TR-0855

Parallel Iterative Aligner with Genetic Algorithm

by
M. Ishikawa, T. Toya, Y. Totoki,
& A. Konagaya (NEC)

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

Parallel Iterative Aligner with Genetic Algorithm

MASATO ISHIKAWA† ishikawa@icot.or.jp

TOMOYUKI TOYA† toya@icot.or.jp

YASUSHI TOTOKI† totoki@icot.or.jp

AKIHIKO KONAGAYA‡ konagaya@csl.cl.nec.co.jp

†Institute for New Generation Computer Technology (ICOT)
1-4-28 Mita, Minato-ku, Tokyo 108 JAPAN

‡C&C Systems Research Laboratories, NEC Corporation
4-1-1, Miyazaki, Miyamae-ku, Kawasaki, Kanagawa 216 Japan

Abstract

This paper proposes a new methodology to improve the performance of multiple sequence alignment by combining a genetic algorithm and an iterative alignment algorithm. Iterative alignment algorithms usually achieve better alignment than other alignment algorithms, such as tournament based multiple alignment. They, also, can incorporate parallelism to improve execution performance. However, they sometimes suffer from being trapped in the local optima and result in relatively low-quality alignments due to their rapid convergence. A genetic algorithm can save this problem by exchanging partial alignment sequences between "individuals". Our experiments show that the combination of a genetic algorithm and an iterative alignment algorithm produces better results than iterative aligners which employ hill-climbing search strategies.

1 Introduction

Computers partly solve the problem of multiple sequence alignment automatically, instead of relying on the hands and eyes of experts. The results obtained by computer, however, have not been as satisfactory as those by human experts. This is because the problem of multiple sequence alignment is particularly time and space consuming. Dynamic programming (DP-matching) [13, 16], 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 in the N-th power of the sequence length. Though a method was proposed to cut unnecessary computation in the dynamic programming algorithm [3], it still needs too much computation to solve any practical alignment problem. A number of heuristic algorithms for multiple alignment problems have been devised [1, 5] in order to obtain approximate solutions within a practical time. Most of these algorithms are based on two-dimensional DP-matching.

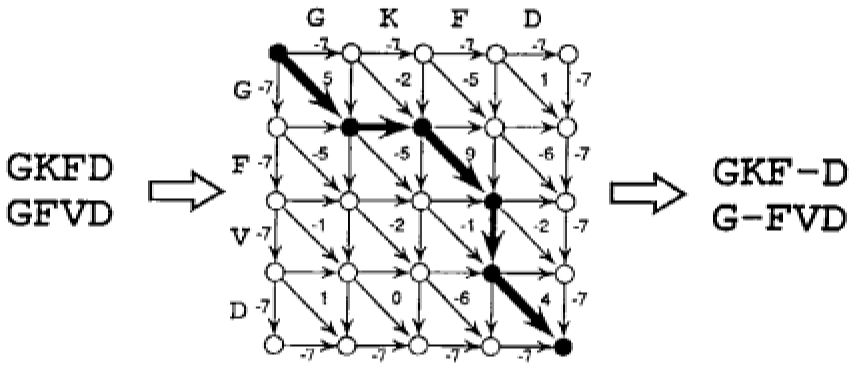

Figure 1: Tow-dimensional DP-matching

Figure 1 shows the algorithm of two-dimensional DP-matching 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 maximizing the total score of arrows. Each score 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 horizontal arrows indicate the insertion of gaps.

We have developed multiple sequence alignment systems using a parallel inference machine [9, 8]. Parallel execution often makes it possible to reduce the execution time required for multiple alignment to a manageable degree. We have focused on iterative alignment algorithms because of their ability to generate a high-quality multiple alignment, and reported on a parallel iterative aligner [10], which employs the parallel extension of an iterative alignment algorithm.

The iterative aligners developed so far have sometimes suffered from relatively low-quality alignments. This is because the search strategy used is the so-called *hill-climbing algorithm* where the search proceeds in a better direction in the search space only. With this algorithm, the search is often trapped in *local optima* in spite of rapid convergence. In this paper, we propose the incorporation of a genetic algorithm with our parallel iterative aligner. A genetic algorithm can save this problem efficiently by exchanging partial alignment sequences between *individuals*.

The organization of the rest of this paper is as follows. In Section 2, we show an overview on iterative aligners, including our parallel iterative aligner. We define a genetic algorithm for solving multiple sequence alignment problems in Section 3. Then, the results of experiments and the evaluation of the genetic algorithm are discussed in Section 4. Finally, conclusions are given in Section 5.

2 Iterative Aligners

In this section, we briefly explain iterative aligners to locate our previous work in their history, and suggest why we use a genetic algorithm.

History

An iterative aligner improves a current alignment iteratively to obtain a final multiple sequence alignment. The prototype iterative aligner originated with Barton and Sternberg [1]. They proposed a constructive alignment method which aligns sequences one by one with DP-matching, and suggested that an iterative method could refine the constructed multiple sequence alignment. The iterative method, being regarded as the prototype iterative aligner, refines an N sequence alignment as follows: it chooses the first sequence to DP-match with the rest of the alignment, then chooses the second sequence to DP-match with the rest of the latest result, ..., then chooses the N-th sequence to DP-match with the rest of the latest result, and repeats this until the score of the alignment converges.

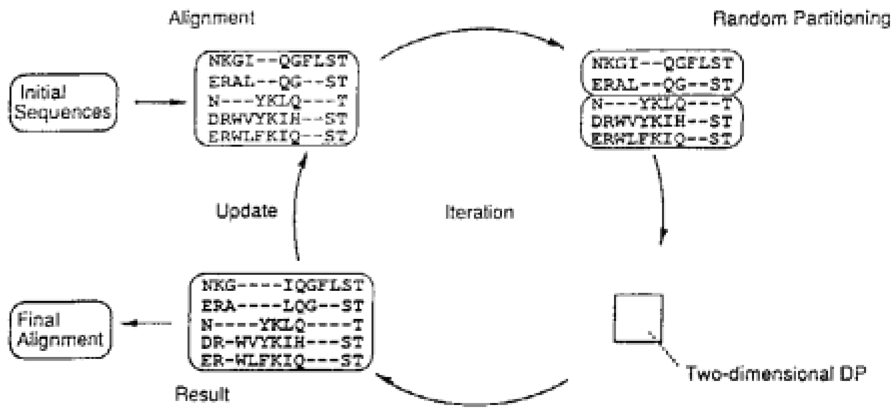

Figure 2: Berger-Munson Method

A real iterative aligner was developed by Berger and Munson [2]. Figure 2 illustrates the iterative strategy in which the initially aligned sequences are randomly divided into two groups. By fixing the alignment within each group we can optimize the alignment between the groups, using two-dimensional DP-matching. The resultant alignment, in turn, is the starting point for the next alignment of a different pair of groups. This procedure is repeated until the score of the alignment converges. Because there are $2^{n-1}-1$ different possible partitionings for n sequences, the method requires a large number of iteration cycles to get a final multiple alignment. It often takes a day to align a practical-scale alignment, although the method ignores columns which contain any number of gaps using Murata's DP-matching [12] in order to reduce computation.

Gotoh [7] discussed the performance of the iterative aligner, emphasizing the importance of the detailed gap cost system. Since the system assigns opening and extending gap costs to individual pairs of compared sequences, it helps to produce a high-quality alignment. However, this requires a large amount of computation for the group-to-group DP-matching.

Tanaka *et al.* indicated the equivalent relation between iterative aligners and Hidden Markov Models [17].

Parallel Iterative Aligner

Though iterative aligners often provide much better multiple alignment than those obtained by conventional algorithms, such as tournament based methods, its randomized iteration needs more than several hours to solve a multiple alignment of a practical scale. When a parallel machine is available, the iterative strategy extended in parallel is fairly helpful for reducing execution time. We developed a parallel iterative aligner as follows [10].

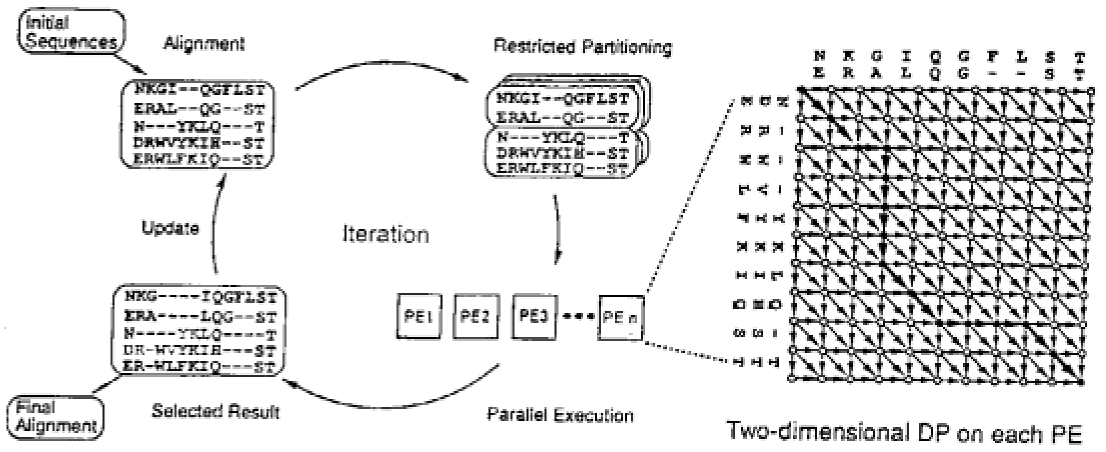

Figure 3: Parallel Extension of Berger-Munson Method

Figure 3 shows the algorithm of our parallel iterative aligner. Every possible partitioning into two groups of aligned sequences can be respectively evaluated with two-dimensional DP matching on a processing element (PE). The detailed gap cost system, described as *Algorithm C* in [7], is used for the calculation of the alignment score. 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 iteration cycle will stop when no improvement occurs. This best-choice search strategy is one of the hill-climbing algorithms.

Furthermore, we developed an effective heuristic search, the *restricted partitioning technique*. Applying the parallel iterative strategy described above, 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 the group-to-group DP-matching. The restricted partitioning technique preferentially selects partitionings which have a small k, such as one or two. This technique resembles the Barton-Sternberg method by DP-matching between plural sequences and a few sequences. It 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 the technique.

Thus, our parallel iterative aligner performs better than the original Berger-Munson method in terms of quality of alignment and the execution time. It does, however, sometimes offer relatively low-quality alignment. This must be because the search is trapped by local optima in the search space. We have incorporated a genetic algorithm in order to get out of such local optima.

3 Genetic Algorithm

In this section, we explain a mechanism of genetic algorithms and our definition for solving multiple sequence alignment problems. Genetic algorithms have been applied to a number of biological problems [11, 14, 15, 18].

Mechanism of Genetic Algorithm

Genetic algorithms are stochastic search algorithms based on the biological evolution process [6]. As in Figure 4, genetic algorithms simulate the survival of the fittest in a population of individuals which represent points in a search space. Each individual, though usually represented by a binary string, corresponds to a possible multiple sequence alignment in our definition. A fitness function corresponds to the alignment score described in detail in Section 4.

The aim of a genetic algorithm is to find the global optimum of the fitness function when given an initial population of individuals by applying genetic operators in each generation. The genetic operators consist of crossover, mutation, and selection.

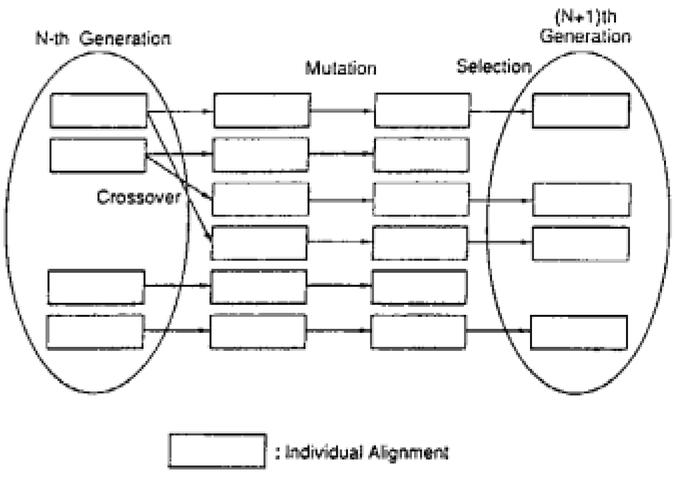

Figure 4: Mechanism of Genetic Algorithm

Crossover

The crossover operator produces two descendants by exchanging parts of two individuals. This operator aims to improve an individual by replacing a part of an individual with a better part from another individual. Figure 5 shows the crossover of two individual alignments. The sequences to be aligned are randomly divided into two groups: the exchange and unexchange groups. The sequence members in the exchange groups are exchanged between the individual alignments by fixing the current alignment within each group. The exchanged and unexchanged groups are re-aligned with two-dimensional DP-matching. Candidate alignments which should be done with crossover operators are chosen randomly from among the higher-scoring individuals.

Mutation

The mutation operator changes certain parts of an individual. Figure 6 shows the mutation of an individual alignment. A sequence is randomly chosen from among the aligned sequences,

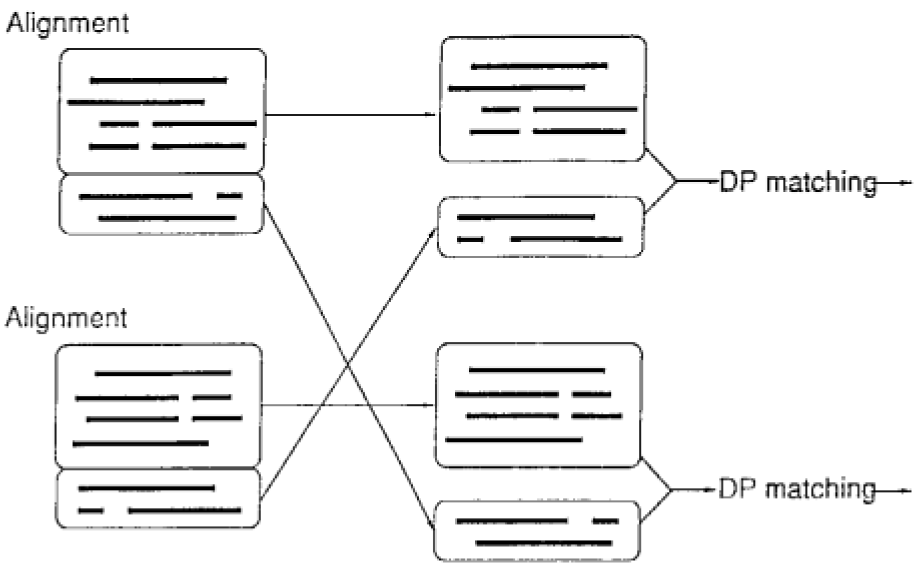

Figure 5: Crossover of Alignments

and is re-aligned against the other sequences with two-dimensional DP-matching. Although mutation means a random perturbation under the orthodox concept of a genetic algorithm, it is considered an iterative cycle of improvement in our definition.

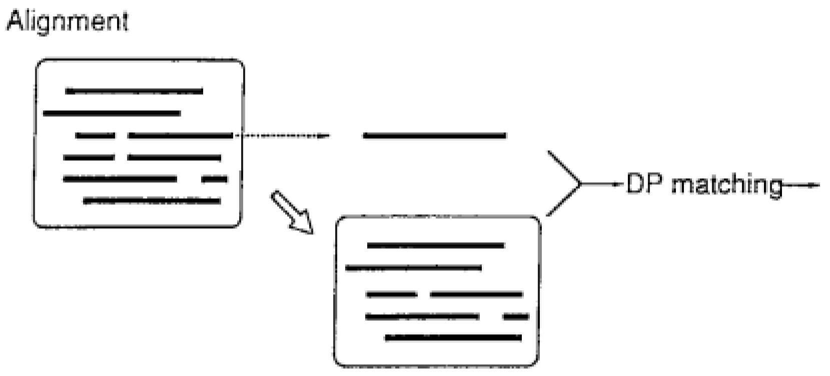

Figure 6: Mutation of Alignment

Selection

The selection operator chooses good individuals from a population according to fitness and the given selection strategy. This operator aims to increase the quality of individuals in the population while maintaining a certain level of diversity. The operator first calculates the relative fitness of all individuals, then discards several of the lesser individuals as determined by a parameter value. The same number of individuals are compensated for by crossover operators in the next generation so that the number of individuals is constant generation after generation.

4 Experimental Results

In this section, we explain implementation of our system and compare the performance of a parallel iterative aligner with the genetic algorithm and a parallel iterative aligner with the hill-climbing algorithm.

Implementation

Iterative aligners gradually improve global multiple alignment. Improvement is evaluated by the alignment score defined 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 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 (*Algorithm C* in [7]).

We use Dayhoff's odds matrix [4], each value of which is a logarithm of the mutation probability of a character pair (zero is the neutral value). The gap penalty imposed on a row of k gaps is a linear relation: $a + bk$ where a and b are parameters. We set $a = -7$ and $b = -1$ as default values. Character pairs *gap vs. gap* and *outside gap vs. any character* are ignored; they are assigned the neutral value zero.

Our multiple sequence alignment systems work on PIM, a MIMD parallel machine equipped with up to 256 processing elements (PEs). We have implemented two algorithms for comparison analysis: the genetic algorithm and the hill-climbing algorithm.

Results

Figure 7 compares the histories of the alignment scores obtained by the algorithms. Every algorithm solves the same practical-scale alignment problem, which consists of twenty-two sequences with eighty amino-acid letters each. Each sequence is the beginning of a different kind of protein kinase. The initial state of the alignment problem has no gaps inside the sequences.

(a) genetic algorithm: The iterative strategy of the genetic algorithm is applied to the alignment problem. Each individual alignment is assigned to a PE, so the size of population is 255. A PE is used as a manager process to select discarded individuals and crossover candidates. In each generation, ten percent of the population is discarded by the selection operator and this same number of individuals are compensated for by the crossover operator. The period of each generation is controlled with the system timer, whose default value is eighty seconds. When each PE finds that the system timer has passed eighty seconds, it does not apply another mutation operator but sends the current alignment to the manager PE. On average, four mutations occur in a generation with the crossover operation, or six mutations occur in a generation without it. In Figure 7, MAX shows the history of the maximum alignment score among the population, and AVE shows the history of the average alignment score of the population. The MAX score was decided as having converged when it was invariable through more than one hundred generations. It took about two hours to obtain the converged MAX score, 15775, and the final AVE was 15121.

(b) hill-climbing algorithm: The best-choice iterative strategy is applied to the alignment problem. In each iteration, possible partitions of aligned sequences are distributed to

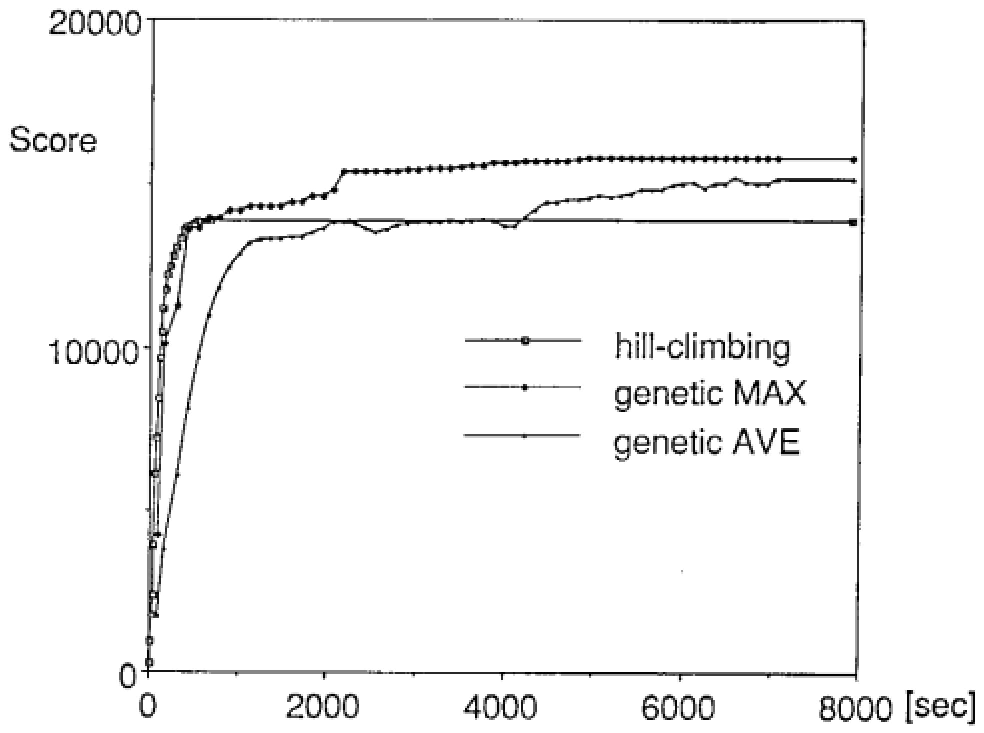

Figure 7: Comparing Alignment Score Histories

the PEs so that they can be evaluated at the same time. We use the restricted partitioning technique; the number of sequences in the smaller divided group is restricted to one or two. 253 PEs ($_{22}C_1 = 22$ plus $_{22}C_2 = 231$) are necessary to execute all restricted partitioning in parallel. Additionally, a PE is used as a manager process to select the best alignment in every iteration. In total, 254 PEs are employed in this execution. Each iteration cycle takes thirty seconds on average. Execution stops if no variation in alignment score is found. The final alignment, which is obtained at the twenty-fifth cycle with score 13903 in eleven minutes.

We made the following observations from the results.

1. The MAX alignment score obtained by the genetic algorithm reached the level of the final alignment score, 13903, obtained by the hill-climbing algorithm in almost the same time. The MAX score was not trapped at a relatively low level but increased up to 15775.

2. The AVE alignment score obtained by the genetic algorithm surpassed the final alignment level, 13903, within seventy minutes. This means that more than half of the population could escape local optima at a relatively low level.

3. The AVE score increased closer to the MAX score, even after the MAX score converged to the final value. This means that the number of individuals who had the final 15775 alignment increased. In fact, more than eighty percent of the population had the same alignment after five hours of execution.

Figure 8 compares the scores of resultant alignments obtained by the algorithms. Both algorithms solved thirty different alignment problems, whose size were the same as the problem used in the previous experiment. Figure 8 shows that the scores obtained by the genetic

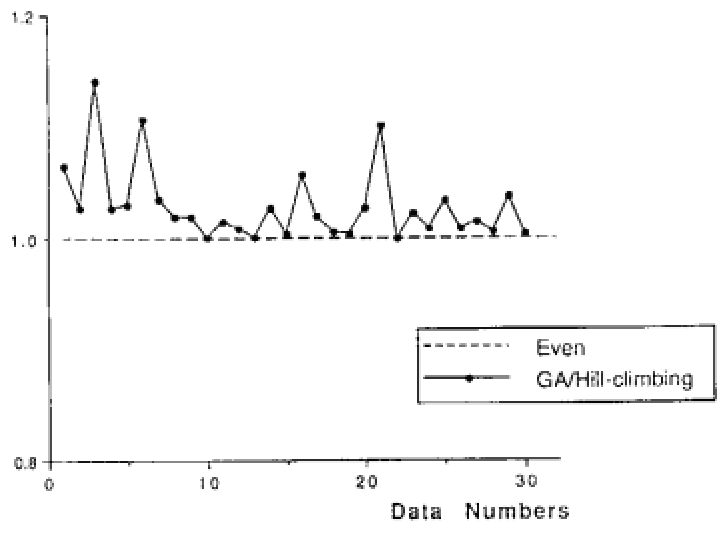

Figure 8: Comparing Alignment Scores

algorithm were better than those obtained by the hill-climbing algorithm in all experiments except three, in which both scores were equal. This result suggests that the genetic algorithm always escape the annoying local optima.

5 Conclusion

The parallel iterative aligner we reported last year sometimes produced low-quality alignments whose score might be trapped in local optima in the search space. We assumed that the hill-climbing algorithm brought the defect in spite of its rapid convergence. So, we incorporated a genetic algorithm, instead of the hill-climbing algorithm, into our parallel iterative aligner. As a result, we found that the parallel iterative aligner with a genetic algorithm could improve an alignment score as rapidly as that with the hill-climbing algorithm, and that the aligner gradually increased the score to a higher level which might be close to the optimal alignment score.

The reason why the genetic algorithm showed such high performance seemed to be the modularity of multiple alignment problems. If we can replace a part of a sequence alignment with a better part from another, we obtain a better multiple sequence alignment. Crossover operators achieve this replacement in a statistical manner. In fact, about five percent of crossover descendants had better alignment scores than both their parents' scores in early execution stage.

Thus, we consider the parallel iterative aligner with the genetic algorithm to be the most suitable system for solving multiple sequence alignment problems of a practical scale.

Acknowledgments

The authors wish to thank Masaki Hoshida, Makoto Hirosawa, Katsumi Nitta, and the people in the GIP project at ICOT for the fruitful discussions we had with them and their great efforts in realizing our systems.

References

[1] Barton, G.J. and Sternberg, M.J.: "A Strategy for the Rapid Multiple Alignment of Protein Sequences: Confidence Levels from Tertiary Structure Comparisons", *J. Mol. Biol.*, 198, pp.327-337 (1987).

[2] Berger, M.P. and Munson, P.J.: "A Novel Randomized Iterative Strategy for Aligning Multiple Protein Sequences", *Comput. Applic. Biosci.*, Vol.7, No.4, pp.479-484 (1991).

[3] Carrillo, H. and Lipman, D.: "The Multiple Sequence Alignment Problem in Biology", *J. Appl. Math.*, 48, pp.1073-1082 (1988).

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

[5] Feng, D. F. and Doolittle, R. F.: "Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees", *J. Mol. Evol.*, 23, pp.267-278 (1986).

[6] Goldberg, D. E.: "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Publishing Company, Inc., (1989).

[7] Gotoh, O.: "Optimal Alignment between Groups of Sequences and its Application to Multiple Sequence Alignment", *Comput. Applic. Biosci.*, Vol.9, No.3, pp.361-370 (1993).

[8] Hirosawa, M., Hoshida, M., Ishikawa, M. and Toya, T: "MASCOT: Multiple Alignment System for Protein Sequences Based on Three-way Dynamic Programming", *Comput. Applic. Biosci.*, Vol.9, No.2, pp.161-167 (1993).

[9] Ishikawa, M., Toya, T., Hoshida, M., Nitta, K., Ogiwara, A. and Kanehisa, M.: "Multiple Sequence Alignment by Parallel Simulated Annealing", *Comput. Applic. Biosci.*, Vol.9, No.3, pp.267-273 (1993).

[10] Ishikawa, M., Hoshida, M., Hirosawa, M., Toya, T., Onizuka, K. and Nitta, K.: "Protein Sequence Analysis by Parallel Inference Machine", *Proc. Int'l Conf. Fifth Generation Comput. Syst. '92*, pp.294-299 (1992).

[11] Konagaya, A. and Kondou, H.: "Stochastic Motif Extraction using a Genetic Algorithm with the MDL Principle", *Proc. 26th Hawaii Int'l Conf. Syst. Sci.*, Vol.1, pp.746-755 (1993).

[12] Murata, M., Richardson, J.S. and Sussman, J.L.: "Simultaneous Comparison of Three Protein Sequences", *Proc. Natl. Acad. Sci. USA*, 82, pp.3073-3077 (1985).

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

[14] Parsons, R., Forrest, S. and Burks, C.: "Genetic Algorithms for Sequence Assembly", *Proc. 1st Int'l Conf. Intelli. Syst. Mol. Biol.*, (1993).

[15] Platt, D. and Dix, T.I.: "Construction of Restriction Maps Using a Genetic Algorithm", *Proc. 26th Hawaii Int'l Conf. Syst. Sci.*, Vol.1, pp.756-762 (1993).

[16] Smith, T.F. and Waterman, M.F.: "Identification of common molecular subsequences" *J. Mol. Biol.*, 147, pp.195-197 (1981).

[17] Tanaka, H., Asai, K. and Ishikawa, M.: "Hidden Markov Models and Iterative Aligners: Study of their Equivalence and Possibilities" *Proc. 1st Int'l Conf. Intelli. Syst. Mol. Biol.*, (1993).

[18] Unger, R. and Moult, J.: "On the Applicability of Genetic Algorithms to Protein Folding", *Proc. 26th Hawaii Int'l Conf. Syst. Sci.*, Vol.1, pp.715-725 (1993).