Interactive visualisation and exploration of biological data

David Gilbert and **Michael Schroeder**
City University, London, UK
{drg,msch}@soi.city.ac.uk

Jacques van Helden
EBI, Cambridge, UK
jvanheld@ebi.ac.uk

Introduction

In this paper we report on the design of, and background to an experimental system we are developing to perform interactive visualisation and exploration of biological data. This research is motivated by the need to analyse the large and rapidly increasing amount of complex data now available to researchers working in the field of bioinformatics. The situation has arisen because advances in technology have resulted in a great output of bio-data; however improvements in methods for analysing and displaying this data have not kept up with this increase.

Biologists often wish to analyse these databases in order to understand the relationship between the items that they contain. The traditional approach is to discover the evolutionary relationship between genes or proteins by analysing their sequence and structure similarities. More recently, similar methods have been applied to discover functional relationships between genes by comparing their transcriptional response to environmental modification. The usual approach to this is: Given a set of n objects and some pairwise comparison, which outputs a similarity or difference measure, (1) perform an all against all pairwise comparison, (2) cluster the items (possibly hierarchically), and (3) display the clusters in some visually meaningful way, normally as rooted (dendrograms) or unrooted trees

Dendrograms

Tree representations were first used in the biological field to represent systematic classifications and their evolutionary interpretation. This has been motivated by the similarity of molecular mechanisms which has let to the widely held view that all organisms diverged from a common ancestor. Specifically, the relationship between any set of species is termed a *phylogeny*; this relationship can be represented by an evolutionary, or *phylogenetic tree*. The task of phylogenetics is to infer this tree from observations of living organisms (Durbin *et al.* 1998).

In their simplest form such trees are binary and can be used to approximate general n-ary trees. The leaves of the tree are labelled by the observed data, or *taxa* (species, biomolecular sequences, protein structures etc) and internal nodes can be labelled by hypothesised or known ancestors.

Each edge of the tree has a certain amount of evolutionary divergence associated with it, defined by the distance between the data items. Thus the distance between nodes (clusters) is biologically meaningful. Although a true phylogenetic tree has a root, i.e. common ancestor for all the species represented at the leaves, some algorithms give no indication about its position and thus return unrooted trees.

Trees can be constructed from pairwise distances by a variety of methods, including UPGMA (unweighted pair group method using arithmetic averages) (Sokal & Michener 1958) and parsimony e.g (Fitch 1971). Since one, or in the case of parsimony several, optimal trees can be generated by tree building algorithms, an approach such as the bootstrap method (Feldenstein 1985) is commonly used to assess the significance of some phylogenetic feature and thus give some measure of confidence for the tree.

Tree/Cluster display

In general each cluster, or node in a tree, can be associated with a classifier or conservation function (pattern) (Brazma *et al.* 1998), a representative member of the cluster, or other information, for example data sources etc.

For example, hierarchically structured databases of protein structures (SCOP (Murzin *et al.* 1995), CATH (Orengo *et al.* 1997), annotate the nodes in the tree at certain levels with representative structures, as well as other (functional or structural) descriptions.

There two main problems associated with trees as a means of visualising relationships. First, there are isomorphisms if the comparison relation is symmetric, and second, the trees do not scale up for large amounts of data.

Given these shortcomings of the classical approach, we have set out to develop a different visualisation technique avoiding the above problems.

Clustering and Interactive Exploration

In contrast to dendrograms, our method uses a 3D space to visualise distance between objects directly. To accom-

plish this task, we have to deal with three subproblems: (1) we have to design distance metrics, (2) given such a distance metric we have to find points in space according to the required distances and (3) we have to visualise these points. Regarding (1), we need a design methodology for mathematical distances which satisfy among others triangle inequality, which states that the direct distance between two objects is the shortest. This is important, as it is our intuitive conception of space and therefore desirable to apply to the distance metric used. Regarding (2), we will develop two approaches, one based on spring embedding (Quinn & Breuer 1979) and the other one on singular value decomposition (Schroeder 1999; Goldberg 1991). Regarding (3), we show how to facilitate interactive exploration of the generated worlds using VRML.

A design methodology for distance metrics

In this section we develop a design methodology for distance metrics. A distance metric is defined as follows:

Definition 1 *Distance Metric/Table*
A matrix $D = (d_{ij}) \in \mathcal{R}^{n,n}$ is a distance metric/table
if it is non-negative, i.e. $d_{ij} \geq 0$ if $i \neq j$ and $d_{ii} = 0$, and
if it is symmetric, i.e. $d_{ij} = d_{ji}$, and
if it satisfies the triangle inequality, i.e. $d_{ij} \leq d_{ik} + d_{kj}$.

Often it is desirable to transform a distance table given the distance property is not violated. For example, one may simply want to re-scale a distance table by a factor of 1000. Or one may want to scale-up clusters and but still leave the bigger picture untouched. To this end, one can simply apply a logarithmic function, which increases small distances relatively more than it does large ones.

The following three general operations do not affect the property of being a distance table: Addition of two distance tables, multiplying with a scalar, and applying a monotonic and concave function. Intuitively, a function f is concave iff the image of the function is below its tangents; formally, this means that $f'' \leq 0$. Two examples of monotonic and concave functions are square root and logarithm. As outlined above, the latter is extremely useful to scale-up clusters.

To apply any of the above operations to a distance table, we need methodologies for converting raw data into a meaningful distance table. Although many approaches will do this ad hoc and possibly violating some of the distance properties, there are various functions which will lead to a distance table. A commonly used similarity measure, which forms a distance is the cardinality of symmetric set difference. We use this approach for example to compare hydrogen bonds and chiralities in the TOPS systems (Gilbert *et al.* 1999). Furthermore, the edit distance of two strings (Levenshtein 1965) - as the name suggests - is a distance. However, an exception are asymmetrically defined derived edit distance measures, which penalise mismatches with different weights. A disadvantage of the basic edit distance is that it is not normalised. A relatively close match of two very long strings may be, for example, much higher than a complete mismatch of two small strings. As it turns out, we can normalise edit distance by dividing by the maximal length of the two strings. Interestingly, other candidate normalisations with the minimal length or the sum of the two lengths does not work.

Two visual clustering algorithms

Given a distance table, we are interested in clusters and their visualisation. In this section, we describe two alternative clustering and visualisation approaches by Schroeder (Schroeder 1999) to the classical dendrograms and related techniques (Durbin *et al.* 1998). Rather than clustering objects directly based on their distance table, we want to find points in a possibly higher-dimensional space, which satisfy the required distances and visualise this space directly for the user, who can then explore the visualisation.

Since we require a mathematical distance, we can guarantee that such points exist. Unfortunately, their dimension may be higher than three, so that we additionally aim to find a three-dimensional solution with least error. Further requirements are a rating of the layout, i.e. how reliable is it, and an anytime-behaviour, which allows us to improve layout with more computational resources available.

Problem Statement So, given the distance table $D = (d_{ij}) \in \mathcal{R}^{n,n}$ we want to position the objects. Using Euclidian distance, which is defined as $||v,w||_2 = \sqrt{\sum_{h=1}^{m}(v_h - w_h)^2}$ for $v,w \in \mathcal{R}^m$, we can formulate our problem formally: We have to define an algorithm which computes a matrix $X = (x_1,\ldots,x_n) \in \mathcal{R}^{m,n}$ for a distance table $D = (d_{ij}) \in \mathcal{R}^{n,n}$ such that $||x_i,x_j||_2 = d_{ij}$, i.e. the distance between x_i and x_j is d_{ij}.

Singular Value Decomposition (SVD) First of all, let us note that there are always many solutions to the problem as we can shift a solution and it still is a solution. This means that we can always find a solution, which is centered around null. We make our way from the original distances D to the sought points in space X through a matrix A, which is derived from D by assigning to an entry at position i,j the value $-\frac{1}{2}$ times the squared distance minus the average squared distances to point i and j plus the average overall squared distances. This rather complicated matrix A turns out to be exactly X^2, so that we have reduced the problem of find X to taking the root of A. For this task, matrix theory provides singular value decomposition (see e.g. (Goldberg 1991)), given the matrix is positive semidefinite, which is the case for A.

The full algorithm works as follows:

1. Compute A as defined above. If A is not positive semidefinite then exit.
2. Compute $A = USU^T$ by singular value decomposition with S having the eigenvalues λ_i in decreasing order on its diagonal.
3. $X = (\sqrt{\lambda_1}u_1^T, \sqrt{\lambda_2}u_2^T, \sqrt{\lambda_3}u_3^T)^T$ where $U = (u_1, \ldots, u_n)$
4a. If $\lambda_1 > \ldots > \lambda_m > 0$ and $\lambda_{m+1} = \ldots = \lambda_n = 0$, then there is only a solution in $\mathcal{R}^m$ and X contains a mapping to $\mathcal{R}^3$ which is best since it is based on the greatest eigenvalues λ_1 and λ_3.
4b. Else X is the solution, i.e. $||x_i, x_j||_2 = d_{ij}$.

Spring Embedding An alternative to the above layout algorithm is spring embedding (Quinn & Breuer 1979). We use the physical metaphor of springs connecting the objects and leading to attractive forces based on the desired distance between them. Additionally, there are repulsive forces between any pair of objects. The algorithm starts with a random layout and then computes for a number of iterations the forces for each object and moves it a bit into the direction of the overall force.

The advantage of this approach is that it is a flexible anytime-algorithm which comes up with an approximation of a solution with any time limitations. However, it is difficult to theoretically evaluate the quality of solutions and to optimally adjust the springs automatically. They are application dependent and thus difficult to find.

Comparison of the Two Approaches As described in (Schroeder 1999), the SVD approach computes the least error when mapping down the higher dimensional solution onto three dimensions. However, in practice it turned out that the spring embedder was sometimes more accurate, which may have to do with accuracy of internal computations. However, the spring embedder may get stuck in locally optimal solutions. Since the spring embedder is an anytime-algorithm, it can be interrupted at any time to provide an approximated solution. In our current implementation this is not so for the SVD algorithm, but it could be added by computing eigenvalues incrementally using the power method. For the spring embedder testing of the layouts' reliability can be achieved by running it several times and comparing the variances of the resulting layout distances. For SVD, we can permute the original distance matrix to check for layout variance.

Interactive Exploration

In data visualization, one distinguishes intrinsic and extrinsic approaches (Benedikt 1991). The former maps object relations to spatial distance, the latter maps objet properties to colour, shape, texture, etc. The theory developed above caters for the intrinsic approach. In the current prototype,

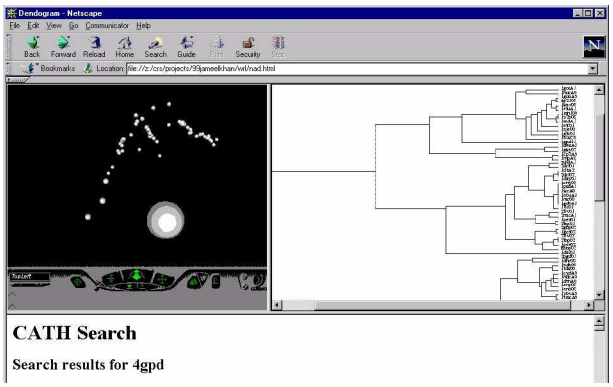

Figure 1: Screenshot of case study system showing integrated view of classical dendrograms, 3D view and textual representation. The sphere in the foreground is 4gpd.

we generate from the layout computed by the above algorithms VRML code. In accordance with (Ware & Franck 1994), we argue that such three dimensional virtual worlds are far more flexible than 2D diagrams, such as traditional dendrograms (Durbin *et al.* 1998), as they are scalable (the space is infinite, and therefore suitable for large amounts of data), as they provide continuous local and global views (the user can focus on details and get the big picture at the same time), as they provide navigation and exploration aids (labels, specific viewpoints, guided tours, interactivity).

Besides VRML's excellent visualisation properties and our intrinsic layout approach, the visualisation benefits even more from extrinsic features. In the case of gene expression data for example, it desirable to interactively colour interesting functional families in the same colour. This feature allows the user to check whether the distance measure chosen reflects the functional properties of the objects.

To maximize the benefits to the user, it is desirable to combine existing visualisation approaches such as dendrograms with novel ideas. In a case study, we created a web-enabled user-interface including dendrograms, our novel 3D visualisation, and text window for detailed information on specific objects. Figure 1 shows a screenshot of our case study.

The above technique can also be applied to multi-attribute data and the next section discusses some preliminary results.

Multi-attribute clustering

Classifications of organisms were originally based on morphological and anatomical criteria, but with the advent of molecular biology, the concept expanded to include comparison of protein sequences and structures. With the emergence of functional genomics (Hieter & Boguski 1997), a

big challenge became to compare genes on the basis of their functional similarities rather than structural features. The development of large-scale gene expression measurement methods (deRisi, Iyer, & Brown 1997) has resulted in the need to analyse and visualise multi-dimensional data (i.e. each item in the database is associated with more than one attribute), and a lot of effort is currently being put into the development of dedicated clustering and visualization tools (e.g. (Eisen *et al.* 1998)).

Gene expression data have been collected from different experiments, where each experiment can itself be a time series with several scalar values. In the case of yeast, 6200 genes are considered, and for each of them, the number of expression measurement under different conditions is rapidly growing. Currently, about 60 values per genes are publicly available, and probably many more will soon be published. For higher organisms, the order of magnitude is of approximately 100,000 genes. Eisen's clustering algorithm (Eisen *et al.* 1998) permits the weighting of the different experimental values through a linear transform of the expression measurement vector. The gene expression profiles can thus generate alternative trees, depending on the precise experiments one wants to emphasize. This is an indirect way of mimicking non-hierarchical clustering, which would possibly be more appropriate to analyze such multidimensional data.

We are in the process of adapting our experimental prototype in order to treat gene expression, and other multi-attribute data. The adaptations include the ability to group and weight different attributes using linear transforms, and the use of colour in the VRML display in order to highlight those parts of the 3D cluster whose grouping has been caused by different attributes. Initial results on clustering gene expression data are promising. This ongoing work has already provided us with some insights into the ways in which different visual attributes can be exploited in order to display this highly complex data.

Conclusion

Interactive display is clearly a useful method to aid the comprehension and analysis of large amounts of complex data; developing such approaches is a significant challenge for computer scientists (and cognitive psychologists) working in bioinformatics. However the bottom line is that biologists must find such methods useful.

In this paper, we aimed to provide an alternative visualisation technique to dendrograms. We have developed a design methodology for distance metrics and two layout algorithms. In contrast to dendograms our approach is scalable. As an essential pratical requirement, we have also outlined how to compute confidence values for the layout. Finally, we have described our experimental prototype, which caters for interactive exploration and we have outlined some ongoing work on multi-variate data visualisation.

References

Benedikt, M. 1991. Cyberspace: Some proposals. In Benedikt, M., ed., *Cyberspace: First Steps*. MIT Press. 273–302.

Brazma, A.; Jonassen, I.; Eidhammer, I.; and Gilbert, D. R. 1998. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology* 5(2):277–303.

deRisi, J.; 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.

Durbin, C.; Eddy, S.; Krough, A.; and Mitchison, G. 1998. *Biological Sequence Analysis*. CUP.

Eisen, M. B.; Spellman, P. T.; Brown, P. O.; and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *PNAS* 95(25):14863–14868.

Feldenstein, J. 1985. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution* 39:783–791.

Fitch, W. M. 1971. Toward defining the course of evolution: minimum change for a specified tree topology. *Systematic Zoology* 20:406–416.

Gilbert, D.; Westhead, D.; Nagano, N.; and Thornton, J. 1999. Motif-based searching in tops protein topology databases. *Bioinformatics*. to appear.

Goldberg, J. L. 1991. *Matrix Theory with Applications*. Mcgraw-Hill.

Hieter, P., and Boguski, M. 1997. Functional genomics : it's all how you read it. *Science* 278:601–602.

Levenshtein, V. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii nauk SSSR (in Russian)* 163(4):845–848. Also in Cybernetics and Control Theory, vol 10, no. 8, pp 707–710, 1996.

Murzin, A.; Brenner, S.; Hubbard, T.; and Chotia, C. 1995. **scop**: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* 247:536–540.

Orengo, C. A.; Michie, A. D.; Jones, D. T.; Swindells, M. B.; and Thornton, J. M. 1997. CATH – a hirearchic classification of protein domain structures. *Structure* 5(8):1093–1108.

Quinn, N. R., and Breuer, M. A. 1979. A force directed component placement procedure for printed cicuit boards. *IEEE Transactions on Circuits and systems* CAS-26(6):377–388.

Schroeder, M. 1999. Using singular value decomposition to visualise relations within multi-agent systems. In *Proc. of the Conf. on Autonomous Agents*. Seattle, USA: ACM Press.

Sokal, R. R., and Michener, C. D. 1958. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin* 28:1409–1438.

Ware, C., and Franck, G. 1994. Viewing a graph in virtual reality display is 3 times as good as a 2D diagram. In *Proc. of Visual Languages*, 182–183. IEEE Press.