EXCAVATOR: a computer program for efficiently mining gene expression data

Dong Xu1,*, Victor Olman1, Li Wang1 and Ying Xu1,2

1Protein Informatics Group, Life Sciences Division and 2Computer Sciences and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6480, USA

Received June 22, 2003; Revised August 1, 2003; Accepted August 18, 2003

ABSTRACT
Massive amounts of gene expression data are generated using microarrays for functional studies of genes and gene expression data clustering is a useful tool for studying the functional relationship among genes in a biological process. We have developed a computer package 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) data-constrained clustering; (ii) identification of genes with similar expression profiles to pre-specified seed genes; (iii) cluster identification from a noisy background; (iv) computational 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 as colored figures and 2-dimensional plots. Moreover, EXCAVATOR provides a wide range of options for data formats, distance measures, objective functions, clustering algorithms, methods to choose number of clusters, etc. The effectiveness of EXCAVATOR has been demonstrated on several experimental data sets. Its performance compares favorably against the popular K-means clustering method in terms of clustering quality and computing time.

INTRODUCTION
As we are moving into the post genome sequencing era, various high throughput experimental techniques have been developed to characterize biological systems at the genome scale. Unlike traditional approaches, where genes/proteins are 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 manner. Among various high throughput methods, 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 laboratories worldwide and to extract biological information hidden in the noisy and unstructured expression data. Computational analysis is often carried out using gene expression profiles, i.e. the expression level versus experimental condition or over a time course. Since genes with the same cellular function or in the same biological pathway often show similar patterns of expression profiles, one can often infer functions of unknown genes based on the known functions of genes with similar expression patterns. One can also assign new players in a biological pathway through identifying genes exhibiting similar expression patterns to the known genes in the pathway. That is why expression profiles clustering is often the first step in biological knowledge discovery from gene expression data. Gene expression data clustering can also be used for disease sub-typing (3) (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 GeneCluster (4), using weighted voting and K-nearest neighbor algorithms, Cleaver (5), using K-means clustering, and Treeview (6), using hierarchical clustering. Some computer tools, such as the R Packages for Gene Expression Analysis (http://www.stat.uni-muenchen.de/~strimmer/Rexpress.html), J-Express (7), Genesis (8) and Stanford’s Xcluster (http://genome-www.stanford.edu/~shergolx/cluster.html), implement a suite of classical algorithms for data clustering such as hierarchical clustering (6), self-organizing maps (9), K-means clustering...
paper, we will focus on EXCAVATOR from the software package that can run from a user-friendly Java interface or a Web browser. EXCAVATOR is freely available through a Web browser. Even researchers with little computer expertise can develop a server that allows users to run EXCAVATOR remotely through a DOS/Unix/Linux command line. In addition, we have developed a stand-alone package that can run from a user-friendly Java interface or a Web browser. EXCAVATOR is freely available to academic users at http://compbio.ornl.gov/structure/excavator/

To overcome such problems 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) (13), a concept from graph theory. The basic idea of a MST-based clustering includes the construction of a MST and clustering by cutting edges on the MST, as illustrated in Figure 1. To construct a 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 length of an edge is calculated by a certain measure, e.g. the Euclidean distance between two gene expression profiles. A MST is a tree structure that connects all the nodes together with the minimum total distance. Conceptually, a MST provides a skeleton of the graph. Through this representation, the problem of clustering a multi-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 proved (12). This representation has made our data clustering problem much easier to tackle.

Based on the MST representation of a set of gene expression data, we have developed a number of rigorous and efficient clustering algorithms. The unique and novel algorithmic techniques of EXCAVATOR include: (i) efficient implementations of clustering algorithms with guaranteed mathematical properties, including global optimality clustering measured by general objective functions; (ii) extracting clusters with guaranteed mathematical properties from a noisy background; (iii) a strong capability in dealing with clusters with complex cluster boundaries. 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 easily analyze gene expression data through the Java interface or a Web browser. EXCAVATOR is freely available to academic users at http://compbio.ornl.gov/structure/excavator/

The algorithmic aspect of EXCAVATOR has been addressed in our previous publications (12,14,15). 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 summarizes the overall design of EXCAVATOR, which consists of a kernel written in C and a graphic user interface (GUI) written in Java. The Java GUI consists of a set of pull-down menus and pop-up panels related to selecting parameters and displaying the output graphics. 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 the MST, MST partition, cluster identification in 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 has been tested thoroughly on Sun, DEC and Pentium PC workstations with Windows or Linux. Running on a computer with 1 Gb memory, EXCAVATOR can handle up to 10 000 genes for clustering. There is no restriction on the number of experimental conditions, which has little effect on the memory requirement and computing time. Although a whole-genome microarray may contain tens of thousands of genes, a specific analysis often focuses on only several hundred differentially expressed genes or fewer, which generally takes less than 100 Mb memory and 10 min of computing time or less on a desktop PC. The kernel can be used as a stand-alone tool from the DOS/Unix/Linux command line. 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 interface at the client side is similar to the GUI of the stand-alone version, while all the computation is done on 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.

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.
Figure 2. EXCAVATOR design flowchart.

In the following, we illustrate the design and features of EXCAVATOR using a small set of gene expression data (68 genes in total) from the budding yeast *Saccharomyces cerevisiae* (6), with each gene having 79 conditions (represented by a vector in 79-dimension space). The data set is made up of four clusters which had been previously determined experimentally, i.e. protein degradation, glycolysis, protein synthesis and chromatin.

**Data input**

The main input to EXCAVATOR is a file containing gene expression data, which can be obtained from public databases or through experimentation. EXCAVATOR recognizes several widely used data formats for gene expression profiles. The most common format (as the default EXCAVATOR input format) represents each gene using one 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 for every gene. A user can choose how a missing data point should be handled in clustering analysis. The default in EXCAVATOR is to set the missing data to 1 for 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 from a data analysis. Such a selection can be done using two cut-off values $v_1$ and $v_2$. Specifically, if the minimum value among the expression levels of a gene under all conditions is larger than $v_1$ and/or the maximum value among its expression levels is less than $v_2$, this gene will be removed.

**Similarity measure**

The similarity measure is used to calculate the distance between gene expression profiles. EXCAVATOR has several options for the similarity measure: 1.0 – correlation coefficient (default); 1.0 – square of correlation coefficient; 1.0 – absolute value of correlation coefficient; Euclidean distance; sine square of the angle between two vectors of expression profiles; Manhattan distance (16); Mahalanobis distance (17).

**Clustering methods**

EXCAVATOR offers the following methods for MST-based clustering algorithms based on the selected similarity measure and objective function, which were described in detail in our previous publication (12).

- 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, through an iterative procedure. 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 globally 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 than the other methods.
- Single-linkage clustering by simply cutting the longest edges in the MST. It is the fastest method and it works well for obvious clusters. However, the result may not be the desired one 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 the number of clusters based on the transition profile generated by EXCAVATOR (12). The transition profile $T(K)$, where $K$ is the number of clusters, is calculated based on the minimum value, $Q(K)$, of the objective function for $K$, 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, the user can specify the number of clusters based on a local maximum of $T(K)$, which may provide 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 the user to apply constraints during the clustering process so that certain specified genes will stay in the same cluster. The constraints can be specified using a file in which genes in the same line (separated by spaces or tabs) are required to belong to the same cluster. Another related option is to use a set of ‘seed’ genes (specified in the constraint file) for EXCAVATOR to find additional genes having a similar pattern to that of the seed genes, as demonstrated in Results.
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 and extract data clusters from a noisy background. Such a feature is achieved through establishing an ordered representation plot (15,18,19) based on the relationship between data and the result of Prim’s algorithm (20) for constructing a MST, as shown in Figure 3. The ordered representation plot provides a 1-dimensional profile for edge distances in the MST in the order determined by the selection order of 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 in the example given in Figure 4.

Comparison between different clustering results

EXCAVATOR provides many options to satisfy different needs of users. Most data analysis methods work best using the default options. However, in some cases other parameter settings may be more suitable; 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 data set but with different parameters and clustering methods. For this purpose, EXCAVATOR provides a capability to facilitate such comparison and it outputs a value (12) between 0 (most different clustering results) and 1 (identical clustering results) to measure the similarity between two clustering results. 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. The comparison can also help address the issue of overlapping clusters, where some genes can appear in different clusters at the same time. If different clustering results only differ by a few genes when using different parameters, these genes are likely to be the overlapping genes in different clusters.

RESULTS

We have tested EXCAVATOR on several applications (12,14,15). In addition, we applied EXCAVATOR to analyze expression data for genes involved in chitin elicitation in Arabidopsis thaliana (21), which led to the discovery of novel genes related to the process. In this paper, we will also compare EXCAVATOR with the popular K-means clustering method in terms of clustering quality and computing time. We will also show two examples of applications 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 paper (12). Here we provide a benchmark on a single CPU Linux workstation. As shown in Figure 5a 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 long (<70 s for >500 genes). In this case, as shown in Figure 5b, all the methods finished the clustering in 10 s, i.e. 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. The computing time plots in Figure 5 have the expected computational complexity for MST-based methods, as discussed in our previous paper (12).

Quality comparison between EXCAVATOR and K-means clustering

Other than offering more features than other gene expression analysis tools, we found that EXCAVATOR generally outperforms the most popular clustering method, K-means (5), in partitioning expression profiles. As shown in our previous paper (12), other than MST iterative clustering, all the other clustering algorithms used in EXCAVATOR mathematically guarantee global optimal solutions of some objective functions. In contrast, the K-means method does not guarantee global optimality of the problem. Practical comparisons of clustering quality between different methods can be tricky and currently there is no gold standard for this purpose. The two common goals or criteria for clustering are: (i) to put elements that are close to each other, with respect to an appropriate distance measure, in the same cluster; (ii) to separate elements that are far apart into different clusters. A clustering method often performs well on one criterion but does poorly on another. Here we try to compare the methods on both criteria using the same data set [the rat CNS gene expression data (23)]. We applied an open source implementation of K-means (http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/index.html). For the first criterion, we applied...
a method developed by Yeung et al. (24) for validating clustering of gene expression data. This method used a jackknife approach (25) 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. The quality of clustering is measured by the $\ell_2$-norm figure of merit (24) as a function of the number of clusters. The $\ell_2$-norm figure of merit $FOM(m,n)$ for $n$ clusters with the $m$th condition as the left-out is defined as follows:

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

where $N$ is the total number of genes, $C_i$ is the set of genes in cluster $i$, $R(x,m)$ is the expression level of gene $x$ at the $m$th condition and $R_i(m)$ is the average gene expression level at the $m$th condition for the genes in 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 Figure 6, for $FOM(n)$, all three algorithms used in EXCAVATOR outperform the K-means method. For the second criterion, we measured the number of genes whose closest neighbors were in different clusters versus the number of clusters. The higher the number of closest neighbors in different clusters, the poorer the clustering quality. Again, all three algorithms used in EXCAVATOR outperform the K-means method using this measure. In short, our results show that EXCAVATOR performs better than K-means as measured by 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 the yeast *S.cerevisiae*, 104 of which are known to be cell cycle regulated (26). It was estimated that about 250 cell cycle-regulated genes might exist (27). 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. identifying...
the rest of the genes associated with a particular biological process, knowing a subset of the genes associated with this 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://cellcycle-www.stanford.edu/. In this data set, expression levels were collected at 82 different time points for each gene. The distance between two gene expression profiles, \( A \) and \( B \), is defined as \( 1 - \text{cc}(A, B) \), where \( \text{cc}(A, B) \) 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 gene genome. The basic idea is as follows. We first label all the 104 genes as being in the same cluster. We then select a threshold \( H \) to remove all edges in the MST 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 to be 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 performed a data constrained clustering using 52 of the 104 genes as the ‘seeds’, which were randomly selected from the 104 genes. Any of the three EXCAVATOR algorithms, i.e. MST hierarchical (blue long dashed lines), MST iterative (brown dot-dashed lines) or MST global optimal (red dotted lines), outperforms the K-means algorithm (black solid lines) in both measurements. In both figures, the lower the \( y \)-value, the better the clustering quality.
The shape of the dashed line and the additional 27 identified genes from the 104 gene set suggest that we should have some confidence in the 263 genes being cell cycle-regulated genes. For the general number of seeds to be included, Figure 7b 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 calculated the $P$ value for finding 27 (i.e. 79 genes minus 52 seeds) correct genes or more out of 52 non-seed (known) genes by chance. For this purpose, we calculated $P(i,N,M,Q)$, i.e. the probability of obtaining $i$ known genes out of $M$ genes in a random sample of volume $Q$ from $N$ total genes is

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

where $N = 6178$, $M = 52$, $Q = 211$ (i.e. 263 genes minus 52 seeds) and $i = 27, 28, 29, \ldots, 52$. By summing $P(i,N,M,Q)$ for all $i$, we obtained $P = 1.55 \times 10^{-15}$, which means that our result is highly significant statistically.

**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 a noisy background as they are designed to partition a data set into ‘clusters’ regardless of whether any ‘clusters’ exist or how many exist. Our cluster identification method using ordered representation plots, as described in Materials and Methods, 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 experimental conditions that allow us to identify genes possibly involved in the amino acid transport pathway. Figure 8 shows a part of the ordered representation for the data set. We can clearly see 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 (28), we know that these five genes are part of the amino acid transport pathway in yeast. This information, which cannot be obtained by other methods, is very important in understanding the pathway.

**SUMMARY**

In this paper, we have presented a new software package EXCAVATOR for analyzing gene expression data. Although many software packages are available to analyze 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 properties (12), 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 to aid the user in evaluating and understanding the results. The Web server can obviate the need to install the software and provide the computing power on 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 together. With all the unique features, we believe EXCAVATOR will become a powerful tool for the gene expression data analysis community.

ACKNOWLEDGEMENTS

We thank Manesh Shah, Yu Chen, Shauna Somerville, Keith M. Goldstein, Jeffrey M. Becker, Robin Zimmer and Morey Parang for helpful discussions and evaluation of the software. This work was supported in part 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-genomestolife.org) under project ‘Carbon Sequestration in Synechococcus sp.: From Molecular Machines to Hierarchical Modeling’ (www.genomes-to-life.org).

REFERENCES