Minimum Spanning Trees for Gene Expression Data Clustering*

Ying Xu[1] **Victor Olman**[1] **Dong Xu**[1]
xyn@ornl.gov vo4@ornl.gov xud@ornl.gov

[1] Computational Protein Structure Group, Life Sciences Division, Oak Ridge National Laboratory, 1060 Commerce Park Drive, Oak Ridge, TN 27831-6480, USA.

Abstract

This paper describes a new framework for microarray gene-expression data clustering. The foundation of this framework is a minimum spanning tree (MST) representation of a set of multi-dimensional gene expression data. A key property of this representation is that each cluster of the expression data corresponds to one subtree of the MST, which rigorously converts a multi-dimensional clustering problem to a tree partitioning problem. We have demonstrated that though the inter-data relationship is greatly simplified in the MST representation, no essential information is lost for the purpose of clustering. Two key advantages in representing a set of multi-dimensional data as an MST are: (1) the simple structure of a tree facilitates efficient implementations of rigorous clustering algorithms, which otherwise are highly computationally challenging; and (2) as an MST-based clustering does not depend on detailed geometric shape of a cluster, it can overcome many of the problems faced by classical clustering algorithms. Based on the MST representation, we have developed a number of rigorous and efficient clustering algorithms, including two with guaranteed global optimality. We have implemented these algorithms as a computer software EXCAVATOR. To demonstrate its effectiveness, we have tested it on two data sets, i.e., expression data from yeast *Saccharomyces cerevisiae*, and *Arabidopsis* expression data in response to chitin elicitation.

Keywords: microarray gene expression data, clustering, minimum spanning trees

1 Introduction

As probably the most explosively expanding tool for genome analysis, microchips of gene expression have made it possible to simultaneously monitor the expression levels of tens of thousands of genes under different experimental conditions. This provides a powerful tool for studying how genes collectively react to changes in their environments, providing hints about the structures of the involved gene networks. One of the basic problems in interpreting the observed expression data is to cluster genes with correlated expression patterns over some time series and/or under different conditions.

A number of computer algorithms/software have been developed for clustering gene expression patterns. The most prevalent approaches include (i) hierarchical clustering [1, 2], (ii) K-means clustering [3], and (iii) clustering through self-organizing maps (SOMs) [4]. While all these approaches have clearly demonstrated their usefulness in applications [5], some basic problems remain – (1) none of these algorithms can, in general, rigorously guarantee to produce a globally optimal clustering for any non-trivial objective function; (2) both K-means and SOM heavily depend on the "regularity" of the geometric shape of cluster boundaries; they generally do not work well when the clusters cannot be contained in some non-overlapping convex sets – just to name a few.

We have developed a framework for representing a set of multi-dimensional data as a *minimum spanning tree* (MST), a concept from the graph theory. A tree is a simple structure for representing

*Correspondence author: Ying Xu. This work is supported by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC.

binary relationships, and any connected component of a tree is called a *subtree*. Through this MST representation, we can convert a multi-dimensional clustering problem to a tree partitioning problem, i.e., finding a particular set of tree edges ("long" edges from either local or global point of view) and then cutting them. Representing a set of multi-dimensional data points as a simple tree structure will clearly lose some of the inter-data relationship. However we have rigorously demonstrated that no essential information is lost for the purpose of clustering. This is achieved through a rigorous proof that each cluster corresponds to one subtree, which does not overlap the representing subtree of any other cluster. Hence a clustering problem is equivalent to a problem of identifying these subtrees through solving a tree partitioning problem. Because of the simplicity of a tree structure, many tree-based optimization problems can be solved efficiently in a similar but generalized fashion to that of their corresponding 1D problems. We will describe, in the following sections, a number of efficient and rigorous tree-based clustering algorithms, some of which have guaranteed global optimality.

In addition to being able to facilitate efficient clustering algorithms, an MST representation also allows us to deal with clustering problems that classical clustering algorithms have problems with. As these algorithms rely on either the idea of grouping data around some "centers" or the idea of separating data points using some regular geometric curve like a hyperplane, they generally do not work well when the boundaries of the clusters are very complex. An MST, on the other hand, is quite invariant to detailed geometric changes in the boundaries of clusters. For example, the MST representation will be quite stable under a large class of geometric transformations to the shapes of the cluster boundaries (detailed discussion will be provided elsewhere). This implies that the shape complexity of a cluster has very little effect on the performance of our MST-based clustering algorithms.

MSTs have been used for data classification in the field of pattern recognition [6] and image processing [7, 8, 9]. We have also seen some limited applications in biological data analysis [10]. One popular form of these MST applications is called the *single-linkage cluster analysis* [11, 12]. Our study on these methods has led us to believe that all these applications have used the MSTs in some heuristic ways, e.g., cutting long edges to separate clusters, without fully exploring their power and understanding their rich properties related to clustering. In this paper, we will provide in-depth studies for MST-based clustering. Our major contributions include a rigorous formulation for general clustering problems, the discovery of new relationship between MSTs and clustering, and novel algorithms for MST-based clustering.

We have implemented the MST-based clustering algorithms, along with the MST representation, as a computer program EXCAVATOR (EXpression data Clustering Analysis and VisualizATiOn Resource). We have tested the program on a number of data sets.

2 Spanning Tree Representation of a Data Set

We will use a minimum spanning tree to represent a set of expression data and their significant inter-data relationships to facilitate fast rigorous clustering algorithms. Let $D = \{d_i\}$ be a set of expression data with each $d_i = (e_i^1, ..., e_i^t)$ 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 $d_i, d_j \in D$ and $i \neq j\}$. Hence $G(D)$ is a complete graph. Each edge $(u, v) \in E$ has a weight that represents the distance (or dissimilarity), $\rho(u, v)$, between u and v, which could be defined as the Euclidean distance, the correlational coefficient, or some other distance measures.

A *spanning tree* T of a (connected) weighted graph $G(D)$ is a connected subgraph of $G(D)$ such that (i) T contains every vertex of $G(D)$, and (ii) T does not contain any cycle. A *minimum spanning tree* is a spanning tree with the minimum total distance. A minimum spanning tree of a weighted graph can be found by a *greedy* method, as illustrated by the following strategy used in the classical Kruskal's algorithm (see page 222 in [12]). A simple implementation of the Kruskal's algorithm [13]

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.

runs in $O(\|E\| \log(\|E\|))$ time, where $\|\cdot\|$ represents the number of elements in a set. Figure 1 shows an example of a minimum spanning tree of a 2D data set, consisting of four "natural" clusters.

By examining examples like Figure 1, we have observed that data points of the same cluster are connected with each other by short tree edges (without data points from other clusters in the middle) while long tree edges link clusters together. We found this is generally the case with an MST representation of any multi-cluster data set. To rigorously prove this, we need a formal definition of a cluster. So what constitutes a cluster in a data set? Here we provide a **necessary** condition for a subset of a set to be a *cluster*. Let D be a data set and ρ represent the distance between two data points of D.

> *$C \subseteq D$ forms a cluster in D only if for any arbitrary partition $C = C_1 \cup C_2$, the closest data point d to C_1, $d \in D - C_1$, is from C_2. Formally, this can be written as*
>
> $$\arg \min_{d \in D - C_1} \{\min \{\rho(d, c) | c \in C_1\}\} \in C_2, \tag{1}$$

where $D - C$ represents the subset of D by removing all points of C. We call this the *separability condition* of a cluster. In essence, by this definition, we are trying to capture our intuition about a cluster; that is distances between neighbors within a cluster should be smaller than any inter-cluster distances. Clearly, each of the four "natural" clusters in Figure 1 satisfies this necessary condition. So does the whole data set. However the subset formed by the cluster in the up-left corner plus any proper subset of the cluster in the up-right corner does not form a cluster.

Now we can rigorously prove that any cluster, C, corresponds exactly to one subtree of its MST representation. That is

> *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.*

This statement can be proved rigorously. We only give a sketch of the proof here. Let's assume that the statement is incorrect. Hence there exists a point a in path P, which does not belong to C (see Figure 2). Without loss of generality, we assume that a is right next to c_1 on P so that (c_1, a) is an edge in P. We define a data set A as follows. Initially $A = \{c_1\}$. We then repeatedly expand A 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 (1) 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 $\rho(c_1, a)$ (see Figure 2(b)). We know that

Figure 2: (a) A path connecting two vertices c_1 and c_2 of the same cluster C (C's boundary is given by the dashed line) with one vertex a from a different cluster. (b) A schematic of the result of the *expansion* operation.

at least one edge of P' is not in the current minimum spanning tree. For the simplicity of discussion, we assume that exactly one edge, e, of P' is not in the current minimum spanning tree (the case with multiple such edges can be reduced to the case with only one edge). So $P \cup P'$ contains a cycle with one edge of P' not in the minimum spanning tree. By removing edge (c_1, a) and adding e, we get another spanning tree with smaller total distance. This contradicts the fact that a minimum spanning tree has the minimum total distance among all spanning trees. By having this contradiction, we have proved the statement.

The above statement implies that clustering (of multi-dimensional data) can be achieved through tree partitioning. So to cluster, all we have to do is to find the *right* set of edges of the MST representation of the data set and cut them; the connected subtrees will give us the desired clusters.

3 MST-based Clustering Algorithms

Apparently, different clustering problems may need different objective functions, in order to achieve the best clustering results. In this section, we describe three objective functions and their corresponding clustering algorithms. All algorithms presented here are for partitioning a tree into K subtrees, for a specified integer $K > 0$.

3.1 Clustering through removing long MST-edges

One simple objective function is to partition an MST into K subtrees so that the total edge-distance of all the K subtrees is minimized. This objective function intends to capture the intuition that two data points with a short edge-distance should belong to the same cluster (subtree) and data points with a long edge-distance should belong to different clusters and hence be cut. It is not hard to rigorously prove that by finding the $K-1$ longest MST-edges and cutting them, we get a K-clustering that achieves the global optimality of the above objective function. This simple algorithm works very well as long as the inter-cluster (subtree) edge-distances are clearly larger than the intra-cluster edge-distances.

To determine automatically how many clusters there should be, the algorithm examines the optimal K-clustering for all $K = 1, 2, ...,$ up to some large number to see how much improvement we can get as K goes up. Typically after K reaches the "correct" number (of clusters), the quality improvement levels off, as we can see in Figure 4(a). By locating the transition point, our program can automatically choose the number of the clusters for the user.

3.2 An iterative clustering algorithm

We now describe another clustering algorithm that attempts to partition the minimum spanning tree T into K subtrees, $\{T_i\}_{i=1}^{K}$, to optimize a more general objective function than the previous one:

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

that is to optimize the K-clustering so that the total distance between the *center* of each cluster and its data points is minimized – this is a typical objective function for data clustering. The center of a cluster is the position which satisfies the condition that the sum of the distances between the position and all the data points in the cluster is minimized.

Our iterative algorithm starts with an arbitrary K-partitioning of the tree (selecting $K-1$ edges and removing them gives a K-partitioning). Then it repeatedly does the following operation until the process converges: *For each pair of adjacent clusters (connected by a tree edge), 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 (2).* Our experience with this iterative algorithm indicates that the algorithm converges to a local minimum very quickly.

3.3 A globally optimal clustering algorithm

We now present an algorithm that finds the globally optimal solution of the clustering problem defined as follows. We use a slightly different objective function than the objective function (2). In the previous one, we want to group data points around the center of each cluster (to be clustered). Here we want to group data points around the "best" representatives from our data set. The representatives are not pre-selected but rather they are the results of the optimization process, i.e., our optimization algorithm attempts to partition the tree into K subtrees and simultaneously to select K representatives in such way to optimize the objective function (3). More formally, for a given minimum spanning tree T, we want to partition T into K subtrees, $\{T_1, ..., T_K\}$, and to find a set of data points $d_1, ..., d_K \in D$ such that the following objective function is minimized:

$$\sum_{i=1}^{K} \sum_{d \in T_i} \rho(d, d_i). \tag{3}$$

where $\rho()$ is the distance function used. The rationale for using a "representative" rather than the "center" is that a center may not belong to, or even close to, the data points of its cluster when the shape of the cluster boundary is not convex, which may result in biologically less meaningful clustering results. The representative-based scheme provides an alternative when center-based clustering does not generate desired results. A good property of the representative-based objective function is that it facilitates an efficient global optimization algorithm.

The very basic idea of our algorithm can be explained as follows. It first converts the minimum spanning tree into a *rooted* tree [12] by arbitrarily selecting a tree vertex as the root. Now the *parent-child* relationship is defined among all tree vertices. At each tree vertex v, we define the following: *$S(v, k, d)$ is defined to be the minimum value of the objective function (3.3) on the subtree rooted at vertex v, under the constraint that the subtree is partitioned into k subtrees and the representative of the subtree rooted at v is d.* By definition, the following gives the global minimum of objective function (3.3):

$$\min_{d \in D} S(\text{root}, K, d). \tag{4}$$

Our algorithm uses a dynamic programming (DP) approach [12] to calculate the $S()$ values at each tree vertex v, based on the $S()$ values of v's children in the rooted MST. The core of the algorithm is a set of DP recurrences relating these $S()$ values. The *boundary conditions* of this dynamic programming system are given as follows: If a tree vertex v does not have any child, then

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

For each v with children, $S()$ of v is calculated as follows

$$S(v,k,d) = \min_{X \subseteq C_v} \min_{\sum_{i=1}^{\|C_v\|} k_i = k + \|X\| - 1, k_i > 0} \left(\sum_{v_j \in C_v - X} S(v_j, k_j, \overline{d}) + \sum_{v_j \in X} S(v_j, k_j, d) + \rho(v,d) \right), \tag{6}$$

where

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

and C_v represents the set of all children of vertex v. Our algorithm calculates the $S(v,k,d)$ values for all combinations of $v \in T$, $k \in [1,K]$, and $d \in D$.

The correctness of these DP recurrences can be proved based on the observation that $S(v,k,d)$ can be decomposed as the sum of some combination of its children's $S()$ values and that the above DP recurrences covers all possible such combinations. We omit the detailed proof.

The computational time of this algorithm can be estimated as follows. It is not hard to see that for each tree vertex v, computing its DP recurrences takes

$$O\left(2^{\|C_v\|} \begin{pmatrix} K + \|C_v\| - 1 \\ \|C_v\| - 1 \end{pmatrix} \|C_v\| \right)$$

time, where $\begin{pmatrix} X \\ Y \end{pmatrix}$ denotes the number of possible ways of selecting Y elements out of X elements. Hence the total time, $\mathcal{T}$, for computing all the DP recurrences for the whole tree T is

$$\mathcal{T} \leq O\left(\sum_{v \in T} 2^{\|C_v\|} \begin{pmatrix} K + \|C_v\| - 1 \\ \|C_v\| - 1 \end{pmatrix} \|C_v\| \right).$$

Since

$$\begin{pmatrix} K + \|C_v\| - 1 \\ \|C_v\| - 1 \end{pmatrix} \leq (K+1)^{s-1},$$

we have

$$\mathcal{T} \leq 2^s K^s \sum_v \|C_v\|,$$

where s is the maximum number of children of any tree vertex. Since $\sum_{v \in T} \|C_v\| = n - 1$, we have shown that it takes $O(n(2K)^s)$ time to compute all the $S()$ values, where n is the number of data points in our data set and K is the maximum number of clusters we want to consider. To get the actual clustering that achieves the global minimum value, we need some simple bookkeeping to trace back which tree edges are cut. This can be done within the computational time needed for calculating the $S()$ values. We omit further discussions.

This algorithm runs in exponential time only in the maximum number of children, s, of a tree vertex. To get a sense about how large s could be for a typical application, we have done a number of simulations to estimate s. In the simulation, we have randomly generated a set of 60-dimensional (60 is chosen arbitrarily) data points, and constructed an MST representation of the set. Then we count

(a) (b)

Figure 3: (a) The distribution of the number of children of the MST representing a data set with 1000 random data points in 60-dimensional Euclidean space. (b) The maximum number of children versus the total number of data points ranging from 50 to 9,000.

the number of children of each vertex in this MST. Figure 3 summarizes these counts. This study shows that this global optimization algorithm runs efficiently for a typical clustering problem with a few hundred data points consisting of a dozen or so clusters.

Note that our algorithm finds the optimal k-clustering for all k's simultaneously, $k \leq K$, for some pre-selected K. For a particular application, if we set K to, say, 30 or to certain percentage of the total number of vertices, we will get the optimal objective values for any $k = 1, 2, ..., K$. By comparing these values, we can automatically select the number of clusters that is most "natural" as we will discuss in Section 4.1.

4 Results

4.1 Key features of EXCAVATOR

The core of the EXCAVATOR program is a set of MST-based clustering algorithms. While detailed description of EXCAVATOR will be discussed elsewhere (manuscript in preparation), we now highlight a few key and unique features of the EXCAVATOR program, in addition to the MST-based rigorous and efficient clustering algorithms that we have described above.

- In EXCAVATOR, we provide a number of different ways of measuring the "distance" between two expression profiles. Based on a user's selection of the distance measure, the program constructs the MST representation of the data set. These distances include (a) Euclidean distance, (b) correlational distance, defined as "1 - the correlational coefficient between two vectors", and (c) Mahalanobis distance.

- For a user-selected objective function and an integer value K, EXCAVATOR calculates the optimal k-clustering for all $k \in [1, K]$, and then compares these values, as shown in Figure 4. Let $Q(k)$ represent the objective value for the optimal k-clustering for our selected objective function. It selects the $k \in [1, K]$ with the highest following value (see Figure 4(b)) as the most "natural" number of clusters:

$$\frac{Q(k-1) - Q(k)}{Q(k) - Q(k+1)}, \tag{7}$$

  where we define $Q(0) = 0$. This function defines a *transition profile* of $Q()$.

- EXCAVATOR allows a user to specify if any genes should (or should not) belong to the same cluster, based on the user's *a priori* knowledge, and finds the optimal clustering that is consistent

with the specified constraints. This feature is implemented as follows. If data points are specified to belong to the same cluster, the algorithm marks the whole MST-path connecting the two points as "cannot be cut" when doing the clustering. So every data points on this path will be assigned to the same cluster of these two points. Similar is done for two genes that should belong to different clusters.

- EXCAVATOR provides different distance measures and different clustering algorithms. For different clustering results, the program has a capability for measuring the similarity of two clustering results, for comparison purposes. Let $\mathcal{D}_1 = \{D_1^1, D_2^1, ..., D_N^1\}$ and $\mathcal{D}_2 = \{D_1^2, D_2^2, ..., D_M^2\}$ be two clusterings of data set D, one with N clusters and the other with M clusters. We define the measure of similarity between these two clusterings as

$$P_{diff}(\mathcal{D}_1, \mathcal{D}_2) \;=\; \sum_{i,j} \frac{\|D_i^1 \bigcap D_j^2\|}{\|D_i^1 \bigcup D_j^2\|} \Big[\, \|D_i^1\| + \|D_j^2\| \Big] \;. \tag{8}$$

It can be shown that P_{diff} has the following upper and lower bounds, $P_{min} \;\leq\; P_{diff}(\mathcal{D}_1, \mathcal{D}_2) \;\leq\; P_{max}$, where

$$P_{min} \;=\; \|D\| \;+\; \min\left(\sum_i \frac{\|D_i^1\|^2}{(M-1)\|D_i^1\| + \|D\|}, \sum_j \frac{\|D_j^2\|^2}{(N-1)\|D_j^2\| + \|D\|)} \right) \;; \tag{9}$$

$$P_{max} \;=\; 2\|D\| \;. \tag{10}$$

The following quantity, which ranges from 0 to 1, gives a good measurement on the similarity between the two clustering results $\mathcal{D}_1$ and $\mathcal{D}_2$,

$$\frac{P_{diff}(\mathcal{D}_1, \mathcal{D}_2) - P_{min}}{P_{max} - P_{min}} \;. \tag{11}$$

The value is 1 if and only if the two partition results are the same. The closer the value is to 0, the more dissimilar the two partition results are.

4.2 Application results

We now outline the application results to two data sets.

4.2.1 Yeast data

Our first application is on a set of gene expression data in the budding yeast *Saccharomyces cerevisiae* [1], with each gene having 79 data points (or 79 dimensions). We selected four clusters (68 genes in total) determined in the paper [1]. These are (1) protein degradation (cluster C), (2) glycolysis (cluster E), (3) protein synthesis (cluster F), and chromatin (cluster H). Genes in each of these four cluster share similar expression patterns and are annotated to be in the same biological pathway. The goal of this application is compare our clustering results with known cluster information.

For this application, we have applied all three clustering algorithms, using both the Euclidean distance and the correlational distance as the distance measure. The computering time on a PC was less than 1 second for clustering through removing long MST-edges, less than 7 seconds for the iterative algorithm, and less than 20 seconds for the globally optimal algorithm. We have achieved virtually identical clustering results, using any combination of these algorithms and functions. Here we show the clustering result obtained, using our first clustering algorithm with the Euclidean distance as the distance measure. Figure 4 shows how the objective function values improve as the number of clusters

(a) (b)

Figure 4: (a) Objective function values versus the number of clusters. (b) The transition profile value, calculated by Function (7), versus the number of clusters. The dashed line shows the transition profile for a set of random data.

increases. This provides a profile similar to the "Scree Test" [14]. Based on the transition profile in Figure 4(b), the program decides a 4-clustering gives the most "natural" number of clusters for this problem. Figure 5 gives the 4-clustering results, which is 100% in agreement with the annotated results in [1].

Figure 5: Expression profiles and clustering results of the yeast data. Red indicates high expression and green indicates low expression.

4.2.2 *Arabidopsis* data

Our second application is on a set of gene expression data of *Arabidopsis* in response to chitin elicitation [15]. The data was averaged over two experiments. Each gene had 6 data points (collected at 10 min., 30 min., 1 hr., 3 hr., 6 hr., and 24 hr.). 68 genes were selected for clustering, each containing at least one data point with a 3-fold change of expression level by chitin elicitation. We used both the second and third algorithms for this problem. Here we present the clustering results by the third algorithm, with the Euclidean distance as the distance measure. From Figure 6(a), we can see there are two high peaks in the transition profile, indicating that there are at least two levels of clustering, one with four clusters and one further dividing the four clusters into seven clusters. Figure 6(b) shows the clustering results for both the optimal 4-clustering and optimal 7-clustering. Through searching the regulatory regions of these genes, we found that a known *cis*-acting element of chitin-responsive genes, i.e., the W-box hexamer, was over-represented in genes of one of 7 clusters. This suggests that these genes are not only co-expressed, but also co-regulated through the W-box motif [15].

(a) (b)

Figure 6: Clustering results for the *Arabidopsis* data. (a) The transition profile versus the number of clusters; (b) Clustering results for optimal 4-clustering and optimal 7-clustering.

References

[1] Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). *Proc. Natl. Acad. Sci. USA*, **95**:14863–14868.

[2] Wen, X., Fuhrman, S., Carr, G. S. M. D. B., Smith, S., Barker, J. L., and Somogyi, R. (1998). *Proc. Natl. Acad. Sci. USA*, **95**:334–339.

[3] Herwig, R., Poustka, A. J., Mller, C., Bull, C., Lehrach, H., and O'Brien, J. (1999). *Genome Res.*, **9**:1093–1105.

[4] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). *Proc. Natl. Acad. Sci. USA*, **96**:2907–2912.

[5] Sherlock, G. (2000). *Curr. Opin. Immunol.*, **12**:201–205.

[6] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, 1973.

[7] Gonzalez, R. C. and Wintz, P. *Digital Image Processing (second edition).* Addison-Wesley, Reading, MA, 1987.

[8] Xu, Y. and Uberbacher, E. C. (1997). *Image and Vision Computing*, **15**:47–57.

[9] Xu, Y., Olman, V., and Uberbacher, E. C. (1998). *Pattern Recognition Letters*, **19**:1213–1224.

[10] States, D. J., Harris, N. L., and Hunter, L. (1993). *ISMB*, **1**:387–394.

[11] Gower, J. C. and Ross, G. J. S. (1969). *Applied Statistics*, **18**:54–64.

[12] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[13] Kruskal Jr., J. B. (1956). *Proc. Amer. Math. Soc.*, **7**:48–50.

[14] Cattell, R. (1966). *Multivariate Behavioral Research*, **1**:245–276.

[15] Ramonel, K. M., Zhang, B., Ewing, R., Chen, Y., Xu, D., Gollub, J., Stacey, G., and Somerville, S. (2001). Submitted.