

EXCAVATOR: A Computer Program for Efficiently Mining Gene-Expression Data

Dong Xu ^{1,*}, Victor Olman¹, Li Wang¹, and Ying Xu^{1,2}

¹Protein Informatics Group, Life Sciences Division

²Computer Sciences and Mathematics Division

Oak Ridge National Laboratory, Oak Ridge, TN 37831-6480, USA

Keywords: gene expression, gene regulation, biological data mining, clustering, minimum spanning tree, Java GUI.

*To whom correspondence should be addressed. Tel: +1 865 574 8934; Fax: +1 865 241 1965; Email: xud@ornl.gov.

ABSTRACT

Massive gene-expression data are generated using microarrays, and clustering gene-expression data is useful for studying functional relationship among genes in a biological process. We have developed a computer package, EXCAVATOR (<http://compbio.ornl.gov/structure/excavator/>), for clustering gene-expression profiles based on our new framework for representing gene-expression data as a minimum-spanning tree. EXCAVATOR uses a number of rigorous and efficient clustering algorithms. This program has a number of unique features, including capabilities for (i) automatic selection of the most "natural" number of clusters, (ii) data-constrained clustering, (iii) identification of genes with similar expression profiles to pre-specified seed genes, (iv) cluster identification from a noisy background, and (v) automated comparison between different clustering results of the same data set. EXCAVATOR can be run from a Unix/Linux/DOS shell, from a Java interface, or from a Web server. The clustering results can be visualized in colored figures and 2D plots. Moreover, EXCAVATOR provides a wide range of options for data formats, distance measures, clustering algorithms, and methods to choose number of clusters, etc. The effectiveness of EXCAVATOR has been demonstrated on several experimental data sets. Its performance also compares favorably against the popular K-means clustering method in terms of clustering quality and computing time.

INTRODUCTION

As we entered the post genomic era, various high-throughput experimental techniques have been developed to characterize biological systems at the genome scale. Unlike traditional approaches, where a gene/protein is generally studied one at a time, high-throughput approaches can provide a global view of all the genes in a genome in a relatively fast and cost-effective way. Among various high-throughput methods, the microarray technology^{1,2} provides a unique approach to simultaneously observe expression changes of thousands of genes under a set of experimental conditions or over a time course. A challenging issue is to effectively "mine" the enormous amount of gene-

expression data being generated by various research labs worldwide and to extract biological information hidden in the noisy and unstructured expression data. Computational analysis is often carried out using gene-expression profile, i.e., the expression level versus experimental condition or over a time course. Since the genes with the same cellular function or in the same biological pathway often show similar pattern in their expression profiles, one can infer functions of unknown genes based on the known functions of the genes with similar expression patterns. One can also assign new players in a particular pathway through identifying genes with similar expression pattern to the known genes in the pathway. That is why clustering expression profiles is often the first step in biological knowledge discovery from gene-expression data. Clustering gene-expression data can also be used for disease sub-typing³ (i.e., to categorize a disease). In this case, instead of clustering expression profiles, patients with a certain disease (e.g., leukemia) can be clustered into several groups according to their expression patterns in a set of related genes. Then each group of patients can be given customized medicine to maximize the effectiveness of the treatment while minimizing potential side effects. In this case, one can apply similar methods of clustering gene-expression profiles for disease sub-typing.

A number of computer packages have been developed for clustering gene-expression patterns, including GeneCluster⁴ using weighted voting, k-nearest neighbors algorithms, Cleaver⁵ using K-means clustering, and Treeview⁶ using hierarchical clustering. Some computer tools, such as the R Packages for Gene Expression Analysis⁷, J-Express⁸, Genesis⁹, and Stanford's XCluster¹⁰, implement a suite of classical algorithms for clustering such as hierarchical clustering⁶, self-organizing maps¹¹, k-means clustering¹², and principal component analysis¹³. While these packages have demonstrated their usefulness in applications, some basic problems remain in terms of the algorithms applied¹⁴: (1) None of these methods can, in general, guarantee a globally optimal clustering for any non-trivial objective function. (2) Most methods, such as K-means and self-organizing maps, depend on the “regularity” of the geometric shape of cluster boundaries. These methods generally do not work well when the clusters cannot be contained in some non-overlapping convex sets. (3) Given the above two problems, the

clustering results by these methods are often sensitive to noise. In addition, none of the methods can detect clusters from a noisy background. The noise-related problem is particularly important for analyzing gene-expression data, which are typically very noisy. (4) All these methods require a predefined number of clusters. For gene-expression data analysis, such information is generally unknown. Hence, users have to determine the number of clusters based on manually assessing clustering results.

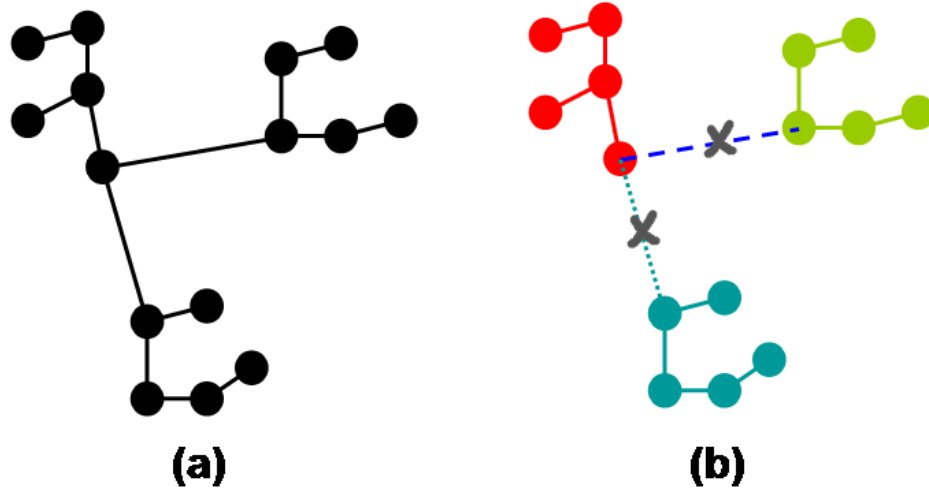


Figure 1. Basic idea of MST-based clustering. (a) Data representation using MST. Each node represents a gene in a multi-dimensional space, where the value of each dimension shows the expression level of a given experimental condition or at a particular time point. (b) Clustering of genes by cutting edges on the MST. Cutting the dotted edge and the dashed edge of the MST creates three clusters, i.e., clusters of red, green, and blue nodes.

To overcome the problems for clustering in the existing packages for gene-expression data analysis, we have developed a computer system EXCAVATOR (EXpression data Clustering Analysis and VisualizATIOn Resource). Unlike any other gene-expression analysis tool, EXCAVATOR represents a set of gene-expression data as a minimum spanning tree (MST)¹⁵, a concept from the graph theory. The basic idea of an MST-based clustering includes the construction of MST and clustering by cutting edges on the MST, as illustrated in Fig. 1. To construct an MST, we first build a complete graph, where each node of the graph represents a gene and every pair of nodes is connected by an edge. The distance of the edge can be calculated by a certain measure, e.g., the Euclidean distance.

An MST is a tree structure that connects all the nodes together with the minimum total distance. The MST basically provides a skeleton of the graph. Through this representation, the problem of clustering a multiple-dimensional data set is rigorously reduced to a tree-partitioning problem without losing any essential information for the purpose of data clustering, as we have mathematically proved¹⁴. This has made the computational problem much easier to tackle.

Based on the MST representation, we have developed a number of rigorous and efficient clustering algorithms. The unique and novel algorithmic techniques of EXCAVATOR include (1) efficient implementations of clustering algorithms with guaranteed mathematical properties, including global-optimality clustering measured by general objective functions; (2) extracting clusters out of a noisy background with guaranteed mathematical properties; (3) a strong capability in dealing with clusters with complex cluster boundaries; and (4) automated determination of number of clusters. We have implemented a stand-alone package that can run from a user-friendly Java interface or DOS/Unix/Linux command line. In addition, we also developed a server that allows users to run EXCAVATOR remotely through a Web browser. Even researchers with little computer skill can analyze gene-expression data through the Java interface or a Web browser easily. EXCAVATOR is available freely to academic users. Information pertaining to academic or governmental licenses can be found at <http://compbio.ornl.gov/structure/excavator/>. Commercial license of EXCAVATOR is available from ApoCom Genomics at <http://www.apocom.com>.

The algorithmic aspect of EXCAVATOR has been addressed in our previous publications^{14,16,17}. In this paper, we will focus on EXCAVATOR from the software perspective, including its design, functionality, and comparison against other methods.

MATERIALS AND METHODS

Overview of EXCAVATOR

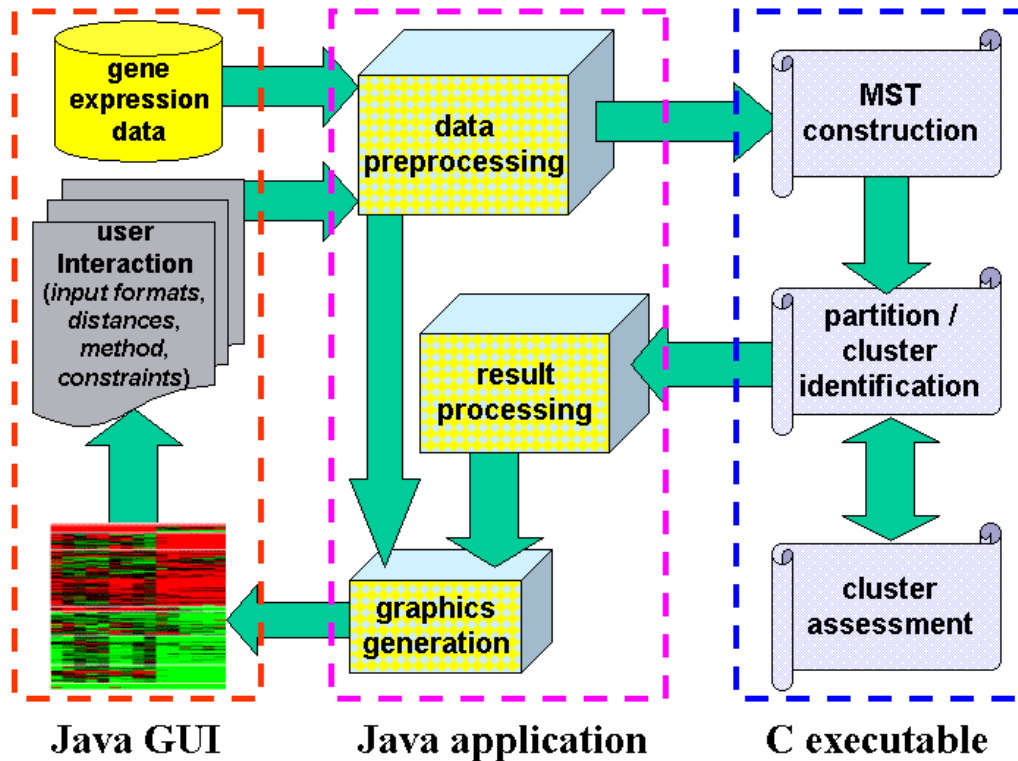


Figure 2. EXCAVATOR design flowchart.

Figure 2 summarizes the overall design of the EXCAVATOR, which consists of a kernel written in the C programming language and a graphic user interface (GUI) written in Java. The connection between the two parts is done through a Java application. Such decomposition allows fast computing with the efficient kernel, while having flexible graphics features at the interface. All the intensive computations, including constructing MST, MST-partition, cluster identification of gene-expression profiles, and cluster assessment, are done by the kernel. The whole package is bundled using JDK1.3 (<http://java.sun.com/j2se/1.3/download.html>), and it has been tested thoroughly on Sun, DEC, and Pentium PC with Windows or Linux. With a large memory (~1GB), the current version of EXCAVATOR can handle up to 10,000 genes for clustering. The kernel can be used as a stand-alone tool from the command line on DOS/Unix/Linux. Through the interface, a user can input the data and customize the kernel parameters. Java system calls are then made to preprocess the data and feed data into the kernel. After the computing is done, the user can select the graphics for visualization. In addition, all the results are kept in individual output files. The Web display an interface at the client side

similar to the GUI of the stand-alone version, while all the computation is done at the server side at Oak Ridge National Laboratory. EXCAVATOR provides an easy-to-use way to analyze the data, while providing numerous options for users to choose.

In this section, we demonstrate the features and design of EXCAVATOR using a subset of gene-expression data (68 genes in total) in the budding yeast *Saccharomyces cerevisiae*⁶, with each gene having 79 conditions (represented by a vector in 79-dimensions space), as an example. The data set includes four annotated clusters, i.e., protein degradation, glycolysis, protein synthesis, and chromatin.

Data input

The main input to EXCAVATOR is a file containing gene-expression profiles, which can be obtained from public databases or new experiments. Though there is no standard format for recording gene-expression profiles, several formats are widely used and EXCAVATOR can read these formats with specification from users. The most common format of the input file (as the default of EXCAVATOR input format) represents each gene with each line and all the entries for a gene are separated by tabs, including gene name, gene annotation, and a set of expression levels.

Sometimes the gene-expression data may not be complete for all data points in every gene. A user can choose how a missing data point is handled in clustering analysis. The default is to set the missing data as 1 for the ratio of the gene-expression level. The missing data can also be replaced by the average over other genes at the same column of the data series, the average over all the other known data points of the same gene, or the average over two neighboring known data points of the same gene.

A user can also select only differentially expressed genes and remove other genes for a data analysis. Such a selection can be set by two cutoff values v_1 and v_2 . If the minimum value among the expression levels of all the conditions of a gene is larger than v_1 and the maximum value among the expression levels is less than v_2 , this gene will be removed.

Similarity measure

The similarity measure represents the means to calculate the distance between gene-expression profiles. EXCAVATOR has several options for similarity measure:

- 1 - correlation coefficient (default)
- 1 - square of correlation coefficient
- 1 - absolute value of correlation coefficient
- Euclidean distance
- square of Euclidean distance
- sine square of the angle between two vectors of expression profiles
- Manhattan distance¹⁸
- Mahalanobis distance¹⁹

Clustering methods

EXCAVATOR offers the following methods for MST-based clustering algorithms based on the selected similarity measure, which were described in details in our previous publication¹⁴:

- MST-hierarchical clustering (default) to minimize hierarchically the sum of the distances between a gene and the center of its cluster.
- MST-iterative clustering (non-hierarchical) to minimize the sum of the distances between a gene and the center of its cluster iteratively; the clustering result, while reaching a local minimum, may not reach the global optimal solution for the objective function.
- MST-optimal clustering (non-hierarchical) to minimize the sum of the distances between a gene and the best representative gene from the cluster. The global optimal solution is guaranteed, but it takes much longer time than other methods.

- Single-linkage clustering by simply cutting longest edges on the MST. It is the fastest method, and it works well for obvious clusters. But the result may not be desired for complicated clusters.

Number of clusters

EXCAVATOR provides three options to determine the number of clusters:

- A user specifies the number of clusters.
- A user determines a number of clusters based on the transition profile generated by EXCAVATOR¹⁴. The transition profile $T(K)$, where K is number of clusters, is calculated based on an objective function $Q(K)$, i.e.,

$$T(K) = \frac{Q(K-1) - Q(K)}{Q(K) - Q(K+1)},$$

where we define $Q(0) = 0$. Typically the highest $T(K)$ value indicates the most “natural” number of clusters. However, a user can specify a number of clusters based on a local maximum of $T(K)$, which may provides an alternative meaningful number of clusters.

- EXCAVATOR automatically determines the best number of clusters based on the maximum value of $T(K)$.

Constraints

EXCAVATOR allows a user to add constraints so that certain specified genes will stay in the same cluster. The constraints can be specified by a file in which genes in the same line (separated by spaces or tabs) are forced into the same cluster. Another related option to cut long edges with distance longer than a threshold and then select the subtrees of MST which contain the “seed” genes specified in the constraint file. In this way, EXCAVATOR can capture the genes having the similar pattern of to that of the seed genes. In addition, a user can also choose the number of genes to be selected instead of the threshold for selecting genes, as demonstrated in the **RESULTS** section.

Cluster identification using ordered representation plot

As a unique feature of gene-expression data analysis packages, EXCAVATOR is able to not only partition all the genes in a data set but also identify clusters from a noisy background. Such a feature is achieved through establishing an ordered representation plot^{17,20,21} based on the relationship between data clusters and the Prim's algorithm²² for constructing MST, as shown in Fig. 3. The ordered representation plot provides a one-dimensional profile for distances of edges on the MST in the order determined by the Prim's algorithm. The plot gives a set of clusters, each corresponding to a “valley” in the ordered representation plot. The statistical significance of a cluster can be assessed by a reliability value using the distances of the edges associated with the valley. All the related information is shown in the Java GUI, as an example given in Fig. 4.

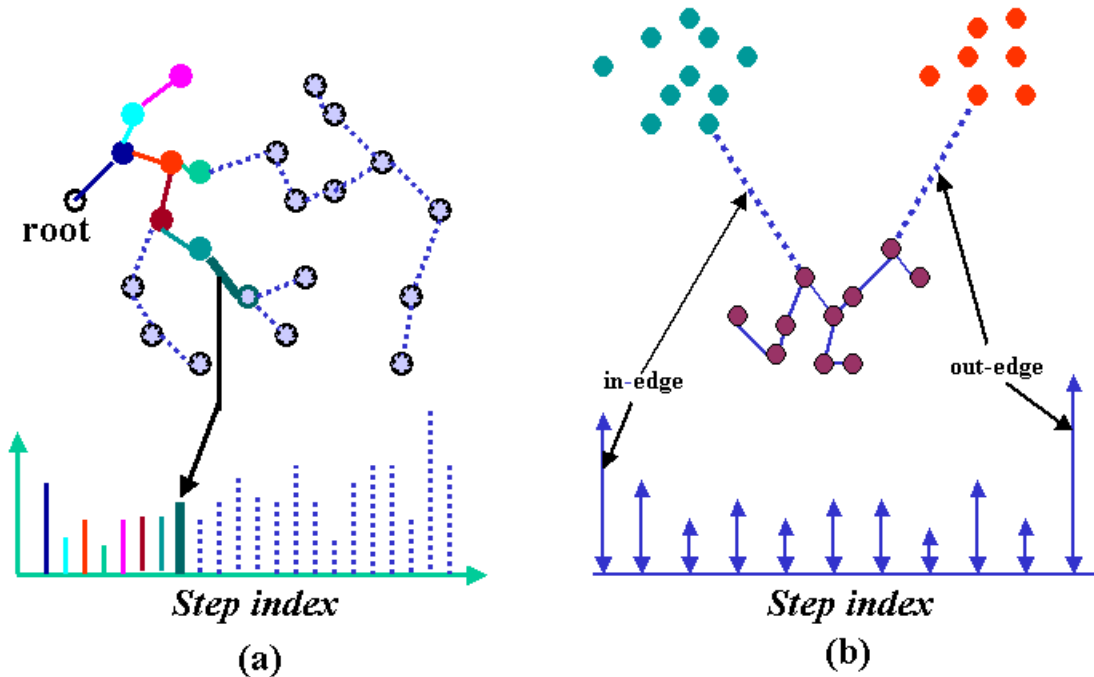


Figure 3. Ordered representation plot. (a) Construction of an ordered representation plot. The construction starts from a complete graph of all the genes in a data set and establishes the order of edges of the MST through a series of steps (partial solutions) in the Prim's algorithm. An initial partial solution is a singleton set containing an arbitrary node as the root. In each step, the current partial solution is repeatedly expanded by adding the node (not in the current solution, i.e., nodes connected by the dotted lines in the figure) that has the shortest edge to any node in the current solution, until all the

nodes are in the current solution. The shortest edge added by each step forms a one-dimensional profile as the ordered representation plot. (b) The relationship between cluster and ordered representation plot. A cluster is generally indicated by the nodes connected through a series of short edges bounded by two long edges (forming a valley) on an ordered representation plot. The statistical significance of the cluster can be assessed by the distribution of the edge distances of such a valley on an ordered representation plot based on the null hypothesis of Dirichle distribution²⁰.

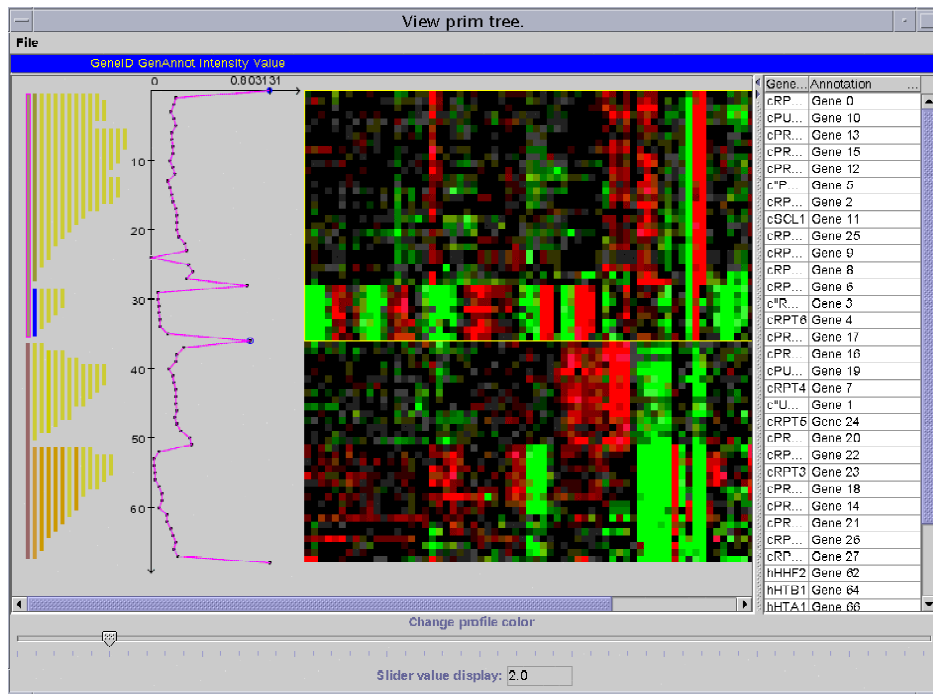


Figure 4. The ordered representation plot for gene-expression profiles. The middle red/green graphics show the expression profiles sorted by Prim's algorithm, where red indicates up-regulation, green indicates down-regulation, and black indicates no significant change in the expression level. To the left of the red/green graphics is the ordered representation plot. The left-most figure shows identified clusters. Each bar indicates a cluster, where darker colors (especially blue) indicate more significant clusters, corresponding to deeper valleys in the ordered representation plot. All the genes in the cluster marked in blue, and only these genes among the data set, are related to chromatin.

Comparison between different clustering results

EXCAVATOR provides many options to satisfy different needs from users. Most data analysis works the best using the default options. However, in some cases other parameter settings may be more suitable and it may not be obvious which set of parameters is the best for a certain problem. Therefore, it is important to compare clustering results from the same input but different sets of parameters and methods. If these clustering results are very similar, it means that the clustering is stable and likely to be reliable; otherwise the clustering results may require more manual evaluations. For this purpose, EXCAVATOR outputs a similarity measure¹⁴ between 0 (most different clustering results) and 1 (identical clustering results).

Design of GUI

The Java GUI consists of a set of pull-down menus and pop-up panels related to selecting parameters and displaying the graphics. The following pull-down menus and related pop-up panels are available, as shown an example in Figure 5:

- File: loads and views a dataset, saves the output in a file, resets all the parameters to the default, and quit the program.
- Input: specifies the input format and whether to apply logarithm to the data, handles missing data, and selects differentially expressed genes.
- Distance: chooses a distance measure and adds constraints.
- Method: selects a clustering algorithm, defines how number of clusters is determined, identifies clusters from background, and finds a set of related genes given predefined seeds (see the bottom-right graph in Figure 5).
- Run: runs EXCAVATOR through a system call to the C executable with the specified parameters and dataset.
- Plot: parses resulting output files generated by the C executable and displays them in graphics, including colored pattern of clustered genes, 2-D plots of the expression profile values for each cluster, dendrogram, and ordered representation

- plot. Mouse-over on a data point of colored pattern shows the gene name and expression level of the data point. A mouse-click on a picture pulls out a list of genes in the cluster clicked. 2-D plots can be enlarged by double click.
- Utility: compares clustering results, with more utilities to be added in the future.
 - Help: provides software information and connection to the EXCAVATOR Web site.

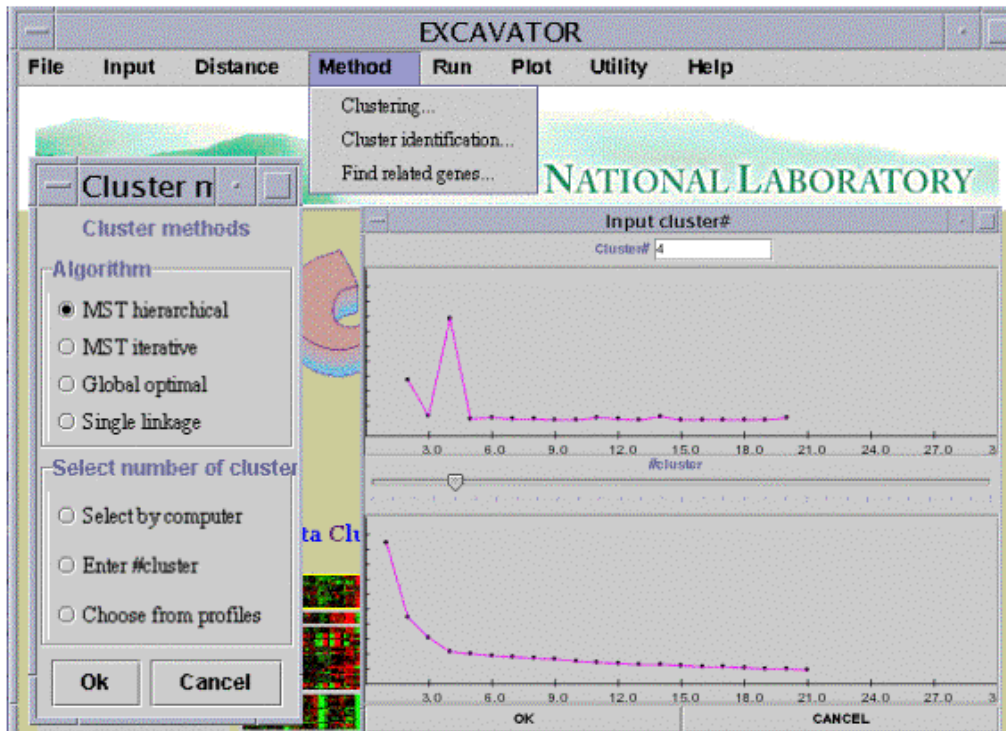


Figure 5. An example of EXCAVATOR interface. In this case, a user chose clustering methods from the pull-down menu, and then selected an MST-hierarchical algorithm for clustering (shown in the lower left panel) and manually determine number of clusters from a profile (shown in the lower right panel).

Java classes

The implementation of Java application and Java GUI was through a series of Java classes, which were built upon the standard Java libraries. Though the C executable generates all the results, they need to be parsed and prepared in a format readily usable by the Java GUI. For this purpose, some calculations are needed. For example, in a 2D plot

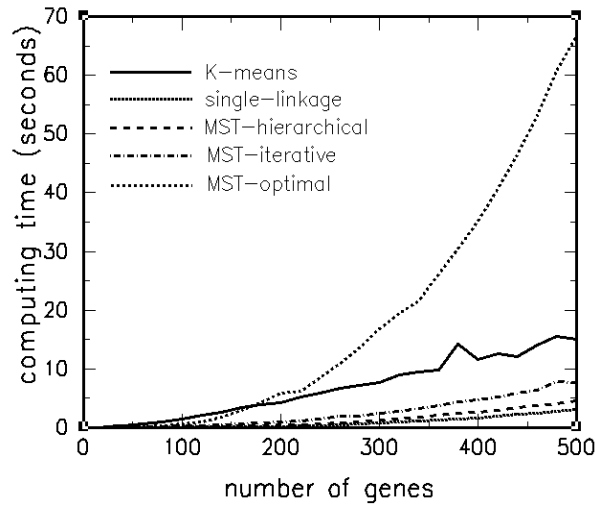
to show the range of gene-expression profiles, a Java class is required to determine the upper bound and the lower bound of the expression profiles at a specific experimental condition. The EXCAVATOR package has dozens of Java classes in four categories: (1) utility classes as general-purpose tools; (2) data readers to handle input/output; (3) graphical components to prepare the data needed for full graphs and implement basic graphic functions; and (4) window classes to display the full pictures of EXCAVATOR outputs.

RESULTS

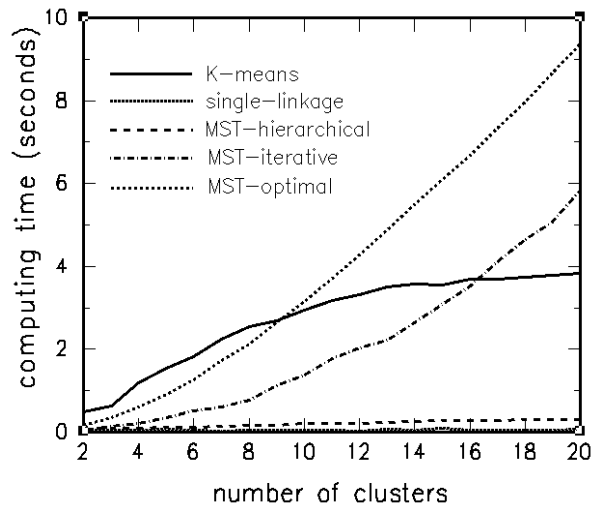
We have tested EXCAVATOR on several applications^{14,16,17}. In addition, we applied EXCAVATOR in analyzing gene-expression data of chitin elicitation in *Arabidopsis thaliana*²³, which lead to discovery of novel genes related to the process. In this paper, we will also show a comparison between EXCAVATOR and the popular K-means clustering method in terms of clustering quality and computing time. We will also show two application examples of EXCAVATOR with unconventional use of clustering.

Computing time

EXCAVATOR is very efficient in terms of computing time. The computational complexity of the algorithms used in EXCAVATOR was analyzed in our previous paper¹⁴. Here we provide a benchmark on a single-CPU Linux workstation. As shown in Fig. 6(a) for different number of genes in 4 clusters, all the EXCAVATOR algorithms other than the MST-optimal clustering are much faster than the K-means method. The CPU time for the MST-optimal clustering is not very long (less than 70 seconds for more than 500 genes). Typically researchers are interested only in the highly regulated genes. The rat CNS data set with 111 genes is a typical size for gene-expression data analysis. In this case as shown in Fig. 6(b), all the methods finish the clustering in 10 seconds, i.e., practically different methods do not make much difference. However, with comparable computing time to the K-means method, EXCAVATOR can deliver better results with well-defined mathematical properties.

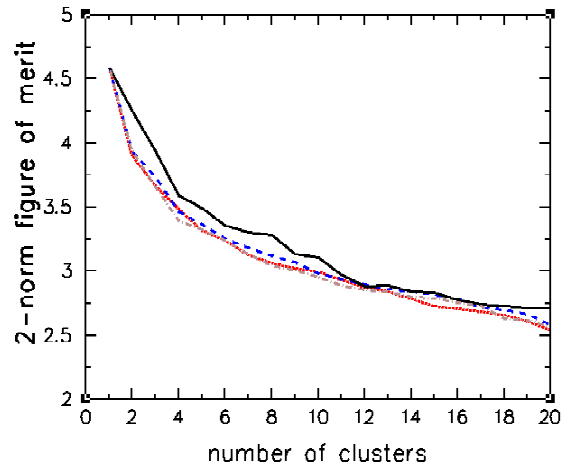


(a)

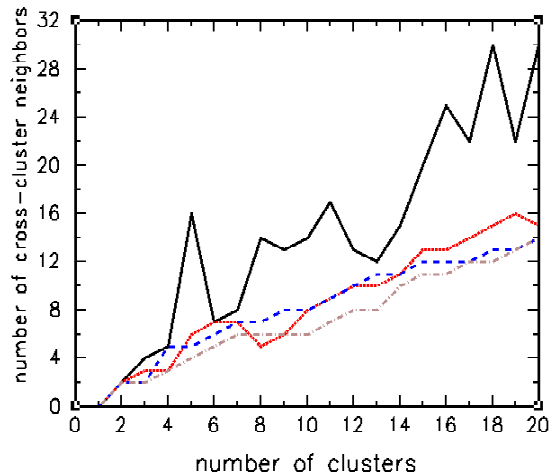


(b)

Figure 6. Computing time benchmark for comparing the EXCAVATOR algorithms and the K-means method. (a) CPU time vs. number of genes for the serum gene-expression data with 517 genes and 21 conditions^{24,25}. The set of genes with a particular quantity were randomly selected from the whole data set. The number of clusters is 4 for all cases. (b) CPU time vs. number of clusters for the rat CNS data with 111 genes and 9 conditions. It is worth mentioning that the computing time of the K-means method involves some randomness due to the randomization process in the method.



(a)



(b)

Figure 7. A comparison between the EXCAVATOR clustering algorithm and the K-means clustering algorithm using the rat CNS data with 111 genes and 9 conditions. (a) The predictive power of the clustering algorithm measured by the 2-normal figure of merit vs. number of clusters based on a jack-knife approach; (b) the separability quality of clusters measured by the number of genes whose closest neighbors are in different clusters vs. number of clusters. Any of the three EXCAVATOR algorithms, i.e., MST-hierarchical (blue long dashed lines), MST-iterative (brown dot-dashed lines), and MST-

global optimal (red dotted lines) outperforms the K-means algorithm (black solid lines) in both measurements.

Quality comparison between EXCAVATOR and K-means clustering

Other than offering more features than other gene-expression analysis tools, we found EXCAVATOR generally outperforms in partitioning expression profiles compared with the most popular clustering method K-means⁵. As shown in our previous paper¹⁴, other than the MST-iterative clustering, all the other clustering algorithms used in EXCAVATOR have the mathematical guarantee for a global optimal solution of a certain objective function. In contrast, the K-means method does not guarantee the global solution of the problem. The practical comparison of clustering quality between different methods can be tricky and currently there is no gold standard for this purpose. The common goals or criteria for clustering are two folds: (1) to put elements that are close to each other, with respect to an appropriate distance measure, into the same cluster; (2) to separate elements that are far apart into different clusters. A clustering method often performs well in one criterion but does poorly in another. Here we try to compare the methods in both criteria using the same data set (the rat CNS gene-expression data²⁶). We applied an open-source implementation of K-means²⁷. For the first criterion, we applied a method developed by Yeung et al²⁸ for validating clustering of gene-expression data. This method used a jack-knife approach²⁹ that applies a clustering algorithm to the data from all but one experimental condition. The remaining condition is used to assess the predictive power of the resulting clusters. Meaningful clusters should exhibit less variation in the remaining condition than clusters formed by chance. The quality of clustering is measure by “2-norm figure of merit”²⁸ as a function of number of clusters. The 2-norm figure of merit $FOM(m, n)$ for n clusters and the m -th condition as the left-out as is defined as follows:

$$FOM(m, n) = \sqrt{\frac{1}{N} * \sum_{i=1}^n \sum_{x \in C_i} [R(x, m) - \bar{R}(i)]^2} ,$$

where N is the total number of genes, C_i is the set of genes in cluster i , $R(x, m)$ is the left-out column (the m -th condition) for gene x , and $\bar{R}(i)$ is the average of R for cluster i . The total

FOM for all the conditions is $FOM(n) = \sum_{m=1}^M FOM(m, n)$, where M is the total number of columns. For a given n , a lower $FOM(n)$ value indicates better clustering quality in terms of the first criterion described above. As shown in Fig. 7 for $FOM(n)$, all three algorithms used in EXCAVATOR outperform the K-means method. For the second criterion, we measure the number of genes whose closest neighbors are in different clusters vs. number of clusters. More genes whose closest neighbors are in different clusters represent a separation of closely related genes, giving a poorer clustering quality. Again, all three algorithms used in EXCAVATOR outperform the K-means method using this measure. In short, our result shows that EXCAVATOR can perform better than K-means for both criteria at the same time.

Identifying cell-cycle regulated genes

Here we show an example of using EXCAVATOR to identify yeast genes whose transcription levels are cell-cycle regulated. There are 6178 genes in yeast *Saccharomyces cerevisiae* and 104 of them are known to be cell-cycle regulated³⁰. It was estimated that about 250 cell-cycle regulated genes might exist³¹. The challenge is to identify the remaining 150 or so unknown cell-cycle regulated genes, based on the 104 known ones and the expression profiles of the 6178 genes. This type of problem occurs often in gene-expression data analysis, i.e. to identify the rest of the genes associated with a particular biological process, knowing a subset of the genes associated with this same process. Our working assumption is that genes that are cell-cycle regulated have correlated expression patterns. We have used the gene-expression data from <http://cellycycle-www.stanford.edu/>. In this data set, expression levels are collected at 82 different time points for each gene. The distance between two genes' expression profiles, A and B , is defined as $1 - cc(A, B)$, where $cc(A, B)$ is the correlation coefficient of vectors A and B . We have applied EXCAVATOR to identify the remaining 150 or so cell-cycle regulated genes from the 6178 ones. The basic idea is as follows. We first label all the 104 genes being in the same cluster. We then select a threshold H to remove all edges with distance $> H$, in such a way that the cluster containing the 104 genes has approximately 250 genes. This procedure produced a cluster with 263 genes including the 104 genes. We predict these genes as the cell-cycle regulated genes. Since there is no

experimental data to verify our prediction, we conducted the following computational exercise, trying to estimate how reliable our above prediction is. We did a data-constrained clustering using 52 of the 104 genes as the “seeds”, which were randomly selected from the 104 genes. Figure 8 (a) shows how the threshold H affects the composition of our target cluster. When the threshold value H reach 0.2, our target cluster consists of a total of 279 data points, of which 79 are from the 104-gene set. When this values goes beyond 0.2, a large number of “junks” get included in our target cluster. The threshold 0.2 clearly represents a “transition point”. The shape of the dashed line and the 27 additional identified genes from the 104-gene set suggest that we should have some confidence in the 263 genes being cell-cycle regulated genes. For a general number of seeds to be included, Figure 8(b) shows the prediction accuracy when choosing ~250 genes from all the yeast genes using different numbers of “seeds” from the 104-gene set.

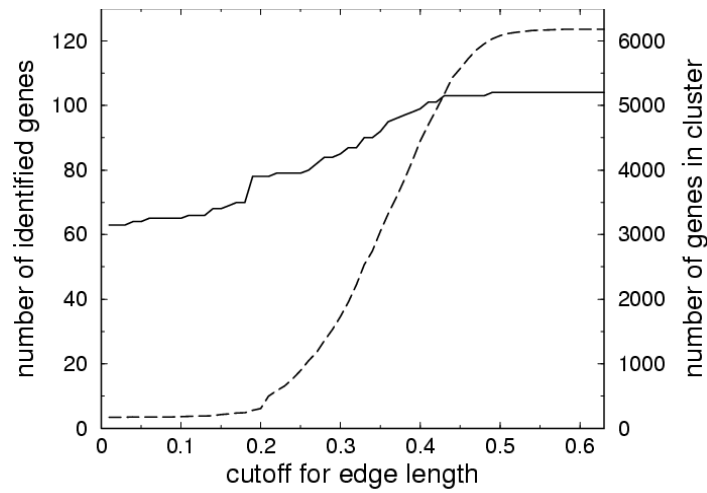
To verify the statistical significance of our results, we used the simple technique of statistical hypothesis testing, where the null-hypothesis is that suggested procedure for reconstruction of a cluster is random. In other words it means that for expansion of original set of 52 genes we, according to our hypothesis, used random sampling from original set of genes with the equal probabilities each gene to be chosen. In this model $P(i, N, M, Q)$, the probability to get i known genes out of M ones in a random sample of volume S from N genes is

$$P(i, M, N, Q) = \frac{\binom{M}{i} \binom{N-M}{Q-i}}{\binom{N}{i}}$$

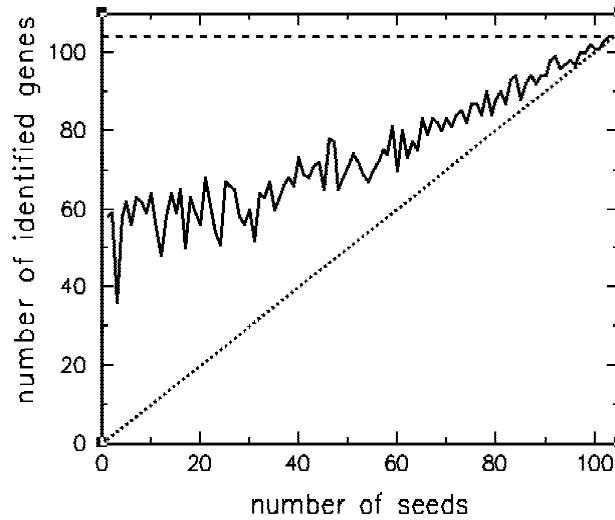
for $i \geq \max(0, M + Q - N)$. Therefore choosing the value " i_0 " such that

$$\sum_{i \geq i_0} P(i, N, M, Q) < \alpha < \sum_{i \geq i_0 - 1} P(i, N, M, Q),$$

we have the first type error α for rejection of the true hypothesis. In our case $N=6126$, $M=52$, $Q=211$, and choosing $\alpha = 0.01$, we get $i_0=5$. So for the statistical significance 0.01, p-value is 1.55E-15, that means we would reject the null-hypothesis even with the statistical significance = p-value. In other words, our result is highly significant statistically.



(a)



(b)

Figure 8: Identification of cell-cycle regulated genes in yeast. (a) Gene identification as a function of threshold H (x-axis) using 52 out of 104 genes as the seeds. Solid line shows the number of genes (y-axis on the left), from the 104 genes, included in our target cluster, and the dashed line shows the total number of genes included in this cluster (y-axis on the right). (b) The number of genes from the 104-gene set that are included in the target cluster as a function of number of seeds used, when we chose ~ 250 genes in total from all the yeast genes (solid line).

Cluster identification in gene-expression profiles

One of the main problems with existing clustering techniques is that they are generally inadequate in identifying “dense” data clusters from the noisy background as they are designed to partition a data set into “clusters” no matter if any “clusters” exists or how many exist. Our cluster identification method using ordered representation plot, as described in the “MATERIALS AND METHODS” section can solve this problem. We have applied the method to a number of gene-expression data sets. The example we show here is a set of 145 differentially expressed genes from yeast under the experimental conditions to identify genes possibly involved in the amino acid transport pathway. Figure 9 shows a part of the ordered representation for the data set. We can see clearly that there is a “dense” cluster in the middle of the figure. Five genes are in this dense cluster: PHO5, BAP2, BAP3, AGP1, and TAT1. Based on our previous knowledge and experimental results, we know that these five genes are part of the amino acid transport pathway in yeast. This information, which cannot be obtained from other methods, is very important to understand the pathway.

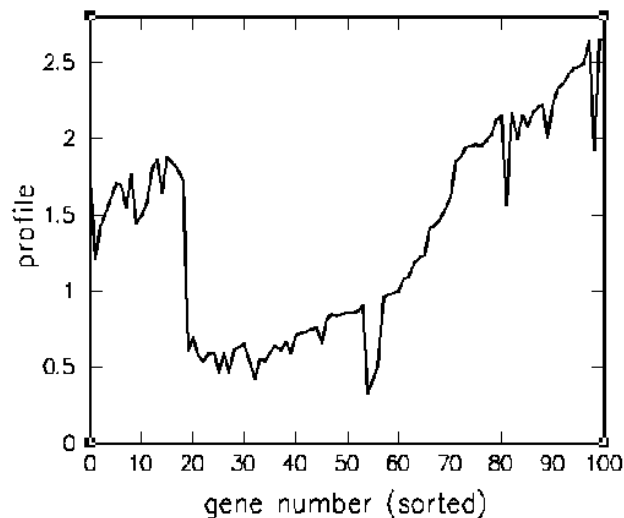


Figure 9. The ordered representation plot for gene-expression data related to the yeast amino acid transport, where the valley in the middle of the plot indicates a “dense” cluster.

SUMMARY

In this paper, we have presented a new software package EXCAVATOR for analyzing gene-expression data. Although many software packages are available for analyzing gene-expression data, EXCAVATOR is unique in many ways. EXCAVATOR is based on a new data-representation framework, which provides a solid mathematical foundation for clustering and classification. It has efficient implementations of clustering algorithms with guaranteed mathematical properties¹⁴, including global optimality. It also has strong capabilities in handling data clusters with complex cluster boundaries and overcoming problems caused by background noise. EXCAVATOR also offers a wide range of options and features, including various definitions of distance between expression profiles, different clustering algorithms, data-constrained clustering, automatic selection of the most plausible number of clusters, removal of background noise, identification of genes with similar expression profiles to a set of specified seed genes, comparison of clustering results using different parameters and algorithms, etc. The Java GUI interface of the stand-alone version and the Web interface allow a user with little computing experience to use the program easily and produce many graphics for the user to evaluate and understand the results. The Web server can save the effort to install the software and provide the computing power at the server side. To our knowledge this is the only gene-expression data analysis package that provides a Unix/Linux/DOS command line option, a stand-alone version with a GUI, and a Web interface altogether. With all the unique features, we believe EXCAVATOR will become a powerful tool for the gene-expression data analysis community.

ACKNOWLEDGMENTS

We thank Manesh Shah, Yu Chen, Shauna Somerville, Keith M. Goldstein, Jeffrey M. Becker, Robin Zimmer, and Morey Parang for helpful discussions and evaluations of the software. This work was supported by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC. It was also funded in part by the US Department

of Energy's Genomes to Life program (www.doe-genomes-to-life.org) under project, "Carbon Sequestration in *Synechococcus Sp.*: From Molecular Machines to Hierarchical Modeling" (www.genomes-to-life.org).

REFERENCES

-
- ¹ Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O., and Herskowitz, I. (1998). The transcriptional program of sporulation in budding yeast. *Science*, **282**, 699-705.
 - ² DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, **278**, 680-686.
 - ³ Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E.S. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531-537
 - ⁴ Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Dmitrovsky, E., Lander, E.S., Golub, T.R. (1999) Interpreting gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA*, **96**, 2907-2912.
 - ⁵ Raychaudhuri, S., Sutphin, P. D., Chang, J. T., and Altman, R. B. (2001) Basic microarray analysis: grouping and feature reduction. *Trends Biotechnol.* **19**, 189-193.
 - ⁶ Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, **95**, 14863-14868.
 - ⁷ <http://www.stat.uni-muenchen.de/~strimmer/rexpress.html> .
 - ⁸ Dysvik, B. and Jonassen, I. (2001) J-Express: Exploring Gene Expression Data using Java. *Bioinformatics*, **17**, 369-370.
 - ⁹ Sturn, A., Quackenbush, J., and Trajanoski, Z. (2002) Genesis: cluster analysis of microarray data. *Bioinformatics*, **18**, 207-208.
 - ¹⁰ <http://genome-www.stanford.edu/~sherlock/cluster.html>.
 - ¹¹ 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.

¹² Herwig, R., Poustka, A. J., Mller, C., Bull, C., Lehrach, H., and O’Brien, J. (1999). Large-scale clustering of cdna-ngerprinting data. *Genome Res.* **9**, 1093–1105.

¹³ Yeung, K. Y. and Ruzzo, W. L. (2001) Principal component analysis for clustering gene expression data. *Bioinformatics*, **17**, 763-774.

¹⁴ Xu, Y., Olman, V., and Xu, D. (2002) Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Trees. *Bioinformatics*, **18**, 526-535.

¹⁵ Graham R. L. and Hell, P. (1985) On the history of the minimum spanning tree problem, *Annals of the History of Computing*, **7**, 43-57.

¹⁶ Xu, Y., Olman, V., and Xu, D. (2001) Minimum Spanning Trees for Gene Expression Data Clustering. In *Proceedings of the 12th International Conference on Genome Informatics (GIW)*, edited by S. Miyano, R. Shamir and T. Takagi. Universal Academy Press, Tokyo. Pages 24-33.

¹⁷ Olman, V., Xu, D., and Xu, Y. (In press) Solving data clustering problem as a string search problem. *Proceedings of Conference on Statistical Data Mining and Knowledge Discovery* (ed by Hamparsum Bozdogan), Chapman & Hall/CRC.

¹⁸ Skiena, S. (1990) Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Reading, MA: Addison-Wesley.

¹⁹ Bar-Shalom, T. and Fortmann, T. E. (1988) Tracking and Data Association. New York: Academic.

²⁰ Olman, V., Xu, D., and Xu, Y. (2003) Identifications of Regulatory Binding Sites Using Minimum Spanning Trees. In *Proceedings of the 2003 Pacific Symposium on Biocomputing (PSB)*. Pages 327-338.

²¹ Olman, V., Xu, D., and Xu, Y. (In press) CUBIC: Identifications of Regulatory Binding Sites through Data Clustering. *Journal of Bioinformatics and Computational Biology*.

²² R. C. Prim. (1957) Shortest connection networks and some generalizations. *Bell System Technical Journal*, **36**, 1389-1401.

-
- ²³ Ramonell, K. M., Zhang, B., Ewing, R. M., Chen, Y., Xu, D., Stacey, G., and Somerville, S. (2002) Microarray analysis of chitin elicitation in *Arabidopsis thaliana*. *Molecular Plant Pathology*, **3**, 301-311.
- ²⁴ Iyer, V. R., Eisen, M. B., Ross, D. T., Schuler, G., Moore, T., Lee, J. C. F., Trent, J. M., Staudt, L. M., Hudson Jr, J., Boguski, M. S., Lashkari, D., Shalon, D., Botstein, D., and Brown, P. O. (1999) The transcriptional program in the response of human fibroblasts to serum. *Science*, **283**, 83-87.
- ²⁵ <http://genome-www.stanford.edu/serum/fig2data.txt>
- ²⁶ Wen, X., Fuhrman, S., Michaels, G. S., Carr, D. B., Smith, S., Barker, J. L., and Somogyi, R. (1998) Large-Scale Temporal Gene Expression Mapping of Central Nervous System Development. *Proc. Natl. Acad. Sci. USA*, **95**, 334-339.
- ²⁷ <http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/index.html>
- ²⁸ Yeung, K.Y., Haynor, D.R., and Ruzzo, W. L. (2001) Validating Clustering for Gene Expression Data. *Bioinformatics*, **17**, 309-318.
- ²⁹ Efron, B. (1982) *The Jackknife, the Bootstrap, and Other Resampling Plans*. Society for Industrial and Applied Mathematics.
- ³⁰ Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. of the Cell*, **9**, 3273–3297.
- ³¹ Price, C., Nasmyth, K., and Schuster, T. (1991). A general approach to the isolation of cell cycle-regulated genes in the budding yeast, *saccharomyces cerevisiae*. *J. Mol. Biol.* **218**, 543–556.