Published online March 17, 2006

Nucleic Acids Research, 2006, Vol. 34, No. 5 ***1571–1580***
doi:10.1093/nar/gkj515

Spectral clustering of protein sequences

Alberto Paccanaro*, James A. Casbon and Mansoor A. S. Saqi

Bioinformatics Group, The Genome Centre, Barts and The London School of Medicine, Queen Mary, University of London, Charterhouse Square, London EC1M 6BQ, UK

Received November 9, 2005; Revised and Accepted January 31, 2006

ABSTRACT

An important problem in genomics is automatically clustering homologous proteins when only sequence information is available. Most methods for clustering proteins are local, and are based on simply thresholding a measure related to sequence distance. We first show how locality limits the performance of such methods by analysing the distribution of distances between protein sequences. We then present a global method based on spectral clustering and provide theoretical justification of why it will have a remarkable improvement over local methods. We extensively tested our method and compared its performance with other local methods on several subsets of the SCOP (Structural Classification of Proteins) database, a gold standard for protein structure classification. We consistently observed that, the number of clusters that we obtain for a given set of proteins is close to the number of superfamilies in that set; there are fewer singletons; and the method correctly groups most remote homologs. In our experiments, the quality of the clusters as quantified by a measure that combines sensitivity and specificity was consistently better [on average, improvements were 84% over hierarchical clustering, 34% over Connected Component Analysis (CCA) (similar to GeneRAGE) and 72% over another global method, TribeMCL].

INTRODUCTION

An important problem in today's genomics is that of grouping together homologous proteins when only sequence information is available. This problem is difficult since sequence similarity is a very noisy measure of evolutionary relatedness.

In spite of this, reasonable results have been obtained with algorithms which are relatively simple. In fact, the core of most methods proposed so far in the literature is based on simply thresholding a measure related to the distance between the sequences. These methods could be roughly divided into two categories. A first group uses fully connected graphs, in which the vertices represent the proteins while the edges are labelled with the distance between the two proteins they connect. These algorithms proceed by first removing those edges whose labels are below a certain fixed threshold, and then obtaining a grouping by collecting those proteins that are still linked. This technique, known in the vision literature as Connected Component Analysis (CCA) (1), is used for example by GeneRAGE (2). Clearly, it is crucial to set the threshold to a value that will provide useful groupings. Notice that if the threshold is set to a value which is too conservative (namely two sequences must have a very high sequence similarity in order for their link to be retained) then only very close sequences will be assigned to the same cluster. This type of grouping would then not be very informative, since it is already well established that proteins that are very similar in sequence are very likely to be evolutionary related; moreover, a conservative threshold is likely to generate many singleton clusters. On the other hand, a relaxed threshold would have the opposite effect of including many unrelated proteins into the same cluster.

A second group of methods uses single linkage clustering to organize the proteins on a tree according to the distances between them. A hard threshold on such distances is then used to separate the clusters. An advantage of this approach is that it provides the user with an hierarchical organization of the proteins. These ideas are used, e.g. by SYSTERS (3), ProtoMap (4) and ProClust (5) among others.

All the methods outlined above are 'local', in the sense that they assign a protein to a cluster taking into account only the distances between that protein and the other proteins in the set. We begin this paper by analyzing what results can be achieved in this way. Using sequences and their classification in the SCOP (Structural Classification of Proteins) (6) database, we are able to show what limitations arise for such methods, owing to their locality.

Spectral methods differ from the ones described above in the sense that they are 'global', since they assign a protein to a

*To whom corresponding should be addressed at Molecular Biophysics and Biochemistry Department, Yale University, 266 Whitney Avenue, New Haven, CT 06520-8114, USA. Tel: +1 203 4325065; Fax: +1 203 432 5175 Email: albertopaccanaro@yale.edu

cluster taking into account all the distances between every pair of proteins in the set. We explain why global methods are useful for clustering protein sequences where related proteins can have low sequence identity.

We use a random subset of the SCOP dataset to learn a mapping from sequence distances to probabilities of evolutionary relatedness, and we apply a Spectral Clustering algorithm to these probabilities. We show how these results are superior to the ones obtained using a CCA-related method [similar to GeneRAGE (2)], a hierarchical clustering method and also another global method [TribeMCL (7)] by running these algorithms on several datasets and comparing them in terms of a performance measure defined in terms of precision and recall. Finally, we discuss and explain the reasons for the differences in performance of the various methods.

RESULTS

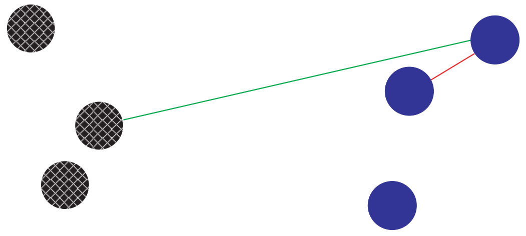
How difficult is clustering using sequence distances?

In this paper we consider the problem of clustering proteins according to their evolutionary relatedness and particularly we are interested in those cases in which some related proteins have very low sequence similarity. As a characterization of evolutionary relatedness we used SCOP's superfamily grouping. The SCOP (6) database is an expert, manually curated database where proteins are grouped together on the basis of their 3D structures. It is organized in a hierarchical manner at four main levels: class, fold, superfamily and family. Proteins in the same superfamily are believed to be evolutionary related, and for this reason we chose such superfamily groupings as the correct groupings, our 'ground truth'. At the superfamily level homology relationships may not be apparent from sequence considerations alone since proteins in the same superfamily can display varying degrees of sequence similarity. Therefore, at superfamily level, SCOP provides an excellent benchmark for testing how algorithms perform in cases in which some related proteins have very low sequence similarity. Also, considering SCOP domains rather than multi-domain proteins allows us to keep the analysis simple while focusing on the comparison between global and local methods. The complete SCOP dataset contains many redundant domains, whose sequences are very similar, so we used the ASTRAL compendium for sequence and structure analysis (8) to select non-redundant subsets. In the study presented here we chose the SCOP subset Astral-95 where no two proteins share 95% sequence identity.

Following many authors before us, we chose BLAST *E*-values as a distance measure between two sequences. BLAST (9) is fast, widely used and, unlike custom distance measures, *E*-values are directly understood by both biologists and bioinformaticians.

Our first goal was to ascertain the difficulty of clustering protein sequences. We performed an all-against-all BLAST comparison of the Astral-95 dataset and we then examined the distribution of BLAST *E*-values within and between superfamilies. In particular, we wanted to understand how far members of the same superfamily can be from each other, and how these distances compare with the distances to members of other superfamilies. This would allow us to analyze the limitations of local methods that involve using a hard threshold on such distances to decide whether proteins should be clustered together.

For each non-singleton domain in Astral-95 we calculated the *E*-value to the nearest neighbour from its own superfamily and the *E*-value to the nearest neighbour from any other superfamily. In other words, these are the minimum *E*-value to a domain from its own superfamily and the minimum *E*-value to a domain from any other superfamily (Figure 1). [A similar figure is calculated in (5). However, here we study the distribution of minimum *E*-values rather than all *E*-values since minimum *E*-values are the ones that ultimately affect



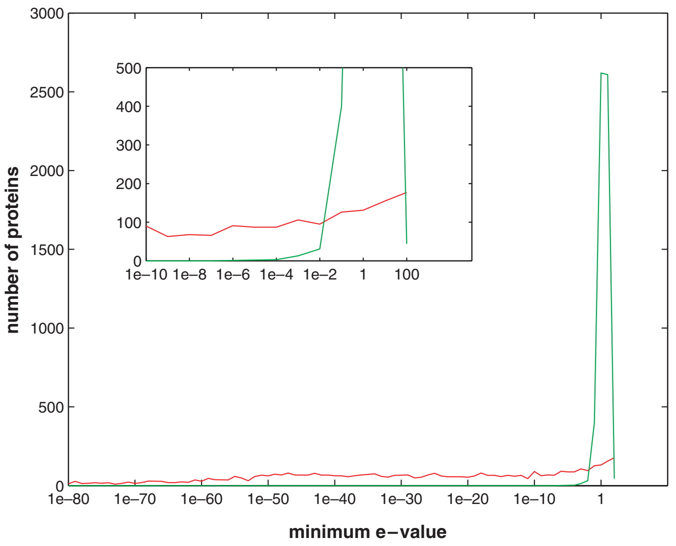


Figure 1. (Left) Pictorial description of how the plot to the right was generated. A protein is represented by a circle. Assume that there are two super-families, identified by the two different colours, blue (solid) and black (pattern). For each protein in turn we computed the distance to the closest protein with the same colour (and we used it for the red plot) and the distance to the closest protein with a different colour (and we used it for the green plot). In the figure, the distances used for one of the blue proteins are shown. (Right) Distribution of minimum *E*-values within (red) and across (green) super-families in Astral-95, for *E*-values between $1e^{-80}$ and 100.

the clustering accuracy.] Furthermore, we counted for how many domains the closest domain within a certain distance belonged to the same superfamily, and conversely, for how many domains the closest domain within that distance belonged to a different superfamily. Figure 2 (left) shows the curves obtained for different values of the distances. From these plots we can understand both why the problem is difficult and what are the limitations of local clustering methods based on placing a hard threshold on the *E*-values.

First of all, we can see that the curves in Figure 1 (right) overlap. This means there does not exists a threshold that will allow a perfect clustering. Take for example the CCA algorithm: a conservative threshold, placed where the green curve of Figure 1 is zero, say at 10^{-6}, will provide all pure clusters; i.e. it will never happen that proteins that belong to different superfamilies are assigned to the same group (in fact the green curve of Figure 2 is zero). At the same time, however, such threshold will place into different groups proteins that really should have been grouped together. Such a cut-off will produce many singleton clusters. A looser threshold, say at 10^{-1}, will produce impure clusters, while continuing to place into different groups proteins that should have been grouped together. We can see then that there is no perfect threshold, and the best results are obtained using a conservative cut-off, as it is usually done by the algorithms found in the literature.

However, we want to point out that such conservative cut-offs solve only the 'easy' part of the problem: i.e. assigning very similar proteins to the same cluster. It cannot solve the most interesting part of the problem, i.e. grouping correctly those proteins having low sequence similarity with the other members of their own class—thus recognizing evolutionary relatedness among proteins which are wildly different in sequence. Interestingly, Figure 2 (right) shows that using psi-BLAST (10), a more sensitive search algorithm, will not improve things—even in this case the problem cannot be solved using conservative cut-offs. Although the error rate is lower for a higher coverage, there are still errors.

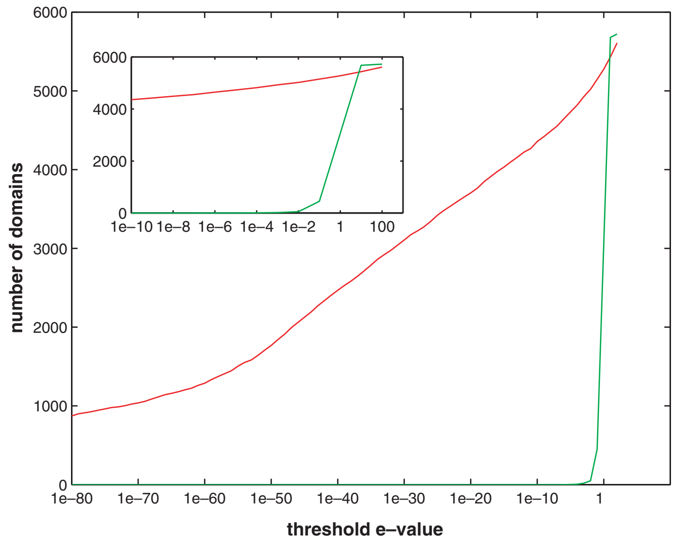
This discussion has elucidated the shortcomings of local methods that use hard thresholds. One way to try to get better results is to use global methods, that assigns a protein to a cluster taking into account not only its distance to every other protein in the set, but also the distance between any pair of proteins in the set.

Spectral clustering results

Spectral methods allow one to study global properties of a dataset by making only pairwise similarity measurements between data points. Here we present the results obtained by our spectral clustering algorithm and we compare them with the ones obtained with three other methods from the literature: GeneRAGE (2) (our implementation), hierarchical clustering and TribeMCL (7). The first two are examples of the two common types of local methods described in the Introduction. TribeMCL was chosen because it is an interesting attempt to use global information.

Considering the grouping provided by SCOP superfamilies as the 'ground truth', it is possible to use the so called external quality measures to evaluate a clustering. In order to evaluate the performance of the different algorithms, we used the *F*-measure, which combines Precision and Recall with equal weights. Refer to the Materials and Methods section for details on the algorithms and on the performance measure.

Here we present the results obtained on two sets of experiments; the results of other two sets of experiments are presented in the Supplementary Data (Experiment 1 and Experiment 2 files). The first set of experiments was done on a group of proteins extracted from SCOP, where superfamilies were hand-chosen in such a way that they would be challenging to cluster, and at the same time the dataset had a simple structure, thus enabling the performance of the algorithm to be appreciated visually. The dataset consists of 507 sequences belonging to 6 super-families, namely Globin-like (88 proteins), EF-hand (83), Cupredoxins (78), (*Trans*)glycosidases



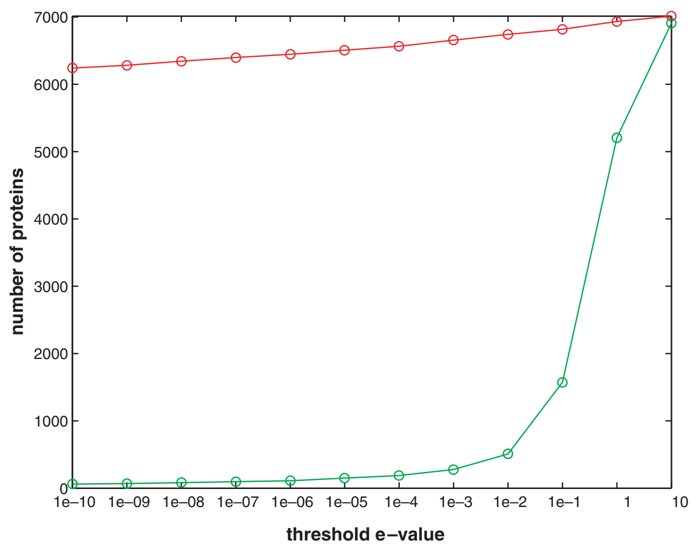


Figure 2. Plots showing the number of non-singleton domains in Astral-95 that hit a member of the same super-family (red line) or different super-family (green line) beneath a threshold *E*-value. The left plot was generated using BLAST, considering *E*-values between $1e^{-80}$ and 100. The right plot was generated using psi-BLAST (five rounds with Astral-95 embedded in non-redundant protein database), considering *E*-values between $1e^{-10}$ and 10.

(83), Thioredoxin-like (81), Membrane all-alpha (94). The last two superfamilies contained 12 and 13 families, respectively. This set was extracted from Astral-95, so the maximum pairwise identity was 95%.

Figure 3 shows the results obtained using the four methods. The spectral clustering method clearly outperforms the other three. First of all, it detects a number of clusters which is close to the correct number of superfamilies, since it detects eight clusters; at the same time, our implementation of GeneRAGE detects 152 clusters, the hierarchical clustering detects 205 clusters and TribeMCL 50 (with the inflation parameter set to 1.58). Only the 30 most populated clusters are shown in Figure 3. The better quality of the clustering is quantified by the *F*-measure: for the spectral clustering it is equal to 0.8132; our implementation of GeneRAGE has a score of 0.4739, the hierarchical clustering 0.2609 and TribeMCL 0.3173.

Looking at Figure 3 we can see that GeneRAGE and the hierarchical clustering algorithms are sometimes able to detect families, but they are inferior to our spectral clustering algorithm in terms of being able to group separate families into superfamilies. This is a direct consequence of our analysis in the Introduction: since proteins in SCOP families have a high degree of sequence similarity they are easy to detect with methods that use a conservative threshold. On the other hand SCOP superfamilies are constructed by grouping families for which a pairwise comparison of individual sequence members may reveal low sequence identity, but whose structures and possibly functional features suggest

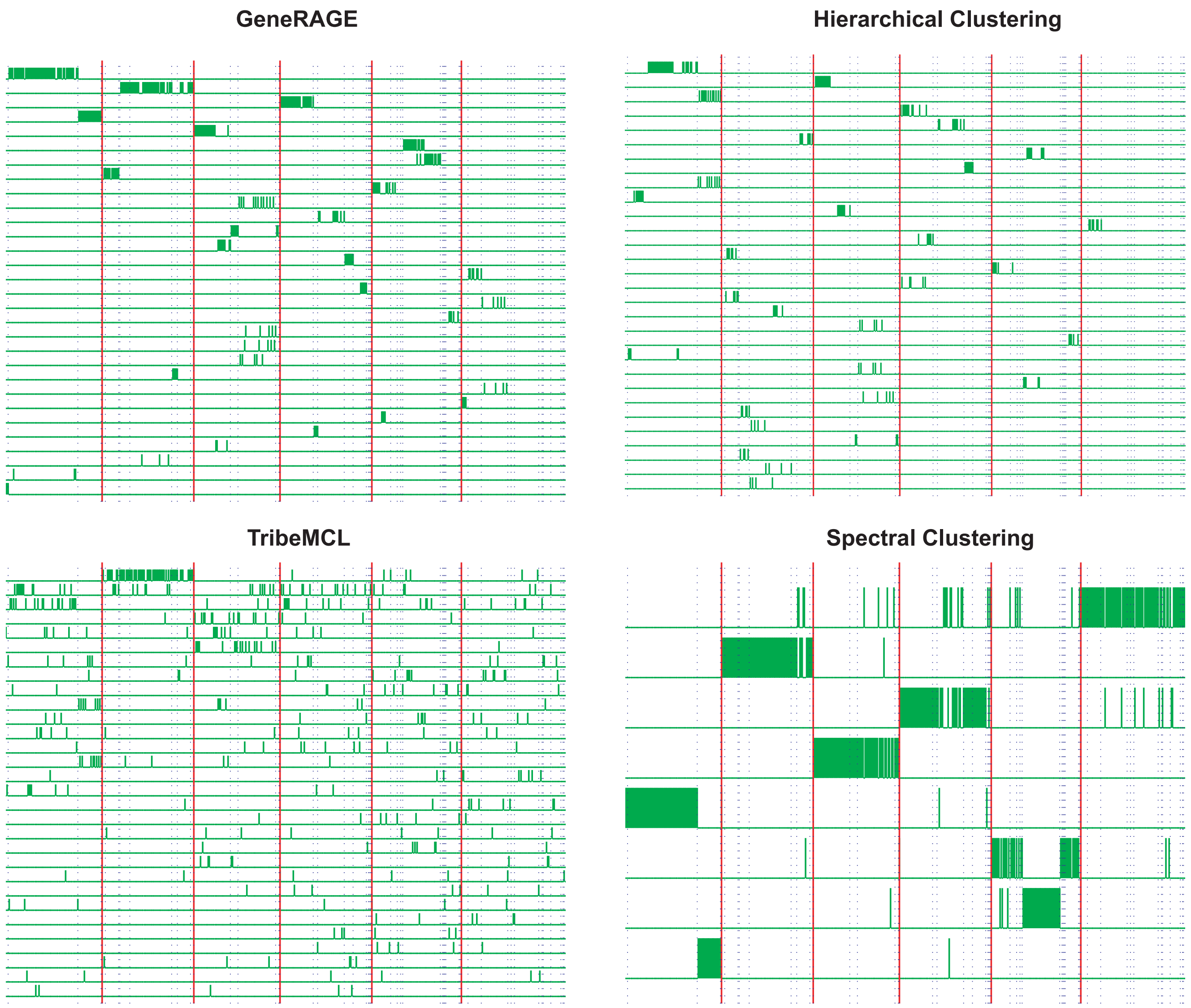


Figure 3. Clustering results on the 507 dataset with our implementation of GeneRAGE (Top Left), hierarchical clustering (Top Right), TribeMCL (Bottom Left) and our Spectral Clustering algorithm (Bottom Right). The figures show only the top 30 most populated clusters returned by each algorithm and 8 for the spectral clustering, since it returned only 8 clusters. Each row in the diagrams corresponds to a different cluster. Short (green) bars represent the assignment of each protein sequence to a cluster. Each protein has one of these bars in only one of the rows (clusters); the presence of the bar means that the protein is assigned to that cluster. Boundaries between super-families are shown by vertical thick (red) lines; boundaries between families within each super-family are shown by dotted (blue) lines. The dataset has 6 super-families, orderly from left to right: Globin-like (88), EF-hand (83), Cupredoxins (78), (*Trans*)glycosidases (83), Thioredoxin-like (81), Membrane all-alpha (94).

that a common evolutionary origin is probable. Therefore methods that use a conservative threshold will not perform well, since related proteins have low sequence similarity.

The cluster plots for the 507 dataset reveal that our spectral method (Figure 3 bottom right) correctly groups together almost all the members of the EF-hand, Cupredoxins and Membrane all-alpha superfamilies. Most of the members of the (*Trans*)glycosidases superfamily are correctly grouped although a few (belonging to one family) are wrongly assigned to the Membrane all-alpha cluster. The Globin superfamily is split into two clusters corresponding to the Globin and Phycocyanin families, and these two clusters include hardly any wrongly assigned members. For those superfamilies where most of the members are correctly assigned to a cluster [e.g. the (*Trans*)glycosidases] a next step would be to analyse in detail the relationships of those incorrectly assigned members to the rest of their superfamily.

Our implementation of GeneRAGE (Figure 3 top left) grouped many of the Globin and EF-hand proteins and also part of the Cupredoxins, (*Trans*)glycosidases and Thioredoxins. Hierarchical clustering (Figure 3 top right) grouped many of the Globins, and part of the Cupredoxins and some families within the (*Trans*)glycosidase superfamily. However, neither of these methods seem to be able to capture superfamily relationships effectively. As we predicted earlier, using a conservative cut-off, both of them create many singletons and all the clusters are pure.

As regards TribeMCL (Figure 3 bottom left), the main clusters it identifies appear to group disparate families. Moreover, TribeMCL also tends to create many spurious clusters containing only few proteins. However it did group together most of the sequences in the EF-hand superfamily.

The second set of experiments was performed on a group of 10 different datasets which were generated from Astral-95 by adding random superfamilies to a dataset until it contained at least 500 proteins. To ensure a fair selection of superfamilies, these were chosen by selecting a random protein from Astral-95 and then including all members of the corresponding superfamily in the dataset. The number of superfamilies in the datasets thus obtained varied between 13 and 23.

Again the spectral clustering algorithm outperformed the other three methods. Figure 4 summarizes the results by showing the *F*-measure obtained on each of the 10 datasets for each of the four methods. For some problems, the *F*-measure obtained by TribeMCL is better than the one obtained by our implementation of GeneRAGE. The hierarchical clustering algorithm often has a better performance than TribeMCL, but is generally less effective than GeneRAGE. However, our algorithm consistently offers an improvement in performance over the other three.

In the Supplementary Data (Experiment 1 file) we present the results obtained by all four methods on a dataset which consists of 511 sequences belonging to 7 superfamilies, namely Globin-like (88), Cupredoxins (78), Viral coat and capsid proteins (106), Trypsin-like serine proteases (73), FAD/NAD(P)-binding domain (64), MHC antigen-recognition domain (51), Scorpion toxin-like (51).

In the Supplementary Data (Experiment 2 file) we present the results obtained on a dataset which consists of 430

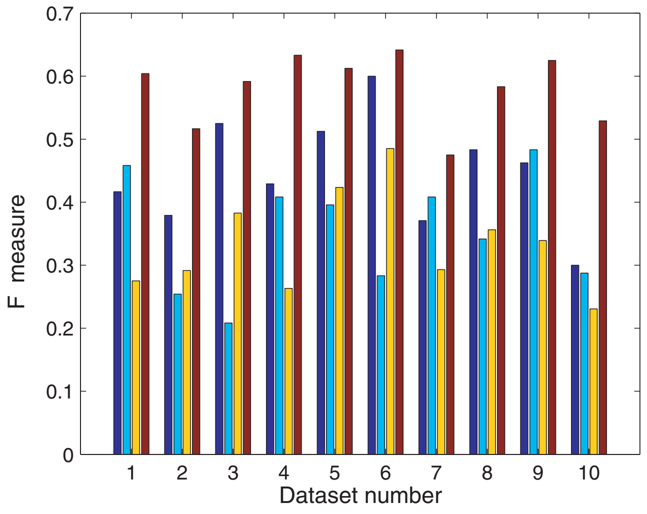


Figure 4. *F*-measure of the cluster quality on the 10 randomly drawn subsets from SCOP. For each dataset the bars represent the performance respectively, from left to right: our implementation of GeneRAGE (dark blue), TribeMCL (light blue) with inflation parameter set to 1.60, hierarchical clustering (yellow) and our spectral method (red).

sequences belonging to 5 superfamilies, namely NAD(P)-binding Rossmann-fold domains (208), Triosephosphate isomerase (TIM), (15) Nucleotide-binding domain (8), Globin-like (97) and EF-hand (102). All the datasets used in the experiments presented in the paper and in the Supplementary Data are available in the Supplementary Data (Datasets file).

Finally, we wanted to test whether the clustering of a certain superfamily was influenced by the other superfamilies in the set, i.e. whether the four clustering algorithms produced consistent groupings of a given superfamily when it was part of different datasets. This is the reason why the Globin-like and the EF-hand superfamilies were included in more that one experiment—the first appears in all three datasets, while the second one appears in the dataset presented above and in the Experiment 2 in the Supplementary Data. In both cases the spectral clustering method gave very consistent results. GeneRAGE and the hierarchical clustering also gave reasonably consistent results, while TribeMCL gave quite inconsistent results for the Globin-like superfamily and more consistent results for the EF-hand one.

In our experiments, on average, the value of the *F*-measure given by our method is 84% better than hierarchical clustering, 72% better than TribeMCL and 34% better than our implementation of GeneRAGE.

DISCUSSION

Our earlier analysis had anticipated what sort of results one might expect using a local method based on placing a hard threshold on the distances between sequences. The results shown in section 2 have confirmed our analysis: all clusters obtained with such methods are pure, but often homologous proteins are placed in different clusters and there are many singletons.

We pointed out earlier that the problem with local methods is that a protein is assigned to a cluster using ultimately only a few measurements, the distances to its closest neighbours. Global methods, on the other hand, assign a protein to a cluster taking into account the distances between every pair of proteins in the set, and for this reason they should be less sensitive to the error which might be present in a few measurements.

We have suggested that spectral clustering methods are global methods, and we have seen that results obtained with a particular implementation are much better than the ones obtained with some local methods. But what exactly do we mean when we say that spectral clustering methods are global? And can we intuitively explain why results are so much better?

Spectral methods use the leading eigenvectors of a matrix derived from the distance matrix between the points. And we know that the eigenvectors of a matrix depend on the whole matrix: change one value in the matrix, and its eigenvectors will be different. This fact ensures the globality of the method.

To intuitively understand how this leads to better results, let us think of a situation as the one depicted in Figure 5. Here we see proteins from two different superfamilies, identified by the two different colours, green (solid) and blue (pattern). Four of the green proteins are very close together and form a tight cluster, and so do the four blue ones. Then there is another green protein, node 5, which shows only very weak similarities to other proteins in the dataset. Four of these weak similarities are with members of its own family, but it also has a weak similarity to one of the blue proteins; moreover, let us assume that such similarity is slightly stronger than any of the similarities to a green protein, that is the distances $a > b$.

Since the closest protein to node 5 is a blue protein, in general, a local method will either place protein 5 together with the blue proteins, or at best in a cluster by itself. To understand what a spectral method will do instead, let us make use of the random walk interpretation of spectral clustering. We can imagine that at any given time there are some particles placed on the vertices of the graph, and at each time step these particles jump from one vertex onto another with a probability related to how similar the two vertices are: the more similar they are, the more likely it is for the particles to jump between them (refer to the Materials and Methods section for details on the random walk interpretation of spectral clustering). According to this, the algorithm will assign node 5 to one of the two clusters, depending on whether a particle travelling on the graph will spend more time within the set $\{1, 2, 3, 4, 5\}$ or within the set $\{5, 6, 7, 8, 9\}$. And although a particle starting from node 5 at any given time step will always be more likely to travel to node 6, for certain values of a and b, it will on average travel more often within the green cluster, since there are four connections to the green cluster, but only one to the blue cluster. In our figure, if for example $a = 0.5$ and $b = 0.3$, protein 5 would be clustered together with the green proteins. We see that the spectral clustering algorithm takes into account the fact that protein 5 has weak similarities to four green proteins, and this is considered a stronger clue of its evolutionary grouping than the fact that there is just one slightly stronger similarity to one of the blue proteins.

Furthermore, due to the globality of the method and according to the random walk interpretation of spectral clustering, the distance between every pair of proteins in the dataset plays a role in the clustering. This means that, in general, the distances within the proteins in the groups $\{1, 2, 3, 4\}$ and $\{6, 7, 8, 9\}$ will influence the assignment of protein 5, although such distances do not directly involve protein 5.

One final question remains to be addressed, which is why the results obtained with our spectral method are better than the ones obtained using TribeMCL. In fact, both methods are global and the starting point for both is the Markov transition matrix. The two algorithms differ in the way in which they propagate the Markov chain on the graph. Our spectral method analyses the perturbations to the stationary distribution of a Markovian relaxation process defined in terms of similarity weights (refer to the Materials and Methods section for details). The Markovian relaxation process never needs to be explicitly carried out; instead, it is analytically expressed using the eigenvectors corresponding to the leading eigenvalues of the Markov transition matrix. TribeMCL does something different since it actually modifies the random walks to promote the emergence of clusters in the graph (refer to the Materials and Methods section for details). While the expansion operators just amounts to one iteration of the relaxation process, the inflation parameters modifies the random walk, boosting probabilities on strong intra-cluster walks and demoting weak inter-cluster walks. While this results in an extremely efficient algorithm, an error is introduced, as the process is only an approximation to the relaxation process implied by the data.

CONCLUSIONS

In this paper we have shown that the problem of correctly grouping together evolutionary related proteins using only sequence information is a difficult one. Our main goal was to point out that if we want to cluster correctly those cases in

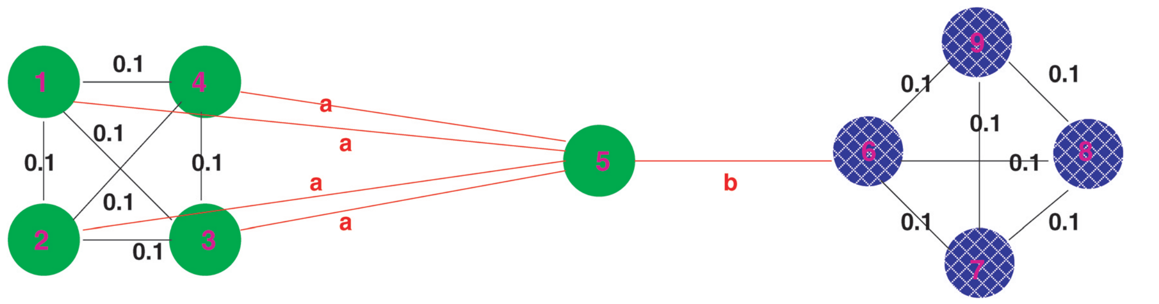


Figure 5. Pictorial representation of proteins belonging to two different super-families, identified by green (solid) and blue (pattern) circles, respectively. Numbers on the connection represent distances between the proteins.

which related proteins have very low sequence similarity then local methods will not provide a satisfactory answer, and global methods should instead be used. Here we have proposed one method to do so, and we have presented results on difficult sets of problems. However, other global methods based on spectral clustering have been proposed (see for example http://crd.lbl.gov/~cding/Spectral/ for a good reference list), and their performance for clustering protein sequences should be investigated.

The spectral clustering algorithm that we used is very simple to implement. The eigenvectors of the normalized symmetric matrix are obtained by singular value decomposition (SVD), which is a very stable process and there exist numerical procedures to compute it efficiently. A Matlab implementation of the algorithm, running on a 1.8 GHz Pentium 4, took only few seconds to cluster the biggest dataset presented here. Our procedure gave very stable results for several different runs. The Matlab code of our implementation is available from the authors upon request.

We have compared the results obtained by our algorithm with those obtained by a connected component method (similar to GeneRAGE), hierarchical clustering and TribeMCL. Our algorithm gives an improved performance over the other three methods in terms of the quality of the clusters as measured by the F-measure. Most importantly, the number of clusters returned by the algorithm is in general much closer to the correct one than the one returned by the other two methods, and particularly there are much fewer singletons. This is particularly relevant for biological applications since it means that the algorithm is able to detect the evolutionary relatedness of proteins which are distant in sequence space.

In our study we have chosen SCOP as our gold standard because it is an expertly curated categorization of proteins which takes into account structural information. Proteins in the same SCOP superfamily may be very distantly related and the similarity may not be apparent from consideration of the sequences alone. We remain aware that the SCOP categorization is not perfect and it may also change. For example, one of the superfamilies considered in this study, viral coat and capsid proteins, (refer to Supplementary Data, Experiment 1 file) was reclassified after version 1.63 of SCOP. In addition to SCOP other classifications exist which could also have been used [e.g. CATH, (11)]. It would be interesting to investigate how the different clustering methods perform with respect to different protein classification schemes.

So far we have not addressed the problem of multi-domain sequences. Also we need to see how our spectral method scales to datasets that are much larger in size.

MATERIALS AND METHODS

The spectral clustering algorithm

In order to apply spectral methods to our problem of clustering protein sequences we re-formulate it as the problem of partitioning a graph.

We consider partitioning a weighted undirected graph G into a set of discrete clusters. Each node in the graph corresponds to a protein sequence and the weight on each edge corresponds to the similarity between the two protein sequences it connects. Ideally, we are looking for areas in the graph, the clusters, in which the nodes are connected with highly-similar edges; and at the same time the connections between such areas should be weak, constituted by edges with low similarity. The problem is to identify these tightly coupled clusters, and cut the inter-cluster edges.

Following the formulation in (12,13) (http://books.nips.cc/papers/files/nips15/AA26.pdf) (citeseer.ist.psu.edu/article/meila01random.html), we consider an undirected graph $G = (V,E)$ with vertices $v_i \in V$, for $i = 1, \ldots, n$ and edges $e_{i,j} \in E$ with non-negative weights $s_{i,j}$ (the similarity between vertices v_i and v_j). The edge weights are assumed to be symmetric, i.e. $s_{i,j} = s_{j,i}$. We shall use the following notation: all vectors are column vectors, and are denoted by bold letters; capital letters denote matrices; $\mathbf{v}^T$ denotes the transpose of column vector $\mathbf{v}$. Similarities are collected into a symmetric $n \times n$ matrix S with elements $s_{i,j}$. We shall use $\mathbf{d}$ to denote the vector of degrees of the nodes, i.e.: $d = (d_1, \ldots, d_n)$, where $\mathbf{d}_i = \sum_j s_{i,j}$ $(= \sum_j s_{i,j}$, since S is symmetric); D will denote the diagonal matrix of degrees: $D = \mathrm{diag}(d_1, \ldots, d_n)$.

Spectral methods use the leading eigenvectors of a matrix derived from the similarity information. There are various ways in which this can be done. We used a method which has been proposed recently (14) (http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2001/papers/psgz/AA35.ps.gz), and was shown to give good results in a variety of difficult problems. The algorithm, depicted in Figure 1 in the Supplementary Data (Appendix file), is the following:

(i) From the affinity matrix S construct a symmetric normalized matrix $L = D^{-1/2} SD^{-1/2}$.
(ii) Find a matrix U of eigenvectors: $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K]$, corresponding to the K largest eigenvalues of L.
(iii) Build a matrix Y by renormalizing each of U's rows to have unit length: $Y_{i,j} = \frac{U_{i,j}}{(\sum_j U_{i,j}^2)^{1/2}}$.
(iv) Treating the rows of Y as points in $\mathbb{R}^K$, cluster them into K clusters using K-Means.
(v) Assign node i to cluster k if and only if row i of the matrix Y was assigned to cluster k.

Here we shall give an intuitive explanation of how the algorithm works. To do this let us think of a graph as a system with some dynamics. We can imagine that at any given time there are some particles placed on the vertices of the graph, and at each time step these particles jump from one vertex onto another with a probability related to how similar the two vertices are: the more similar they are, the more likely it is for the particles to jump between them. The path that the particle travels is called random walk and the dynamical process is called Markovian relaxation process.

It is possible to prove that for a fully connected, undirected graph with positive non-zero weights, any particle starting from any position after an infinite number of iterations will always reach the same stationary distribution (15). The stationary distribution, however, is not very interesting for us: in fact, being the distribution that is reached after an infinite number of iterations, it does not give us a lot of information about which areas of the graphs are tighter and relatively isolated from the rest of the graph. However, if the graph exhibits such areas, we expect that during the Markovian relaxation process a particle would spend there some time before eventually jumping onto a different area of the graph. In other words, looking at areas where the particles 'get trapped' and thus spend most of their time, we can identify the tight clusters in the graph that we are looking for [see Figure 2 in the Supplementary Data (Appendix file)]. Now we shall see that we can study the random walk of a particle on the graph and particularly where it spends most of its time before reaching the stationary distribution by analysing the eigenvectors and eigenvalues of a matrix which is derived from the similarity matrix.

Given a graph with n vertices, we can describe the initial position of a particle as a discrete probability distribution over the vertices, that can be written out as a vector $\boldsymbol{p}_0 \in \mathbb{R}^n$ whose components are all positive and sum to 1. Then the probability distribution of the particle at the next time step is given by:

$$\boldsymbol{p}_1 = M \cdot \boldsymbol{p}_0$$

where:

$$M = SD^{-1}$$

M is called Markov transition matrix, and it completely describes the random walk of the particle.

We show in the Supplementary Data (Appendix file) that the probability distribution of the particle after β iterations is given by M^{β}. Therefore M^{β} describes the dynamics of the particle before we reach the stationary distribution and we expect that during this time the particle will spend longer time travelling within the clusters than across clusters. So we would expect to be able to discern in the M^{β} matrix some structure relating to the clusters in the graph. In the appendix we show that M^{β} can be nicely decomposed as:

$$\begin{aligned} M^{\beta} &= D^{1/2}\mathbf{u}_1\ \mathbf{u}_1^{\mathrm{T}}D^{-1/2} + \sum_{i=2}^{n} D^{1/2}\mathbf{u}_i\lambda_i^{\beta}\mathbf{u}_i^{\mathrm{T}}D^{-1/2} \\ &= M^{\infty} + \sum_{i=2}^{n} D^{1/2}\mathbf{u}_i\lambda_i^{\beta}\mathbf{u}_i^{\mathrm{T}}D^{-1/2} \end{aligned} \qquad \mathbf{1}$$

where $\mathbf{u}_i$ and λ_i, $i = 1, \ldots, n$, are the eigenvectors and eigenvalues of the L matrix, which is similar to M.

Therefore we can think of the probability distribution of a particle after β iterations as the sum of two terms: the first term is the distribution in which the particle would end up if the random walk was allowed to run for an infinite number of iterations; while the second term accounts for the fact that the random walk is stopped only after β iterations—it is therefore a perturbation to the stationary distribution.

The matrix constituting the second term of Equation 1 is thus the most 'responsible' for the fact that the particles spend most of their time in certain regions of the graph. Therefore we expect it to contain blocks of positive values, roughly corresponding to the clusters in the graph. And given that such matrix is constituted by a weighted sum of outer products of eigenvectors, these eigenvectors should therefore exhibit the property of being roughly piecewise constant, and components corresponding to elements in the same cluster should have approximately the same value. In fact in (13), it was shown that for K weakly coupled clusters the leading K eigenvectors will be roughly piecewise constant. Finally, notice that the contribution of each eigenvector to the summation in the right hand side of Equation 1 is weighted by the β power of its eigenvalue. Therefore, since all eigenvalues (except the first) are strictly less than one, only the contribution of the first few eigenvectors will be relevant. Figure 3 in the Supplementary Data (Appendix file), shows the eigenvectors and the $\mathbf{u}_i \cdot \mathbf{u}_i^{\mathrm{T}}$ for a few values of i and the matrix constituting the second term of Equation 1 for a simple toy problem.

Therefore it is clear why a simple recipe that has been quite successful at partitioning the graph into two clusters is to assign points based on the sign of the elements of the second eigenvector of the Markov matrix M. This algorithm is called Normalized Cut (or NCut) (16) (citeseer.ist.psu.edu/shi97normalized.html). And the spectral clustering procedure described earlier can be seen as a particular manner of employing the standard K-Means algorithm on the elements of the leading K eigenvectors to extract K clusters simultaneously.

One final comment to be made regards how to choose the number of clusters K. In our implementation, we analysed the eigenvalues of M. We computed the eigengaps which are the ratios of successive eigenvalues. We then applied a predefined threshold ϵ on the eigengaps to select the number of clusters: $K = \min\{i : \lambda_i/\lambda_{i+1} > \epsilon\}$. While this method of selecting K is not perfect, we found it to be adequate for the protein datasets on which we tried it.

Learning a similarity function

In our experiments, in order to cluster a set of proteins, we began by computing the BLAST E-values for each pair of sequences in the set, and collected them into a matrix P. In general P will not be symmetrical, since there are no guarantees that the E-value obtained when aligning protein a with b will the same which is obtained when aligning protein b with a. We transformed P into a symmetrical matrix, S, by assigning to each $s_{i,j}$ and $s_{j,i}$ the higher of the two values $p_{i,j}$ and $p_{j,i}$ This amounts to a conservative interpretation of the E-values. (We also tried assigning to each $s_{i,j}$ the average of the two values $p_{i,j}$ and $p_{j,i}$, but we found the results to be roughly the same.)

We began by applying the spectral clustering algorithm described earlier directly to S [these earlier experiments are described in (17)]. We then realized that results could be improved by integrating some background knowledge into our method. This can be done easily by analysing the statistics of the E-values for the same and different superfamilies.

To do this we randomly extracted superfamilies from SCOP until we had collected about 1000 proteins. We used these sequences to create two sets of distances, one of intra-class distances (which contained 15 544 elements), and one of inter-class distances (with 981 670 elements). We then used this data to train a simple logistic regression model to discriminate between the two classes. We can interpret the posterior probabilities returned by the model as probabilities of evolutionary relatedness. Such probabilities are then fed as affinities into the spectral clustering algorithm. Note that the training of the logistic regression model needs to be done only once. Also we point out that those proteins which were used during this phase were not used later for testing the performance of the clustering algorithm. A scheme of the complete method that we used is shown in Figure 6.

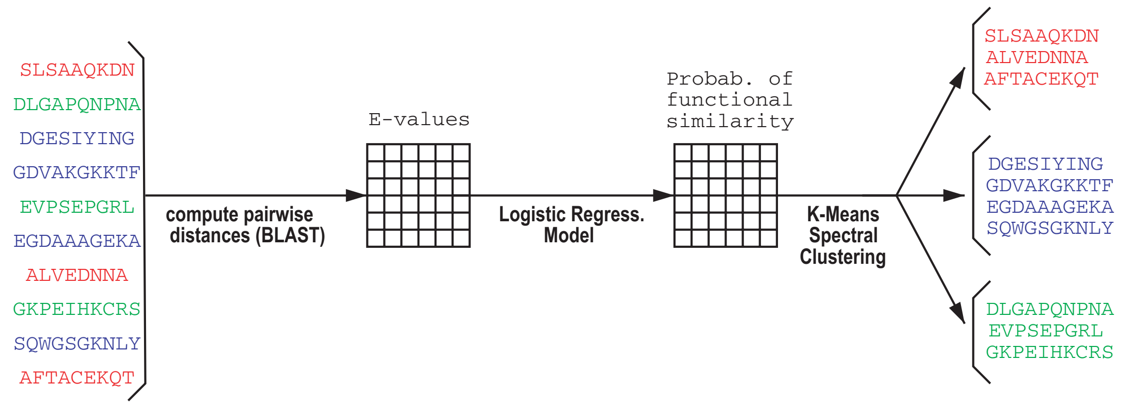


Figure 6. The scheme of the method that we used in our experiments. Proteins of the same colour are evolutionary related.

Performance measure

For a certain protein set, let us call K the categorization into superfamilies provided by SCOP, and let us denote by λ the clustering returned by a certain clustering algorithm for that set. Let n be the total number of proteins in the dataset; n^h the number of proteins in superfamily K_h according to K; n_l the number of proteins in cluster C_l according to λ; and n_l^h the number of proteins that are in cluster l according to λ as well as in class h given by K. The Precision is then defined as the fraction of correctly retrieved proteins out of all the proteins in the cluster:

$$P(C_l, K_h) = \frac{n_l^h}{n_l}$$

The Recall is defined as the fraction of correctly retrieved proteins out of all the proteins in the superfamily:

$$R(C_l, K_h) = \frac{n_l^h}{n^h}$$

The F-measure combines Precision and Recall with equal weights. For the entire clustering, the total F-measure is defined as:

$$F(\lambda, K) = \frac{1}{n} \sum_h n^h \max_l \frac{2n_l^h}{n_l + n^h}$$

Algorithms for comparison: GeneRAGE, hierarchical clustering and TribeMCL

We implemented a simplified version of GeneRAGE (2), a CCA-related method which has recently been introduced for clustering protein sequences. The starting point of the algorithm is a set of pairwise BLAST E-values between sequences. It then binarizes such matrix using a threshold value of 10^{-6}. A connected component search is then performed on the binarized matrix to retrieve the clusters. The original GeneRAGE presents also other features. For example, it uses the CAST algorithm to mask composition bias; it solves the problem of asymmetric E-values by performing an additional dynamic programming alignment between asymmetric elements; and putative multi-domain proteins are identified considering where transitivity does not hold. However these aspects are not relevant for the discussion addressed here, and so were not included in our implementation.

Hierarchical clustering (18) is a clustering method that begins with the individual data points and then builds a tree by iteratively merging the closest points until only one is left. The links in the resulting tree are then cut in order to obtain separate clusters. There are therefore two matters to be addressed by a hierarchical clustering procedure. The first one is how to determine the distance from a newly merged data point to the rest of the dataset [various choices for this distance have been analyzed e.g. in (19)]. The second one is how to cut the tree into different clusters.

We tried several different choices for these parameters, and here we presented the best results that we obtained on our datasets. For these we used the average distance metric, in which the distance between two clusters is given by the average distance between all pairs of items where one member of a pair belongs to each cluster. This distance metric was chosen after comparing it with several other metrics using the cophenetic correlation coefficient (20). We then cut those links in the tree that were greater than 10^{-6}.

As for our spectral method, the starting point of the Markov Cluster (MCL) algorithm (21) is to build a Markov transition matrix, M. The two algorithms differ in the way in which they propagate the Markov chain on the graph: while our spectral clustering analyses perturbations to the stationary distribution of M, MCL modifies the random walks to promote the emergence of clusters in the graph.

We discussed earlier that a cluster in a graph is characterized by the presence of many strong edges between its members and perhaps few weak connections with other clusters. If a graph contains such clusters, then it is unlikely that random walks based on Markov transition probabilities will jump between two clusters too frequently. The idea behind the MCL algorithm is the following: if the random walks can somehow be biased, say by pruning weak edges and reinforcing strong edges simultaneously, clusters may emerge from the graph.

MCL induces this bias in random walks by alternating between two operators called expansion and inflation. Expansion is similar to the propagation we discussed in the introduction, where the Markov transition matrix M is raised to a power. Inflation corresponds to taking powers of M entry-wise and it is followed by a normalization step, so that the matrix elements in each column sum up to 1 again. Expansion makes the differences between nodes less distinguishable, while inflation has the effect of boosting probabilities on strong intra-cluster walks and demoting weak inter-cluster walks. The inflation process is controlled by a parameter r. Increasing r has the effect of increasing the tightness of the clusters. Iterative application of expansion and inflation operators approaches an equilibrium state and the resulting graph is then examined for cluster information. We tried several different choices for the r parameter, and here we presented the best results that we obtained on our datasets.

MCL constitutes the core component of TribeMCL, an algorithm used for clustering protein sequences (7). In TribeMCL, the similarity measure between two proteins is built on the BLAST E-value. In particular, a similarity matrix S is put together by taking the average of the pairwise $-\log_{10}$ (E-value) values between two proteins, thus resulting in a symmetric matrix. The similarity matrix is then converted into a Markov transition matrix for the application of the expansion and inflation operators.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

This project began as a collaboration with Chakra Chennubhotla. The authors would especially like to thank Chakra for many crucial discussions and for always sharing his enthusiasm, inspiration and criticism. Also thanks to Paul Grosvenor, Mike Curtis, Mark Gerstein, Rajkumar Sasidharan, Madan Babu, Aroul Selvam, Zoubin Ghahramani, Jaz Kandola and Allan Jepson for many important and stimulating discussions. Thank you to Allan Jepson for the code for Connected Component Analysis and to Ian Nabney for the Netlab package (22). Funding to pay the Open Access publication charges for this article was provided by Queen Mary, University of London. A.P. was supported from MRC/DTI (grant G0100109) and J.A.C. from the Special Trustees of St Bartholomews and the Royal London Hospital.

Conflict of interest statement. None declared.

REFERENCES

1. Ballard,D. and Brown,C. (1982) *Computer Vision*. Englewood Cliffs: Prentice-Hall.
2. Enright,A.J. and Ouzounis,C.A. (2000) GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, **16**, 451–457.
3. Krause,A., Stoye,J. and Vingron,M. (2000) The SYSTERS protein sequence cluster set. *Nucleic Acids Res*., **28**, 270–272.
4. Yona,G., Linial,N. and Linial,M. (2000) ProtoMap: automatic classification of protein sequences and hierarchy of protein families. *Nucleic Acids Res*., **28**, 49–55.
5. Pipenbacher,P., Schliep,A., Schneckener,S., Schonhuth,A., Schomburg,D. and Schrader,R. (2002) ProClust: improved clustering of protein sequences with an extended graph-based approach. *Bioinformatics*, **18**, S182–S191.
6. Murzin,A.G., Brenner,S.E., Hubbard,T. and Chothia,C. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol*., **247**, 536–540.
7. Enright,A.J., Van Dongen,S. and Ouzounis,C.A. (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*., **30**, 1575–1584.
8. Brenner,S.E., Koehl,P. and Levitt,M. (2000) The ASTRAL compendium for protein structure and sequence analysis. *Nucleic Acids Res*., **28**, 254–256.
9. Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Bio*., **215**, 403–410.
10. Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389–3402.
11. Pearl,F., Todd,A., Sillitoe,I., Dibley,M., Redfern,O., Lewis,T., Bennett,C., Marsden,R., Grant,A., Lee,D. *et al*. (2005) The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis. *Nucleic Acids Res*., **33**, 247–251.
12. Chennubhotla,C. and Jepson,A. (2002) Half-lives of eigenflows for spectral clustering. In Becker,S., Thrun,S. and Obermayer,K. (eds), *Advances in Neural Information Processing Systems 15, NIPS*. pp. 689–696.
13. Meila,M. and Shi,J. (2001) A random walks view of spectral segmentation. *Proceedings International Workshop on AI and Statistics, AISTATS*.
14. Ng,A., Jordan,M. and Weiss,Y. (2001) On spectral clustering: analysis and an algorithm. In Dietterich,T.G., Becker,S. and Ghahramani,Z. (eds), *NIPS 14, Advances in Neural Information Processing Systems 14*. pp. 849–856.
15. Fan,R.K. (1997) Chung Spectral Graph Theory. *Am. Math. Soc*., **92**, 1–212.
16. Shi,J. and Malik,J. (2000) Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*. **8**, 888–905.
17. Alberto,P., Chennubhotla,C., Casbon,J. and Saqi,M. (2003) Spectral clustering of protein sequences. *International Joint Conference on Neural Networks, IJCNN*. Portland, OR, USA.
18. Everitt,B.S. (1993) *Cluster Analysis*. 3rd edn. Edward Arnold, London.
19. Sasson,O., Linial,N. and Linial,M. (2002) The metric space of proteins: comparative study of clustering algorithms. *Bioinformatics*, **18**, S14–S21.
20. Farris,J.S. (1969) On the cophenetic correlation coefficient. *Syst. Zool*., **18**, 279–285.
21. van Dongen,S. Graph Clustering by flow simulation. Ph. D. Thesis, University of Utrecht, 2000, The Netherlands.
22. Nabney,I.T. Netlab: Algorithms for Pattern Recognition Springer, 2002.