ICOT Technical Memorandom: TM-1219

TM-1219

Protein Sequence Analysis by Parallel Inference Machine

by
K. Nitta

September, 1992

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Protein Sequence Analysis by Parallel Inference Machine

Katsumi Nitta

Institute for New Generation Computer Technology
4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan

Institute for New Generation Computer Technology (ICOT) is the research center of the Fifth Generation Computer Project in Japan. ICOT has developed and applied its knowledge processing technologies based on logic programming into such things as a parallel computer, PIM; a database management system, Kappa; a deductive and object oriented database language, Quixote; and various knowledge processing tools. In order to evaluate the effectiveness of our technologies, we developed genome analysis software such as protein sequence analysis programs, a protein folding simulator and an integrated protein knowledge base.

1 Introduction

The Fifth Generation Computer Systems project (FGCS project) started in 1982 as an eleven-year project supported by Ministry of International Trade and Industry. The purpose of this project is to develop infrastructures for knowledge information processing in the 1990s. Though computers have become smaller and become faster, serious limitations are still felt when developing knowledge information processing systems (KIPS). To develop an infrastructure for KIPS, a wide range of basic research into hardware, basic software, and knowledge processing techniques and the like is needed. Institute for New Generation Computer Technology (ICOT) is the center for research in the FGCS project. ICOT has developed knowledge processing technologies based on logic programming such as a parallel computer, PIM; a logic programming language, KL1; a parallel operating system, PIMOS; and various knowledge processing support tools. We applied these technologies to the field of molecular biology to show the effectiveness of logic programming and parallel inference.

In Section Two, overview of FGCS project and the architecture of PIM are introduced. In Sections Three and Four, a parallel protein sequence analysis progran and a parallel protein folding simulator are introduced. In Section Five, the knowledge representation language Quixote and its application to biological databases are introduced.

2 Overview of FGCS Project and PIM

There are various approaches to realize KIPS. A significant feature of the FGCS project is that "*predicate logic*" is adopted as the underlying principle because predicate logic is a convenient tool for formalizing human knowledge processing and human reasoning. The machine languages of our computers are based on predicate logic. Therefore, we call our computers "*inference machines*".

In the first stage of the FGCS project (1982-1984), ICOT developed a sequential inference machine PSI, a logic programming language ESP and an operating system SIMPOS. Using these tools, ICOT developed knowledge processing technologies based on logic programming.

In the intermediate stage (1985-1988), ICOT has developed an experimental parallel inference machine Multi-PSI, a parallel logic programming language KL1 and a parallel operating system PIMOS. A Multi-PSI has 64 processing elements (PEs), each of which consists of a processor, a local memory and a communication unit.

In the final stage (1989-1992), we developed five models of the parallel inference machine

- PIM/m, PIM/p, PIM/k, PIM/c and PIM/i [1]. They employ different hardware technologies. Figure 1 shows the architecture of PIM/m. PIM/m is composed of 256 PEs with the PEs connected in a two dimensional network.

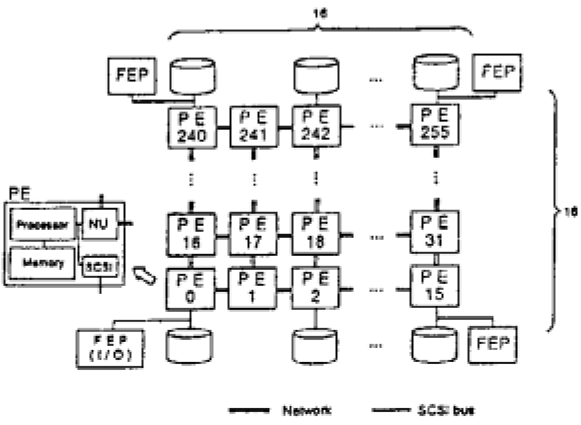

Figure 1: Architecture of PIM/m

In addition to the parallel operating system, PIMOS, a database management system, Kappa, and knowledge processing tools such as a theorem prover, a constraint solver and a knowledge representation language have been developed on PIMs (Figure 2). To show the effectiveness of these technologies, the genetic information processing group (M.Ishikawa, M.Hirosawa, M.Hoshida, T.Toya, K.Onizuka and H.Tanaka) developed a parallel multiple sequence aligner, a parallel protein folding simulator and experimental protein knowledge bases.

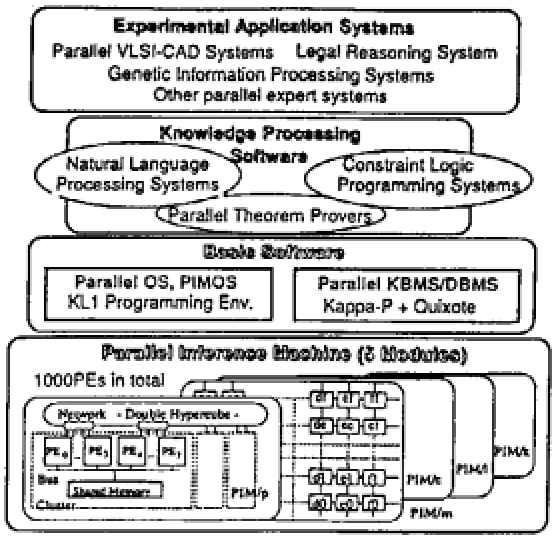
Experimental Application Systems
Parallel VLSI-CAD Systems
Legal Reasoning System
Genetic Information Processing Systems
Other parallel expert systems
Knowledge Processing Software
Natural Language Processing Systems
Constraint Logic Programming Systems
Parallel Theorem Provers
Basic Software
Parallel OS, PIMOS
KL1 Programming Env.
Parallel KBMS/DBMS
Kappa-P + Quixote
Parallel Inference Machine (5 Modules)
1000PEs in total
Network - Double Hypercube -
Bus
Shared Memory
Cluster
PIM/p
PIM/c
PIM/i
PIM/k
PIM/m

Figure 2: Organization of prototype system

3 Protein Sequence Aligner

We have developed a multiple alignment system for protein sequence analysis on PIM [2]. The system consists of two components: a parallel iterative aligner and an intelligent refiner[3]. 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

We developed a *parallel iterative aligner* in order to improve the quality of automatic multiple sequence alignment. The algorithm of this parallel iterative aligner is based on the Berger-Munson algorithm. Firstly, we introduce the B-M algorithm, and then we explain the parallel iterative aligner.

The B-M algorithm features a novel randomized iterative strategy so as to generate a high-score multiple sequence alignment (Figure 3). It works by randomly dividing the initially aligned sequences into two groups. By fixing the alignment of sequence members within each group, we can optimize the alignment between the groups, using two-dimensional dynamic programming (DP). The resultant alignment, in turn, is the starting point for the next alignment of a different pair of groups. Each iteration that improves the alignment between any two sequence groups will also improve the global alignment.

This iterative strategy often results in much better multiple alignments than those obtained

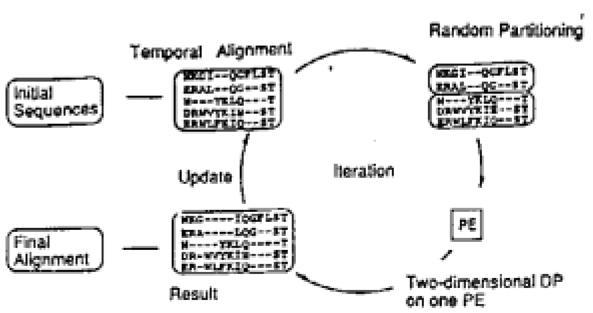

Figure 3: Original B-M algorithm

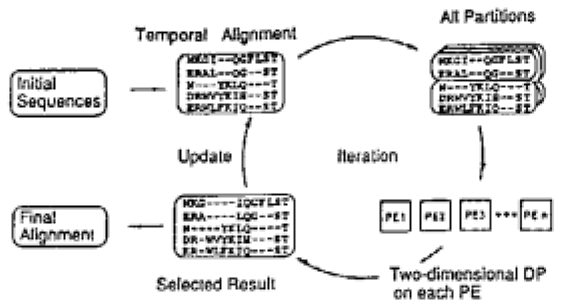

Figure 4: Parallel B-M algorithm

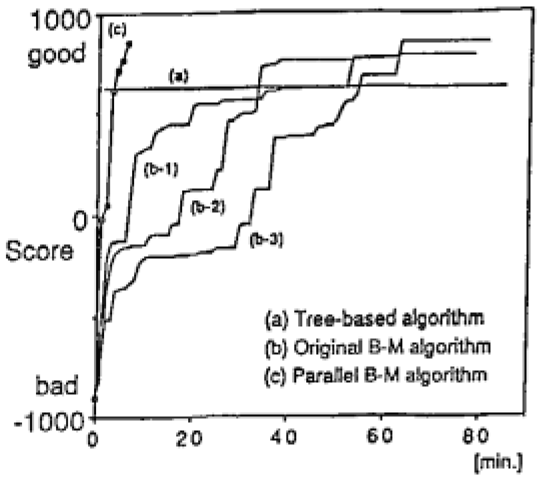

Figure 5: Performance comparison

by conventional algorithms. In Figure 5, (b) shows the performance when this B-M algorithm is applied to seven sequences. Lines (b-1), (b-2) and (b-3) are different in respect to random numbers. The quality of the results is superior to that obtained by a conventional tree-based algorithm, which is shown as horizontal line (a). However, the B-M algorithm needs large amounts of time as the number of sequences grows.

We can reduce the execution time when a parallel machine is available. Figure 4 shows the algorithm of our parallel iterative aligner. Every possible partitioning into two groups of aligned sequences can be respectively evaluated by two-dimensional DP in a parallel way. 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. The performance of this method is shown as line (c) in Figure 5. This shows that the parallel iterative aligner performs better than the original B-M method in terms of execution time.

Furthermore, we have developed an effective heuristic search, the *restricted partitioning technique*. Applying the iterative strategy, we realized that the number of sequences in the divided groups is important. As partitioning divides N sequences into k sequences and $N-k$ sequences, a smaller k tends to provide a larger improvement when using two-dimensional DP. The restricted partitioning technique preferentially selects partitionings which have a small k such as one or two. This can restrict the search space and reduce the execution time remarkably. Parallel iterative aligners with this technique can manage more sequences at the same time than those without.

3.2 Intelligent refiner

The objective of multiple sequence alignment is to identify the biologically important portions of sequences. However, an alignment with an optimal score is not necessarily the best alignment for the purpose. The intelligent refiner iteratively refines the alignment produced by the aligner and gradually identifies the biologically important portions.

Alignment experts refine alignment by using their alignment know-how and knowledge on biology. We interviewed experts and extracted their know how and knowledge. Then, based on this extracted information, we implemented an initial version of the intelligent refiner.

The overall framework of the intelligent refiner is shown in Figure 6. *The control module* iteratively modifies the alignment by calling rules (represented using KL1) stored in *the refinement rule base* and produces a biologically optimal alignment. The refinement rules consult with *biological knowledge base* whenever necessary.

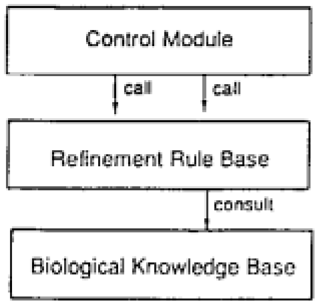

Figure 6: Framework of Intelligent Refiner

4 Protein Folding Simulator

Folding simulation is a research topic that has fascinated hundreds of theoretical bio-chemists for a quarter of a century. The molecular dynamic method is, theoretically, able to solve folding simulation problem. The method precisely simulates the movement of each atom driven by kinetic forces. However, it requires such huge amounts of computational time that actual folding simulation problems cannot be solved (it can simulate pico-second movements of a protein whereas the whole folding process takes a few seconds or more). To make the computational time tractable, we have to seek effective approximation methods.

We introduced a water-counting model to approximate folding simulation concisely and to formulate folding simulation as an optimization problem based on the model [4]. The water-counting model employs a lattice representation for protein and water.

The main chain of an amino acid serves to connect adjacent amino acid. The relative location between two adjacent amino acids is like the move that a knight in chess makes, but on a 3-dimensional lattice (Figure 7), $(\pm 3, \pm 1, \pm 1)$. Every main chain of amino acid occupies 27 ($= 3^3$) lattice cells.

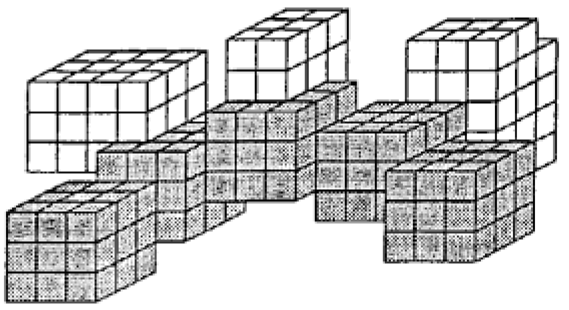

Figure 7: Representation of a part of protein: main chains(shaded) and side chains(unshaded)

The energy of states is a function of hodrophobicity and numbers of adjacent cells occupied by other amino acid. The energy can be reduced both by increasing the amount of water around the hydrophilic amino acid and by reducing the amount of water around the hydrophobic amino acid. The minimization of energy has the effect of inviting hydrophobic amino acids toward the center of the protein where there is less water and to oust hydrophilic amino acids to the surface of the protein where water is abundant.

As a transition from one state to another, we introduce two classes of transition - rotational transition and translational transition (Figure 8).

After a new state is created by the transition selected, a collision check is executed. If, in the next possible state, there is no multiply occupancy of any lattice cell by different parts of the protein, this state is acceptable. Otherwise, the state is discarded and new transitions are tested until some that is accepted is found.

We formulated folding simulation as the problem to search for the minimum energy in a solution space. We employed temperature parallel simulated annealing as an algorithm to find a global optimal solution. Temperature parallel

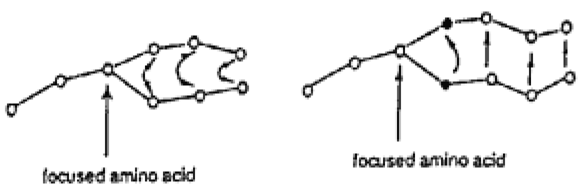

Figure 8: Rotational transition (left) and Translational transition (right)

simulated annealing is an algorithm that can circumvent a scheduling problem of simulated annealing (SA), by introducing the concept of parallelism in temperature.

The outline of the algorithm is as follows. Each processor maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs between processors. After every k annealing steps, each pair of processors with adjacent temperatures performs a *probabilistic exchange of solutions* (Figure 9).

The probability has been defined such that solutions with lower energy tend to be at lower temperatures. Hence, the solution at the lowest temperature is expected to be the best solution so far. The cooling schedule is invisibly embedded in the parallel execution.

5 Biological Database

In this section, we briefly introduce the effectiveness of knowledge representation language Quixote[6] in representing biological knowledge, as well as the efficiency of the parallel nested relational DBMS Kappa-P [7], and the plan for a privately integrated knowledge base[5].

When we think of the analysis of the relationship among protein sequences, structures, and functions, for example, a protein knowledge base to help it should contain at least existing public protein databases (PIR, PDB, ProSite, etc.), private databases of experiments, and their cross-references.

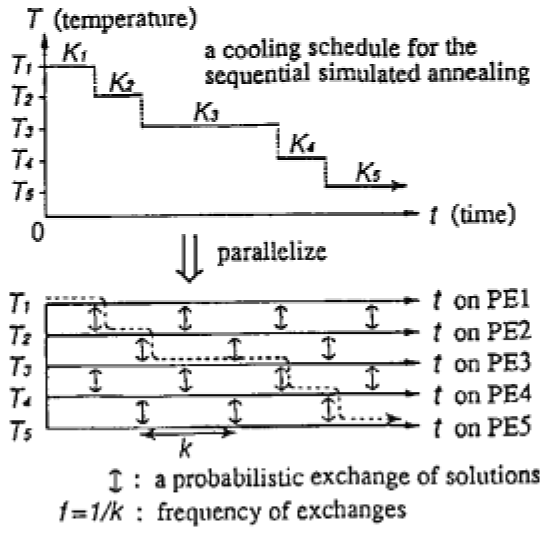
T (temperature)
a cooling schedule for the sequential simulated annealing
T1
T2
T3
T4
T5
K1
K2
K3
K4
K5
0
t (time)
parallelize
t on PE1
t on PE2
t on PE3
t on PE4
t on PE5
k
: a probabilistic exchange of solutions
f=1/k : frequency of exchanges

Figure 9: Temperature Parallel SA

Moreover, we have to develop the way to represent and utilize data of structures and functions, and biological knowledge to define similarity of structures and functions. It is also important to distinguish public knowledge and personal hypotheses in a knowledge base.

In our system, existing databases are stored efficiently in Kappa-P, while other descriptions are represented in Quixote. Here are small examples of 'motif' representation in Quixote.

Examples in Quixote

```
(1) zinc_finger /
    [function="nucleic acid-binding structure"];;
(2) zinc_finger / [pattern = Y];;
(2') zinc_finger [subname = X] / [pattern = Y];;
(3) zinc_finger [function = "exception_1"] / [...];;
```

As shown above, we can write several attributes separately (see (1) and (2)). The object terms (the left-hand side of '/') are regarded as object identifiers and the properties written in the right-hand side of '/' are collected. We can also add properties in object identifiers (see (2) and (2')).

The object (2') inherits property 'function' in (1) according to their implicit order. These facilities are suitable to write knowledge briefly and to catch up up-to-date knowledge. Moreover, exceptions are represented as (3). (2') only shows the scheme of the object. The facts for (2') are as follows.

ProSite R9 zinc finger motif

```
zinc_finger[subname="C2H2"] /
  [pattern="C-x(2,4)-C-x(12)-H-x(3,5)-H"];;
zinc_finger[subname="C3HC4"] /
  [pattern="C-x-H-x-[LIVMFY]-C-x(2)-C-[LIVM]"];;
zinc_finger[subname="GATA"] /
  [pattern="C-x-N-C-x(4)-T-x-L-W-R-R-x(3)-G-x(3)-
            C-N-A-C"];;
zinc_finger[subname="Poly(ADP-ribose) polymerase"] /
  [pattern="C-K-x-C-x-[EQ]-x(3)-K-x(3)-R-x(16,18)-
            W-[YH]-H-x(2)-C"];;
```

Functions are represented as follows. We have to provide a content retrieval facility, though it is not implemented yet.

zinc finger motif (functions)

```
(1) zinc_finger / [function="'Zinc finger' domains
 [1-5] are nucleic acid-binding protein structures
 first identified in the Xenopus transcription factor
 TFIIIA. These domains have since been found in
 numerous nucleic acid-binding proteins...."];;
(2) reference [id="TWXL3", no=1] /
      [keywords+ = {"...", ..},
       authors+ = {"Ginsberg A.M.", ...},
       journal = "Cell (1984) 39:479-489",
       abstract = ".....", ... ];;
```

6 Conclusion

In this paper, I gave an overview of FGCS technologies and their application to molecular biology. By using PIM/m, we were able to obtain high performance, and we were able to use computational models, such as parallel simulated annealing, which can't be employed on sequential computers. By using Kappa and Quixote, we showed that these are useful tools for developing unified biological databases.

ICOT started to develop the KL1 language processor on the UNIX system. Therefore, our software tools will be available on UNIX machines next year.

References

[1] Shunichi Uchida : "Summary of the Parallel Inference Machine and its Basic Software", Proc. Int. Conf. on Fifth Generation Computer Systems 1992.

[2] Masato Ishikawa et al. : "Protein Sequence Analysis by Parallel Inference Machine", Proc. Int. Conf. on Fifth Generation Computer Systems 1992.

[3] Makoto Hirosawa et al. : "Protein Sequence Alignment using Knowledge", ICOT Technical Report (ICOT TR 793), 1992.

[4] Makoto Hirosawa et al. : "Folding Simulation using Temperature Parallel Simulated Annealing", Proc. Int. Conf. on Fifth Generation Computer Systems 1992.

[5] Hidetoshi Tanaka : "Integrated System for Protein Information Processing", Proc. Int. Conf. on Fifth Generation Computer Systems 1992.

[6] Yasukawa, H., Tsuda, H. and Yokota, K.: "Objects, Properties, and Modules in Quixote", Proc. International Conference on Fifth Generation Computer Systems 1992.

[7] Kawamura, M., Sato, H., Naganuma, K. and Yokota, K.: "Parallel Database Management System : Kappa-P", Proc. International Conference on Fifth Generation Computer Systems 1992.