Optimal Arrangement of Leaves in the Tree Representing Hierarchical Clustering of Gene Expression Data

Therese Biedl[1], Broňa Brejová[1], Erik D. Demaine[1], Angèle M. Hamel[2], and Tomáš Vinař[1]

[1] Department of Computer Science, University of Waterloo, ON, Canada,
{biedl,bbrejova,eddemaine,tvinar}@uwaterloo.ca
[2] Dept. of Physics and Computing, Wilfrid Laurier University, Waterloo, ON, Canada, ahamel@wlu.ca

Technical report 2001-14
Dept. of Computer Science, University of Waterloo

Abstract. In this paper, we study how to present gene expression data to display similarities by trying to find a linear ordering of genes such that genes with similar expression profiles will be close in this ordering. In general, finding the best possible order is intractable. Therefore we assume that hierarchical clustering has been applied to the gene expression profiles and show that the best order respecting the clustering can be computed efficiently. We perform experiments comparing the optimal order to several other methods. The implementation of the algorithm, as well as a simple program for viewing hierarchically clustered expression array data and the complete results of our experiments are available at http://monod.uwaterloo.ca/supplements/01expr/.

1 Introduction

Microarray technology can provide scientists with genome-scale insight into the state of a cell under different conditions. A single microarray experiment measures the relative level of expression of every gene in the organism at the same time. To monitor certain cellular processes we can perform several microarray experiments in a time series or under different conditions. Such a series of observations can enhance understanding the role of a gene in this process.

With thousands of genes and tens of experiments, the immense amount of biological data produced by microarrays cannot be evaluated purely by humans. Computer analysis and visualization of this information is therefore an important area of research. We can either apply automated data mining techniques to the data, or try to present the information in a way that can be handled and used by scientists on genomic scale.

This paper explores efficient algorithms for one step of displaying such data in a way that it can be analyzed by humans: ordering the genes on the display to best illustrate trends in gene expression. A general formulation of this unrestricted problem is computationally intractable (see Section 2.2). A natural alternative approach, taken by Eisen et al. [5], is to restrict the orderings to those illustrating a high-quality hierarchical clustering of the genes. Figure 1 shows an example of a diagram resulting from their technique. In such a diagram one can easily observe both the expression profiles of similar genes and the clusters formed by the clustering algorithm. As a consequence, this approach has been used successfully to analyze many experiments; see for example [1, 13, 16, 20, 21] and many others.

In this paper, we show that the ordering phase in Eisen et al.'s technique [5] can be solved optimally by an efficient polynomial-time algorithm. In comparison to the heuristic solution suggested by Eisen et al., our experimental results show that the optimal ordering can improve a standard measure of quality by 25%, which facilitates better understanding of gene expression in microarray technology.

To understand the ordering phase that we solve, we first give a general description of the method taken by Eisen et al. [5], which is divided into three steps.

First, hierarchical clustering is performed on the data, producing a binary tree in which genes are represented by the leaves of the tree. Genes with "similar" expression profiles should be located in the same subtree. Variations of hierarchical agglomerative clustering algorithm are most commonly used for hierarchical clustering, such as average linkage clustering [24]. Other options include neural networks [7]. For surveys on clustering, see e.g. [3, 11, 22].

Second, the list of genes is reordered in a way consistent with the tree, i.e. so that each subtree of the tree corresponds to a contiguous sequence of genes in the list. Thus genes with "similar" expression profiles will be located close to each other in the list.

Third, the resulting ordered collection of numbers is presented to the user in a way that is easy to assimilate. Each data point is represented by a coloured box, where the colour represents a number. The coloured boxes are organized in a table, where rows represent genes (ordered according to the previous step), and columns represent single experiments. The hierarchy of the tree is drawn on the side of the colour box.

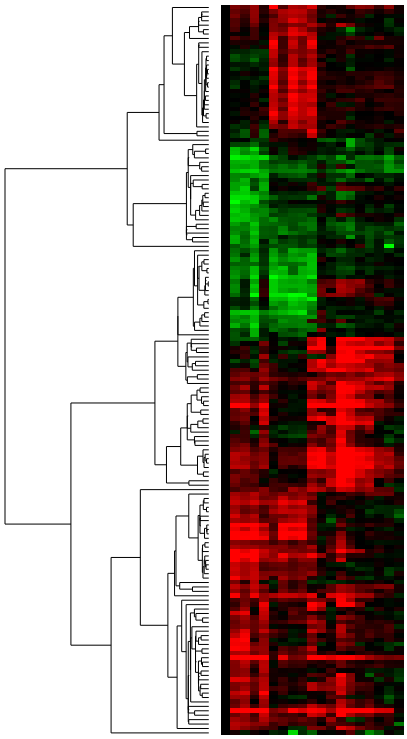


Fig. 1. Response of human fibroblasts to serum. Genes with high expression levels were selected from the data set [10]. The hierarchical clustering tree was then generated by program Cluster [4]. The leaves of the resulting tree (genes) have been rearranged by our algorithm (see Section 2). The gene expression profiles were then visualized as lines of colour boxes, each box corresponding to one data point.

To construct the tree, Eisen et al. need to express similarity of behaviour between two genes as a number. It is possible to use either some similarity measure (i.e., a higher number for more similar genes) or a distance measure (i.e., a lower number for more similar genes). Often it is possible to transform a similarity measure to get a distance measure. Eisen et al. use the standard correlation coefficient – the dot product of two normalized vectors – as this similarity measure seems to be biologically meaningful. Here we assume that a distance measure between the genes is given, either as a function or in tabular form. Furthermore, we assume that the tree itself is given, constructed either by the methods of Eisen et al. or by other techniques, and we focus on the gene ordering. In particular then, the results of this paper hold for other distance measures and other trees as well.

In this setting, it is possible to formulate the problem as follows: find an optimal linear ordering of genes, such that genes with similar expression profiles are close together. In particular, we want to minimize the sum of distances between gene expression profiles adjacent in the linear ordering. As mentioned above, the unrestricted form of this problem cannot be solved efficiently, for as we will see in Section 2.2 the problem is NP-complete. However, in the context of Eisen et al.'s method, the ordering must satisfy a desired hierarchical clustering, and subject to that constraint minimize the sum of distances between adjacent gene expression profiles. If there are n genes, then there are 2^{n-1} possible linear orderings that are consistent with the hierarchical clustering. Hence the obvious algorithm of trying all possible linear orderings is infeasible.

Since this appears to make the problem computationally intractable, Eisen et al. therefore proceeded to compute a linear ordering with a simple heuristic: assign some weight to each gene (for example the average expression over all experiments or a position of the gene in chromosome). Then for any two children of a

node in the tree, compute the average weight of the two subtrees, and put the subtree with smaller average weight to the top.

As we will show in this paper, finding the optimal linear ordering that is consistent with the hierarchical clustering in fact *is* computationally tractable. We give an $O(n^3)$ time algorithm to solve this problem.

Our results show that among the different orderings we consider—optimal ordering, Eisen's ordering, and random ordering—no one ordering is superior either in cost or appearance. Quantitatively, the optimal ordering improves on the cost of Eisen's ordering by around 25%. Furthermore, the TSP heuristic ordering has a slight advantage over the others in terms of cost. Qualitatively, however, there are only small visible variations in the pictures, and, in particular, the TSP ordering appears to be the most disordered of the orderings.

The rest of the paper is organized as follows. In Section 2 we give a description of our algorithm, as well as notes on how the algorithm can be extended to other different possible settings and metrics. We have implemented our algorithm together with several other methods to rearrange the list of genes and a simple tree viewing program. We used these programs, together with program Cluster [4] to perform several experiments on real data. The results of these experiments are summarized in Section 3. We give concluding remarks in Section 4. Our programs, as well as the complete results of experiments are available as a web supplement to the article at `http://monod.uwaterloo.ca/supplements/01expr/`.

2 Computing an optimal linear ordering

In this part of the paper, we explore how to compute the optimal linear ordering of gene expression profiles, assuming that a distance metric and a hierarchical clustering have been fixed. We first briefly discuss the computational intractability in the absence of a hierarchical clustering. Then we provide a dynamic programming algorithm and discuss how to improve it with matrix multiplication. Finally, we consider possible improvements and variations.

2.1 Notation

We assume that there are n gene expression profiles, numbered $1, 2, \ldots, n$. For $i \neq j$, denote by $d_{i,j}$ the distance between gene i and j. We assume that $d_{i,j} \geq 0$ and $d_{i,j} = d_{j,i}$.

For most of this paper, we assume that the hierarchical clustering has been given, in the form of a tree T with the n genes at its leaves. For any node v in T, define $T(v)$ to be the subtree of v rooted at v.

2.2 Ordering without clustering

Ideally, one would like to obtain a linear order of all genes that puts similar genes close to each other. So one would like to obtain an ordering such that for any two consecutive genes the distance between them is small. Put mathematically, we would like to find a permutation π of $\{1, \ldots, n\}$ such that

$$\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)}$$

is minimized.

Unfortunately, this problem is NP-complete, and hence appears to be computationally intractable. Note in fact that the problem is the same as the problem of finding a tour among n cities such that the total distance travelled is minimized. This so-called *Traveling Salesman Problem* is well-known to be NP-complete [6].

If the distance measure is a metric, i.e., if the distance measure satisfies the *triangle inequality* $d_{i,j} \leq d_{i,k} + d_{k,j}$, then we can apply approximation algorithms to find an ordering of genes that is provably not far away from the optimum. Indeed, the problem can be approximated within ratio $\frac{3}{2}$ in $O(n^3)$ (see [8]).

2.3 A dynamic programming solution

We now assume that a hierarchical clustering in form of a tree T has been fixed. We can compute an optimal ordering that respects the clustering using dynamic programming to compute the optimal orderings for all subtrees. Note that we could use the same dynamic program to minimize the maximum distance instead of minimizing the average distance.

The basic idea is to create $(n-1) \times n \times n$ table A with the following meaning. For any internal node v of T, and any two genes i and k that are at leaves in the subtree $T(v)$, define $A(v,i,k)$ to be the cost of the best linear order of the leaves in $T(v)$ that begins with i and ends with k. We allow $i = k$ in the case when $T(v)$ contains only one leaf, i.e., when $v = i = k$ is a leaf.

To compute the values of A, we start with the leaves and define $A(i,i,i) = 0$. Now assume that v is an interior node with children w and x. Let i and k be two genes for which we want to compute $A(v,i,k)$. If (say) both i and k belong to $T(w)$, then the optimum order is undefined, and we set $A(v,i,k) = \infty$. So assume that (say) $i \in T(w)$ and $k \in T(x)$. We then set

$$A(v,i,k) = \min_{j_1 \in T(w), j_2 \in T(x)} A(w,i,j_1) + d_{j_1,j_2} + A(x,j_2,k). \tag{1}$$

This formula is explained as follows: Assume that we knew the optimal linear order of $T(v)$ that starts with i and ends with k. Note that since $i \in T(w)$, and because the order respects the hierarchical clustering, the first elements of the order are an order of $T(w)$, and the remaining elements are an order of $T(x)$. Set j_1 to be the last leaf (in this order) of $T(w)$, then $A(w,i,j_1)$ describes the cost of this order of $T(w)$. Set j_2 to be the first leaf of $T(x)$, then $A(x,j_2,k)$ describes the cost of the order of $T(x)$. Finally, we must add the distance between j_1 and j_2, as this distance is added for the order of $T(v)$.

Since we do not know which element of $T(w)$ and $T(x)$ will be last/first in the optimal order, we simply try all possible elements and take the minimum value obtained.

The time complexity of this algorithm is $O(n^5)$, since there are $O(n^3)$ entries of A to be filled, and filling each entry involves testing $O(n^2)$ combinations of j_1 and j_2.

2.4 An $O(n^3)$ solution

To improve the time complexity of the dynamic programming algorithm, we look at ways to compute Equation (1) more efficiently. Let us replace momentarily the '+' by a multiplication, and the 'min' by a summation. The formula then becomes

$$A(v,i,k) = \sum_{j_1 \in T(w)} \sum_{j_2 \in T(x)} A(w,i,j_1) \cdot d_{j_1,j_2} \cdot A(x,j_2,k).$$

In this form, the equation becomes a double matrix multiplication. More precisely, assume that there are n leaves in $T(v)$, of which k leaves are in $T(w)$ and $n-k$ leaves are in $T(x)$. To compute the $n \times n$-matrix $A_v = (A(v,i,j))$, we must "multiply" the $k \times k$-matrix $A_w = (A(w,i,j_1))$ with the distance-matrix (d_{j_1,j_2}), which in turn is "multiplied" with the $(n-k) \times (n-k)$-matrix $A_x = (A(x,j_2,k))$.

We will from now on use the term "matrix multiplication" where in reality we mean taking the minimum of a collection of sums for each entry of the matrix. Such a matrix multiplication can clearly be done in $O(n^3)$ time. Overall this leads to an $O(n^4)$ algorithm, because we may have to do the matrix multiplication once per level of the tree, and there may be $O(n)$ levels in the tree.

But in fact, the time complexity is only $O(n^3)$, as can be shown by a tighter analysis. Let $t(n)$ be the time to compute the $n \times n$-matrix $(A(v,i,k))$ of a tree $T(v)$ with n nodes.

To compute this matrix, we first must recursively find the matrix for the left subtree which we assume to have k nodes. This takes $t(k)$ time. We also must compute the matrix for the right subtree which takes $t(n-k)$ time. Finally, we must do a double matrix multiplication. To multiply a $k \times k$-matrix with a $k \times (n-k)$-matrix takes time $O(k^2(n-k))$. To multiply the resulting $k \times (n-k)$-matrix with an $(n-k) \times (n-k)$-matrix takes $O(k(n-k)^2)$ time. Hence $t(n)$ obeys the recurrence relation

$$t(n) = t(k) + t(n-k) + c \cdot (k^2(n-k) + k(n-k)^2),$$

for a suitable constant $c > 0$. By induction on n we can prove that $t(n) \leq c \cdot n^3$. For $n = 1$ the claim holds by choosing c appropriately. For the induction step, observe that both k and $n-k$ are smaller than n. Using induction, we then have

$$\begin{aligned} t(n) &\leq c \cdot (k^3 + (n-k)^3 + k^2(n-k) + k(n-k)^2) \\ &= c \cdot (k^3 + (n^3 - 3n^2k + 3nk^2 - k^3) + (nk^2 - k^3) + (n^2k - 2nk^2 + k^3)) \\ &= c \cdot (n^3 - 2n^2k + 2nk^2) = c \cdot (n^3 + 2nk(k-n)) < c \cdot n^3 \end{aligned}$$

since $k < n$. Thus we can compute the matrix $(A(r,i,k))$, where r is the root of the tree, in $O(n^3)$ time. The optimal order is then the one that creates the minimum possible entry in $(A(r,i,k))$. Actually finding the order can be done with usual dynamic programming techniques that store in an additional table the indices for which the minimum has been found.

The memory required to compute the matrix can be reduced to $O(n^2)$ as follows: Observe that for given leaves i, j, there exists only one vertex v in the tree for which $A(v,i,j)$ is not infinity, namely, the nearest common ancestor of i and j. Hence, it suffices to compute a matrix $B(i,j)$ that contains the unique entry $A(v,i,j)$ that is not infinity.

Theorem 1. *The optimal linear order of gene expression profiles that is consistent with a given hierarchical clustering can be computed in $O(n^3)$ time and $O(n^2)$ memory.*

2.5 Improvements and variations

Given the close connection to matrix multiplication, it is natural to attempt to speed up the algorithm in the previous section by applying fast matrix multiplication. Namely, Strassen's algorithm [25] and improvements thereof [2] that allow (in principle) multiplying two $n \times n$ matrices in $O(n^\omega)$ time, where $\omega < 3$. Unfortunately, there are two impediments to this approach.

The first problem is that fast matrix multiplication relies on existence of additive inverses and on distributivity of multiplication over addition [14]. However, in our situation, addition is mapped to min, and multiplication is mapped to +, so neither of these properties hold. Indeed, in this case, there is a lower bound on the number of arithmetic operations proving that no improvement to standard matrix multiplication is possible using these operations, even for rectangular matrices [14]. However, in the special case that all distances are either zero or one, and the goal is to find an ordering with total distance zero, the multiplication operation + changes to max. On values 0 and 1, min is equivalent to a boolean and, and max is equivalent to a boolean or. Thus, additive inverses exist and distributivity holds, so fast matrix multiplication could in principle be applied.

The second problem is that the matrices we multiply are inherently nonsquare. Fast matrix multiplication has been generalized to certain rectangular matrices [9]; for example, asymptotic improvement is possible when multiplying an $n \times n$ matrix by an $n \times n^r$ matrix, where r is any constant greater than 0. But when multiplying an $n \times n$ matrix by an $n \times c$ matrix for some constant c, no improvement is possible; there is an $\Omega(n^2)$ lower bound [18], [9]. In our context, we multiply a $k \times k$ matrix with a $k \times (n-k)$ matrix, and multiply a $k \times (n-k)$ matrix with an $(n-k) \times (n-k)$ matrix. If k is $O(1)$ or $\Omega(n)$, and such imbalance is possible, fast matrix multiplication gives us no improvement on at least one of the multiplications. If we can guarantee a consistent constant balance between the two subtrees, then fast matrix multiplication yields an asymptotic improvement in the running time of $O(n^\omega)$, where ω is the best matrix multiplication exponent, currently approximately 2.376 [2]. But again this result relies on distributivity of the operations as described above.

Other variations of the problem may lead to more efficient dynamic programming algorithms. For example, if each leaf of the tree stores a single representative number instead of an entire vector, is there an algorithm running in less than $\Theta(n^3)$ time? What about the special (but unrealistic) case in which each representative number is either 0 or 1?

3 Experiments

We have performed several experiments to determine whether it is useful to consider different methods for reordering leaves in hierarchical clustering tree.

3.1 Methods

Data sets. We have used 11 publicly available microarray data sets. Data sets [1, 16, 15, 19–21] were obtained from the Stanford Microarray Database [23]. Additional data sets [5, 10, 13] were obtained from web supplements corresponding to the articles. In most cases the data was first filtered to exclude any genes that were missing 20% or more of their values or did not have at least one experiment with log ratio at least 2. Data set [5] was combined with the upper-most level of the function classification from MIPS yeast genome database [17].

Implementation. We have implemented several algorithms for ordering genes in a program TreeArrange. All except the last algorithm order genes consistent with a given hierarchical clustering tree.

- *Optimal ordering* obtains the best-possible ordering that respects the clustering, as described in section 2.
- *Eisen's heuristics* is the heuristic given in [5]. It starts by assigning a weight to each leaf equal to the average expression level of that gene over all experiments. The weight of a subtree is the average of the weights of its leaves. In each internal node of the tree the subtree with higher weight is placed on top.
- *Random ordering* assigns random weights to leaves and then proceeds in the same way as Eisen's heuristics.
- *Traveling salesman (TSP)* is an implementation of the 2-OPT heuristic for the traveling salesman problem (see [12] for details). To adapt this heuristic to compute the best path rather than the best cycle we have added an extra gene connected with distance 0 to all other genes. The resulting cycle is then cut at this extra gene.

The optimal ordering and TSP need a distance measure between genes. We used three different distance measures. Let $X = x_1, x_2 \ldots, x_k$ and $X = y_1, y_2 \ldots, y_k$ be the expression levels of two genes.

- *Pearson correlation:*

$$s_{X,Y} = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{x_i - \overline{X}}{\sigma_X} \right) \left(\frac{y_i - \overline{Y}}{\sigma_Y} \right)$$

  where $\overline{X}$ is the mean of the expression levels and

$$\sigma_X = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (x_i - \overline{X})^2}.$$

  The Pearson correlation has value between -1 and 1, with 1 indicating a linear relationship between the two vectors. If we want to use it as a distance measure, we take $d_{X,Y} = 1 - s_{X,Y}$.
- *Uncentered Pearson correlation:* This is essentially the same as the Perason correlation, except that vectors are not centered to have zero mean, i.e.,

$$s_{X,Y} = \frac{1}{k} \sum_{i=1}^{k} \frac{x_i}{n_X} \cdot \frac{y_i}{n_Y}$$

  where

$$n_X = \sqrt{\frac{1}{k} \sum_{i=1}^{k} x_i^2}.$$

  In fact, $s_{X,Y}$ is the the dot product of vectors X and Y normalized to length 1. Again we use $d_{X,Y} = 1 - s_{X,Y}$ as our distance measure.
- *Euclidean distance.*

$$d_{X,Y} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (x_i - y_i)^2}$$

  The $\frac{1}{k}$ term is used to accommodate missing values.

In case of missing values we use only positions that are not missing in both vectors. k is then the number of such positions. The formulas were adjusted also for the case where different experiments have different weights.

We have also implemented a program ShowTree, presenting the results of experiments graphically in encapsulated postscript. The program is partially based on source code of the TreeView program [4].

Data processing. First, hierarchical clustering was applied to all data sets using program Cluster [4] (choosing as options average-linkage clustering method with uncentered Pearson correlation). Second, TreeArrange was applied to rearrange the leaves, using all 12 combinations of methods and distance measures. Finally, we used program ShowTree to present the results graphically. The complete results of our experiments are available in the web supplement to this article.

3.2 Results

Our goal was to find an ordering of the genes such that genes with similar expression profiles were grouped together, and the ordering was consistent with a given hierarchical clustering tree. We also show results from the TSP heuristic for comparison (note that the TSP ordering does not obey the given tree structure and that the TSP results presented here are non-optimal solutions of the TSP problem). We used the sum of distances between the neighbouring genes in the ordering as a quantity to optimize. Note that our algorithm can be applied to other similar optimization problems (e.g. maximum instead of sum).

Quantitative analysis. Numerically, in terms of cost for uncentered Pearson correlation, the effect of random ordering and Eisen's heuristic is similar (see Figure 2). The optimal ordering is significantly cheaper than both Eisen's heuristic and random ordering, on average by a factor of 25%, and the TSP heuristic ordering is consistently the best of all. The results for other distance metrics are similar, the only notable difference being that for Euclidean distance, optimal ordering only achieves a 12% improvement over Eisen's heuristic.

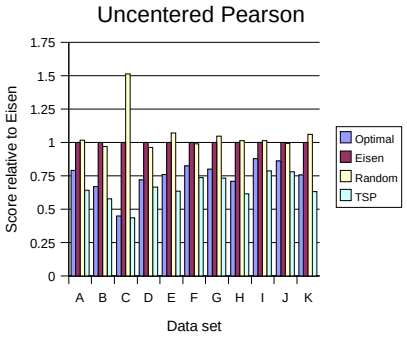
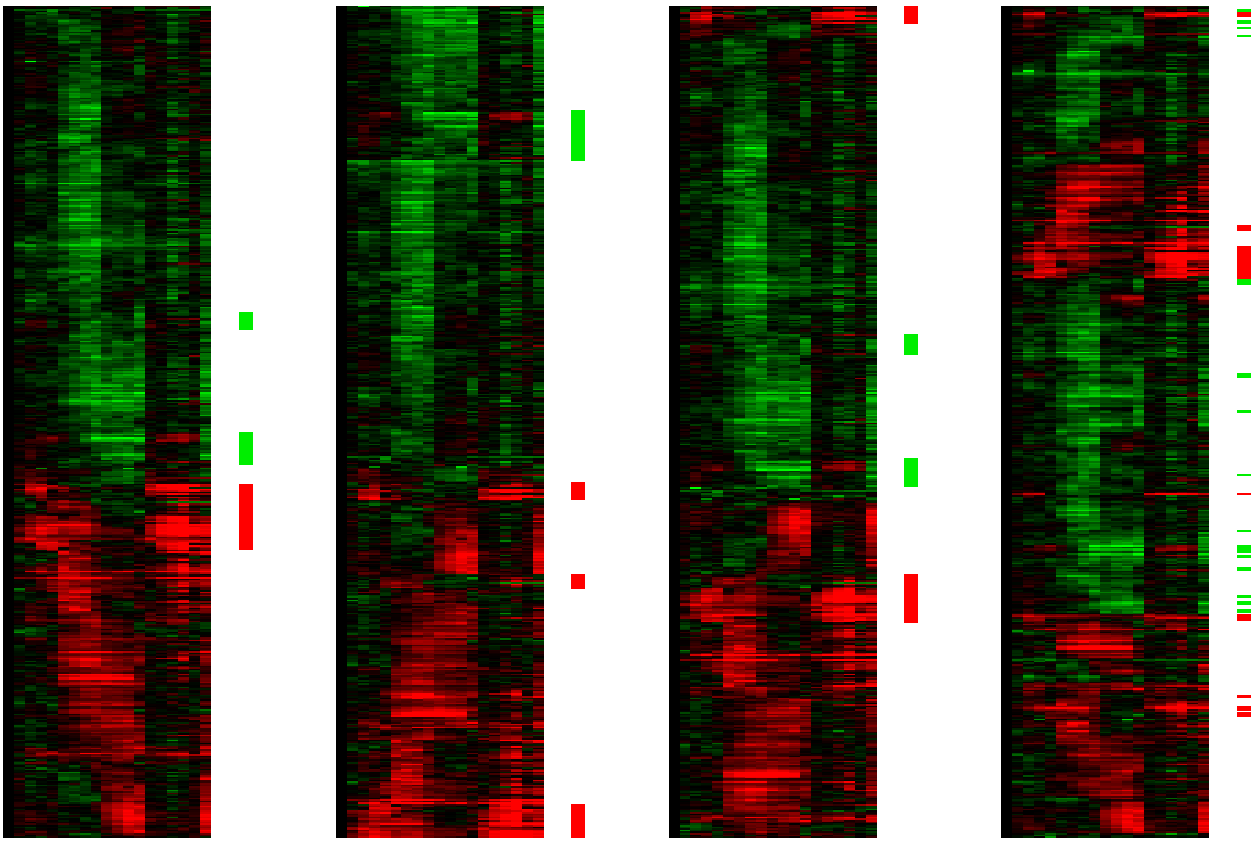


Fig. 2. Cost comparison for 11 data sets using uncentered Pearson correlation. Orderings on 11 data sets have been computed using the optimal algorithm, Eisen's heuristic, random ordering, and TSP heuristic. The resulting costs are presented relative to Eisen's heuristic. *Data sets:* A: C. elegans [19]; B: Rice (salt stress) [13]; C: Arabidopsis (small experiment set) [21]; D: Arabidopsis (large experiment set) [21]; E: E. coli (tryptophan metabolism) [16]; F: E. coli (topoisomerase function) [15]; G: H. pylori [20]; H: Human (DLBCL - small experiment set) [1]; I: Human (DLBCL - large experiment set) [1]; J: Yeast (time series) [5]; K: Human (serum) [10]

Qualitative analysis. Pictorially, in terms of red–green–black colour variation, the differences between the approaches are more subtle. In the following few paragraphs we show examples of such differences on two data sets. For other data sets the results are similar and they can be found in the web supplement.

Figure 3 shows the resulting ordering of data set K [10] using the optimal algorithm, Eisen's heuristic, random ordering, and TSP heuristic. We used the Pearson correlation as a distance measure; the results are similar for other measures.

Although using the optimal ordering leads to substantial improvements in overall cost, visually the results are remarkably similar. In particular, even a random ordering of the tree leaves leads to a visual presentation comparable to Eisen's heuristic or the optimal algorithm. The TSP heuristic produces a completely different ordering, since it does not obey the tree constraints. The results produced by TSP seem to be more disorganized; however, one should take into account that not the best-possible solution for the TSP problem was found, and thus the result is not fully optimized. In fact, locally the TSP heuristic achieves very good results.



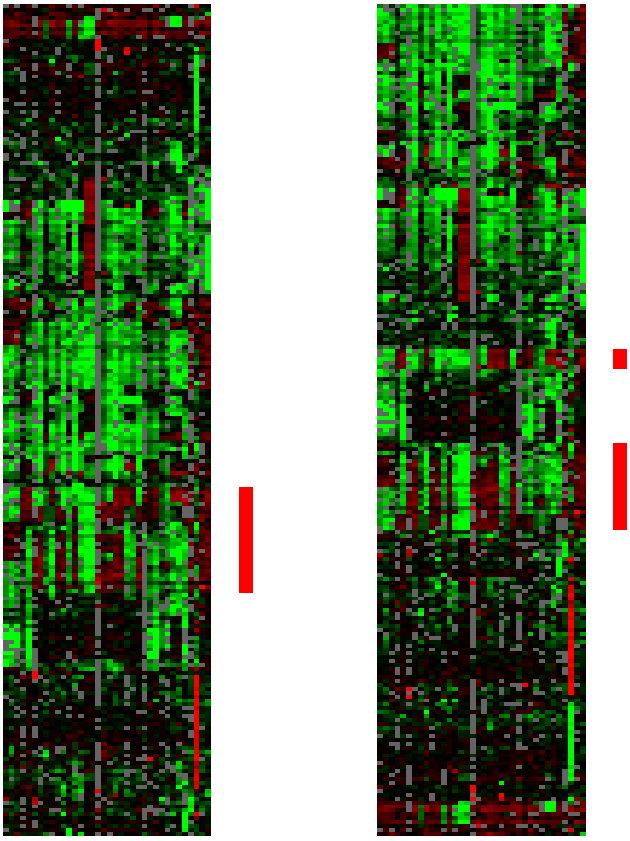
(a) Optimal **(b) Eisen's** **(c) Random** **(d) TSP**

Fig. 3. Response of human fibroblasts to serum. The human serum data represents the response of human fibroblasts to serum. Primary cultured fibroblasts from human neonatal foreskin were deprived of serum for 48 hours and then exposed to a medium containing serum [10]. 517 of the most expressive genes in response to serum were selected. The figure shows the gene ordering using four different algorithms. We used the Pearson correlation as a distance measure. The same hierarchical clustering tree was used to produce (a)-(c), however the tree is not shown here. Ordering (d) does not satisfy the tree constraints.

To study local differences, we have chosen two groups of genes. They are highlighted by red and green bars on the right side of the diagrams. The first group (red) can be characterized by high expression levels (red colour) in the first and last third of the time series. These genes are displayed together in the optimal solution, whereas Eisen's heuristic and all other methods divide them into several groups.

The second group (green) is characterized by low expression levels (green colour) in the middle and at the very end of time series, and they are slightly overexpressed in the first and last third of time series. These genes are displayed together using Eisen's heuristic, whereas the optimal algorithm divides them into two groups.

Figure 4 shows results for data set G [20]. We show only the orderings produced by Eisen's heuristic and the optimal algorithm (using the Pearson correlation as a distance measure).



(a) Optimal **(b) Eisen's**

Fig. 4. Different strains of Helicobacter pylori. The helicobacter pylori is the bacteria responsible for a number of gastric disorders, including ulcers. The DNA microarray technique was used to analyze the genomic composition of different strains of H. pylori to determine the variability between strains [20]. The figure shows the gene ordering using two algorithms with the Pearson correlation as a distance measure. The same hierarchical clustering tree was used to produce both orderings.

In this case, the two pictures look quite different. The result of Eisen's heuristic looks more organized, roughly starting with underexpressed genes (green colour) and ending with overexpressed genes (red colour). In the optimal solution, underexpressed genes are located in the middle of the diagram. Due to the common tree structure, we see large blocks of genes that are ordered similarly in both solutions.

As an example of differences between the orderings we have chosen a group of genes (highlighted by a red bar) that is characterized by interleaving stripes of high and low expression levels. This set is grouped together in the optimal ordering, but split into two parts using Eisen's heuristic.

Although we have chosen to highlight these two data sets in particular, the results from the other nine data sets are similar.

3.3 Discussion

The quantitative data clearly demonstrates that the optimal ordering is superior to Eisen's heuristic in terms of cost. The TSP heuristic ordering is further superior to the optimal ordering, but by a very small amount. However, these cost differences are small and not apparent in the resulting picture (i.e. TSP has the smallest cost, but the pictures appear more disordered).

The qualitative data is not nearly so clear cut. There are only small visible variations in the resulting pictures and we cannot always determine which approach is consistently superior. However, this conclusion is significant in itself on three fronts.

First, it suggests that another distance metric or optimality criterion should be considered to see if it would provide some variation or superior performance. For example, one could use the maximum distance instead of the sum of distances. We have tried three traditional distance measures, and the results in all three cases were very similar. However, other variations and non-traditional distance measures could lead to interesting results. For example, we could consider a 0/1 distance measure, where 0 means "similar" genes, and 1 means "different". In this case we are trying to minimize number of neighbouring "different" genes. We could also have a distance measure that assigns high similarity to genes that are similar only on subset of experiments, even though the rest of the expression profile can be different.

Second, best results would appear to be tied closely to the clustering tree since once a tree is chosen, the leaf ordering techniques appear to have little real effect. Further support is lent to this conclusion by the fact that the TSP orderings, although they yield the smallest sum of distances, appear the most disordered, and these are the only ones that does not rely on the same underlying clustering tree.

Third, it would appear it is not worth expending much effort in optimizing the order of the leaves as this results in a small overall change. Eisen's heuristic (or no ordering at all) is the fastest approach to the problem. However, our approach is still feasible (our algorithm takes about 6 minutes on 350Mhz Pentium on a data set with more than 2099 genes and 34 columns and about 30 seconds for 1257 genes and 29 columns).

This said, there is still an advantage in looking at the data from different points of view as this highlights different dependencies in the data.

4 Conclusions

In this article we have studied the problem of ordering leaves of the tree representing hierarchical clustering of the gene expression profiles. We have presented an efficient $O(n^3)$ algorithm solving the problem. Our algorithm is suitable for different distance measures and various optimization criteria. Note that this approach could be applied to other tree-ordering problems unrelated to gene expression arrays. For example, one could use it to re-order the subtrees of a phylogeny tree under some similarity criterion.

In the second part of the article we have presented results of a small experimental study. The results from all of our experiments point to one conclusion: different orderings may create slightly different results, but no one ordering seems to be superior. Quantitatively, sharp differences can be seen, with TSP and optimal ordering outperforming Eisen's heuristic and random ordering over several distance metrics. However, these advantages, particularly in the case of TSP, may have to be examined in light of qualitative results.

Qualitatively, differences are subtle, and while we can identify individual variations on individual data sets, it is difficult to make general determinations, other than to remark that the TSP approach appears generally more disordered. This could be due to the fact that we have not optimized the TSP implementation.

Clearly, then, the underlying hierarchical clustering tree, and not the reordering or the distance metric, is having the largest effect on the pictures (recall that all but the TSP heuristic used the same tree), and the reordering is of little significance.

On the other hand, our examples show that different methods of ordering leaves in the tree can uncover different local dependencies between the gene profiles. Therefore we suggest that it may be useful to use several different approaches to reordering of the leaves in the tree while studying gene expression data.

Future Work. There are a number of avenues that can be investigated for future work:

1. Our algorithm is quite general. However, for some special cases of distance measures, it may be possible to find more efficient algorithms (see Section 2.5).

2. We have only implemented three different distance metrics. It is also possible that an additional metric or optimization criterion would produce better results. For some optimization criteria (e.g. maximum instead of sum) our algorithm can be used without changes. Other criteria (e.g. if the distance measure depends not only on adjacent genes, but on more distant neighbours in the list as well) can lead to interesting algorithmic questions.
3. Our implementation of the TSP approach uses very simple 2-OPT heuristic to optimize the cost function. There are several other known heuristics that can perform better. For special metrics satisfying the triangle inequality approximation algorithms could also be used. Whether or not such improvements would surpass the results of the other orderings is an important question.

Acknowledgements

We thank all people participating at the Bioinformatics problem sessions at the University of Waterloo for many useful comments on this problem. Angèle M. Hamel acknowledges the support of this research by a research grant from the Leverhulme Foundation, UK. All authors are supported by NSERC.

References

1. A. A. Alizadeh, M. B. Eisen, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.
2. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.
3. B. S. Duran and P. L. Odell. *Cluster Analysis, A Survey*, volume 100 of *Lectures Notes in Economics and Mathematical Systems*. Springer, 1974.
4. M. B. Eisen. Cluster v. 2.11 and TreeView v. 1.5, 2000. `http://rana.lbl.gov/EisenSoftware.html`.
5. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the U.S.A.*, 95(25):14863–14868, 1998.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
7. J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17:126–136, 2001.
8. J. A. Hoogeven. Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Operation Research Letters*, 10(5):291–295, 1991.
9. X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14:257–299, 1998.
10. V. R. Iyer, M. B. Eisen, et al. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283(5398):83–87, 1999.
11. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
12. D. S. Johnson and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, 1997.
13. S. Kawasaki, C. Borchert, et al. Gene expression profiles during the initial phase of salt stress in rice. *Plant Cell*, 13(4):889–906, 2001.
14. Leslie Robert Kerr. *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication*. PhD thesis, Department of Computer Science, Cornell University, June 1970. Technical Report 70-75.
15. A. B. Khodursky, B. J. Peter, et al. Analysis of topoisomerase function in bacterial replication fork movement: use of DNA microarrays. *Proceedings of the National Academy of Sciences of the U.S.A.*, 97(17):9419–9424, 2000.
16. A. B. Khodursky, B. J. Peter, et al. DNA microarray analysis of gene expression in response to physiological and genetic changes that affect tryptophan metabolism in Escherichia coli. *Proceedings of the National Academy of Sciences of the U.S.A.*, 97(22):12170–12175, 2000.
17. H. W. Mewes, D. Frishman, and other. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research*, 28(1):37–40, 2000. `http://mips.gsf.de/proj/yeast/`.
18. V. Pan. *How to Multiply Matrices Faster*, volume 179 of *Lecture Notes in Computer Science*. Springer, 1984.
19. V. Reinke, H. E. Smith, et al. A global profile of germline gene expression in C. elegans. *Molecular Cell*, 6(3):605–606, 2000.

20. N. Salama, K. Guillemin, et al. A whole-genome microarray reveals genetic diversity among Helicobacter pylori strains. *Proceedings of the National Academy of Sciences of the U.S.A.*, 97(26):14668–14673, 2000.
21. R. Schaffer, J. Landgraf, et al. Microarray Analysis of Diurnal and Circadian-Regulated Genes in Arabidopsis. *Plant Cell*, 13(1):113–123, 2001.
22. R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In T. Jiang, T. Smith, Y. Xu, and M. Q. Zhang, editors, *Current Topics in Computational Biology*. MIT press, 2001. To appear.
23. G. Sherlock, T. Hernandez-Boussard, et al. The Stanford Microarray Database. *Nucleic Acids Research*, 29(1):152–155, 2001. `http://daisy.stanford.edu/MicroArray/SMD/`.
24. R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.
25. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.