Biophysics, Vol. 48, Suppl. 1, 2003, pp. S68–S70.

ProMult: Prediction of the Exon–Intron Structure by Spliced Alignment with Several Proteins

D. V. Vinogradov[1] and A. A. Mironov[1,2]

[1]*Department of Mathematics, Moscow State University, Vorobievy gory, Moscow, 119992 Russia*
[2]*State Scientific Center GosNIIGenetika, 1 1st Dorozhny proezd, Moscow, 117545 Russia*

Received November 7, 2003; in final form, January 6, 2004

Abstract—All existing similarity-based gene recognition algorithms can use only one protein as a template. The proposed enhancement of the ProFrame algorithm allows one to use structural information about several related proteins in multiple alignment. The new algorithm, named ProMult, was tested on a sample of human genes and demonstrated improved reliability of predictions.

Key words: gene recognition, exon–intron structure, multiple alignment

INTRODUCTION

Prediction of the exon–intron structure is an important step in genome annotation. Of all existing approaches, the best results are produced by algorithms based on analysis of similarity to related proteins [1–6]. The main idea of the similarity-based algorithms is to choose an exon chain most similar to a given related protein. In algorithms such as Procrustes [2, 3], GeneWise [4], INFO [5, 6] it is done by a variant of dynamic programming named spliced alignment.

If a sufficiently close homolog exists, predicted exon–intron structure is close to the true one (correlation 96–99% [7]). Even with errors in the genomic sequence (up to 6%), reliable predictions are still possible (correlation 90%) [8].

Rapid growth of available sequence data in many cases leads to availability of multiple related, although rather distant, proteins for a given gene. Exon–intron structures obtained by comparison with any particular protein may differ, especially if the evolutionary distance between the proteins is large. At that, construction of the correct structure requires application of additional considerations. Instead, we suggest to align all related proteins using Clustal [9] or a similar program, and then to do multiple alignment of the genomic sequence with the obtained protein profile.

The computation time of the algorithm (with ready multiple alignment) is $O((k + n)m)$, the memory is $O(km)$, where n is the length of the nucleotide sequence, m is the alignment length, k is the number of proteins in the alignment.

ALGORITHM

The algorithm is based on the ProFrame program [8], but instead of a single related protein, it performs multiple alignment with a protein profile, that is, a set of aligned proteins. Thus, in order to apply the dynamic programming technique, one has to define the weighting function for one element of the nucleotide sequence (translated codon) and one element of the profile, a column of amino acids (Fig. 1). We use a standard profile function:

$$w(\alpha, \overline{B}) = \sum_{\beta \in \overline{B}} Match(\alpha,\beta)P_\beta, \quad P_\beta = \log\frac{n_\beta}{N},$$

where $Match(\alpha, \beta)$ is the weight of matching two amino acids α and β, n_β is the number of symbols β in column $\overline{B}$, N is the height of column $\overline{B}$ (that is, the number of proteins in the multiple alignment), and the sum is taken over all β, encountered in $\overline{B}$. At that, P_β

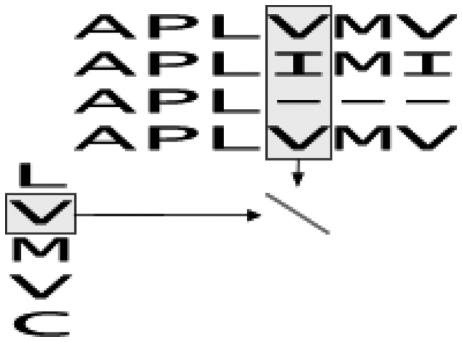

Fig. 1. Defining the weight function. Left: the protein profile, right: the studied sequence (translated into amino acids).

was linearly transformed to a function with the mean 0 and variance 1.

As a rule, the number of available related proteins is not too large, and thus a random mutation may strongly influence the distribution of amino acids in the multiple alignment. To offset this, we use pseudocounts

$$n'_\beta = n_\beta + \sum_\gamma \kappa\sqrt{N} P(\gamma \to \beta) P(\gamma), \quad N' = \sum_\beta n'_\beta,$$

$$w(\alpha, \overline{B}) = \sum_{\beta \in \overline{B}} Match(\alpha, \beta) P'_\beta, \quad P'_\beta = \log \frac{n'_\beta}{N'},$$

where κ is the smoothing coefficient, and $P(\gamma \to \beta)$ is the probability of mutation of amino acid to amino acid. The probabilities $P(\gamma \to \beta)$ were computed using the BLOSUM62 substitution matrix [11].

OPTIMIZATION

To decrease the computation time, one can use the technique of anchors [12]. Initially, for each position in the protein profile, tabulate the weight function for all 20 amino acids. Select the amino acid of the highest weight and set it in correspondence with this position. Now, having a sequence P instead of a profile, we make a hash table of all words of length k occurring in P with the hash function

$$H(C_k) := \sum_{i=0}^{k-1} Nim(C_k[i]) 20^i,$$

where *Num*() is the function making a correspondence between an amino acid and its number. The value of this function changes in a simple fashion when a letter at one end of a word is deleted, and a letter on the other end is added. Thus for each sequence D it is simple to generate a list of words of length k common to this sequence and the sequence P. Indeed, considering all subwords of D left to right, we compute the value of the hash function in constant time, and then use the hash table to find all occurrences of this subword in P. Any common subword of sequences P and D can be considered as a local alignment without mutations, insertions and deletions.

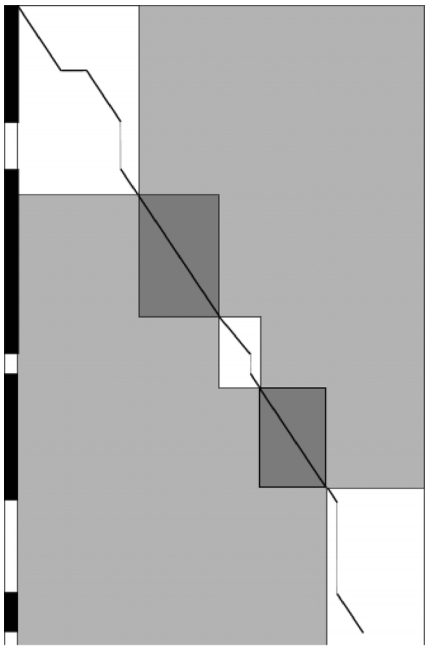

Fig. 2. The optimization scheme. The dynamic programming graph is shown. The Y axis represents the DNA, the X axis represents the protein profile. Dark gray rectangles: a set of non-contradictory local alignments. White rectangles: regions where the basic algorithm should be applied. The line at the left represents exons (black) and introns (white).

Now we try to extend these alignments on both ends, maximizing their score (still only single symbols are matched). We obtain a set of local alignments of length at least k with comparatively large score, and each of these alignments is defined by coordinates of its beginning on the graph of dynamic programming, and its length. It is clear that some of them cannot be part of one alignment, e.g., if one their co-ordinate and the other is different. Besides, some contradictions can be resolved if the length of one alignment is reduced, whereas others are not (see Fig. 2). Thus one needs to find the maximal set of non-intersecting alignments (maybe after the length correction) with the highest total score.

Define a pre-order relation: alignment A precedes alignment B if both coordinates of A are less

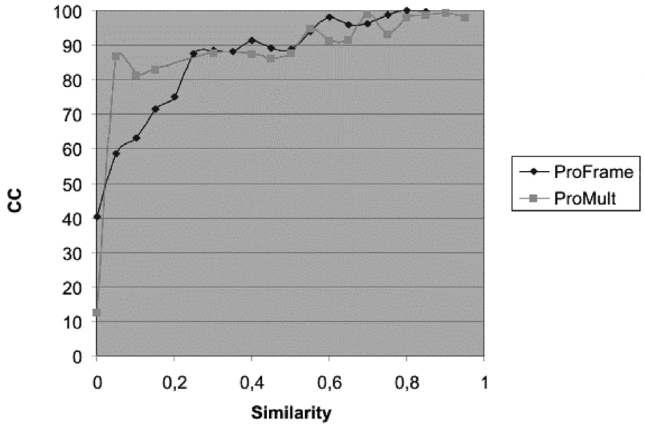

Fig. 3. Testing of ProMult and ProFrame on a sample of human genes. Horizontal axis: average similarity between the gene and the related proteins (rounded to 0.05). Vertical axis: the correlation coefficient.

than the corresponding coordinates of *B*. It is clear that *A* precedes *B* if it does not contradict *B* or if the contradiction is resolvable.

Now we reduce the problem to finding an optimal path in a graph. A weighted graph of dynamic programming is constructed as follows. Its vertices are alignments, with two special vertices (alignments of length zero in the upper left and lower right corners) being the source and sink respectively. Two vertices *A* and *B* are linked if *A* precedes *B*. The weight of an edge *AB* is the score of alignment *B*, maybe after correction required by *A*. Then the desired optimal subset of alignments corresponds to a route of the maximal total weight. It can be found by backtracking by the time proportional to the number of edges. Given a set of non-contradictory alignments, one can simply apply the initial algorithm to the regions between them (Fig. 3).

RESULTS AND DISCUSSION

ProMult was tested on a sample of 88 human genome fragments and 256 related amino acid sequences of average similarity to the genomic sequence equal 80%. The average length of the homologs was 450 amino acids, whereas the largest one was longer than 4500 amino acids. The number of related proteins per genomic sequence was 2 through 7. Multiple alignments were obtained using ClustalW with standard parameters.

All 88 predictions were made in about one hour on a PC with Athlon XP 2000+ processor under OS Linux. The quality of predictions was measured by the correlation coefficient [8]. The comparison of results with those of ProFrame is given in Fig. 3. It shows that ProFrame works better when close homologs are available, whereas ProMult is superior in working with only distant homologs.

ACKNOWLEDGMENTS

We are grateful to P. Novichkov for the ProFrame program.

This study was partially supported by HHMI (55000309) and LICR (CRDF RB0-1268).

REFERENCES

1. Gish, W. and States, D.J., *Nature Genetics*, 1993, vol. 3, no. 3, pp. 266–272.
2. Gelfand, M.S., *et al*., *PNAS*, 1996, Aug 20, vol. 93, no. 17, pp. 9061–9066.
3. Mironov, A.A., *et al*., *Genomics*, 1998, Aug 1, vol. 51, no. 3, pp. 332–339.
4. Birney, E. and Dubrin, R., *Proceedings International Conference Intelligent Systems in Molecular Biology*, 1997, vol. 5, pp. 56–64.
5. Hultner, M., *et al*., *Journal of Molecular Evolution*, 1994, vol. 38, no. 2, pp. 188–203.
6. Laub, M.T. and Smith, D.W., *Journal of Computational Biology*, 1998, vol. 5, no. 2, pp. 307–321.
7. Guigo, R., *et al*., *Genome Research*, 2000, vol. 10, no. 10, pp. 1631–1642.
8. Mironov, A.A., Novichkov, P.S, and Gelfand, M.S., *Bioinformatics*, 2001, vol. 17, no. 1, pp. 13–15.
9. Higgins, D.G. and Sharp, P.M., *Gene*, 1988, vol. 73, no. 1, pp. 237–244.
10. Henikoff, S. and Henikoff, J.G., *PNAS*, 1992, vol. 89, no. 22, pp. 10915–10919.
11. Nazipova, N.N., *et al*., *Comput. Applic. Bioscience*, 1995, vol. 11, pp. 423–426.