Online journal ISSN - 1676-5680
GMR Genetics and Molecular Research
EVOLUTION AND TECHNOLOGY

Application of latent semantic indexing to evaluate the similarity of sets of sequences without multiple alignments character-by-character

B.R.G.M. Couto[1,2], A.P. Ladeira[1,3] and M.A. Santos[4]

[1]Programa de Doutorado em Bioinformática,
Departamento de Bioquímica e Imunologia,
Universidade Federal de Minas Gerais, UFMG, Belo Horizonte,
MG, Brasil
[2]Curso de Ciência da Computação, Centro Universitário de Belo Horizonte,
UNI-BH, Belo Horizonte, MG, Brasil
[3]Escola de Ciência da Informação, Universidade Federal de Minas Gerais,
UFMG, Belo Horizonte, MG, Brasil
[4]Departamento de Ciência da Computação,
Universidade Federal de Minas Gerais, UFMG, Belo Horizonte,
MG, Brasil
Corresponding author: B.R.G.M. Couto
E-mail: bcouto@acad.unibh.br

Genet. Mol. Res. 6 (4): 983-999 (2007)
Received August 03, 2007
Accepted September 25, 2007
Published October 05, 2007

ABSTRACT. Most molecular analyses, including phylogenetic inference, are based on sequence alignments. We present an algorithm that estimates relatedness between biomolecules without the requirement of sequence alignment by using a protein frequency matrix that is reduced by singular value decomposition (SVD), in a latent seman-

tic index information retrieval system. Two databases were used: one with 832 proteins from 13 mitochondrial gene families and another composed of 1000 sequences from nine types of proteins retrieved from GenBank. Firstly, 208 sequences from the first database and 200 from the second were randomly selected and compared using edit distance between each pair of sequences and respective cosines and Euclidean distances from SVD. Correlation between cosine and edit distance was -0.32 ($P < 0.01$) and between Euclidean distance and edit distance was +0.70 ($P < 0.01$). In order to check the ability of SVD in classifying sequences according to their categories, we used a sample of 202 sequences from the 13 gene families as queries (test set), and the other proteins (630) were used to generate the frequency matrix (training set). The classification algorithm applies a voting scheme based on the five most similar sequences with each query. With a 3-peptide frequency matrix, all 202 queries were correctly classified (accuracy = 100%). This algorithm is very attractive, because sequence alignments are neither generated nor required. In order to achieve results similar to those obtained with edit distance analysis, we recommend that Euclidean distance be used as a similarity measure for protein sequences in latent semantic indexing methods.

Key words: Bioinformatics, Molecular comparisons, Sequence alignments, Latent semantic indexing

INTRODUCTION

Many molecular analyses, including phylogenetic inferences, are based on character-by-character comparisons (Krawetz and Womble, 2003). These standard methods use alignment algorithms that are intrinsically highly subjective and usually employ cut-off values and gap penalties that are difficult to define (Stuart et al., 2002a). According to Thorne (2000), the most significant error in molecular phylogenies is due to inaccurate alignments. Furthermore, once an alignment is obtained, it is necessary to discard a fraction of the original sequences compared, which restricts the postulated homology to a few selected domains (Thorne, 2000; Stuart et al., 2002a). Besides the difficulties with the alignment algorithm itself, as whole genome sequences continue to accumulate in public databases, with billions of sequence characters, effective methods for comparing and categorizing these genes are crucial. Actually, the complexity involved in estimating relatedness between large numbers of biomolecules is enormous, and methods based on character-by-character comparisons to produce large-scale alignments become impractical, far beyond the scope of currently available computational systems (Stuart et al., 2002a,b; Stuart and Berry, 2003, 2004).

In this report, we present an algorithm to compare and to categorize genes that are based on the methodology developed by Stuart et al. (2002a) for generating whole genome phylogenies using vector representations of protein sequences. The algorithm estimates relat-

edness between large numbers of biomolecules without the requirement of multiple sequence alignment. The original method (Stuart et al., 2002a) uses a tool from numerical analysis, called singular value decomposition (SVD), to process a peptide frequency matrix, a large sparse data matrix in which each protein is uniquely represented as a vector. As the comparisons among sequences are made by vector pairwise comparisons instead of sequence alignments, before applying the proposed method, we analyzed the relationship between the vector properties (cosine and Euclidean distance values) and edit distance measures, which allowed the validation of the methodology.

MATERIAL AND METHODS

A biomolecular sequence can be viewed as a complex written language, so that its analysis can be very similar to that used by information retrieval (IR) systems, where large amounts of textual information are organized, compared and categorized (Berry et al., 1999; Stuart et al., 2002a). In the IR field, commonly used models are the boolean, vector space, probabilistic model, and latent semantic indexing (LSI), which combine the vector space model with singular value decomposition (Cöster, 1999).

The method proposed by Stuart et al. (2002a) to evaluate the similarity of sequences is an LSI method, where individual protein sequences correspond to a "passage" of text, whereas peptides of a given size serve as n-gram "words". In this approach, protein sequences are recoded as p-peptide frequency values using all possible overlapping p-peptides (Stuart et al., 2002a; Rodrigues et al., 2004). With 20 amino acids, a 20^p x n matrix is generated (20^p rows and n columns or vectors, one for each n protein under analysis). For instance, by using a tripeptide, there are 20^3 = 8000 possible peptides, and if 4 amino acids are used, there are 20^4 = 160,000 possible tetrapeptides. The simplest situation, illustrated by Figure 1, occurs when only one amino acid is used for each peptide. In this case, the frequency matrix has only 20 rows and n columns, each one representing the protein vectors. These n vectors are composed of the frequency of each amino acid in the protein (f1,1 = frequency of alanine in the first protein). When all combinations of size 3 amino acids are used to build the matrix (Figure 2), each vector has the frequency of each tripeptide in the protein (f1,1 = frequency of tripeptide 1 in the first protein). In these matrices, proteins are treated as documents and peptides as terms, which allows the problem to be solved by information retrieval methods.

Programs and datasets

Programs implemented for this analysis were written in MATLAB (The Mathworks, 1996), using its built-in functions (SVD, sparse matrix manipulation subroutines, etc.). Two datasets were used in this paper. The first database evaluated had 64 vertebrate mitochondrial genomes composed of 832 proteins from 13 known gene families (ATP6, ATP8, COX1, COX2, COX3, CYTB, ND1, ND2, ND3, ND4, ND4L, ND5, and ND6). This curated protein database was downloaded from the online information at http://mama.indstate.edu/users/stuart/gaspipe/index.html from Stuart et al. (2002b).

The second database was composed of sequences from proteins retrieved from GenBank on April 19, 2006 (Figure 3). A random sample of 100 sequences was obtained of each

Terms: amino acids	Documents: proteins			
	Protein 1	Protein 2	...	Protein n
V1 = Alanine	f1,1	f1,2	...	f1,n
V2 = Arginine	f2,1	f2,2	...	f2,n
V3 = Asparagine	f3,1	f3,2	...	f3,n
V4 = Aspartic acid	f4,1	f4,2	...	f4,n
V5 = Cysteine	f5,1	f5,2	...	f5,n
V6 = Glutamine	f6,1	f6,2	...	f6,n
V7 = Glutamic acid	f7,1	f7,2	...	f7,n
V8 = Glycine	f8,1	f8,2	...	f8,n
V9 = Histidine	f9,1	f9,2	...	f9,n
V10 = Isoleucine	f10,1	f10,2	...	f10,n
V11 = Leucine	f11,1	f11,2	...	f11,n
V12 = Lysine	f12,1	f12,2	...	f12,n
V13 = Methionine	f13,1	f13,2	...	f13,n
V14 = Phenylalanine	f14,1	f14,2	...	f14,n
V15 = Proline	f15,1	f15,2	...	f15,n
V16 = Serine	f16,1	f16,2	...	f16,n
V17 = Threonine	f17,1	f17,2	...	f17,n
V18 = Tryptophan	f18,1	f18,2	...	f18,n
V19 = Tyrosine	f19,1	f19,2	...	f19,n
V20 = Valine	f20,1	f20,2	...	f20,n

Figure 1. Protein frequency matrix built with 1-letter string of amino acids.

type of protein (globin, cytochrome, histone, cyclohydrolase, pyrophosphatase, ferredoxin, keratin, and collagen) and 200 other proteins from lymphocytes and bacteriophages, totaling 1000 sequences.

Construction of the protein matrix

Terms, documents, queries, and weights are fundamental components of any IR system (Cöster, 1999). A term is an individual word or a phrase that reflects a particular concept or key word (Berry et al., 1995). Terms are extracted from either the body of a text or a surrogate text (e.g., abstract). In the context of biomolecular sequences, terms are the p-peptide strings (usually, tripeptides or tetrapeptides). Documents are the text itself, composed of terms. Here, proteins are the documents analyzed. The information needed by an IR user is called a query (Cöster, 1999). In this report, a query will be an unknown gene sequence whose category or family we need to determine. A weight is a value reflecting the importance of a term in a document or query (Cöster, 1999). For this analysis, all terms (p-peptide) have the same weight, assumed to be one. The elements of the term document or protein matrix are the occurrences of each peptide (of size p) in a particular protein.

Terms: all peptides with 3 amino acids	Documents: proteins			
	Protein 1	Protein 2	...	Protein n
ABC	f1,1	f1,2	...	f1,n
ABD	f2,1	f2,2	...	f2,n
ABE	f3,1	f3,2	...	f3,n
ABF	f4,1	f4,2	...	f4,n
ABG	f5,1	f5,2	...	f5,n
ABH	f6,1	f6,2	...	f6,n
ABI	f7,1	f7,2	...	f7,n
ABJ	f8,1	f8,2	...	f8,n
ABL	f9,1	f9,2	...	f9,n
ABM	f10,1	f10,2	...	f10,n
ABN	f11,1	f11,2	...	f11,n
ABO	f12,1	f12,2	...	f12,n
ABP	f13,1	f13,2	...	f13,n
ABQ	f14,1	f14,2	...	f14,n
ABR	f15,1	f15,2	...	f15,n
ABS	f16,1	f16,2	...	f16,n
ABT	f17,1	f17,2	...	f17,n
ABU	f18,1	f18,2	...	f18,n
ABT	f19,1	f19,2	...	f19,n
...	...	...	...	...
...	**f8.000,1**	**f8.000,2**	...	**f8.000,n**

Figure 2. Protein frequency matrix built with 3-letter string of amino acids.

Type of sequence	Number of GenBank sequences
Globin	1,958
Cytochrome	164,423
Histone	9,985
Cyclohydrolase	2,670
Pyrophosphatase	2,313
Ferredoxin	8,338
Lymphocyte	15,535
Bacteriophage	19,663
Keratin	459
Collagen	2,922

Figure 3. Number of sequences retrieved from GenBank from different types of proteins.

The document-term matrix construction is based on the protein sequences that are re-coded as p-peptide frequency values using all possible overlapping p-peptides, which generates the frequency matrix. Matrices are built using p = 1, p = 2, p = 3, and p = 4 peptides. These sparse matrices have dimensions of 20 x n, 400 x n, 8000 x n, and 160,000 x n, respectively, where *n* is the number of sequences analyzed. A larger number of peptides is not used because it will produce huge matrices, with more than 3 million rows (20^5 = 3,200,000 rows). The MATLAB codes in Figure 4A and B build the protein matrix using sequence data in a text file, for example, in a file named "mitgenes_M.stu". The first line of the file has the number of sequences to be analyzed (n), and the other lines have the string sequences of each protein in the dataset.

It is important to note that, with four amino acids in the p-peptide, there will be 160,000 possible tetrapeptides in the protein matrix, most of which will have zero frequency. Actually, the matrix produced by the algorithm 4A and B will be very sparse, which is computationally good in terms of memory requirements.

Figure 5 shows the protein frequency matrix in the simplest case (variable n_pep = 1), when the peptide is composed of only one amino acid. In this situation, we have 20 terms, and in analyzing 5 proteins, the document-term matrix has 20 rows and 5 columns. The five proteins correspond to 2 genes (COX3 and COX2) from different vertebrate mitochondrial genomes. The original amino acid frequency for each protein varies across each vector (columns of the protein matrix).

Latent semantic indexing

LSI, developed by Deerwester et al. (1990), is an IR method that uses singular value decomposition and a vector space model to retrieve information (Orengo, 2004). In a vector space representation of information, vectors that form a frequency term-by-document matrix, as illustrated in Figures 1 and 2, are used to represent each document or proteins. The aim of LSI is to perform the retrieval of a query in terms of conceptual content, rather than literally matching terms (Deerwester et al., 1990; Berry et al., 1995; Orengo, 2004). Due to synonymy, where the same concept can be expressed in many different ways, and polysemy, where a word can have multiple meanings, in the traditional IR systems individual words provide unreliable evidence about the meaning of the document (Orengo, 2004). To overcome the synonymy and polysemy problems, LSI estimates the usage of terms across documents, revealing its underlying semantic structure. Terms that occur frequently together are associated, which in practice means that a query may retrieve documents which have none of the query terms (Deerwester et al., 1990).

In a mathematical way, synonymy and polysemy are solved by applying an SVD in the term-by-document matrix, followed by a rank matrix reduction. After the SVD, the matrix reduction is performed by replacing the original matrix with another that is as close as possible to the original but whose column space is only a subspace of the column space of the original matrix (Berry et al., 1999). The objective of breaking down the term-document matrix is to remove extraneous information or noise from the original database.

SVD is performed by many software, including MATLAB (The Mathworks, 1996) used in this study. Given a (m x n) term-by-document matrix *M*, the SVD of *M* is defined using Equation 1 (Deerwester et al., 1990):

$$M = USV^T \qquad \text{(Equation 1)}$$

where *U* is the m x m orthogonal matrix having the left singular vectors of *M* as its columns, *V* is the n x n orthogonal matrix having the right singular vectors of *M* as its columns, and *S* is the m x n diagonal matrix with the singular values $\sigma 1 \geq \sigma 2 \geq \sigma 2 \ldots \geq \sigma r$ of *M* in order along its diagonal (*r* is the rank of *M* or the number of linearly independent columns or rows of *M*).

A

```
n_pep = input('Number of amino acids in the p-peptide (1, 2, 3 or 4): ');
m = 20^n_pep;
MAT=sparse(m,n);

% Obs.: "mitgenes_M.stu" is an example of file with the protein sequences
archive = 'mitgenes_M.stu'
fid = fopen(archive,'rt');
n=fscanf(fid,'%d \n',1);

% building each protein vector
for i=1:n
    fprintf('\n %6i of %4i',i,n);
    [protein] = fgetl(fid);
    size      = length(protein);
    column    = i;
    [MAT]     =montaMAT(MAT, protein,column,size,n_pep);
end
fclose(fid);
```

B

```
function [MAT]=montaMAT(MAT,protein,column,size,n_pep)

amino acids = 'ACDEFGHIKLMNPQRSTVWY';
terms  = length(amino acids);

% overlapping window of size n_pep
for k=1:(size-n_pep)
   line  = 0;
   for j=1:(n_pep-1)
       str = protein(k+j-1);
       index  = findstr(amino acids,str);

       % calculating the peptide row in the protein matrix
       line  = line  + (index  - 1)*(terms^(n_pep-j));
   end
   str = protein(k+n_pep);
   index  = findstr(amino acids,str);

   % calculating the peptide row in the protein matrix
   line  = line  + index ;
   MAT(line,column) = MAT(line,column) + 1;
end
```

Figure 4. A. Protein matrix construction subroutine (part I). **B.** Protein matrix construction subroutine (part II).

Terms: amino acids	Documents: proteins				
	Protein 1	Protein 2	Protein 3	Protein 4	Protein 5
	COX3	COX3	COX3	COX2	COX2
V1 = Alanine	23	14	20	16	9
V2 = Arginine	1	2	1	3	3
V3 = Asparagine	5	4	4	13	12
V4 = Aspartic acid	8	7	7	14	12
V5 = Cysteine	24	23	23	8	6
V6 = Glutamine	19	21	18	9	8
V7 = Glutamic acid	18	16	16	10	8
V8 = Glycine	16	14	10	18	19
V9 = Histidine	4	3	2	4	5
V10 = Isoleucine	32	36	35	30	32
V11 = Leucine	9	9	11	8	16
V12 = Lysine	4	7	7	5	5
V13 = Methionine	12	11	12	14	11
V14 = Phenylalanine	6	7	7	7	6
V15 = Proline	5	5	5	5	6
V16 = Serine	18	21	15	21	21
V17 = Threonine	22	22	25	12	21
V18 = Tryptophan	14	15	18	18	11
V19 = Tyrosine	12	12	12	5	5
V20 = Valine	9	12	13	8	11

Figure 5. The 20 x 5-original protein matrix.

The rank reduction of *M* matrix is performed using the k-largest singular values of *M*, or k-largest singular triplet U_k, S_k, V_k, where k ≤ r. The truncated matrix M_k is defined in Equation 2:

$$M = USV^T \approx M_k = U_k S_k V_k^T \quad \text{(Equation 2)}$$

The dimension of the vector in U_k and V_k is equal to *k*, the number of SVD factors used. The extent of dimension reduction, i.e., the choice of *k*, will be detailed in the next sections. This choice is critical, being an open issue in the literature and normally decided via empirical testing (Deerwester et al., 1990; Berry et al., 1999). The truncated SVD has two main advantages. Reduced dimensionality makes the problem computationally approachable, which is crucial in whole genome analysis. Besides, and very importantly, rank reduction improves the accuracy of term-document or protein matrix by discarding noise or variability in term or peptide usage, which can remove possible homoplasy in the data (Stuart et al., 2002b). Another formula (Equation 3) to reconstruct the protein matrix, based on the *k* first singular values is:

$$A_k = \sum_{i=1}^{k} s_i \cdot u_i \cdot v_i^T \quad \text{(Equation 3)}$$

Another advantage of rank reduction is the possibility of graphical analysis and data visualization. Using the two first singular values (k = 2), the data can be analyzed by a 2-dimensional (2-D) plot and, with 3 factors (k = 3), data can be visualized in a 3-D graph.

In Figure 6, we have the *M* protein matrix, reconstructed by using two SVD factors. It is interesting to observe how the data variability, measured by the coefficient of variation, is reduced. The average coefficient of variation of the amino acid frequency for both genes was reduced from approximately 15% in the original matrix to only 3% in the reconstructed matrix. This reduction in variability is optimal for pattern recognition and clustering (Schalkoff, 1992).

Terms: amino acids	Documents: proteins				
	Protein 1	Protein 2	Protein 3	Protein 4	Protein 5
	COX3	COX3	COX3	COX2	COX2
V1 = Alanine	19	19	19	12	12
V2 = Arginine	1	1	1	3	3
V3 = Asparagine	4	5	4	12	13
V4 = Aspartic acid	7	8	7	12	13
V5 = Cysteine	23	23	24	8	6
V6 = Glutamine	19	19	20	9	8
V7 = Glutamic acid	17	16	17	9	9
V8 = Glycine	13	14	13	18	19
V9 = Histidine	3	3	3	4	5
V10 = Isoleucine	34	35	34	31	31
V11 = Leucine	10	10	9	12	12
V12 = Lysine	6	6	6	5	5
V13 = Methionine	12	12	12	12	13
V14 = Phenylalanine	7	7	7	6	7
V15 = Proline	5	5	5	5	6
V16 = Serine	18	18	18	21	22
V17 = Threonine	23	23	23	17	17
V18 = Tryptophan	16	16	16	14	14
V19 = Tyrosine	12	12	12	5	5
V20 = Valine	11	11	11	10	10

Figure 6. The 20 x 5-protein matrix reconstructed with two factors.

Besides homogenizing the amino acid frequency in each gene by eliminating data noise in COX3 and COX2 vectors, dimension reduction allows a data visualization of proteins in a 2-D plot (Figure 7), with two separated clusters (G1 = COX3 and G2 = COX2, from vertebrates A, B and C). The x-coordinate is obtained by multiplying the first column of the matrix *V* (from SVD) by the reduced *S* matrix, with only the two first singular values. The y-coordinate is calculated by the multiplication of the second column of *V* by the reduced *S* matrix, with two SVD factors.

Dimension reduction

As discussed before, the choice of *k*, the number of singular values that must be used in the reconstruction of the protein matrix after SVD, is critical and normally empirically decided. Ideally, the *k* factor or matrix dimension must be large enough to fit all the real structure in the data, and also small enough not to fit the sampling error or unimportant details. According to Deerwester et al. (1990), the best performance of any IR system is achieved when the maximum number of singular values is less than 300.

In this study, we used the method proposed by Everitt and Dunn (2001), who recommend the analysis of the relative variances of each of the singular values (v_i), calculated by Equation 4. Singular values whose relative variance is less than 0.7/n, where *n* is the number of proteins in the document-term matrix, must be ignored (Everitt and Dunn, 2001; Wall et al., 2003).

$$v_i = \frac{S_i^2}{\sum_{j=1}^{r} S_j^2}; i = 1,2,3\cdots r \qquad \text{(Equation 4)}$$

where v_i is the relative variance of the singular value S_i, from *r* singular values of the document-term matrix. The idea is to use only the most significant singular values when the protein matrix is reconstructed. For the 20 x 5-protein matrix in Figure 5, only two singular values are significant (Figure 8). In this case, *k* must be equal to 2, which was done when the 2-D plot was constructed (Figure 7).

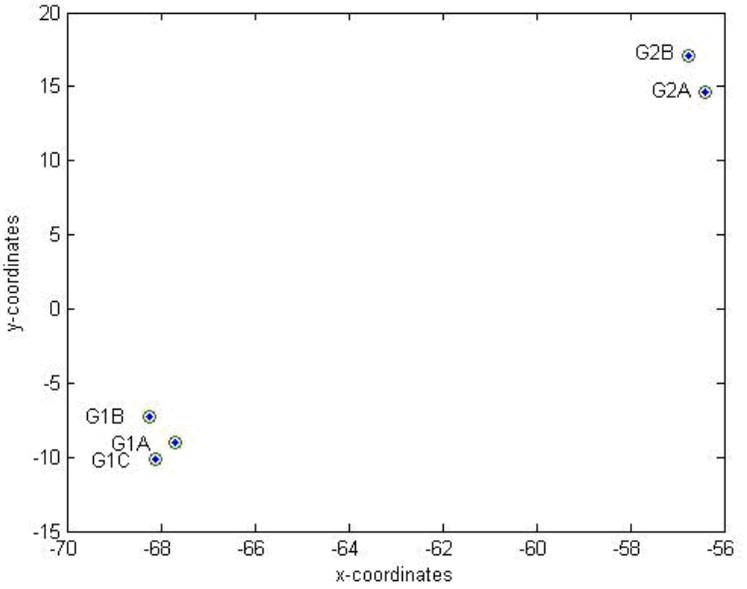

Figure 7. Two-dimensional plot of proteins for the 20 x 5 example.

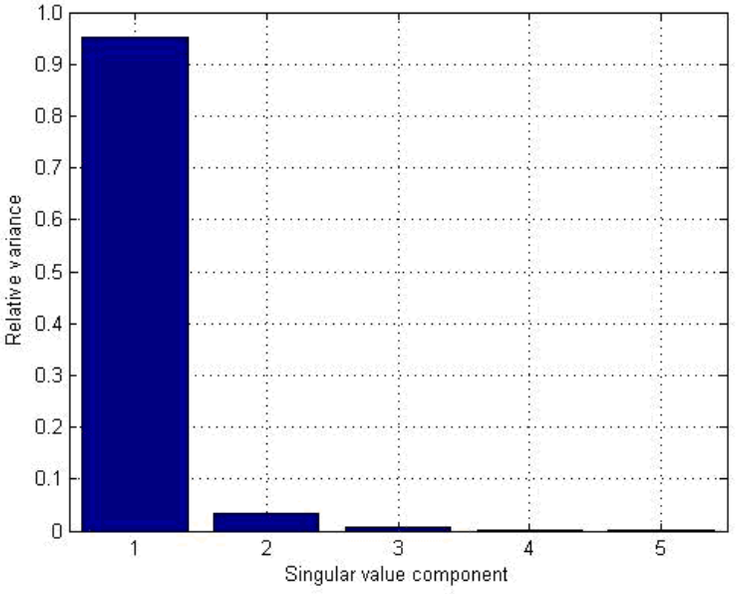

Figure 8. Relative variance plot of the 20 x 5-protein matrix of Figure 5.

Query retrieving algorithm

In the LSI information retrieval system built, it is possible to perform various comparisons: protein-by-protein, peptide-peptide, peptide-protein, and query-protein. Stuart et al. (2002a,b) and Stuart and Berry (2003, 2004) use these comparisons to build gene and species phylogenetic trees and to identify motifs.

Herein, the fundamental operation is the query-to-protein analysis, which allows the classification of the unknown gene (query) in one of the protein categories of the database. In this paper, the classification and retrieving algorithm applies a voting scheme based on the five most similar proteins with the unknown gene.

Since the query is not part of the original protein matrix (M), its vector (q) must be first generated and projected into the same form as a protein vector. The algorithms in Figure 4A and B can be used to generate the query vector q, which is modified according to Equation 5 to become another LSI protein vector:

$$q = q^T US^{-1} \qquad \text{(Equation 5)}$$

To compute similarity between the query vector and each of the protein vectors, to retrieve the most similar proteins with respect to the unknown gene, we can use many measures (Berry et al., 1995). The most often used similarity measures are the cosine of the angle and the Euclidean distance between the vectors. Despite the fact that some authors have recommended

cosine as the most effective similarity measure for text retrieval (Cöster, 1999; Kuruvilla et al., 2002; Orengo, 2004), we evaluated both measures for biomolecular sequence analysis.

The cosine of the angle between two vectors yields a value in the real range [-1.0, +1.0]. If the cosine is close to 1.0, it means that both vectors are in the same direction. A negative value close to -1.0 means that the vectors are in the opposite direction.

Two vectors define two points in the space. The Euclidean distance measures the absolute distance between the points defined by the vectors under comparison. This is a measure of neighborhood between vectors. The higher the similarity is between the two vectors, the smaller the Euclidean distance is.

The top five similar proteins with the query, by using either cosine or Euclidean distance, were used to define the category of the unknown sequence. This query is classified as a gene from a family that includes *t* most of these five sequences. For example, if the five most similar proteins with one query are from two different families A and B (Gene_A, Gene_B, Gene_B, Gene_A, and Gene_A, ordered by similarity with the query), the query is classified as a gene from family A. This method was called the voting algorithm.

The standard methods for comparisons among sequences are based on character-by-character alignments. Before applying the proposed LSI system, we analyzed the relationship between the two similarity measures with the edit distance, obtained from global sequence alignments using dynamic programming (Krawetz and Womble, 2003). In this way, it was possible to validate the method and to determine which similarity measure, cosine or Euclidean distance, is better to produce results approximately equal to the edit distance values. A correlation and a regression analysis (Neter et al., 1996) was performed to evaluate the relationship among the three similarity measures.

RESULTS AND DISCUSSION

To assess the correlation between the cosines, the Euclidean distance and a sequence alignment measure, 208 sequences from the first database and 200 from the second set were randomly selected and compared by using the global edit distance between each pair of sequences and respective cosines and Euclidean distances. The protein matrix was generated with tripeptide terms and reconstructed with 30 SVD factors (the definition of the number of SVD factors followed the relative variance criteria; Equation 4). The pairwise analysis generated 41,428 similarity measures. Despite the fact that we worked with quite different methods (LSI and global distance alignment), the correlation between the cosine and edit distance was -0.32 ($P < 0.01$) and between the Euclidean distance and edit distance was +0.70 ($P < 0.01$). These results indicate that Euclidean distance is better than the cosine in determining the similarity of sequences, when the objective is to achieve the same results as that observed with multiple alignments character-by-character (Figures 9 and 10). Actually, the square root of the Euclidean distance was better than the distance itself, with a Pearson correlation of 0.76 (Figure 10).

The negative correlation between the cosine and edit distance was expected. The higher the cosine of the angle between the two sequence vectors, higher the similarity was and, consequently, the smaller the edit distance. The Euclidean and edit distances showed the same behavior, and thus, the correlation was positive: the higher their values, the lower the similarity was between the two sequences.

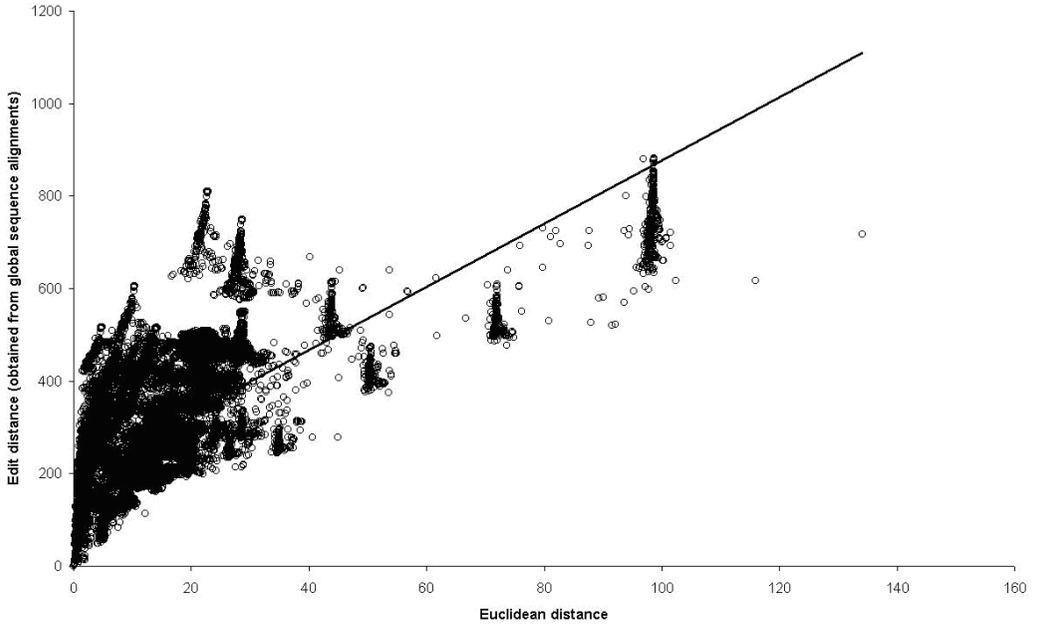

Figure 9. Scatter plot of Euclidean distance and global edit distance.

Similarity measure	Correlation coefficient with **Sij**
Sij = global edit distance of the unknown gene sequence **i** and protein sequence **j**	1.00
Cij = Cos(θ) = cosine of the angle θ between the query vector **q** and the protein vector	-0.32
Dij = Euclidean distance between the query vector **q** and the protein vector	0.70
$r_{ij} = \sqrt{D_{ij}}$	0.76
$DC_{ij} = C_{ij} \times D_{ij}$	-0.43

Figure 10. Correlation coefficient (r) between each singular value decomposition similarity measure and edit distance (Sij).

Despite the moderate correlation between Euclidean distance and edit distance (r = +0.76), it is possible to fit a linear model to estimate edit distance according to the Euclidean distance (Equation 6):

$$S_{ij} = 50 + 69 \times \sqrt{D_{ij}} \qquad \text{(Equation 6)}$$

where S_{ij} = edit distance (from a global sequence alignment), and D_{ij} = Euclidean distance.

After comparing SVD results with edit distance measure, we evaluated the ability of LSI to classify the sequences according to their categories. A sample of 202 sequences from the 13 gene families was randomly chosen as queries and the other proteins (630) were used to generate the p-peptide frequency matrix. For the second database, 735 sequences were selected to build the training set (the p-peptide frequency matrix), and 265 proteins were randomly selected as queries or test set. Figure 11 shows the file format of the original sequences from the first database. In Figure 12, we have part of the protein matrix of these data in the simplest case, where only one amino acid is used in the p-peptide term.

For both datasets, the protein frequency matrix was built by using the subroutines in Figure 4A and B, and the SVD was applied in each matrix that was reconstructed by using a number of factors defined by the relative variance analysis (Equation 4). The number of factors varied from 2 up to 56 (Figure 13). The advantage of the relative variance criteria is that

#Family	Gene and organism	Sequence
1	**COX3_Aame**	MAHQAHSYHMVDPSPWPIFGAAAALLTTSG...
2	**COX2_Aame**	MANHSQLGFQDASSPIMEELVEFHDHALIV...
3	**CYTB_Aame**	MAPNIRKSHPLLKMINNSLIDLPAPSNISA...
4	**ND4_Aame**	MLKIILPTIMLLPTTLLSPPKFLWTNTTMY...
5	**ND5_Aame**	MNATLLINSLTLLTLATLLTPIVFPLLFKN...
6	**ATP6_Aame**	MNLSFFDQFSSPYLLGIPLILLSLLFPALL...
7	**ND3_Aame**	MNMLTFMFSLSLALSAILTALNFWLAQMTP...
8	**ND2_Aame**	MNPHATPILVLSLMLGTTITISSNHWVLAW...
9	**ATP8_Aame**	MPQLNPAPWFSIMIMTWLTLALLIQPKLLT...
10	**ND1_Aame**	MPQMTMMSYLIMSLLYAIPILIAVAFLTLV...
11	**ND4L_Aame**	MSPLHLSFYSAFVLSGLGLAFHRTHLVSAL...
12	**COX1_Aame**	MTFINRWLFSTNHKDIGTLYLIFGAWAGMI...
13	**ND6_Aame**	MTYFVFFLGVCFVVGVLGVASNPSPYYGVV...
1	COX2_Ajam	MAYPFQLGLQDATSPIMEELLHFHDHTLMI...
2	COX1_Ajam	MFISRWFFSTNHKDIGTLYLLFGAWAGMVG...
3	ND4_Ajam	MLKIIIPTIMLMPLTWLSNPKMIWINSTAH...
4	ND6_Ajam	MMTYIVFVLSTIFVLSFVGFSSKPSPIYGG...
5	ATP6_Ajam	MNENLFASFITPTMMGLPIVILIIMFPTIM...
6	ND5_Ajam	MNLVSSMMLLSLMILSMPIMTTMLYPQNHP...
7	ND3_Ajam	MNMAITLLTNTFLASLLVMIAFWLPQTNSY...
8	ND2_Ajam	MNPIIFSMIMTTVILGTTIVMTSSHWLMVW...
9	ATP8_Ajam	MPQLDTSTWFITILATILTLFIIMQLKIST...
10	ND4L_Ajam	MSLTYMNMFMAFTISLLGLLMYRSHMMSSL...
11	COX3_Ajam	MTHQTHAYHMVNPSPWPLTGALSALLLTSG...
12	CYTB_Ajam	MTNIRKTHPLLKIINSSFVDLPAPSSLSSW...
13	ND1_Ajam	MYLMNLLTTIVPVLLAVAFLTLVERKILGY...
...	...	...

Figure 11. File format of the original sequence data from the first database.

p-peptide terms using one amino acid (p = 1)	Proteins																
	COX3 Aame	COX2 Aame	CYTB Aame	ND4 Aame	ND5 Aame	ATP6 Aame	ND3 Aame	ND2 Aame	ATP8 Aame	ND1 Aame	ND4L Aame	COX1 Aame	ND6 Aame	COX2 Ajam	COX1 Ajam	ND4 Ajam	...
A	23	16	30	31	60	17	14	36	3	30	7	46	18	9	42	32	...
C	1	3	5	3	7	0	1	1	0	2	3	1	2	3	1	3	...
D	5	13	7	2	6	1	3	1	0	4	1	15	3	12	14	5	...
E	8	14	7	10	14	4	6	5	0	11	3	10	3	12	10	8	...
F	24	8	31	15	34	8	9	12	2	18	6	42	14	6	43	17	...
G	19	9	23	20	33	8	5	11	0	14	6	47	26	8	47	17	...
H	18	10	11	15	15	4	0	9	0	2	6	19	0	8	18	12	...
I	16	18	30	41	49	19	6	25	3	24	4	43	1	19	36	46	...
K	4	4	10	10	23	4	1	14	3	7	0	9	0	5	9	10	...
L	32	30	63	102	107	61	30	65	8	61	18	63	26	32	60	94	...
M	9	8	9	25	31	10	4	18	3	18	7	25	6	16	29	34	...
N	4	5	21	11	25	9	2	13	2	10	3	15	1	5	17	20	...
P	12	14	25	28	31	17	7	24	10	25	4	30	4	11	29	23	...
Q	6	7	8	12	19	7	4	9	2	6	2	10	0	6	6	11	...
R	5	5	8	12	9	5	2	3	0	8	2	8	6	6	9	10	...
S	18	21	25	39	38	17	8	32	4	29	11	27	11	21	30	35	...
T	22	12	25	48	63	20	6	35	10	17	8	39	4	21	38	42	...
V	14	18	18	10	20	8	1	15	0	17	5	33	36	11	40	12	...
W	12	5	11	12	12	4	5	10	5	8	1	17	5	5	17	14	...
Y	9	8	13	13	11	4	2	8	0	14	1	17	7	11	19	14	...

Figure 12. Protein frequency matrix of the first database (p-peptide = 1 amino acid).

dimension reduction is done according to the information in the protein matrix itself, instead of using external data, as utilized by Stuart et al. (2002a). They used prior categorical information concerning family memberships, which could be difficult for unknown sequences. According to these authors, "the development of a procedure whereby optimal dimension can be approximated without reference to prior information would represent an important advancement" (Stuart et al., 2002b). This is done by using the relative variance criteria.

Number of amino acids in the p-peptide terms	1st database (630 reference sequences)		2nd database (735 reference sequences)	
	#SVD factors for the protein matrix reconstruct	Size of the original protein frequency matrix	#SVD factors for the protein matrix reconstruct	Size of the original protein frequency matrix
1	2	20 x 630	2	20 x 735
2	13	400 x 630	17	400 x 735
3	28	8,000 x 630	32	8,000 x 735
4	35	160,000 x 630	56	160,000 x 735

Figure 13. Dimension reduction according to the relative variance criteria. SVD = singular value decomposition.

In the first database, the best result was achieved with a 3-peptide frequency matrix (size of 8000 rows and 630 columns), reconstructed by SVD with 28 terms: all 202 queries were correctly classified into each of the 13 gene families, with 100% accuracy (Figure 14).

For the second database, 735 sequences were selected to build the p-peptide frequency matrices, and 265 proteins were randomly selected as queries. By using a 3-peptide frequency matrix (size of 8000 rows and 735 columns), reconstructed by SVD with 32 terms, we obtained a global accuracy of 72% in classifying the 265 queries in one of the nine protein categories.

Actual gene family	Classification of the gene query according to the voting algorithm													
	ATP6	ATP8	COX1	COX2	COX3	CYTB	ND1	ND2	ND3	ND4	ND4L	ND5	ND6	Total
ATP6	18													18
ATP8		10												10
COX1			16											16
COX2				12										12
COX3					22									22
CYTB						16								16
ND1							14							14
ND2								16						16
ND3									13					13
ND4										18				18
ND4L											16			16
ND5												15		15
ND6													16	16
Total	18	10	16	12	22	16	14	16	13	18	16	15	16	202

Figure 14. Cross classification table results of the first database.

We had 100% accuracy for cytochrome, 92% for histone, 85% for keratin, 80% for globin, 74% for collagen, 66% for cyclohydrolase, 55% for pyrophosphatase, 52% for ferredoxin, and 65% for other proteins (Figure 15).

Actual gene family	Classification of the gene query according to the voting algorithm									
	Collagen	Cyclohydrolase	Cytochrome	Ferredoxin	Globin	Histone	Keratin	Pyrophosphatase	Other	Total
Collagen	17			3	1	1		1	0	23
Cyclohydrolase	2	19			1			5	2	29
Cytochrome			21						0	21
Ferredoxin	1	1		14				3	8	27
Globin	1	1		1	16			1	0	20
Histone						23		1	1	25
Keratin					1	2	23	1	0	27
Pyrophosphatase		5		2	3	1		17	3	31
Other	3	4	0	6	2	2	0	5	40	62
Total	24	30	21	26	24	29	23	34	54	265

Figure 15. Cross classification table results of the second database.

CONCLUSIONS

The algorithm and methods presented estimate relatedness between large numbers of biomolecules without the requirement of multiple alignments. Proteins are recoded as p-peptide frequency values using all possible overlapping p-peptides, which generates a matrix, reduced by SVD.

The results show that the application of LSI to evaluate the similarity of sets of sequences is a promising method and very attractive, because sequence alignments are neither generated nor required. In order to achieve results similar to those observed using edit distance analysis, we recommend that Euclidean distance be used as a similarity measure for protein sequences in LSI methods.

In a randomly selected GenBank dataset, the results were very promising, with 72% accuracy for classifying unknown gene queries in one of the nine protein categories. However, in a curated protein database, the method was perfect in classifying the unknown genes according to their actual category. Besides using the method in classification analysis, the information retrieval system can be used to generate phylogenetic inferences by using whole genome sequences and global data analysis.

ACKNOWLEDGMENTS

We are thankful to Professor Gary W. Stuart from Indiana State University, Department of Life Sciences, who sent us helpful data.

REFERENCES

Berry MW, Dumais ST and O'Brien GW (1995). Using linear algebra for intelligent information retrieval. *SIAM Rev*. 37: 573-595.

Berry MW, Drmac Z and Jessup ER (1999). Matrices, vector spaces, and information retrieval. *SIAM Rev*. 41: 335-362.

Cöster R (1999). Learning from relevance feedback in latent semantic indexing. Master's thesis (Asker L, orienting professor). Stockholm University and Royal Institute of Technology, Stockholm.

Deerwester S, Dumais S, Furnas G, Landauer T, et al. (1990). Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci*. 41: 1-13.

Everitt BS and Dunn G (2001). Applied multivariate data analysis. 2nd edn. Arnold, London.

Krawetz AS and Womble DD (2003). Introduction to Bioinformatics: a theoretical and practical approach. Humana Press, Totowa.

Kuruvilla FG, Park PJ and Schreiber SL (2002). Vector algebra in the analysis of genome-wide expression data. *Genome Biol*. 3: RESEARCH0011.

Neter J, Kutner MH and Nachstheim C (1996). Applied linear statistical models. 4th edn. Ie-McGraw-Hill, Boston.

Orengo VM (2004). Assessing relevance using automatically translated documents for cross-language information retrieval. PhD thesis (Huyck C, orienting professor), Middlesex University, London.

Rodrigues TS, Pacífico LGG, Teixeira SMR, Oliveira SC, et al. (2004). Clustering and artificial neural networks: classification of variables lengths of *Helminth* antigens in set of domains. *Genet. Mol. Biol*. 27: 673-678.

Schalkoff RJ (1992). Pattern recognition: statistical, structural and neural approaches. 1st edn. John Wiley & Sons Inc., New York.

Stuart GW and Berry MW (2003). A comprehensive whole genome bacterial phylogeny using correlated peptide motifs defined in a high dimensional vector space. *J. Bioinform. Comput. Biol*. 1: 475-493.

Stuart GW and Berry MW (2004). An SVD-based comparison of nine whole eukaryotic genomes supports a coelomate rather than ecdysozoan lineage. *BMC Bioinformatics* 5: 204.

Stuart GW, Moffett K and Baker S (2002a). Integrated gene and species phylogenies from unaligned whole genome protein sequences. *Bioinformatics* 18: 100-108.

Stuart GW, Moffett K and Leader JJ (2002b). A comprehensive vertebrate phylogeny using vector representations of protein sequences from whole genomes. *Mol. Biol. Evol*. 19: 554-562.

The Mathworks (1996). MATLAB: mathematical computation, analysis, visualization, and algorithm development (Version 5.0). Natick, Massachussetts, USA.

Thorne JL (2000). Models of protein sequence evolution and their applications. *Curr. Opin. Genet. Dev*. 10: 602-605.

Wall ME, Rechtsteiner A and Rocha LM (2003). Singular value decomposition and principal component analysis. In: A practical approach to microarray data analysis (Berrar DP, Dubitzky W and Granzow M, eds.). Kluwer, Norwell, 91-109.