ICOT Technical Report: TR-0877

TR-0877

Multiple Sequence Alignment Editor Featured by Constraint-Based Parallel Iterative Aligner

by
M. Ishikawa, Y. Totoki, R. Tanaka (IMS)
& M. Hirosawa (Kazusa DNA)

May, 1994

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5

Institute for New Generation Computer Technology

MULTIPLE SEQUENCE ALIGNMENT EDITOR FEATURED BY CONSTRAINT-BASED PARALLEL ITERATIVE ALIGNER

M. Ishikawa*, Y. Totoki*,
R. Tanaka†, and M. Hirosawa‡

*Institute for New Generation Computer Technology (ICOT)
21F Kokusai bldg., 1-4-28 Mita, Minato-ku, Tokyo 108 JAPAN

†Information and Mathematical Science Laboratory, Inc.

‡Kazusa DNA Research Institute

Abstract

We have developed a multiple sequence alignment editor, on which a parallel iterative aligner works while considering user-defined constraints. The tree-based method, widely adopted as an alignment algorithm, tends to accumulate errors when aligning sequences of low similarities. It was recently found that iterative improvement could correct such errors, but that too many iteration cycles were needed to solve a practical-scale problem. We made the iterative alignment method efficient enough to be incorporated in an interactive alignment editor by employing parallel execution technology. The alignment editor, which features the parallel iterative aligner, realizes alignment which is not only fast and high-quality it is also constraint-based. When a user has some biological knowledge which indicates that some characters might be aligned in a column, a constraint can be defined for those characters. The constraint set is considered simultaneously in each iteration cycle of parallel alignment. Then appropriate multiple alignment is generated by the aligner and displayed on the editor's full-color display. The alignment editor also contains the following characteristic function modules: a phylogenetic tree drawer, a motif-database matcher, and a stem region specifier.

1 Introduction

Multiple alignment of protein/DNA sequences is an important method for predicting function and structure of molecules, and for investigating the phylogenetic relationships among creatures. Many algorithms have been devised to help biologists align sequences. Once a similarity value between characters is determined, Dynamic Programming (DP) [1] can be used to theoretically solve a multiple alignment problem. Applying an N-way DP to an N-sequence alignment problem, however, requires computation on the order of the N-th power of sequence length.

To date, most multiple alignment systems employ 2-way DP as a basic algorithm and combine the results of 2-way DP in a tree-like order for sequence similarity [2, 3, 4]. These algorithms, called tree-based algorithms, are fast enough to be applied to practical-scale problems. If sequence similarity is low, however, they often produce low-quality alignment. This happens because once an error occurs in the alignment process, the error cannot be corrected.

Recently, an iterative improvement algorithm was developed for multiple alignment [5, 6]. In this algorithm, alignment of all sequences is iteratively improved by repetitive application of group-to-group 2-way DP. The algorithm can remedy errors that occur in the alignment process. In our previous study [7], we revised the iterative improvement algorithm by introducing a best-first heuristic search and parallel execution. The Hidden Markov Model for multiple alignment can be considered equivalent to the iterative algorithm [8].

Such iterative algorithms improve an alignment with respect to its score. Whatever scoring system we use, the optimal-score alignment is not always the most significant result from a biological perspective. In addition to optimizing alignments under some scoring system, it is also important to refine them using biological knowledge. Alignment editors [9, 10, 11, 12] have been developed for this purpose. In this paper, we introduce our parallel iterative aligner and an alignment editor which features constraint-based alignment.

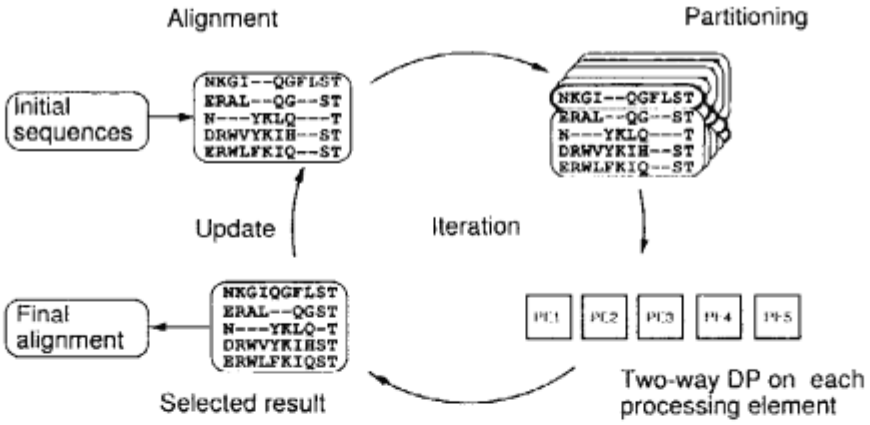

Figure 1: Parallel best-first iterative algorithm

2 Parallel Iterative Aligner

In this section, we propose two parallel iterative algorithms for multiple sequence alignment and evaluate their performance on some test problems with three sequential algorithms.

2.1 Algorithms

We chose a tree-based algorithm to evaluate the conventional sequential algorithms. The algorithm uses 2-way dynamic programming (DP) in a group-to-group manner [2] to align two sub-alignments. In this algorithm, a guided tree is first drawn by the UPGMA method [13] which depends on previous similarity analysis done by all 2-way DP pairs. Sequences are merged based on the branching order of the guided tree by applying group-to-group DP, which optimizes the alignment between sequence groups. The score to be optimized is the summation of all pairwise alignment scores between the groups. The pairwise alignment score is derived from a similarity value between amino acids and a linear relation of gap penalty: $a + bk$ where k is the gap length and a and b are the opening and extending gap cost. In the other algorithms described below, the same type of DP is used to align sequences.

Practical sequential iterative algorithms use round-robin iteration. In the algorithm, group-to-group DP realigns each sequence against the alignment of the other sequences. This process starts from an initial multiple-alignment state which has no gaps inside, and is repeated in a round-robin manner until no change occurs in a round-robin cycle. This algorithm was originally proposed for refining alignment obtained by a tree-based algorithm [14].

We propose the parallel best-first iterative algorithm (Figure 1). First, N sequences which have no gaps are input into an iteration cycle. The sequences are divided into a sequence and $N-1$ alignment sequences. The N different sets produced by the partitioning are then recombined in parallel by group-to-group DP. The results are compared and the best-score alignment is set at the starting point of the next iteration cycle. In this way, the iteration cycle gradually improves alignment of all the sequences. Iteration terminates when none of the N partitions can be improved further. This parallel routine requires N processing elements.

It is also interesting to couple the tree-based and iterative algorithms [15]. We coupled our round-robin algorithm with a tree-based algorithm to create a routine in which every alignment is refined in the round-robin manner until its convergence whenever the two sub-alignments are merged according to the tree method. We also developed a tree-based best-first iterative algorithm, which can be executed in parallel for both the best-first search and tree-based merge sections.

2.2 Experiments and results

The programs, written in the KL1 parallel logic programming language [16], were tested on a distributed-memory parallel machine PIM/m with 256 processing elements [17]. Experimental results are compared under the same scoring system of multiple sequence alignment. The N-sequence alignment score is the summation of $N(N-1)/2$ pairwise alignment scores. The similarity values are from table PAM250 [18]. The gap penalty is defined as a linear relation $a+bk$, where the opening and extending gap costs are $a=-7$ and $b=-1$.

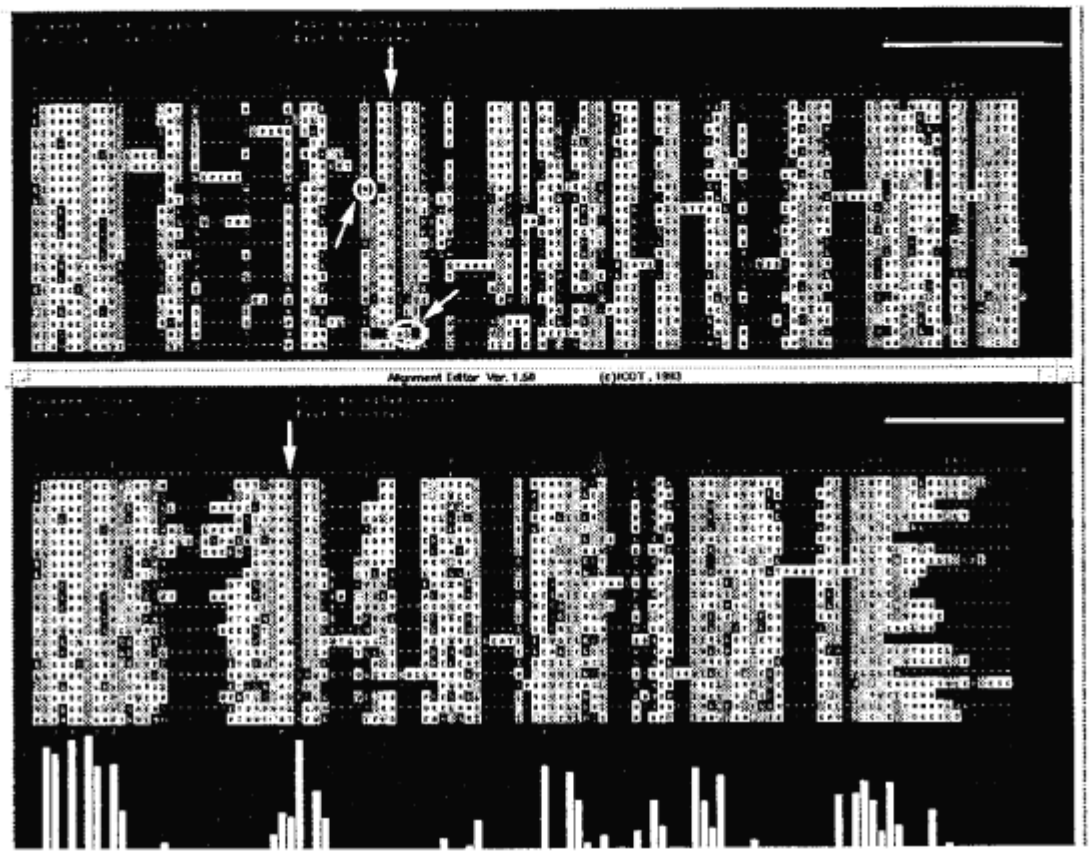

Figure 2: Alignments by tree-based (upper) and iterative (lower) algorithms

As test sequence sets, we used the catalytic domain of the protein kinase [19]. Figure 2 shows a couple of typical test set alignments. The bar graph shows alignment scores for each column. The alignment in the upper window was generated by the tree-based algorithm. The alignment in the lower window was done by the parallel best-first iterative algorithm. They show the merit of iterative alignment; the motif `A-x-K` indicated by an arrow, known as a part of the ATP-binding site, was completely aligned in the lower alignment, whereas two sequences were missed in the upper one shown by the circled positions.

Figure 3 compares the performance of the five algorithms. Each algorithm was executed on thirty test sets to optimize its alignment score. The alignment scores are shown at the top of the figure. The scores obtained from the same test set are connected by dotted lines. Each score is normalized by all connected scores. The bold lines connect the average scores for the thirty test sets. The bottom of the figure shows average execution times for the thirty test sets. Sequential algorithms were executed on a single processing element. Parallel algorithms used twenty-two. The comparison yielded the following information: (1) Although the tree-based algorithm is the fastest, it has the lowest average score. (2) The iterative algorithms coupled with the tree-based approach show better per-

formance in both average score and execution time than the iterative algorithms without the approach. (3) On average, the parallel algorithms take less time and yield slightly better scores than the sequential iterative algorithms.

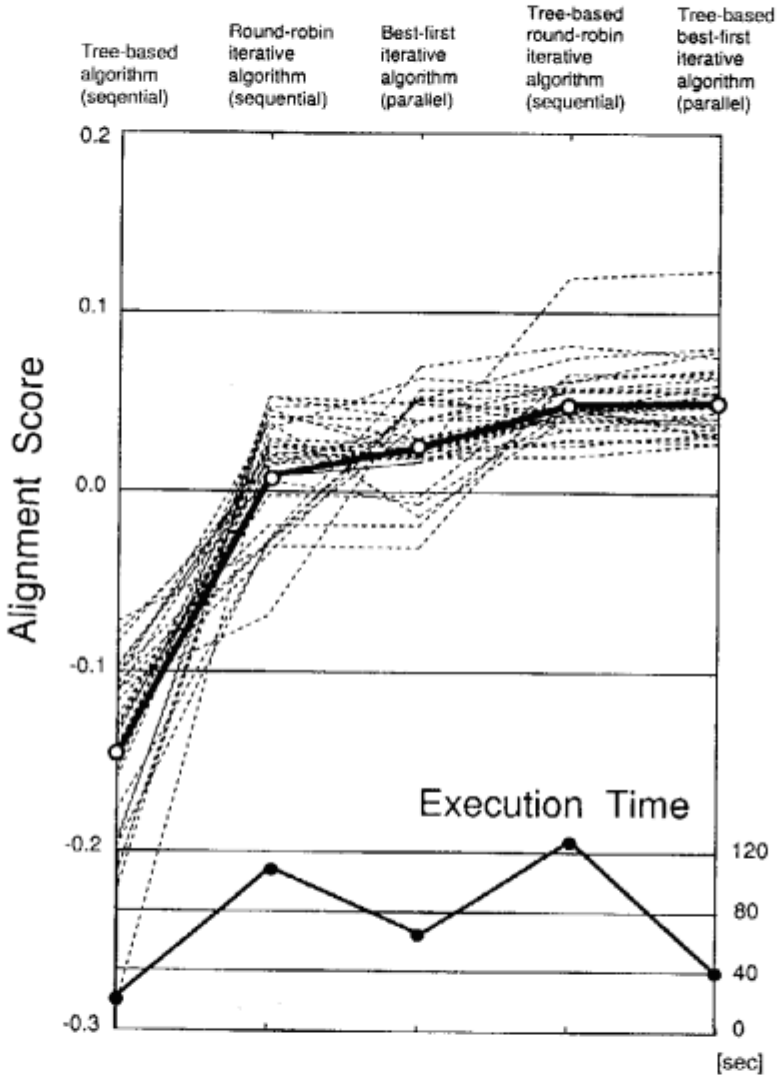

Figure 3: Performance comparison

3 Alignment Editor System

In this section, we introduce our alignment editor. First, we mention a recommended editing process. We then describe the features of constraint-based aligners. Finally, we discuss refinement tools. The entire editor system is written in the C programming language and works on UNIX workstations with OSF/Motif. The exception is the parallel

aligners, which work on parallel inference machines, and are now being implemented on a UNIX-based CM5 parallel computer.

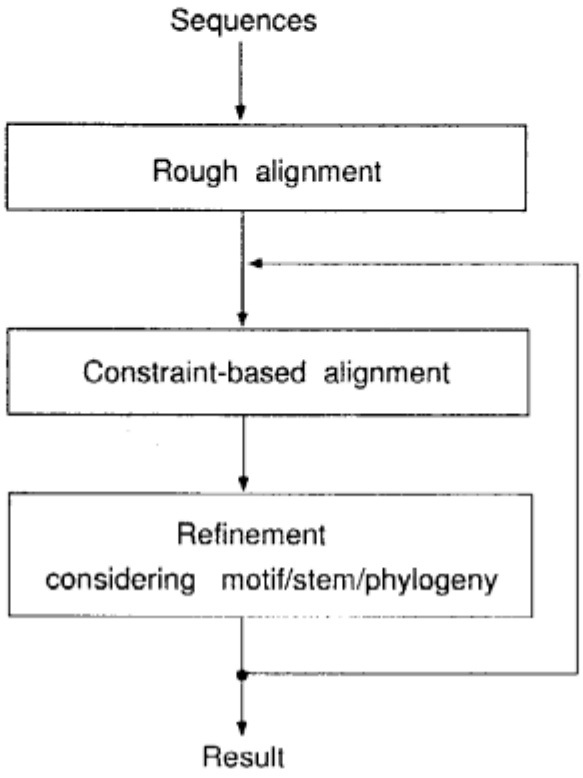

Figure 4: Editing process

3.1 Editing Process

Figure 4 shows a recommended process for using this alignment editor. Sequences are initially aligned roughly with the tree-based aligner, which is rapid enough to deal with a lot of long sequences. The rough alignment often reveals some half-aligned patterns, which can be recognized as constraints and forced into alignment. Sequences are elaborately realigned part by part with constraint-based iterative aligners. Being evaluated with respect to sequence motifs, secondary structures, or phylogenetic relationships, the alignment can then be refined manually by mouse-oriented operations and realigned repeatedly based on some newly imposed constraints. Thus, the final alignment can be arranged to meet user specifications.

3.2 Constraint-based Alignment

The editor features constraint-based alignment, in which the five alignment algorithms described in the previous section are available. The round-robin and best-first iterative

algorithms are useful for keeping a current alignment to some extent, because they are executed starting from the current alignment as the initial state of iteration.

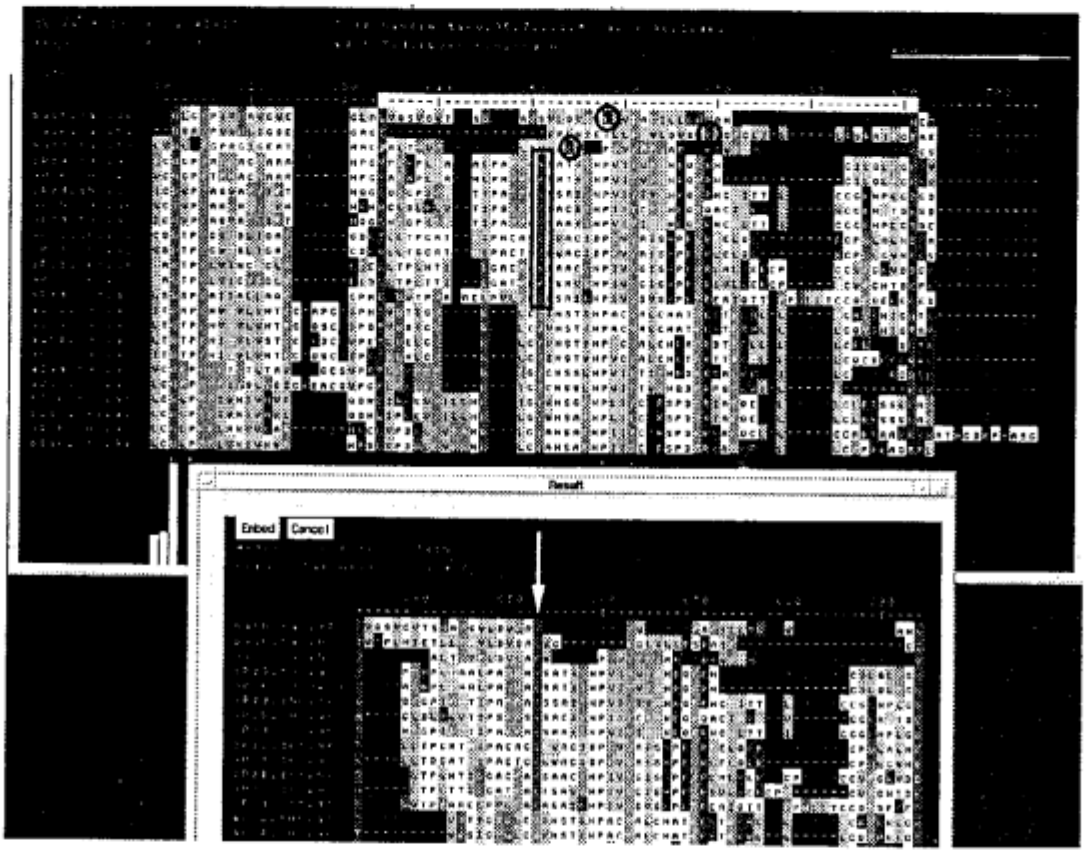

Figure 5: Constraint-based alignment

A typical operation of the constraint-based alignment is shown in Figure 5. In the upper window, protein sequences in the rhodopsin super family [20] are roughly aligned. The displayed alignment corresponds to the G helix of the bacteriorhodopsin family, whose structure has already been mapped. The first three sequences, from the bacteriorhodopsin family, are not similar enough to the other sequences to be well-aligned in the rough alignment. It is, however, known that proteins in the bacteriorhodopsin and rhodopsin families contain a retinal whose binding site is the lysine position in the G helix. This knowledge can improve the alignment. A constraint is imposed on the thirteen lysines, which are indicated by the Ks surrounded by black frames. The other sequences, from the signal receptor family, are not constrained, because proteins in the family have no retinals.

Based on the constraint, the tree-based parallel iterative aligner realigns partial alignment specified by the user. The small window at the bottom of Figure 5 displays the improved alignment, in which the thirteen lysines are aligned at the column indicated by an arrow. If the user accepts the partial alignment, it will be embedded in the main

alignment.

3.3 Refinement Tools

The alignment editor contains three characteristic tools: a motif-database matcher, a phylogenetic tree drawer, and a stem region specifier. Each tool is useful for refining alignment from a biological point of view.

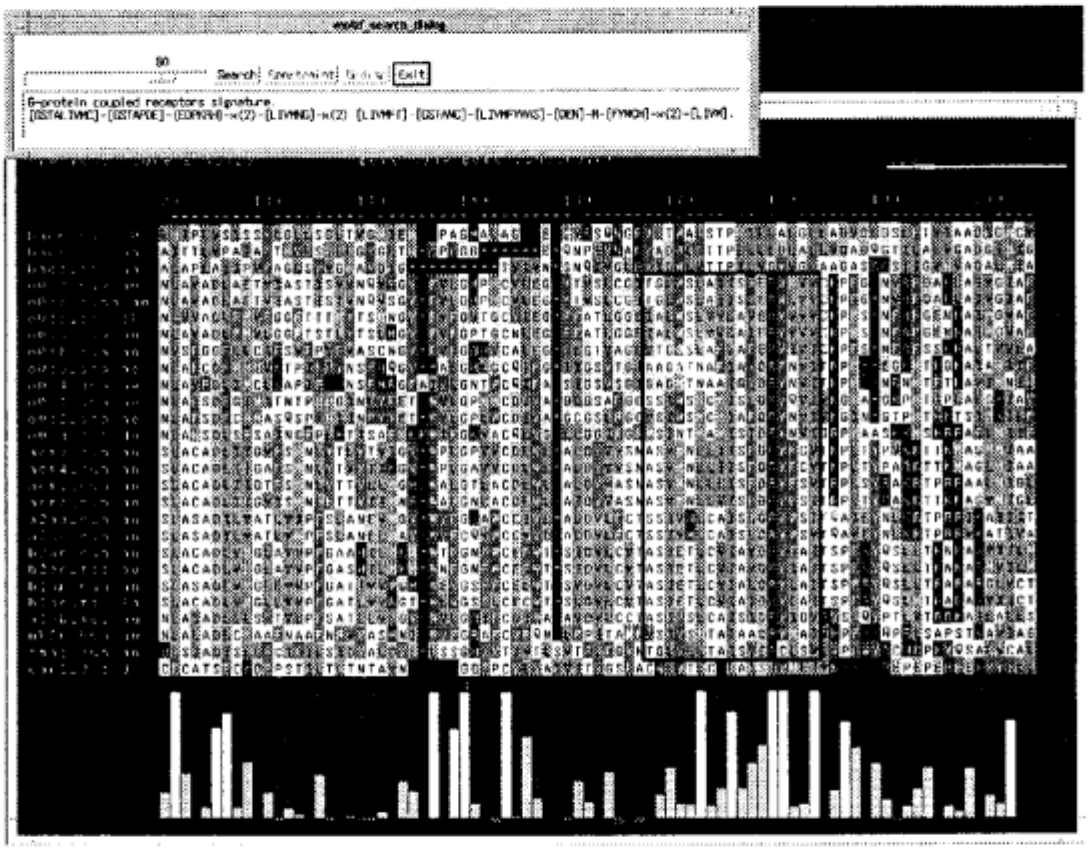

Figure 6: Sequence motif identification

The matcher identifies sequence motifs in a protein sequence alignment, retrieving motif data from the Prosite database [21] (release 9.00). Figure 6 shows a display when a signature motif for G-protein-coupled receptors was found in an alignment of the rhodopsin super family with more than eighty percent consensus. The region is indicated by a black rectangular frame. The motif is represented in Prosite as follows:

```
[GSTALIVMC]-[GSTAPDE]-{EDPKRH}-x(2)-[LIVMNG]-x(2)-[LIVMFT]-[GSTANC]-
[LIVMFYWAS]-[DEN]-R-[FYWCH]-x(2)-[LIVM].
```

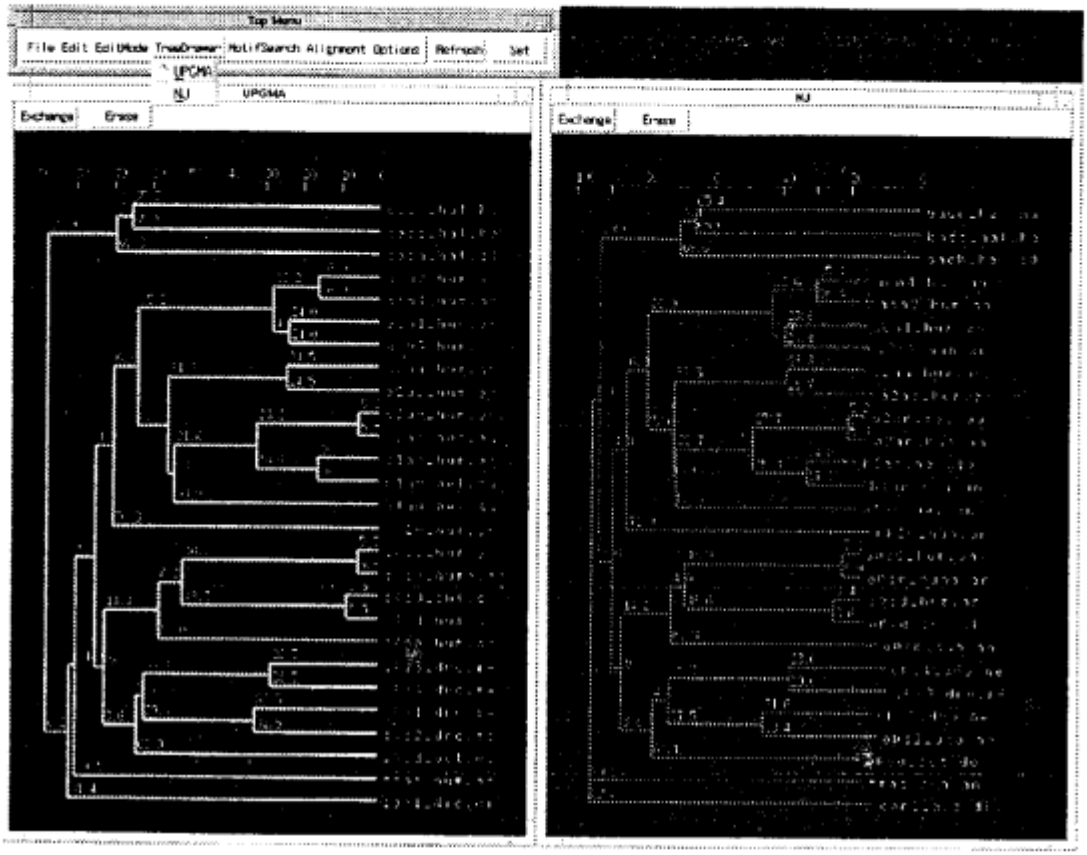

Figure 7: Phylogenetic trees

The drawer constructs an evolutional tree, dependent on the current editing alignment, with UPGMA [13] or NJ [22] method. Trees of aligned sequences shown in Figure 6 are introduced in Figure 7. The left tree is drawn by UPGMA (unweighted pair-group arithmetic average clustering), and the right one is done by NJ (neighbor joining). The estimated number of mutation events is represented on each branch. The order of sequences in an alignment display can be changed according to the evolutional tree.

The stem specifier indicates some possible stacking regions in an RNA sequence alignment by using a circle representation of the secondary RNA structure. Figure 8 shows a rough alignment (top) and its refined alignment (bottom) for Leucine-tRNAs, in which the first eight sequences are mitochondrial and the other are nucleic. In the rough alignment, the four main stems specify less than fifty-four percent consensus in whitened parts of the alignment. Four pseudostems are also displayed in the circle. After refinement considering the real stems as constraints, the stem regions aligned with more than seventy percent consensus and no pseudostems.

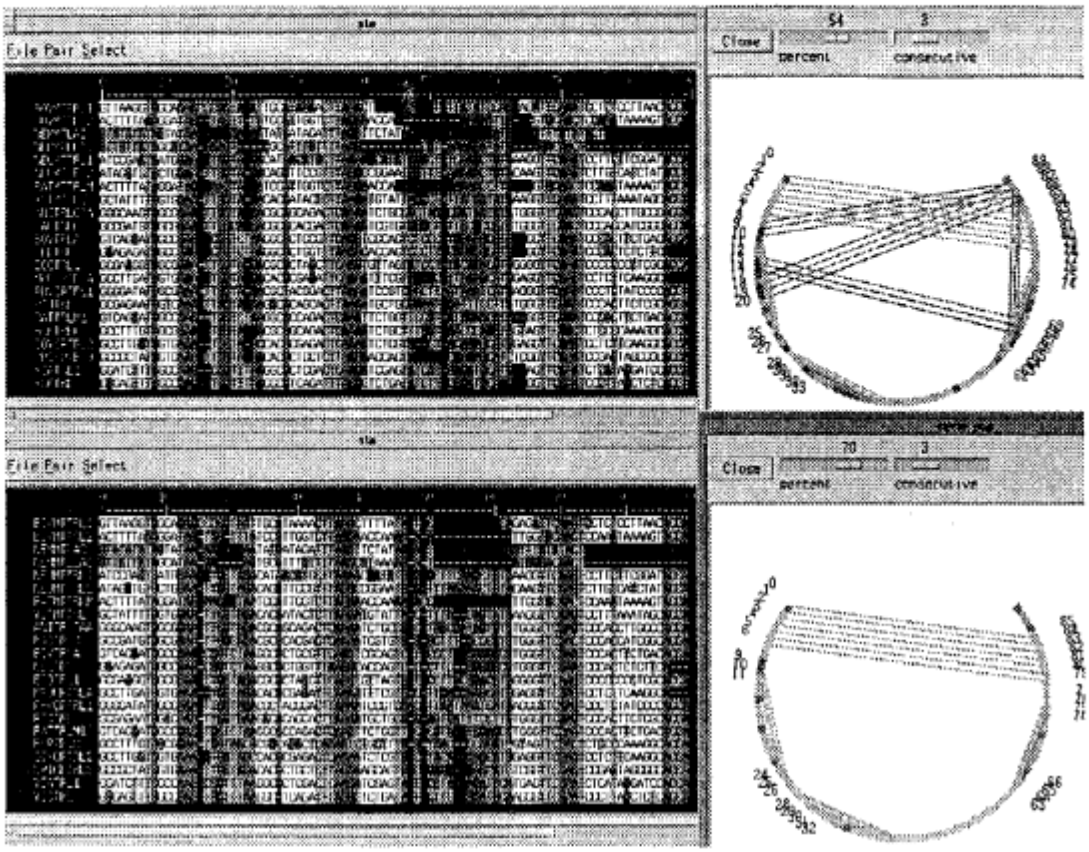

Figure 8: RNA stem specification

4 Conclusion

We have developed a multiple sequence alignment editor which provides three sequential and two parallel aligners. We compared performance and found that the tree-based best-first parallel iterative aligner gave the best results. The aligners work in a constraint-based way which helps users align sequences from various perspectives. The editor also provides a motif-database matcher, a phylogenetic tree drawer, and a stem region specifier to help refine alignments.

The programs are open to the public via Internet as *ICOT Free Software*. Anyone wishing to use this editor should contact **ftp.icot.or.jp** and transfer the file:

/ifs/exper-apps/pimos/editalign.tar.Z

Acknowledgements

We would like to thank Prof. S. Mitaku at Tokyo Univ. of Agriculture and Technology, Dr. K. Kuma at Kyoto Univ., and Prof. M. Lynch at Univ. of Oregon for their discussions and sequence data.

References

[1] Needleman, S.B., and C.D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins", *J. Mol. Biol.*, **48**, 443-453 (1970).

[2] Barton, J.G., "Protein multiple alignment and flexible pattern matching", In R.F. Doolittle (ed), *Methods in Enzymology*, **183**, Academic Press, 403-427 (1990).

[3] Feng, D.F., and R.F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees", *J. Mol. Evol.*, **25**, 351-360 (1987).

[4] Higgins, D.G., A.J. Bleasby, and R. Fuchs, "CLUSTAL V: improved software for multiple sequence alignment", *Comput. Appl. Biosci.*, **8**, 189-191 (1992).

[5] Berger, M.P., and P.J. Munson, "A novel randomized iterative strategy for aligning multiple protein sequences", *Comput. Appl. Biosci.*, **7**, 479-484 (1991).

[6] Gotoh, O., "Optimal alignment between groups of sequences and its application to multiple alignment", *Comput. Appl. Biosci.*, **9**, 361-370 (1993).

[7] Ishikawa, M., M. Hoshida, M. Hirosawa, T. Toya, K. Onizuka, and K. Nitta, "Protein sequence analysis by parallel inference machine", *Proc. Fifth Gener. Comput. Sys. '92*, 294-299 (1992).

[8] Tanaka, H., M. Ishikawa, K. Asai, and A. Konagaya, "Hidden Markov Models and Iterative Aligners: Study of their Equivalence and Possibilities", *Proc. 1st Int'l Conf. Intel. Sys. Mol. Biol.*, 395-401 (1993).

[9] Barber, A.M., and J.V. Maizel Jr, "SequenceEditingAligner: A multiple sequence editor and aligner", *Gene Anal. Tech.*, **7**, 39-45 (1990).

[10] Smith, S., "Genetic Data Environment, version 1.0".

[11] Schuler, G.D., S.F. Altschul, and D.J. Lipman, "A Workbench for Multiple Alignment Construction and Analysis", *PROTEINS*, **9**, 180-190 (1991).

[12] Scharf, M., R. Schneider, G. Casari, P. Bork, A. Valencia, C. Ouzounis, and C. Sander, "GeneQuiz: A workbench for sequence analysis", *Proc. 2nd Int'l Conf. Intel. Sys. Mol. Biol.*, (1994).

[13] Sneath, P.H.A., and R.R. Sokal, "Numerical Taxonomy", Freeman and Company (1973).

[14] Barton, J.G., and M.J.E. Sternberg, "A strategy for rapid multiple alignment of protein sequences", *J. Mol. Biol.*, **198**, 327-337 (1987).

[15] Subbiah, S., and S.C. Harrison, "A Method for Multiple Sequence Alignment with Gaps", *J. Mol. Biol.*, **209,** 539-548 (1989).

[16] Hirata, K. *et al.*, "Parallel and Distributed Implementation of Concurrent Logic Programming Language KL1", *Proc. Fifth Gener. Comput. Sys. '92,* 436-459 (1992).

[17] Nakashima, H. *et al.* "Architecture and Implementation of PIM/m", *Proc. Fifth Gener. Comput. Sys. '92,* 425-435 (1992).

[18] Dayhoff, M.O., R.M. Schwartz, and B.C. Orcutt, "A Model of Evolutionary Change in Proteins", *Atlas of Protein Sequence and Structure 5;3,* Nat. Biomed. Res. Found., Washington DC, 345-352 (1978).

[19] Hanks, K.S., A.M. Quinn, and T. Hunter, "The Protein Kinase Family", *Science,* **241,** 42-52 (1988).

[20] Hargrave, P.A., "Seven-helix receptors", *Current Opinion Struc. Biol.,* **1,** 575-581 (1991).

[21] Bairoch, A., "PROSITE: a dictionary of sites and patterns in proteins", *Nucleic Acids Res.,* **19,** 2241-2245 (1992).

[22] Saitou, N., and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees", *Mol. Biol. Evol.,* **4,** 406-425 (1987).