CLUSS2: an alignment-independent algorithm for clustering protein families with multiple biological functions

Abdellali Kelil* and Shengrui Wang

Faculty of Sciences,
ProspectUS Laboratory,
Department of Computer Sciences,
University of Sherbrooke, Sherbrooke, J1K 2R1 QC, Canada
E-mail: Abdellali.Kelil@USherbrooke.ca
E-mail: Shengrui.Wang@USherbrooke.ca
*Corresponding author

Ryszard Brzezinski

Faculty of Sciences,
Microbiology and Biotechnology Laboratory,
Department of Biology,
University of Sherbrooke, Sherbrooke, J1K 2R1 QC, Canada
E-mail: Ryszard.Brzezinski@USherbrooke.ca

Abstract: CLUSS is an algorithm proposed for clustering both alignable and non-alignable protein sequences. However, CLUSS tends to be ineffective on protein datasets that include a large number of biochemical activities. To overcome this difficulty, we propose in this paper a new algorithm, named CLUSS2 that scales better with the increase of the number of biochemical activities. CLUSS2 differs from CLUSS in many ways including protein sequences representation, conserved motifs extraction and time efficiency. Our experiments show that CLUSS2 more accurately highlights the functional characteristics of the clustered families, especially for those with a large number of biochemical activities.

Keywords: clustering; similarity measure; biological function; non-alignable.

Reference to this paper should be made as follows: Kelil, A., Wang, S. and Brzezinski, R. (2008) 'CLUSS2: an alignment-independent algorithm for clustering protein families with multiple biological functions', *Int. J. Computational Biology and Drug Design*, Vol. 1, No. 2, pp.122–140.

Biographical notes: Abdellali Kelil is currently a PhD candidate at the University of Sherbrooke, and a member of the ProspectUS data mining and bioinformatics laboratory at the same university. His current research interests include protein analysis and classification and functional prediction.

Shengrui Wang received his PhD from the Institut National Polytechnique in Grenoble, France. He is Director of ProspectUS laboratory at the University of Sherbrooke. His research interests include pattern recognition, data mining, artificial intelligence and information retrieval.

Ryszard Brzezinski received his PhD from the University of Warsaw, Poland. He is Director of a laboratory of molecular biotechnology, environmental microbiology and bioinformatics at the University of Sherbrooke.

1 Introduction

To predict the biochemical activity of a newly sequenced or not yet characterised protein sequence, it is necessary to compare its biochemical properties to those of functionally well-characterised protein sequences, to assign this protein to one of the protein families. However, this is not sufficient to attribute a biochemical activity to the protein with a high degree of confidence, since a single family can include a number of biochemical activities. A possible solution for assessing the differences in cases where protein sequences from the same family have different activities is clustering. The literature reports many clustering approaches to the task of grouping protein families into subfamilies of protein sequences that are functionally more closely related. However, clustering protein sequences remains a difficult challenge, especially for sequences whose alignment is not biologically validated (i.e., hard-to-align or totally non-alignable sequences), such as tandem-repeat, multi-domain and circular-permutation proteins, for which alignment-dependent algorithms do not yield biologically plausible clustering results. The main reason is that these algorithms use an alignment process based on matching motifs in corresponding positions, whereas non-alignable proteins often have similar or conserved domains in non-corresponding positions. A more detailed discussion on why these proteins are difficult to align and hard to cluster is given in Kelil et al. (2007a). To the best of our knowledge, the only alignment-independent clustering algorithm which is effective on both alignable and non-alignable protein sequences is the CLUSS algorithm which we proposed recently in Kelil et al. (2007b).

CLUSS is based on a measure named SMS which we designed specifically to compute the similarity between two protein sequences. The SMS measure depends on identical matched motifs and is effective for both alignable and non-alignable protein sequences, a property that plays a key role in CLUSS. Compared to alignment-dependent algorithms, CLUSS highlights the characteristics of the biochemical activities and modular structures of the clustered protein sequences. However, it has a tendency to be less effective when applied to large protein datasets with many biochemical activities. CLUSS also suffers from another problem. Despite the use of optimisation techniques to speed up the matching of motifs, it is still not possible to reduce the worst-case complexity to a linear time in the SMS computation, which remains slow, especially for large protein datasets. All these factors prevent CLUSS from being effective on large protein datasets.

In this paper we propose a new algorithm for clustering protein sequences, which we have named CLUSS2. CLUSS2 is similar to CLUSS in that both are hierarchical clustering algorithms and both aim primarily to cluster hard-to-align sequences. However, CLUSS2 differs significantly from CLUSS in two main respects. First, CLUSS2 is based on a new measure tSMS that extends SMS for computing similarity between protein sequences. The tSMS measure allows the matching of similar motifs, rather than imposing identical matches as in SMS. tSMS is computed based on a new algorithm for extracting matched motifs, which is the main reason for its increased

efficiency. The second major difference from CLUSS is that CLUSS2 applies Singular Value Decomposition (SVD) techniques to the similarity matrix obtained from tSMS, to create a representation of each protein sequence in a vector space. This transformation allows the application of vector operations during the clustering process. One important advantage is that this yields a representative (centroid) for each cluster; another is the possibility of further reducing the runtime by using approximate representations. CLUSS2 is much faster and more effective than CLUSS, especially for large protein datasets with a large number of biological activities.

To show the effectiveness of CLUSS2, we performed extensive clustering experiments on the COG and KOG databases, which contain phylogenetic classifications of proteins encoded in complete genomes (Tatusov et al., 2003), and also on reference sequence proteins encoded by complete prokaryotic and chloroplast plasmids and genomes, known as the Protein Clusters (PC) database, available at the NCBI website: ftp://ftp.ncbi.nih.gov/genomes/Bacteria/CLUSTERS/. To demonstrate its ability to deal with hard-to-align sequences, we tested it on the $(\alpha/\beta)_8$ barrel proteins group, belonging to the Glycoside Hydrolases (GH) family (Coutinho and Henrissat, 1999). In addition, we carried out experimental comparisons with a variety of mainstream algorithms including the BlastClust program (Dondoshansky and Wolf, unpublished), which belongs to the standalone BLAST package used to cluster either protein or nucleotide sequences, available from the NCBI website ftp://ftp.ncbi.nih.gov/blast/, and the well-known algorithms TRIBE-MCL (Enright et al., 2002) and gSPC (Tetko et al., 2005). These comparisons were made on hard-to-align and easy-to-align protein sequences. The results of these experiments show advantages of CLUSS2 in yielding more significant clusters of proteins with similar functional activities, especially for large protein datasets with a variety of biochemical activities.

2 The new similarity measure tSMS

The measure SMS, used in CLUSS to measure the similarity of a pair of protein sequences X and Y, was defined based on a key set of strictly matched subsequences (i.e., identical amino acids) of maximal length between the sequences X and Y, denoted by $E_{X,Y}$. Unlike other word-counting methods, which measure similarity by detecting multiple occurrences and handling them according to their matching scores, as in the well-known algorithm Blast (Altschul et al., 1990), which uses the SHP criterion, SMS takes into account both the position and the inclusion of the matched subsequences.

The fact that we utilise a single similarity value which includes all of the identical matches as well as matched motifs from positions which, while non-equivalent according to the primary structure, might well be equivalent when viewed in terms of secondary and tertiary structure, allows us to take advantage of certain information included in the secondary and tertiary structure. Certainly, taking into account only identical motifs may lead us to overlook some important information in computing similarity. But at the same time, it also filters out noise (i.e., similarities due to chance) from our similarity measure. We believe that for protein datasets which include a small number of biochemical activities, the overlooked information is relatively insignificant compared with the noise-filtering effect.

The experimental results reported in our recent studies (Kelil et al., 2007a, 2007b) seem to confirm the advantage of this strategy for such datasets. On the other hand, these

studies show that the strategy is not very efficient on protein datasets with a large number of biochemical activities. This suggests that the proportion of overlooked similarity information may become more significant as the number of biochemical activities increases, undermining the noise-filtering advantage.

The similarity measure SMS also suffers from problems of speed. Although we utilised a technique to speed up the extraction of significant motifs, the variable length of the matched sequences made it not impossible to reduce the worst-case complexity to a linear time using this technique.

In this paper, we propose a new similarity measure named for 'tolerant SMS' (tSMS) which generalises SMS in terms of tolerance to mismatches and scales well with increase in the number of biochemical activities. Also, tSMS is much faster than SMS; this is made possible by the optimisation techniques used, which have reduced the worst-case complexity to a linear time.

2.1 The matching set

We will use $|\bullet|$ to express the length of a sequence. Let X and Y be two protein sequences whose similarity we want to measure, belonging to the protein family F which contains N protein sequences. Let x and y be two subsequences of the same length, belonging to X and Y, respectively. We use $\Gamma_{x,y}$ to represent the matched subsequence of x and y. We use l to represent the minimum number of matched residues between x and y that $\Gamma_{x,y}$ must include; at the same time, l is also the maximum number of non-matched residues allowed in $\Gamma_{x,y}$. A detailed discussion on the choice of the value of l was given in Kelil et al. (2007a). The length l is used with the aim of detecting and utilising the significant motifs best conserved during evolution and minimising the influence of motifs that occur by chance. We use m (chosen by the user) to represent the minimum substitution score that two matched residues must have in order to be considered similar, or to be considered allowable in $\Gamma_{x,y}$. For X and Y, we define the set of all matched subsequences $\Gamma_{x,y}$ denoted by $E^{l,m}_{X,Y}$, as follows:

$$E^{l,m}_{X,Y} = \left\{ \Gamma_{x,y} \;\middle|\; \begin{array}{l} \left|\Gamma_{x,y}\right| = |x| = |y| \\ \text{card}(\{\Gamma_{x,y} \,|\, x_i = y_i\}) \geq l \\ \text{card}(\{\Gamma_{x,y} \,|\, x_i \neq y_i\}) \geq l \\ \forall i \leq \left|\Gamma_{x,y}\right|, (x_i \in x) \wedge (y_i \in y) \Rightarrow M(x_i, y_i) \geq m \\ (\forall\, _{x',y'} \in E^{l,m}_{X,Y}) \wedge (\Gamma_{x',y'} \neq \Gamma_{x,y}) \Rightarrow (x' \not\subset x) \vee (y' \not\subset y) \end{array} \right. . \quad (1)$$

Here m is one of the substitution matrices (chosen by the user) and i is used to identify the ith position in a subsequence. The variables x_i and y_i are simply the ith amino acids belonging to subsequences x and y, respectively. $M(x_i, y_i)$ is the substitution score of the ith amino acids of the subsequences x and y. The constant m is a minimum value that the score between amino acids x_i and y_i must have to be considered as matched. The symbols x' and y' in the formula are simply used as variables, in the same way as x and y. The expression ($.\not\subset.$) means that the element to the left of the symbol is not included in the one to the right, either in terms of the composition of the subsequences or in terms of their respective positions in their protein sequence. The role of l is to detect and make use of the significant motifs best conserved during evolution and to minimise

the influence of the motifs that occur by chance. The matching set $E_{X,Y}^{l,m}$ thus includes the significant motifs that correspond to matched protein subsequences that are more likely to be similar due to conservation phenomena and not due to chance. The matching set will be used to compute the matching score of the pair of sequences. Here are a few detailed explanations about Formula 1:

- $\left|\Gamma_{x,y}\right| = |x| = |y|$ means that the matched motif $\Gamma_{x,y}$ as well as the matched subsequences *x* and *y* include the same number of amino acids.
- $\text{card}(\{\Gamma_{x_i,y_i} \mid x_i = y_i\}) \geq l$ means that the matched motif $\Gamma_{x,y}$ must include at least *l* identical similar residues according to the threshold *m*.
- $\text{card}(\{\Gamma_{x_i,y_i} \mid x_i \neq y_i\}) \leq l$ means that the matched motif. $\Gamma_{x,y}$ can include at most *l* non-identical residues according to the threshold *m*.
- $\left|\Gamma_{x,y}\right| \geq l$ means that the matched subsequences $\Gamma_{x,y}$ must have the minimum length *l*.
- $\forall i \leq \left|\Gamma_{x,y}\right|, (x_i \in x) \wedge (y_i \in y) \Rightarrow M(x_i, y_i) \geq m$ means that the subsequences *x* and *y* must not include matched residues with a substitution score less than a threshold *m*.
- $(\forall \Gamma_{x',y'} \in E_{X,Y}^{l,m}) \wedge (\Gamma_{x',y'} \neq \Gamma_{x,y}) \Rightarrow (x' \not\subset x) \vee (y' \not\subset y)$ means that for any matched subsequences $\Gamma_{x,y}$ and $\Gamma_{x',y'}$ belonging to $E_{X,Y}^{l,m}$, $\Gamma_{x',y'}$ and $\Gamma_{x,y}$ being different implies that $\Gamma_{x',y'}$ is not included in $\Gamma_{x,y}$ either in terms of the composition of their corresponding subsequences or in terms of their respective positions in their protein sequences according to the partial order induced by set inclusion. In other words, each of the $\Gamma_{x,y}$ in $E_{X,Y}^{l,m}$ is maximal.

To summarise, the formula means that the matching set $E_{X,Y}^{l,m}$ contains all the matched subsequences $\Gamma_{x,y}$ of maximal length (i.e., at least *l* identical matched residues and at most *l* non-identical matched residues) between the sequences *X* and *Y*, with a tolerance to mismatches determined by *m*.

The formula $E_{X,Y}^{l,m}$ adequately describes some known properties of polypeptides and proteins. First, protein motifs (i.e., series of defined residues) determine the tendency of the primary structure to adopt a particular secondary structure, a property exploited by several secondary-structure prediction algorithms. Such motifs can be as short as four residues (for instance, those found in β-turns), but the propensity to form an α-helix or a β-sheet is usually defined by longer motifs. Second, our proposal to take into account multiple occurrences of a particular motif reflects the fact that sequence duplication is one of the most powerful mechanisms of gene and protein evolution. If a motif is found twice or more in a protein, it is more probable that it was acquired by duplication of a segment from a common ancestor than by acquisition from a distant ancestor.

2.2 Definition of the similarity measure tSMS

Our primary concern is to develop an approach that will enable us to cluster hard-to-align protein sequences such as circularly-permuted, multi-domain and tandem-repeat protein sequences. For such sequences, the alignment-dependent approaches usually fail to yield biologically suitable results. In fact, the hard-to-align proteins often have similar and

conserved domains in non-equivalent positions in the primary structure, which makes them difficult to align. However, these domains might well be in equivalent positions when viewed in terms of secondary and tertiary structure. In the absence of explicit identification of such positions in our alignment-free approach to similarity computation, we adopted the strategy of matching all the conserved domains, even those on non-equivalent positions. The reason is that, with a suitable value of the minimum threshold '*l*' for matched motifs, which allows us to detect and make use of the significant motifs best conserved during evolution and to minimise the influence of those motifs that occur by chance, it is more probable that we will effectively match motifs that are similar due to conservation rather than to random phenomena.

For a protein sequence that comprises a number of significant motifs that were better conserved during evolution, each motif contributes in a complex way to provide one or more biological functions. A mutation in one of the conserved motifs can significantly alter or even eradicate the biological activity of the protein, while in another conserved motif it might only slightly decrease the expression of the biological function. So, we make use of a substitution matrix to emphasise the fact that each conserved motif can be involved to a different degree in a biological activity.

Let M be a substitution matrix, and Γ matched subsequence belonging to the matching set $E_{X,Y}^{l,m}$ We define a weight $W(\Gamma)$ for the matched subsequence Γ, to quantify its importance compared to all the other matched subsequences of $E_{X,Y}^{l,m}$ as follows:

$$W(\Gamma) = \sum_{i=1}^{|\Gamma|} M(\Gamma|i|\Gamma|i|) \tag{2}$$

where $\Gamma[i]$ is the ith amino acid of the matched subsequence Γ, and $M(\Gamma[i], \Gamma[i])$ is the substitution score of this amino acid with itself. Here, in order to make our measure biologically plausible, we use the substitution concept to emphasise the relation that binds one amino acid with itself. The value of $M(\Gamma[i], \Gamma[i])$ (i.e., within the diagonal of the substitution matrix) estimates the rate at which each possible amino acid in a sequence remains unchanged over time. For the pair of sequences X and Y, we define the matching score $S_{X,Y}$, understood as representing the substitution relation of the conserved regions in both sequences, as follows:

$$S_{X,Y} = \sum_{r \in E_{X,Y}^{l,m}} W(\Gamma) / \max(|X|, |Y|). \tag{3}$$

Which is our new similarity measure tSMS for a pair of protein sequences X and Y.

2.3 Conservability vs. mutability

The scoring of identical matches with a substitution matrix in SMS reflects the conservability of matched residues. The term conservability is more appropriate than mutability. The nuance is significant for SMS. In fact, protein sequences to be compared contain conservability and mutability information. In the case of easy-to-align protein sequences, both conservability and mutability information can be obtained, while in the case of hard-to-align protein sequences mutability information is difficult to obtain. This is due to some known problems, such as the problem of repeats and the problem of substitutions; for details see Higgins (2004). To the best of our knowledge, existing alignment-based algorithms fail to effectively capture conservability and mutability

information in hard-to-align protein sequences. On the other hand, the experimental results reported in Kelil et al. (2007a) show that the use of only conservability information allows SMS to deal with hard-to-align sequences better than the alignment-based algorithms. Experimental results also show that SMS handles easy-to-align protein sequences equally well as the alignment-based algorithms. This suggests that the utility of conservability might be much more significant than is generally believed. However, the experiments conducted showed that, as the number of biochemical activities increases, the strategy of capturing only the conservability information becomes increasingly insufficient to obtain an accurate similarity measure. Therefore, the use of mutability information becomes inevitable to overcome this drawback. In tSMS, both conservability and mutability information are captured and used to measure the similarity.

2.4 Computational complexity

To compute tSMS, we have made use of a variant of the data structure known as the '*Suffix Tree*' (Weiner, 1973), developed by Cole et al. (2006) and named the '*Suffix Tray*'. The Suffix Tree is a well-known approach to solving the problem of string matching in linear time. Given the question of how many occurrences of a pattern P there are in a string T and where they occur, the Suffix Tree allows an answer to be generated in $O(|P|+z|T|)$ time and with $O(|T|)$ space, where z is the number of occurrences of the pattern P in the text T. With the Suffix Tray, on the other hand, the same task can be performed in $O(|P|+\log\sum)$ with the same space complexity $O(|T|)$ as for the Suffix Tree. Here $\sum$ is the alphabet size. For our case $\sum = 20$, which is the number of amino acids. The fact that the Suffix Tray performs the matching in a time independent of $|T|$ is very advantageous for speeding up our algorithm.

Let X and Y be a pair of protein sequences to be compared. We start by building the Suffix Trays corresponding to the individual sequences, T_X and T_X, which takes time and space $O(|X|)$ and $O(|X|)$, respectively. These Suffix Trays are trees of $O(|X|)$ and $O(|X|)$ nodes, containing all the suffixes of the protein sequences X and Y, respectively. Instead of matching X and Y, which takes time $O(|X|\times|Y|)$ we perform the same task by matching only the suffixes of T_X with those of T_Y, or vice-versa, as follows:

Let $T_X = \{x_1, x_2, \ldots x_t\}$ be the set of all suffixes of T_X, where t is the number of possible suffixes. Finding all the occurrences (i.e., exact matching) of a suffix x_i out of the Suffix Tray T_Y takes time $O(|P|+\log 20)$. Let $\overline{k}$ be the average number of possible matches of all amino acids according to the chosen value of m (in Formula 1) and the chosen substitution matrix. If we consider that we allow a restricted number of matches per residue (see Table 1) and a restricted number of mismatches per matched motif (i.e., $\leq l$), in the worst case, there exist $\overline{k}^l$ possible transformations of x_i, which implies that the pattern x_i will have to be matched $\overline{k}^l$ times with the Suffix Tray T_Y. This has a time complexity of $\overline{k}^l O(|P|+\log 20)$. Since both k and l are constants, and are usually small values, the coefficient $\overline{k}^l$ is also a constant. Performing the matching between all T_X suffixes and the Suffix Tray thus takes time $\overline{k}^l O(|x_1|+\log 20)+\overline{k}^l O(|x_2|+\log 20)+\cdots+\overline{k}^l O(|x_t|+\log 20)=\overline{k}^l O(|X|)$, which is also linear.

Table 1 Number of possible matches for each amino acid with different values of *m*

Amino acids	*BLOSUM62*			*PAM250*		
	m = 0	*m = 1*	*m = 2*	*m = 0*	*m = 1*	*m = 2*
A	6	2	1	10	5	1
C	2	1	1	3	1	1
D	5	3	2	10	6	4
E	8	4	3	10	5	3
F	6	3	2	6	4	3
G	4	1	1	7	4	1
H	6	3	2	9	6	4
I	5	4	3	5	5	5
K	6	4	2	10	4	2
L	5	4	3	5	5	5
M	6	4	2	7	4	4
N	10	4	1	11	7	3
P	1	1	1	7	3	1
Q	9	4	2	9	7	4
R	6	3	2	9	5	4
S	9	4	1	11	6	1
T	5	2	1	11	3	1
V	6	4	2	6	4	4
W	3	3	2	4	2	2
Y	4	4	4	5	2	2
Average $\overline{k}$	5.6	3.1	1.9	7.8	4.4	2.7

Depending on the m value (i.e., column), each amino acid (i.e., row) has a limited number of possible matches; each $\overline{k}$ value is the average of the corresponding column values.

3 The new clustering algorithm CLUSS2

CLUSS2 is composed of three main stages. The first one consists in building a pairwise similarity matrix *S* using our new similarity measure tSMS. The second consists in building a phylogenetic tree according to this matrix, using a new hierarchical clustering approach based on spectral decomposition. The third consists in identifying subfamily nodes from which leaves are grouped into subfamilies.

In the algorithm CLUSS (Kelil et al., 2007b), we used a classical clustering approach by directly making use of the pairwise similarity matrix. In the present version we have developed a new and original hierarchical algorithm, inspired by the LSA approach, for more details see Berry and Fierro (1996). We take advantage of this approach by extracting global information from a large number of protein sequences rather than carrying out a pairwise comparison. We have chosen to keep the name CLUSS, since both versions have the same basic principles, and they are inspired from the same idea.

3.1 Stage 1: Similarity matrix

Using one of the known substitution score matrices, such as BLOSUM62 or PAM250, and our new similarity measure tSMS, we compute S, the $N \times N$ pairwise similarity matrix, where N is the number of sequences of the protein family F to be clustered, and $S_{i,j}$ is the similarity between the ith and the jth protein sequences of F. By using tSMS, the construction of the pairwise similarity measure matrix S becomes much faster, since we transform all the N protein sequences into Suffix Trays only once before the pairwise matching of the protein sequences. Both the transformation of each protein sequence and the matching of two protein sequences take linear time with respect to sequence length, as seen in Section 2.1.

3.2 Stage 2: Phylogenetic tree

Using spectral decomposition on the pairwise similarity matrix S, we obtain a set of vectors. Each of the vectors is used to represent a protein sequence in the new vector space resulting from the decomposition of S. Such a representation is valid in the sense that the similarity between each pair of sequences from the original similarity matrix S is equal or approximately equal to the similarity between the corresponding vectors measured by the inner product function. This representation facilitates the subsequent (hierarchical) clustering. In fact, a cluster will be represented by only one vector; cluster merging can be easily performed by adding two vectors; and the similarity between two clusters can then be estimated by the cosine similarity function. This stage is composed of three steps, as follows.

3.2.1 Step1: Spectral decomposition of the similarity matrix S

We will utilise the theorem in linear algebra, which states that any $R \times C$ matrix A whose number of rows R is greater than or equal to its number of columns C can be written as the product of an $R \times C$ column-orthogonal matrix U, a $C \times C$ diagonal matrix Z with non-negative elements, which are the singular values, and the transpose of an $R \times R$ orthogonal matrix V. This decomposition is named the Singular Value Decomposition (SVD). The matrix A can be written as follows:

$$A = U \times Z \times V^T. \tag{4}$$

We apply the SVD to the squared pairwise similarity matrix $S_{N\times N}$, which is decomposed into the product of three $N \times N$ matrices U, Z and V. The first of these, U, is a left singular matrix describing the original row entities as vectors of derived orthogonal factor values; the second, Z, is a diagonal matrix containing non-negative scaling values; and the third, V, is a right singular matrix describing the original column entities in the same way as the first matrix. Since the matrix Z contains non-negative singular values, the SVD of S can be written in the following form:

$$S = (U \times \sqrt{Z}) \times (\sqrt{Z} \times V^T). \tag{5}$$

For the special case where S is a square and symmetric matrix with a diagonal including much larger values than the rest of the matrix (as is the case here), the matrix S

is very likely to be a semi-definite positive matrix, or at least very close to that. We can thus write Formula 6 in the form:

$$S \simeq (U \times \sqrt{Z}) \times (\sqrt{Z} \times U^T). \tag{6}$$

We can write:

$$S \simeq (U \times \sqrt{Z}) \times (U \times \sqrt{Z})^T. \tag{7}$$

We define an $N \times N$ matrix $E = U \times \sqrt{Z}$, for which each row $E_i = U_i \times \sqrt{Z}$. Now each protein sequence i belonging to the protein family F to be clustered is represented by the vector E_i in the new vector space, mapped by the matrix E. Therefore, the similarity measure $S_{X,Y}$ between a pair of sequences X and Y is now equal or approximately equal to the inner product $<E_X, E_Y>$. The idea of mapping the protein sequences onto a vector space is based on the conservability of distance. This transformation allows us to apply vector operations during the clustering process and obtain (and maintain) a representative for each subcluster. The transformation, as discussed in LSA, also allows us to take advantage of transitivity in the similarities between pairs of proteins (documents, in LSA).

It is possible to take further advantage of this representation. In fact, by taking into account only the K (where $K \leq N$) largest non-negative singular values from the $N \times N$ matrix Z, and their corresponding singular vectors from the $N \times N$ matrices U and V, we get the rank K approximation of S with the smallest error according to the Frobenius norm (Golub and Loan, 1996). The matrices U, Z and V are reduced to $N \times K$, $K \times K$, and $N \times K$ matrices, respectively. Thus, the spectral decomposition approach maps the protein vectors onto a new multidimensional space in which the corresponding vectors are the rows of the $N \times K$ matrix E. Reducing the K value significantly speeds up the clustering process. In the experiments carried out in this paper, we have not exploited the strategy of reducing the value of K, since we set it to $K = N$ because we wanted to concentrate our efforts on the accuracy of the new clustering approach adopted in CLUSS2. However, we will do it extensively in a future work.

3.2.2 Step 2: Phylogenetic tree

Starting from vectors $E_1, E_2, \ldots, E_N$, each of which is considered as the root node of a subtree containing only one node, we initialise the similarity between any pair of nodes by the cosine product of corresponding vectors. We iteratively join a pair of root nodes in order to build a bigger subtree. At each iteration, a pair of root nodes is selected if they are the most similar root nodes (i.e., corresponding vectors have the largest cosine product). This process ends when there remains only one subtree, which is the phylogenetic tree.

Now we introduce the concept of co-similarity for ranking the nodes in the phylogenetic tree. Let L and R be a pair of nodes (L for left and R for right) belonging to the phylogenetic tree. By taking into account information about the neighbourhood around each of the nodes L and R, the concept of co-similarity reflects the cluster compactness of all the sequences (i.e., leaf nodes) in the subtree. In fact, its value is inversely proportional to the within-cluster variance. As the subtree becomes larger, the co-similarity tends to become smaller, which means that the sequences within the subtree become less similar and the difference (i.e., separation) between sequences in different

clusters becomes less significant. In simpler terms, the co-similarity of a particular node is a measure of the balance between its two child nodes. Before the construction of the phylogenetic tree, all co-similarities (of the leaves) are initialised to zero.

Let L and R be the two most similar root nodes at a given iteration step; they are joined together to form a new subtree P (P *f*or parent), which thus has two children, L and R, such that E_P is its corresponding vector. The new root node P has the following definitions:

$$E_P = E_L + E_R \quad \text{and} \quad c_P = \frac{\| E_L \| \times \| E_R \|}{\| E_L \| + \| E_R \|} \tag{8}$$

where E_L, E_R and E_P are vectors corresponding to the root nodes L, R and P respectively, and c_P is the co-similarity of P. The norms $\|E_L\|$ and $\|E_R\|$ depend on the number and proximity of leaves belonging to the subtrees L and R, respectively, and they measure how well F is represented by each one of these particular subtrees. According to this definition, the value of a norm is large if the corresponding subtree is more representative and small if it is less representative.

We assign a '*length*' value to each of the two branches connecting L and R to P. These values are the estimate of the phylogenetic distance from the individual nodes L and R to their parent P in the tree. This distance has no strict mathematical sense; it is merely a measure of the evolutionary distance between the nodes. It is comparative to the notion of dissimilarity. We calculate it as follows:

$$d_{L,P} = \frac{\| E_R \|}{\| E_L \| + \| E_R \|} \quad \text{and} \quad d_{R,P} = \frac{\| E_L \|}{\| E_L \| + \| E_R \|}. \tag{9}$$

3.2.3 Step 3: Separating nodes

This step is exactly the same as in the CLUSS algorithm. However, we give more details about this step here. The CLUSS2 algorithm makes use of a systematic method for deciding which subtrees to retain as a trade-off between searching for the highest co-similarity values and searching for the largest possible clusters. We first separate all the subtrees into two groups, one being the group of low co-similarity subtrees, and the other the high co-similarity subtrees. This is done by sorting all possible subtrees in increasing order of co-similarity and computing a separation threshold according to the maximum interclass inertia method, based on the Koenig-Huygens theorem, which gives the relationship between the total inertia and the inertia of each group relative to the centre of gravity. In our case we have just two groups, the high co-similarity group and the low cosimilarity group. The procedure is described as follows:

Let D be the set of subtrees, D_{Low} the subset of low co-similarity subtrees, and D_{High} the subset of high co-similarity subtrees, such that:

$$D_{\text{Low}} \bigcup D_{\text{High}} = D, \quad D_{\text{Low}} \bigcap D_{\text{High}} = \varnothing \tag{10}$$

$$\forall L, R \in D \mid L \in D_{\text{Low}}, \quad R \in D_{\text{High}} \Rightarrow c_L < c_R. \tag{11}$$

The symbols D_{Low} and D_{High} are simply used as variables representing all possible separations of D according to equations (10) and (11). According to the Koenig-Huygens theorem, we calculate the total inertia as follows:

$$I_{\text{Total}} = \sum_{i \in D_{\text{Low}}} \left(c_i - \overline{c}_{K_{\text{Low}}} \right) + \sum_{j \in D_{\text{High}}} \left(c_j - \overline{c}_{D_{\text{High}}} \right)^2 + \left(\overline{c}_{K_{\text{Low}}} - \overline{c}_{D_{\text{High}}} \right)^2 \quad (12)$$

where c_i and c_j are co-similarity values of subtrees i and j belonging to the subsets D_{Low} and D_{High}, all respectively; and $\overline{c}_{D_{\text{Low}}}$ and $\overline{c}_{D_{\text{High}}}$ are means (i.e., centres of gravity) of subsets D_{Low} and D_{High}, respectively. The best separation of D, the set of sorted subtrees on two subsets D_{Low} and D_{High}, is given by the maximum value of I_{Total}.

3.3 Stage 3: Extracting clusters

From the subset of high co-similarity subtrees belonging to D_{High}, we extract those that are largest. A high co-similarity subtree is largest if the following two conditions are satisfied:

- it does not contain any low co-similarity subtree belonging to the subset D_{Low}
- if it is included in another high co-similarity subtree, the latter contains at least one low co-similarity subtree from the subset D_{Low}.

Each of these largest subtrees corresponds to a cluster and its leaves are then collected to form the corresponding cluster.

4 Experiments

To illustrate its efficiency, we tested CLUSS2 extensively on a variety of protein datasets and compared it both with CLUSS and with several mainstream clustering algorithms. We analysed the results obtained for the different tests with support from the literature and functional annotations. All the data and results cited in this section are available on the CLUSS website http://prospectus.usherbrooke.ca/CLUSS/. To evaluate the quality of the clustering results obtained, in our experiments we used the Q-measure that we introduced in Kelil et al. (2007b).

4.1 Benchmarking

To illustrate the efficiency of CLUSS2 in grouping protein sequences according to their functional annotations and biological classifications, we performed extensive tests on the widely known databases COG (unicellular organisms), KOG (eukaryotic organisms) and PC (microbial protein clusters). The COG and KOG databases include clusters of orthologous groups of proteins that were delineated by comparing protein sequences encoded in complete genomes, representing major phylogenetic lineages. The PC database is a compilation of proteins from the complete genomes of prokaryotes, plasmids and organelles that have been grouped and manually curated and annotated based on sequence similarity and protein function.

In order to evaluate CLUSS2 in a statistical manner, we generated three benchmarks named A, B and C, each containing three different large sets, such that $A = \{A_1, A_2, A_3\}$ $B = \{B_1, B_2, B_3\}$ and $C = \{C_1, C_2, C_3\}$. The nine sets in these benchmarks have been generated in this way; A_1, B_1 and C_1 from the COG database, A_2, B_2 and C_2 from the KOG database and A_3, B_3 and C_3 from the PC database. Each set contains 1000 different, large,

randomly generated subsets of protein sequences. Each subset contains a large number of non-orphan protein sequences (i.e., each protein sequence has at least one similar protein sequence from the same functional classification). Each subset in the benchmark *A* contains a number of proteins with at least five biochemical activities. In the benchmark *B*, each subset contains a number of proteins with at least ten biochemical activities. And finally, in the benchmark *C*, each subset contains a number of proteins with at least 20 biochemical activities. Details about the generated benchmarks are given in Table 2. We tested CLUSS2 and CLUSS on the three benchmarks using both substitution matrices BLOSUM62 and PAM250. The obtained results for both matrices were very similar. The results obtained are shown in Table 3, and discussed below.

Table 2 Generated datasets

Benchmark	*COG* (A_1, B_1, C_1)		*KOG* (A_2, B_2, C_2)		*PC* (A_3, B_3, C_3)	
	Av. No.	*Av. Length*	*Av. No.*	*Av. Length*	*Av. No.*	*Av. Length*
A	298	1087	230	2024	256	815
B	487	1102	458	2043	449	895
C	678	1198	696	2076	628	912

Av. No. is the average number and *Av. Length* is the average length, of all protein sequences within each set (column), in each benchmark (row).

Table 3 Benchmarking results (Time in seconds)

Benchmark	*Algorithm*	*COG* (A_1, B_1, C_1)			*KOG* (A_2, B_2, C_2)			*PC* (A_3, B_3, C_3)		
		Qm	*SD*	*Time*	*Qm*	*SD*	*Time*	*Qm*	*SD*	*Time*
A	CLUSS2	90.32	3.56	11	86.74	3.41	5	96.61	3.86	15
	CLUSS	90.56	4.04	27	86.15	4.63	21	96.28	4.67	60
B	CLUSS2	92.25	5.12	16	87.34	5.76	9	94.45	5.71	18
	CLUSS	90.81	7.87	49	85.16	7.10	39	91.64	7.14	68
C	CLUSS2	91.85	7.92	18	86.56	8.45	14	96.11	7.81	21
	CLUSS	81.39	9.60	61	77.91	11.09	55	88.68	10.94	92

Qm is the average Q-measure, *SD* the standard deviation and *Time* the average execution time, of the clustering results of each set (column) in each benchmark (main row) using each CLUSS version (child row).

4.1.1 Benchmark A with five biological activities (Table 3)

The average Q-measure (*Qm*) and the Standard Deviation (*SD*) values of the clustering results obtained for each database (COG, KOG and PC) are essentially equal with CLUSS2 and CLUSS. However, the execution times (Time) for each database clearly show that CLUSS2 is definitely faster than CLUSS.

4.1.2 Benchmark B with ten biological activities (Table 3)

The Qm and SD values of the clustering results obtained for each of the databases show a small advantage of CLUSS2 compared to CLUSS. However, the Time values for each database show once again that CLUSS2 is faster than CLUSS.

4.1.3 Benchmark C with 20 biological activities (Table 3)

The Qm values of the clustering results obtained for each of the databases using CLUSS2 are clearly higher than those obtained with CLUSS. Also, the SD values of the clustering results obtained for each database using CLUSS2 are visibly lower than those obtained with CLUSS. The Time values for each database using CLUSS2 increase much more slowly than those obtained using CLUSS.

The results obtained clearly show that CLUSS2 is indeed effective in grouping sequences according to the known functional classification of COG, KOG and PC databases more efficiently than CLUSS. Contrary to what was observed for CLUSS, the efficiency of the new algorithm CLUSS2 does not notably decrease with an increase in the number of biochemical functions included in the clustered protein datasets. Another important fact to note is that the optimisation techniques used in the new similarity measure tSMS have significantly improved the time efficiency of the clustering process.

4.2 Comparisons

To compare the efficiency of CLUSS2 to that of alignment-dependent clustering algorithms, we performed tests using CLUSS2, CLUSS, BlastClust, TRIBE-MCL and gSPC on the COG, KOG and PC databases. In all of the tests performed, we used the widely known protein sequence comparison algorithm ClustalW (Thompson et al., 1994) to calculate the similarity measure matrices used by TRIBE-MCL and gSPC. Due to the complexity of alignment, these tests were done on three sets of six randomly generated subsets, named C1 to C6 for COG, K1 to K6 for KOG and P1 to P6 for PC; each generated protein subset includes protein sequences with at least 20 biological activities.

The results obtained are summarised in Tables 4–6. The experiments show clearly that CLUSS2 obtained the best Q-measure, compared to the other algorithms tested. Even if we compare the results of CLUSS2 with those of CLUSS, we can see that CLUSS2 has obtained better clustering results. This is because each of the subsets tested contains a number of proteins with a large number of biological functions (each subset includes protein sequences with at least 20 biological functions). Globally, the clusters obtained using our new algorithm CLUSS2 correspond better to the known characteristics of the biochemical activities and modular structures of the protein sequences according to the COG, KOG and PC classifications. The execution times reported in Tables 4–6 for algorithm comparison, show clearly that the fastest algorithm is BlastClust, closely followed by the CLUSS2 algorithm, and then by CLUSS, while TRIBE-MCL and gSPC, which use ClustalW as a similarity measure, are much slower.

Table 4 Clustering results on the COG database (Time in seconds)

Protein subsets	*CLUSS2*		*CLUSS*		*BlastClust*		*TRIBE-MCL*		*gSPC*	
	Qm	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*
Cl (509)	96.02	33	80.01	109	67.01	20	30.02	422	38.01	451
C2 (448)	98.07	35	68.13	94	42.07	19	35.01	406	31.02	386
C3 (486)	95.06	38	87.02	94	61.03	28	51.04	336	57.01	378
C3 (546)	92.01	36	72.03	114	40.01	22	55.08	479	44.01	492
C4 (355)	98.04	23	86.04	69	69.01	16	40.01	273	42.01	280
C5 (508)	96.01	29	63.04	137	35.03	16	57.01	446	36.10	440
C6 (509)	96.02	33	80.01	109	67.01	20	30.02	422	38.01	451

Table 5 Comparison on the KOG database (Time in seconds)

Protein subsets	*CLUSS2*		*CLUSS*		*BlastClust*		*TRIBE-MCL*		*gSPC*	
	Qm	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*
Kl (317)	97.02	61	82.02	242	33.13	41	54.05	790	40.02	843
K2 (419)	95.02	86	69.02	279	55.02	63	60.01	371	50.01	450
K3 (383)	91.01	161	76.02	381	69.01	134	30.02	1244	30.02	1348
K4 (458)	95.02	54	76.05	310	37.01	37	59.02	1315	47.01	1349
K5 (480)	95.06	60	79.33	324	50.02	34	46.03	1425	43.02	1409
K6 (388)	93.02	76	80.03	441	32.01	49	49.01	1269	55.04	1336

Table 6 Clustering results on the PC database (Time in seconds)

Protein subsets	*New CLUSS*		*Prev. CLUSS*		*BlastClust*		*TRIBE-MCL*		*gSPC*	
	Qm	Time	Qm	Time	Qm	Time	Qm	Time	Qm	Time
P1 (538)	91.02	29	65.01	84	44.01	16	31.01	447	41.02	441
P2 (392)	94.01	23	73.01	79	31.01	18	35.02	250	57.01	264
P3 (442)	93.02	31	70.01	84	34.06	14	32.01	316	39.01	390
P4 (595)	95.02	46	60.01	152	66.01	35	58.50	711	30.02	633
P5 (561)	91.17	39	81.02	97	68.08	18	54.02	433	34.01	435
P6 (427)	94.02	22	77.08	75	34.01	16	43.03	410	49.02	399

4.3 G-Proteins family

The G-Proteins (for guanine nucleotide binding proteins) that are available at http://www.gpcr.org/ belong to the larger family of GTPases. Their signalling mechanism consists in exchanging Guanosine Diphosphate (GDP) for Guanosine Triphosphate (GTP) as a general molecular function to regulate cell processes, reviewed extensively in (Lodish et al., 2004). This family has been the subject of a considerable number of publications by researchers around the world, so we considered it a good reference classification to test the performance of CLUSS2. The sequences belonging to this family (version of October 6, 2007), including the 2604 sequences used in our experiments, are available on the CLUSS website. The experimental results obtained using both the CLUSS2 and CLUSS algorithms as well as the algorithms BlastClust, TRIBE-MCL and gSPC are summarised in Table 7.

Table 7 Clustering results of the G-Proteins family (Time in seconds)

Protein set	*New CLUSS*		*Prev. CLUSS*		*BlastClust*		*TRIBE-MCL*		*gSPC*	
	Qm	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*	*Qm*	*Time*
G-Proteins	91.78	402	89.32	2199	57.78	372	50.89	32,654	61.45	36,751

The clustering results for the G-Proteins family show clearly that although this family is known to be easy to align, which should have facilitated the clustering task of the alignment-dependent algorithms, CLUSS2 yields a clustering with the highest Qm value

of all the algorithms tested, nearly followed by CLUSS. Thus, the results obtained by CLUSS2 are much closer to the known classification of the G-Proteins family than are those of the other algorithms tested. In Table 7, we can make the same observation about the execution times of the different algorithms as in Tables 4–6.

4.4 The 33 $(\alpha/\beta)_8$-barrel proteins

To show the performance of CLUSS2 with multi-domain protein families which are known to be hard to align and have not yet been definitively aligned, experimental tests were performed on the 33 $(\alpha/\beta)_8$-barrel proteins studied recently by Côté et al. (2006) and (Fukamizo et al., 2006), which form a group in Glycoside Hydrolases family 2 (GH2) from the Carbohydrate Active Enzymes database (CAZy) located at http://www.cazy.org/. The periodic character of the catalytic module known as '*$(\alpha/\beta)_8$-barrel*' makes these sequences hard to align using classical alignment approaches. The difficulties in aligning these modules are comparable to the problems encountered with the alignment of tandem-repeats, which have been exhaustively discussed by Higgins (2004). The FASTA file and full clustering results of this subfamily are available on the CLUSS website. This group of 33 protein sequences includes '*β-galactosidase*', '*β-mannosidase*', '*β-glucuronidase*' and '*exo-β-D-glucosaminidase*' enzymatic activities, all extensively studied at the biochemical level. These sequences are multi-modular, with various types of modules, which complicate their alignment. Clustering such protein sequences using the alignment-dependent algorithms thus becomes problematic. This encouraged us to perform a clustering on this particular group of the GH2 subfamily, to compare the behaviour of both algorithms CLUSS2 and CLUSS with BlastClust, TRIBE–MCL and gSPC in order to validate the use of CLUSS2 on the hard-to-align proteins. An overview of the results is given in Table 8, with a detailed discussion below. The corresponding names and database entries of the 33 $(\alpha/\beta)_8$-barrel proteins group are indicated on the CLUSS website.

The 33 $(\alpha/\beta)_8$-barrel proteins were subdivided by CLUSS2 and CLUSS into five subfamilies, corresponding to their known biochemical activities. However, contrarily to CLUSS, which has classified the two proteins MaC and MaT with the first cluster, CLUSS2 classified all the 33 $(\alpha/\beta)_8$-barrel proteins in the same subfamilies obtained by Côté et al. (2006) that in turn are supported by the structure-function studies of Fukamizo et al. (2006). This shows the superiority of CLUSS2 comparing to CLUSS in clustering protein sequences. The first cluster includes enzymes annotated as '*β-mannosidase*' activities; the second cluster includes enzymes with '*β-mannosidase*' activities; the third cluster includes enzymes with '*β-glucuronidase*' activities; the forth cluster includes enzymes with '*β-galactosidase*' activities; the fifth cluster includes enzymes with '*exo-β-D-glucosaminidase*' activities. While the other algorithms do not succeed to obtain clustering results that correspond to the functional classification of the 33 $(\alpha/\beta)_8$-barrel proteins group obtained by Côté et al. (2006) and Fukamizo et al. (2006). Since, there are a number of well classified proteins (i.e., GaA, GaK, GaC, CsAo, CsN and CsAn) which could not be classified by BlastClust, and a number of proteins which were wrongly classified by TRIBE-MCL and gSPC, for details see Table 8.

Table 8 Clustering results of the 33 $(\alpha/\beta)_8$-barrel proteins group

No.	*Proteins*	*Côté/Fukamizo*	*CLUSS2*	*CLUSS*	*BlastClust*	*T-MCL*	*gSPC*
1	UnA	1	1	1	1	1	1
2	UnBv	1	1	1	1	1	1
3	UnBc	1	1	1	/	1	1
4	UnBm	1	1	1	1	1	1
5	UnBp	1	1	1	1	1	1
6	UnR	1	1	1	1	1	1
7	MaA	2	2	1	2	2	1
8	MaB	2	2	1	2	2	1
9	MaH	2	2	1	2	1	1
10	MaM	2	2	2	2	1	1
11	MaC	2	2	2	2	1	1
12	MaT	2	2	2	2	2	1
13	GIC	2/3	2	2	2	2	1
14	GIE	3	3	3	3	2	2
15	GIH	3	3	3	3	2	2
16	GIL	3	3	3	3	2	2
17	GIM	3	3	3	3	2	2
18	GIF	3	3	3	3	2	2
19	GIS	3	3	3	3	2	2
20	GaEco	4	4	4	4	2	2
21	GaA	4	4	4	/	2	2
22	GaK	4	4	4	/	2	2
23	GaC	4	4	4	/	2	2
24	GaEcl	4	4	4	4	2	2
25	GaL	4	4	4	4	2	2
26	CsAo	5	5	5	/	2	3
27	CsS	5	5	5	5	2	3
28	CsG	5	5	5	5	2	3
29	CsM	5	5	5	5	2	3
30	CsN	5	5	5	/	2	3
31	CsAn	5	5	5	/	2	3
32	CsH	5	5	5	5	2	3
33	CsE	5	5	5	5	2	3

The symbol **'/'** means that the corresponding algorithm (column) was not able to classify the corresponding protein (row) with any one of the other proteins (i.e., orphan protein).

5 Conclusion

Our new similarity measure tSMS makes it possible to measure the similarity between protein sequences much more quickly and effectively than SMS – especially for protein datasets that include proteins with a relatively large number of biochemical

activities – based solely on the conserved motifs, with a certain tolerance to mismatches. Its major advantage over the alignment-dependent approaches is that it gives significant results with protein sequences independent of their alignability, making it effective on both easy-to-align and hard-to-align protein sequences. These properties are inherited by CLUSS2, our new clustering algorithm, which uses tSMS as its similarity measure. Compared to CLUSS, our new clustering algorithm CLUSS2 is a much more effective clustering algorithm for protein sets with respect to the number of biological activities. It more accurately highlights the characteristics of the biochemical activities of the clustered protein sequences than do CLUSS and several mainstream alignment-dependent algorithms.

So far, our similarity measure tSMS has been based on pre-determined substitution matrices. A possible future development is to propose an approach to automatically compute the weights of the matched motifs instead of relying on pre-calculated substitution scores.

References

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) 'Basic local alignment search tool', *J. Mol. Biol.*, Vol. 215, No. 3, pp.403–410.

Berry, M.W. and Fierro, R.D. (1996) 'Low-rank orthogonal decompositions for information retrieval applications', *Numerical Linear Algebra with Applications*, Vol. 3, No. 4, pp.301–327.

Cole, R., Kopelowitz, T. and Lewenstein, M. (2006) 'Automata, languages and programming', *33rd International Colloquium, ICALP 2006, Proceedings, Part I, Chapter Suffix Trays and Suffix Trists: Structures for Faster Text Indexing*, Venice, Italy, 10–14 July, pp.358–369.

Côté, N., Fleury, A., Dumont-Blanchette, E., Fukamizo, T., Mitsutomi, M. and Brzezinski, R. (2006) 'Two exo-β-D-glucosaminidases/exochitosanases from actinomycetes define a new subfamily within family 2 of glycoside hydrolases', *Biochem. J.*, Vol. 394, No. Pt 3, pp.675–686.

Coutinho, P.M. and Henrissat, B. (1999) 'Recent advances in carbohydrate bioengineering', *Chapter Carbohydrate-Active Enzymes: An Integrated Database Approach*, The Royal Society of Chemistry, Cambridge, pp.312.

Enright, A.J., Dongen, S.V. and Ouzounis, C.A. (2002) 'An efficient algorithm for large-scale detection of protein families', *Nucleic Acids Res.*, Vol. 30, No. 7, pp.1575–1584.

Fukamizo, T., Fleury, A., Côté, N., Mitsutomi, M. and Brzezinski, R. (2006) 'Exo-beta-D-glucosaminidase from *Amycolatopsis orientalis*: Catalytic residues, sugar recognition specificity, kinetics, and synergism', *Glycobiology*, Vol. 16, No. 11, pp.1064–1072.

Golub, G.H. and Loan, C.F.V. (1996) *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, USA.

Higgins, D. (2004) *The Phylogenetic Handbook – A Practical Approach to DNA and Protein Phylogeny*, Marco Salemi and Anne-Mieke Vandamme, Chapter Multiple Alignment, pp.45–71.

Kelil, A., Wang, S. and Brzezinski, R. (2007a) 'A new alignment-independent algorithm', *IEEE 7th BIBE, Conference Center at Harvard Medical School*, Cambridge, Boston, Massachusetts, USA.

Kelil, A., Wang, S., Brzezinski, R. and Fleury, A. (2007b) 'CLUSS: clustering of protein sequences based on a new similarity measure', *BMC Bioinformatics*, Vol. 8, p.286.

Lodish, H., Berk, A., Matsudaira, P., Kaiser, C.A., Krieger, M., Scott, M.P., Zipursky, L. and Darnell, J. (2005) *Molecular Cell Biology*, W.H. Freeman and Co., New York and Basingstoke.

Tatusov, R.L., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Kiryutin, B., Koonin, E.V., Krylov, D.M., Mazumder, R., Mekhedov, S.L., Nikolskaya, A.N., Rao, B.S., Smirnov, S., Sverdlov, A.V., Vasudevan, S., Wolf, Y.I., Yin, J.J. and Natale, D.A. (2003) 'The COG database: an updated version includes eukaryotes', *BMC Bioinformatics*, Vol. 4, p.41.

Tetko, I.V., Facius, A., Ruepp, A. and Mewes, H. (2005) 'Super paramagnetic clustering of protein sequences', *BMC Bioinformatics*, Vol. 6, p.82.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) 'CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice', *Nucleic Acids Res.*, Vol. 22, No. 22, pp.4673–4680.

Weiner, P. (1973) 'Linear pattern matching algorithm', *14th Symposium on Switching and Automata Theory*, IEEE Computer Society, ed., Los Alamitos, CA, pp.1–11.