number of clusters along with even size, a proper cluster validation criterion must not allow the horizontal growth to continue forever without the vertical growth. On the other hand, for dataset containing uneven number or uneven size of clusters, a proper cluster validation criterion must be able to detect that uneven behavior and find the best representation in each hierarchical level. The CS criterion we present exactly fulfills this requirement.

To improve the clustering accuracy we proposed a KLD mechanism. The KLD scheme increases the scope in hierarchy for data distribution, which will give the data misclustered at an early stage a chance to be reevaluated. The DGSOT algorithm combined with KLD mechanism overcomes the drawbacks of the traditional neural tree based on hierarchical clustering algorithms.

Like the SOTA, the DGSOT algorithm can be terminated at any hierarchical stage by setting different vertical growing stop threshold. This will save computation time when the user only needs the main pattern of a large dataset. The clustering result of the DGSOT is easily displayed as a dendrogram for visualization. For large datasets, the DGSOT algorithm can display high level main patterns of the dataset only if visualization of the whole hierarchical structure is difficult.

We applied the DGSOT algorithm to cluster 3000 yeast gene cell cycle microarray expression data. We compared the clustering results of DGSOT and SOTA when both have 25 clusters. For these low-level resolution clustering results, we observed that the DGSOT, with the multi-partition strategy, can successfully establish the hierarchical structure of these data, and then get more reasonable results than obtained by SOTA, which is a pure bipartitions method. This is more drastic if the structure and substructure of the data contains an uneven number of clusters and subclusters or contains dramatically different size of clusters and subclusters. Furthermore, the biological functionality enrichment in the clustering result of DGSOT is considerably higher than the clustering result of SOTA and the K-means. We believe that DGSOT can be a robust and accurate framework for the study of patterns among large sets of gene microarray expression data.

5 ACKNOWLEDGEMENTS

We thank Dr Saeed Tavazoie for his help. We also thank the three anonymous reviewers for their helpful suggestion. This study was supported in part by gift from Sun and the National Science Foundation grant NGS-0103709, and by The US Department of Energy under the Genomes To Life and Microbial Genome Programs of the Office of Biological and Environmental Research, Office of Science. Oak Ridge National Laboratory is managed by University of Tennessee-Battelle LLC for the Department of Energy under contract DE-AC05-00OR22725.

REFERENCES

Ben-Dor,A., Shamir,R. and Yakhini,Z. (1999) Clustering gene expression patterns. *J. Comput. Biol.*, **6**, 281–297.

Cho,R.J., Campbell,M.J., Winzeler,E.A., Steinmetz,L., Conway,A., Wodicka,L., Wolfsberg,T.G., Gabrielian,A.E., Landsman,D., Lockhart,D.J. and Davis,R.W. (1998) A genome wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, **2**, 65–73.

Cuevas,A., Febrero,M. and Fraiman,R. (2000) Estimating the number of clusters. *Can. J. Stat.*, **28**, 367–382.

Dopazo,J. and Carazo,J.M. (1997) Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. *J. Mol. Evol.*, **44**, 226–233.

Eisen,M.B., Spellman,P.T., Brown,P.O. and Botstein,D. (1998) Cluster analysis and display of genomewide expression patterns. *Proc. Natl Acad. Sci., USA*, **95**, 14863–14868.

Fritzke,B. (1994) Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Net.*, **7**, 1141–1160.

Gower,J.C. and Ross,G.J.S. (1969) Minimum spanning tree and single linkage clustering analysis. *Appl. Stat.*, **18**, 54–64.

Halkidi,M., Batistakis,Y. and Vazirgiannis,M. (2001) On clustering validation techniques. *J. Intel. Inform. Sys.*, **17**, 107–145.

Halkidi,M., Batistakis,Y. and Vazirgiannis,M. (2002) Clustering validity checking methods: part II. *SIGMOD record*, **31**, 19–27.

Hardy,A. (1996) On the number of clusters. *Comput. Stat. Data Anal.*, **23**, 83–96.

Herrero,J., Valencia,A. and Dopazo,J. (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, **17**, 126–136.

Herrero,J., Shahrour,F.A., Uriarte,R.D., Mateos,A., Vaquerizas,J.M., Santoyo,J. and Dopazo,J. (2003) GEPAS: a web-based resource for microarray gene expression data analysis. *Nucleic Acids Res.*, **31**, 3461–3467.

Khan,L. and Luo,F. (2002) Ontology construction for information selection, *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2002)*, IEEE Computer Society, Washington, D.C. pp. 122–127.

Kohonen,T. (1997) *Self-Organizing Maps*, 2nd edn. Springer-Verlag, Berlin, Germany.

Luo,F., Khan,L., Bastani,F. and Yen,I.L. (2003) A dynamical growing self-organizing tree (DGSOT). *Technical Report*, University of Texas, at Dallas.

Mewes,H.W., Frishman,D., Gruber,C., Geier,B., Haase,D., Kaps,A., Lemcke,K., Mannhaupt,G., Pfeiffer,F., Schuller,C., Stocker,S. and Weil,B. (2000) MIPS: a database for genomes and protein sequences. *Nucleic Acids Res.*, **28**, 37–40.

Pal,N.R. and Biswas,J. (1997) Cluster validation using graph theoretic concepts. *Pattern Recog.*, **30**, 847–857.

Rezaee,R., Lelieveldt,B.P.F. and Reiber,J.H.C. (1998) A new cluster validity index for the fuzzy *c*-mean. *Pattern Recog. Lett.*, **19**, 237–246.

Rousseeuw,P.J. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, **20**, 53–65.

Smet,F., Mathys,J., Marchal,K., Thijs,G. and Moreau,Y. (2002) Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, **18**, 735–746.

Tamayo,P., Slonim,D., Mesirov,J., Zhu,Q., Kitareewan,S., Dmitrovsky,E., Lander,E.S. and Golub,T.R. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl Acad. Sci., USA*, **96**, 2907–2912.

Tavazoie,S., Hughes,J.D., Campbell,M.J., Cho,R.J. and Church,G.M. (1999) Systematic determination of genetic network architecture. *Nat. Genet.*, **22**, 281–285.

Tibshirani,R., Walther,G. and Hastie,T. (2001) Estimating the number of clusters in a dataset via the gap statistic. *J. R. Stat. Soc. Ser. B*, **63**, 411–423.

Vilo,J., Brazma,A., Jonssen,I., Robinson,A. and Ukkonen,E., (2000) Mining for putative regulatory elements in the yeast genome using gene expression data. *Proc. Int. Conf. Intell. Sys. Mol. Biol.*, **8**, 384–394.

Wen,X., Fuhrman,S., Michaels,G.S., Carr,D.B., Smith,S., Barker,J.L. and Somogyi,R. (1998) Largescale temporal gene expression mapping of central nervous system development. *Proc. Natl Acad. Sci., USA*, **95**, 334–339.

Xu,Y., Olmam,V. and Xu,D. (2002) Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, **18**, 536–545.

Yeung,K.Y., Haynor,D.R. and Ruzzo,W.L. (2001) Validating clustering for gene expression data. *Bioinformatics*, **17**, 309–318.