PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

PROTEIN SEQUENCE ANALYSIS BY PARALLEL INFERENCE MACHINE

Masato Ishikawa, Masaki Hoshida, Makoto Hirosawa,
Tomoyuki Toya, Kentaro Onizuka and Katsumi Nitta

Institute for New Generation Computer Technology
4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan
ishikawa@icot.or.jp

Abstract

We have developed a multiple alignment system for protein sequence analysis. The system works on a parallel inference machine PIM. The merits of PIM bring prominent features to the multiple alignment system. The system consists of two major components: a parallel iterative aligner and an intelligent refiner. The aligner uses a parallel iterative search for aligning protein sequences. The search algorithm is the Berger-Munson algorithm with its parallel extension. Our implementation shows that the algorithm extended in parallel can rapidly produce better solutions than the original Berger-Munson algorithm. The refiner uses condition-action rules for refining multiple sequence alignments given by the aligner. The rules help to extract motif patterns from ambiguous alignment patterns.

1 Introduction

Molecular biology and genetic technology have been advancing at an astonishing rate in recent years. Major activities in these fields are closely related to DNA and protein. This is because a set of DNA molecules in a cell contain the genetic information for the complete design of the living organism. This information is embodied as protein to build up the body and to keep its mechanisms alive. Each piece of genetic information, represented by a sequence of nucleic acids, is translated into a sequence of amino acids to form protein. As the method to determine DNA or protein sequences has progressed to its current state, the amount of known sequence data has grown rapidly. For example, *Genbank*, one of the most widely distributed databases, contains information on more than sixty million nucleotides. The growing number of genetic sequences in databases inevitably makes the field of genetic information processing one of the most important application areas for computer science.

The fundamental technique for analyzing genetic sequence data by computer is to examine similarities among sequences. This usually requires large amounts of computation to find the similarities, since there are a lot of sequences in the database to be examined. The computational problem can be partly solved with parallel implementation. There have been some experiments with parallel sequence analysis [Iyengar 1988]. Another approach to the problem is to furnish the analysis program with biological know-how as heuristics. Many consider that logic programming languages are a profitable way of implementing heuristics. Parallel sequence analysis with a logic programming language has been tried [Butler *et al.* 1990].

We have developed a multiple alignment system for protein sequence analysis. The system has been implemented on a parallel inference machine PIM using a parallel logic programming language KL1. The aim of this paper is to show PIM's availability in the field of genetic information processing. The organization of the rest of this paper is as follows. In Section 2, we briefly explain our application problems. We present our multiple sequence alignment system in Section 3. Then, the results of experiments and comparison with other methods are discussed in Section 4. Finally, conclusions are given in Section 5.

2 Protein sequence analysis

As described above, the genetic information, stored in DNA, is translated into sequences of amino acids. A chain of amino acids folds to become protein in water.

The structure of the protein depends on the sequence itself, that is, the same sequence will form the same structure. The function of the protein is chiefly determined by its structure, because proteins whose shapes are complementary can interact with each other.

Every protein is made up of twenty kinds of amino acids which are distinguished by twenty different code letters. A protein has about two hundred amino acids on average and is represented by a linear sequence of code letters. Because every amino acid has its own properties of volume, hydrophobicity, polarity and so on, the order of the amino acids in the protein sequence gives structure and function of the protein.

The protein sequence determination technique has been so established that more than twenty thousand sequences have been specified by the letters; this number is growing day by day. The structures of proteins are also being solved. Methods such as X-ray crystallography reveal how the linear chain of amino acids fold together. But this takes so many months to solve that only three hundred protein structures have been determined so far.

An important way of discovering new genetic information is inferring the unknown structure of a protein from its sequence. We do this by analyzing the sequence of amino acids, because, fortunately, proteins that have similar sequences have similar structures. Multiple sequence alignment is one of the most typical methods of sequence similarity analysis. The alignment of several protein sequences can provide valuable information for researching the function or structure of proteins, especially if one of the aligned proteins has been well characterized.

Let us show an example of multiple sequence alignment. The next set of sequences represents four parts of different protein sequences. Each letter in the sequences means an amino acid. For instance, GDVEK stands for a row of Glycine, Aspartic acid, Valine, Glutamic acid and Lysine.

```
GDVEKGKIFIMKCSQCHTVEKGGKHKTGPNLHGLFG
ASFAEAPAGTTGAKIFKTKCAQCHTVKGHKQGNGLFG
PYAPGDEKKGASLFKTAQCHTVEKGGANKVGPNLHGVFG
PPKARAPLPPGDAARGEKLRAAQCHTANQGGANGVGYGLVG
```

A good multiple sequence alignment for the given sequences is as follows:

```
----------GDVEKG-KIFIMKCSQCHTVEKGGKHKTGPNLHGLFG
--ASFAEAPAG--TTGAKIFKTKCAQCHTV-KG--HKQG---NGLFG
------PYAPGDEKKGASLFKT--AQCHTVEKGGANKVGPNLHGVFG
PPKARAPLPPGDAARGEKL---RAAQCHTANQGGANGVG---YGLVG
          *    *          ****   *      *    * *
```

Each sequence is shifted by gap insertion—dash characters. Each column of the resultant alignment has the same or similar amino acids. An identical pattern such as QCHT is considered to be an important site called a *sequence motif*, or simply a *motif*, because an important protein sequence site has been conservative along with evolutional cycles between mutation and natural selection. Multiple sequence alignment is useful not only for inferring the structure and function of proteins but also for drawing a phylogenetic tree along the evolutional histories of the creatures.

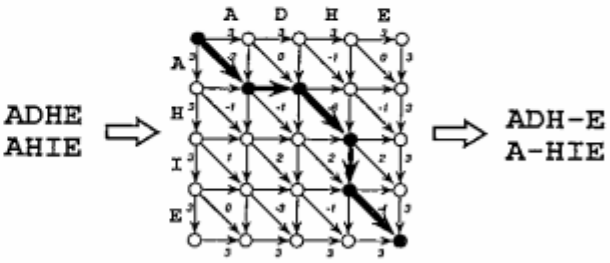

Figure 1: Pairwise dynamic programming

Computers partly solve the problem of multiple sequence alignment automatically, instead of relying on the hands and eyes of experts. The results obtained by computers, however, have not been as satisfactory as those by human experts. That is because multiple sequence alignment is one of the most time and space consuming problems. The dynamic programming algorithm [Needleman and Wunsch 1970, Smith and Waterman 1981,
Goad and Kanehisa 1982], theoretically, provides an optimal solution according to a given evaluation score. This, however, requires memory space for an N-dimensional array (where N is the number of sequences) and calculation time for the N-th power of the sequence length. Though a method was proposed to cut unnecessary computation in the dynamic programming algorithm [Carrillo and Lipman 1988], it still needs too much computation to solve any practical alignment problem. A number of heuristic algorithms for multiple alignment problems have been devised [Barton 1990, Johnson and Doolittle 1986] in order to obtain approximate solutions within a practical time. Most of these algorithms are based on pairwise dynamic programming.

Figure 1 shows the algorithm of dynamic programming applied to a tiny pairwise alignment. The algorithm searches the best path in the figurative network from the top left node to the bottom right node minimizing the total cost of arrows. Each cost indicated on an arrow reflects the similarities between the characters being compared. The best path corresponds to the optimal alignment; each arrow in the path corresponds to each column in the alignment. Vertical and

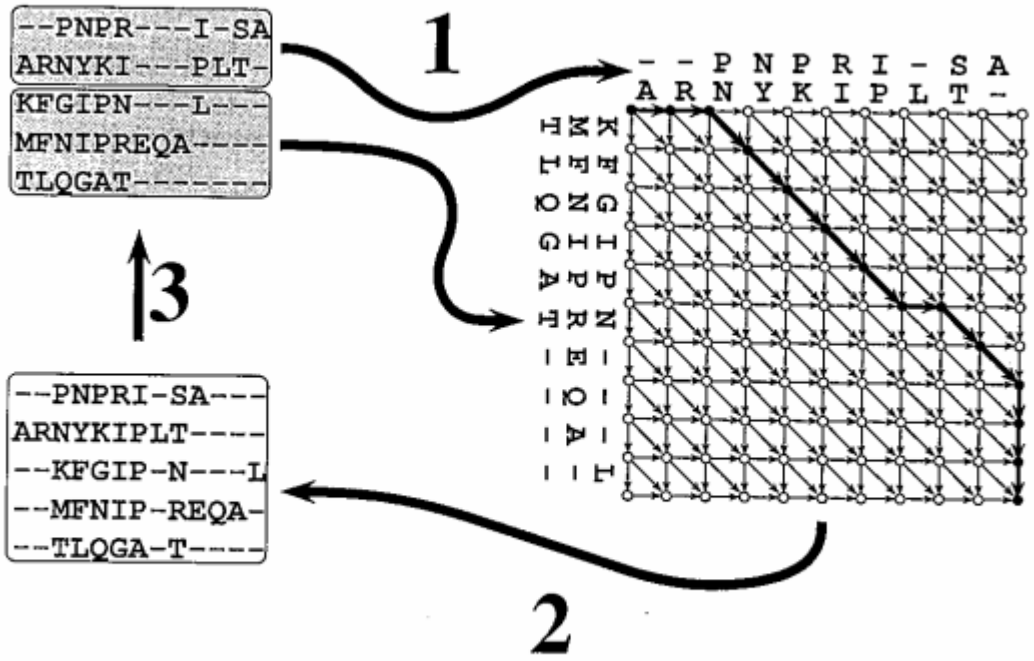

Figure 2: Iterative strategy of Berger-Munson algorithm

horizontal arrows indicate the insertion of gaps.

3 Multiple alignment system

We have developed a multiple alignment system for protein sequence analysis on PIM. The system consists of two components: a parallel iterative aligner and an intelligent refiner. The aligner uses a parallel iterative search for aligning protein sequences. The refiner uses condition-action rules for refining multiple sequence alignments given by the aligner.

3.1 Parallel iterative aligner

The search algorithm in the iterative aligner is the Berger-Munson algorithm extended in parallel. The B-M algorithm [Berger and Munson 1991] is based on the same pairwise dynamic programming method as conventional heuristic algorithms for multiple sequence alignment. The algorithm, however, features a novel randomized iterative strategy so as to generate a high-score multiple alignment.

Figure 2 illustrates the iterative strategy, whose procedure is as follows: the initially aligned sequences are randomly divided into two groups (step 1). By fixing the alignment of sequence members within each group we can optimize the alignment between the groups, using the pairwise dynamic programming method (step 2). The resultant alignment, in turn, is the starting point for the next alignment of a different pair of groups (step 3). Each iteration that improves the alignment between two sequence groups will also improve the global alignment.

Though the B-M algorithm often results in a much better multiple alignment than those obtained by conventional algorithms, its randomized iteration needs more than a few hours to solve multiple alignment of a practical scale. When a parallel machine is available, the iterative strategy extended in a parallel way is fairly helpful for reducing execution time. The B-M algorithm extended in parallel is as follows: all $2^{n-1}-1$ possible partitions of n aligned sequences are respectively evaluated by the pairwise dynamic programming method. In each iteration, the evaluation is executed in parallel and the alignment which has the best score is selected as the starting point for the next iteration.

3.2 Intelligent refiner

Aligning multiple protein sequences requires biological know-how, since the alignment score is not sufficient to evaluate them. The intelligent refiner holds dozens of condition-action rules that reflect the biological know-how for refinement. Part of the biological know-how has been obtained by interviewing human

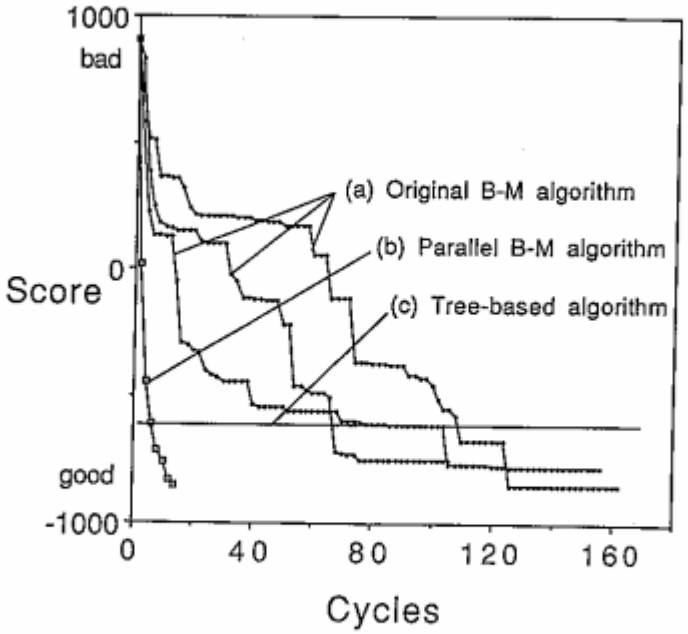

Figure 3: Comparing alignment score histories

experts. Another part of it corresponds to the information contained in a motif database *PROSITE*.

Let us explain an example of the condition-action rule, which features a well-known motif pattern called *Zinc Finger*. Zinc Finger is characterized by two separated Cs, Cysteines, and two separated Hs, Histidines. The condition part of the rule checks whether an alignment has the half-aligned motif pattern of Zinc Finger or not, and if it finds the *weak motif* pattern, it tries, in its action part, to enhance the weak pattern to make it strong (see Figure 4). Every condition-action rule is represented with a parallel logic programming language KL1.

4 Experimental results

Our multiple sequence alignment system works on PIM/m, a MIMD-type parallel machine equipped with up to 256 processing elements (PEs). We have investigated the performance of our system by testing the two components separately.

4.1 Parallel iterative aligner

The B-M algorithm enables us to gradually improve global multiple alignment. Improvement is evaluated by the alignment score. We have defined the alignment score as follows. The alignment score is a total summation of the similarity scores of every pair of aligned sequences, each of which is derived by summing up the similarity values of every character pair in the column. Each similarity value is given by the *odds matrix*. A *gap penalty* corresponding to each row of gaps in the two sequences is added to the similarity score.

We use *PAM*250 [Dayhoff *et al.* 1978] as the odds matrix, each value of which is a logarithm of the mutation probability of a character pair; zero is the neutral value. We have reversed the sign of each value of the matrix to assimilate the habit of optimization problems. So the most similar character pair, `W` *vs.* `W`, gives the lowest value, -17, and the least similar pair, `W` *vs.* `C`, gives the highest value, 8.

The gap penalty imposed on a row of k gaps is a linear relation: $a + bk$ where a and b are parameters. We set $a = 4$ and $b = 1$ as default values. The linear relation is feasible and popular for alignment done by the dynamic programming algorithm [Gotoh 1982]. Character pairs *gap vs. gap* and *outside gap vs. any character* are ignored; they are assigned the neutral value zero.

We have implemented three algorithms for comparison analysis: the original B-M algorithm, the B-M algorithm extended in parallel and the tree-based algorithm. The tree-based algorithm [Barton 1990] is one of the most typical and conventional methods for multiple sequence alignment. Figure 3 compares the histories of the alignment scores obtained by the algorithms.

```
(1)Before:
  ------------ILD---FHE-KLLHPGIQKT---TKLF--GET---YYFPNSQLLIQNIINECSICNLAKTEHRNTDM--P-TKTT
  ------------LLD---F-----LHQLTHLSFSKMKALLERSHSPYYMLNRDRTL-KNITETCKAC--AQVNASKSAVKQG-TR--
  LTDALLIT---PVLQ---LSP-AELHSFTHCG---QTAL--TLQ----GATTTEA--SNILRSCHAC---RGGNPQHQMPRGHI---
  ------VADSQATFQAYPLREAKDLHTALHIG---PRAL--SKA---CNISMQQA--REVVQTCPHC------NSAPALEAG-VN--
  ------------ISD--PIHEATQAHTLHHLN---AHTL--RLL---YKITREQA--RDIVKACKQC---VVATPVPHL--G-VN--
  ------------ILT--ALESAQESHALHHQN---AAAL--RFQ---FHITREQA--REIVKLCPNC---PDWGSAPQL--G-VN--
  (score = -781)           *   ^        ^                      ^   *  *              ^

(2)After:
  ------------ILD---F------HEKLLHPGIQKTTKLF-GET---YYFPNSQLLIQNIINECSICNLAKTEHRNTDM--P-TKTT
  ------------LLD---F-----LHQ-LTHLSFSKMKALLERSHSPYYMLNRDRTL-KNITETCKAC--AQVNASKSAVKQG-TR--
  LTDALLIT---PVLQ---LSP-AELHS-FTHCG---QTAL--TLQ----GATTTEA--SNILRSCHAC---RGGNPQHQMPRGHI---
  ------VADSQATFQAYPLREAKDLHT-ALHIG---PRAL--SKA---CNISMQQA--REVVQTCPHC------NSAPALEAG-VN--
  ------------ISD--PIHEATQAHT-LHHLN---AHTL--RLL---YKITREQA--RDIVKACKQC---VVATPVPHL--G-VN--
  ------------ILT--ALESAQESHA-LHHQN---AAAL--RFQ---FHITREQA--REIVKLCPNC---PDWGSAPQL--G-VN--
  (score = -762)           *    *        *                    ^   *  *
```

Figure 4: Application of intelligent refiner

Every algorithm solves the same small alignment problem which consists of seven sequences with eighty code letters each. The initial state of the alignment problem has no gaps inside the sequences.

(a) **Original B-M algorithm:** The randomized iterative strategy executed by a single PE is applied to the alignment problem. Each iteration cycle takes twenty-eight seconds on average. We set thirty-two as the *convergence condition*; execution stops, if no variation of alignment score is found during thirty-two iteration cycles. Three runs with distinct sequences of random numbers give converged alignment scores: -752, -779 and -851.

(b) **Parallel B-M algorithm:** The best-choice iterative strategy executed by sixty-three PEs is applied to the alignment problem. In each iteration, sixty-three possible partitions of aligned sequences are distributed to the PEs so that they can be evaluated at the same time. Each iteration cycle takes thirty seconds on average. The execution stops if no variation of alignment score is found. The final alignment, which is obtained at the fourteenth cycle with score -851, is the same alignment as one of the three obtained in (a).

(c) **Tree-based algorithm:** The tree-based algorithm is a conventional method to rapidly produce a practical multiple alignment. The algorithm aligns sequences one after another by pairwise dynamic programming. The order in which sequences are aligned depends on the tree-like representation that was previously determined by analyzing the distance of similarity of every pair in the sequences. Our implementation of the algorithm solves the problem in eighty seconds. The alignment score of the solution, -617, is indicated by a horizontal line.

We made the following observations from these results.

1. The parallel B-M algorithm (b) solves alignment problems about ten times faster than the original B-M algorithm (a).
2. The original B-M algorithm (a) gives different alignments depending on the sequence of random numbers, whereas the parallel B-M algorithm (b) gives a constant alignment that often has a better score than obtained by (a).
3. (a) and (b) show that either of the B-M algorithms gives a much better alignment than the conventional tree-based algorithm (c).

Thus, the parallel B-M algorithm can constantly generate high-score alignments in a small number of cycles. And PIM can execute the algorithm in a practical time.

4.2 Intelligent refiner

The refiner holds dozens of condition-action rules and checks a given alignment with the condition parts in parallel. If some condition parts match the alignment, the action parts paired with the condition parts are executed so as to produce candidates for a refined alignment. After evaluation of the candidates, some of them are displayed as refined alignments. Let us show an example of the refinement.

Figure 4 (1) shows an alignment which contains a weak Zinc-Finger motif pattern. `C`s are aligned completely in two columns, but `H`s are not aligned completely in two columns; `Q` exists among identical `H`s in

a column. (* indicates a completely aligned column and ^ indicates an almost completely aligned column.) Application of the intelligent refiner to the alignment produces Figure 4 (2).

The condition-action rule described in Section 3 has worked on the refinement process. The Zinc-Finger motif pattern is brought into full relief in the refined alignment. Although it has a score that is slightly worse than the previous alignment, it is a valuable alignment from a biological point of view.

Thus, the intelligent refiner helps to extract motifs from ambiguous alignment patterns and to produce biologically valuable alignments. Constructing the intelligent refiner on PIM is a profitable way, since KL1, a logic programming language on PIM, is suitable for representing such biological know-how.

5 Conclusions

We have developed a multiple sequence alignment system on PIM. The parallel iterative aligner of this system with the extended Berger-Munson algorithm can constantly generate better alignments than conventional methods in a practical time. The intelligent refiner of this system uses condition-action rules for refining alignments given by the aligner. The rules reflecting biological know-how help us to extract motif patterns from ambiguous alignment patterns. These results show that PIM is fairly available in the field of genetic information processing.

The extended algorithm searches all $2^{n-1}-1$ possibilities in parallel and selects the best one. There is a problem because the number of possibilities increases exponentially as the number of sequences grows. Some practical alignment problems with more than twenty sequences have about a million possibilities. In those cases, preprocessing with *cluster analysis* is useful for reducing the possibilities without reducing the quality of the resultant alignment. The cluster analysis divides given sequences into a few groups based on similarities between sequences; similar sequences gather in the same groups.

One of our future works is to represent complex biological know-how as a combination of simple condition-action rules.

Acknowledgments

We gratefully acknowledge Osamu Gotoh for calling our attention to the B-M algorithm. We would also like to thank Naoyuki Iwabe and Kei-ichi Kuma for providing us with the biological know-how to refine alignments.

References

[Barton 1990] J. G. Barton. Protein multiple sequence alignment and flexible pattern matching. In R. F. Doolittle (ed), *Methods in Enzymology Vol.183*, Academic Press, 1990. pp.403–428.

[Berger and Munson 1991] M. P. Berger and P. J. Munson. A novel randomized iterative strategy for aligning multiple protein sequences. *Computer Applications in the Biosciences*, **7**, 1991. pp.479–484.

[Butler *et al.* 1990] Butler, Foster, Karonis, Olson, Overbeek, Pflunger, Price and Tuecke. Aligning Genetic Sequences. *Strand: New Concepts in Parallel Programming*, Prentice-Hall, 1990, pp.253–271.

[Carrillo and Lipman 1988] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, **48**, 1988, pp.1073–1082.

[Dayhoff *et al.* 1978] M. O. Dayhoff, R. M. Schwartz and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff (ed), *Atlas of Protein Sequence and Structure Vol.5, Suppl.3*, Nat. Biomed. Res. Found., Washington, D. C., 1978, pp.345–352.

[Goad and Kanehisa 1982] W. B. Goad and M. I. Kanehisa. Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries. *Nucleic Acids Res.*, **10**, 1982, pp.247–263.

[Gotoh 1982] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 1982, pp.705–708.

[Iyengar 1988] A. K. Iyengar. Parallel DNA Sequence Analysis. *MIT/LCS/TR-428*, 1988.

[Johnson and Doolittle 1986] M. S. Johnson and R. F. Doolittle. A method for the simultaneous alignment of three or more amino acids sequences. *J. of Mol. Evol.*, **23**, 1986, pp.267–278.

[Needleman and Wunsch 1970] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. of Mol. Biol.*, **48**, 1970, pp.443–453.

[Smith and Waterman 1981] T. F. Smith and M. F. Waterman. Identification of common molecular subsequences. *J. of Mol. Biol.*, **147**, 1981, pp.195–197.