k-Group Multiple Alignment Based on A* Search

H. Imai [1] T. Ikeda [2]
imai@is.s.u-tokyo.ac.jp ikeda@hum.cl.nec.co.jp

[1] Department of Information Science, University of Tokyo
Hongo, Bunkyo-ku, Tokyo 113, Japan

[2] Information Technology Research Laboratories, NEC Corporation

Abstract

This paper proposes a k-group alignment algorithm for multiple alignment as a practical method. In iterative improvement methods for multiple alignment, the so-called group-to-group two-dimensional dynamic programming has been used, and in this respect our proposal is to extend the ordinary two-group dynamic programming to a k-group alignment programming. This extension is conceptually straightforward, and here our contribution is to demonstrate that the k-group alignment can be implemented so as to run in a reasonable time and space under standard computing environments. This is established by generalizing the A* search approach for multiple alignment devised by Ikeda and Imai [8]. The k-group alignment method can be directly incorporated in existing methods such as iterative improvement algorithms (Berger and Munson [2], Gotoh [4]) and tree-based (iterative) algorithms (Hirosawa et al. [6]). This paper performs computational experiments of applying the k-group method to iterative improvement algorithms, and shows that our approach can find better alignments in reasonable time.

1 Introduction

The multiple sequence alignment is the problem to find the alignment of multiple sequences with highest score due to a given scoring criteria between characters. The solution of this problem for multiple sequences of DNA and proteins represents the similarity among them and is applied to various important fields such as the prediction of three dimensional structure of proteins and the inference of phylogenetic tree in molecular biology.

The method based on Dynamic Programming (DP) is a well-known approach for multiple sequence alignment problem. This method searches all vertices in the grid-like acyclic graph,

[1]今井浩：東京大学理学系研究科情報科学専攻，〒 113 文京区本郷 7-3-1
[2]池田崇博： NEC 情報メディア研究所．本研究は著者が東京大学情報科学専攻在学中に行なったものである．

and has $O(n^d)$ time and space complexity for d sequences of length at most n. This approach is effective for small dimension of two or three. In fact, with the increase of computing power, three-dimensional DP became feasible when the length n is not so large. But, it is impractical to apply this method directly to a little larger dimensional problem because n^d becomes enormous then. For large d, approximate algorithms dividing d sequences into two groups and applying two-dimensional DP between the two groups have been used (Berger, Munson [2], Gotoh [4]). The resultant alignment is improved gradually by iteration of dividing and aligning. Another method for multiple alignment is a tree-based (iterative) algorithm [6], and in it two-dimensional DP is also used. In connection with these algorithms, Hirosawa et al. [5] employed three-dimensional DP as the basis for an initial alignment to the subsequent iterative algorithm.

The A* algorithm reduces search space without lack of optimality of the result alignment. There were proposed some methods of reducing the search space in multiple alignment (Carrillo and Lipman [3], Spouge [9], etc.), but the A* algorithm with upper bounding operation would be the best method to derive an optimal alignment [7, 8] (see also [9]). Araki et al. [1] proposed to use an A* algorithm for two-dimensional DP in the Berger-Munson iterative algorithm. They use an estimate derived from a score table which can reduce the search space in the two-dimensional case. The estimate for A* multiple alignment in Ikeda and Imai [8] was demonstrated to be very powerful, and in this respect generalizing the A* algorithm in [8] to k-group alignment is rather natural.

This paper investigates a k-group alignment algorithm for multiple alignment. In the k-group alignment problem, d sequences are given with k disjoint groups of them, each being internally aligned, and a best alignment among these k groups should be found with only inserting a gap simultaneously in the same position for the alignment of each group. First, it is noted that the same approach in [8] can be applied to the group alignment problem. Several ways of applying A* search to this problem are discussed. Then, its connection with the standard iterative improvement algorithm is described. Through computational experiments, it is demonstrated that the k-group alignment can be performed for $k = 3, 4, 5$ in a practical time depending on the problem size, and this produces better alignments. For example, for 9 sequences of length about 750, whose similarity is relatively low, 3-group alignment can be performed very fast even starting with a bad initial alignment, and 4-group alignment can be executed for mildly good alignments. For 21 sequences of length about 430, whose similarity is quite high, 5-group alignment can be performed iteratively. About the alignment quality, 3-group alignment yields better solutions compared with 2-group DP almost with a little additional time. 4- and 5-group alignment methods can find better solutions in most cases but require more time. The practicality of k-group alignment is thus shown, and further investigation of elaborating this with other methods and enhancing itself should be done.

2 A* Algorithm for Multiple Alignment

The multiple alignment problem can be solved by finding the shortest path on some directed acyclic graph. Suppose that S_k denotes the k-th sequence of length $n_k = O(n)$ and d denotes the dimension, the number of sequences. Then in the directed acyclic graph $G = (V, E)$ such that $V = \{(x_1, \ldots, x_d) \mid x_i = 0, 1, \ldots, n_i\}$ and $E = \cup_{e \in \{0,1\}^d} \{(v, v+e) \mid v, v+e \in V,\ e \neq 0\}$, a path from the vertex $s = (0, \ldots, 0)$ to the vertex $t = (n_1, \ldots, n_d)$ corresponds to an alignment of sequences. In case of $d > 2$, the sum of all scores for pairwise sequence alignments is used as the score for the multiple sequence alignment in general. This corresponds to defining each edge length in G as the sum of all corresponding edge length in the graphs for pairwise alignments. Let $G_{ij} = (V_{ij}, E_{ij})$ denote the graph for the alignment of S_i and S_j $(i < j)$, that is, $V_{ij} = \{v_{ij} = (x_i, x_j) \mid v = (x_1, \ldots, x_d) \in V\}$ and $E_{ij} = \{(u_{ij}, v_{ij}) \mid (u, v) \in E, u_{ij} \neq v_{ij}\}$. Then the length of edge (u, v) in E is defined as $l(u, v) = \sum_{1 \leq i < j \leq d} l(u_{ij}, v_{ij})$ where $l(u_{ij}, v_{ij})$ denotes the length of edge (u_{ij}, v_{ij}) in graph G_{ij} and is defined from the score table between characters. Thus, the multiple alignment problem can be formulated as a shortest path problem on G. The shortest path on such a graph can be computed by dynamic programming in a direct way, but its complexity $\Theta(n^d)$ is intractably large for large d.

The A* algorithm can find a shortest path without searching the whole graph if good estimates on the shortest path length from each vertex to t are at hand. Ikeda and Imai [8] show a method of obtaining such good estimators by computing pairwise two-dimensional $\binom{d}{2}$ subproblems for d sequences. For the case $d > 2$ (the 2-group alignment in the sequel can be handled similarly), it uses the following estimator h: $h(v) = \sum_{1 \leq i < j \leq d} L^*(v_{ij}, t_{ij})$, where $L^*(u, v)$ denotes the shortest path length from u to v. This estimator uses the shortest path length in the pairwise alignment problem as the estimate for the length of the path corresponding to the shortest path in the multiple alignment problem by making use of relations of G and G_{ij}. In higher dimensional problem, necessary time and space for solving pairwise problems is negligible. It is clear that each estimate $h(v)$ does not exceed the actual shortest path length from v to t, and moreover h is dual feasible, i.e., for any edge (u, v) in E,

$$l(u, v) + h(v) = \sum_{1 \leq i < j \leq d} (l(u_{ij}, v_{ij}) + L^*(v_{ij}, t_{ij})) \geq \sum_{1 \leq i < j \leq d} L^*(u_{ij}, t_{ij}) = h(u).$$

Hence the A* algorithm using this estimator is reduced to the following simple one.

1. For arbitrary pair of i and j satisfying $1 \leq i < j \leq d$, apply DP to graph G_{ij} from vertex t_{ij} and calculate $L^*(v_{ij}, t_{ij})$ for any v_{ij} in V_{ij}.

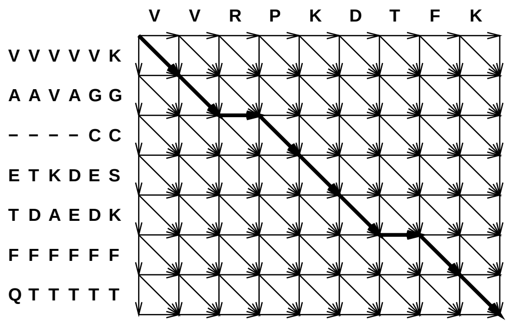
2. Apply the Dijkstra method to graph G from vertex s with the length of edge (u, v) modified as $l(u, v) + h(v) - h(s)$ where $h(v) = \sum_{1 \leq i < j \leq d} L^*(v_{ij}, t_{ij})$.

3 Group Alignment and Its Use in Iterative Algorithms

3.1 k-Group Alignment

Since the multiple alignment problem becomes hard to solve when d is large, as a subproblem, the group alignment is considered. Originally, in the group alignment, d sequences are divided into two groups, say d' sequences and $d - d'$ sequences $(0 < d' < d)$, and then fixing the alignment in each group, it solves a two-dimensional alignment problem between two groups. In this two-dimensional problem, since the alignment in each group is fixed, when a gap is inserted into a group, it is simultaneously inserted in the same position. See Figure 1.

For general $k > 2$ $(k \leq d)$, the k-group alignment problem can be defined similarly. In this problem, d sequences are given as k disjoint groups, and each group is associated with some alignment of sequences in the group. In a typical case, for an alignment of d sequences, k aligned



(a)

```
Thc VVRPKDTFK
Tha VV--KA-FT
Hal VA--ET-FQ
Met VA--TD-FT
Sul VA--DE-FT
Ent VG-CED-FT
Pla KG-CSK-FT
```

(b)

Figure 1: (a) Aligning seven sequences by a 2-group alignment for {Thc} and others, (b) an obtained alignment (- is a newly inserted gap)

groups can be obtained simply by dividing this alignment into k groups (and removing trivial gaps inside each group alignment). Then, the k-group alignment problem finds a best-score alignment of d sequences under a condition that, in each group, each column of its alignment should be fixed. Hence, when a gap is inserted into a sequence in some group, the same gap should be inserted in the same position in every sequence in the group. In this case, a k-dimensional grid-like graph is used to solve the k-group alignment problem, as in the original k-dimensional alignment problem. See an example of $k = 2$ and $d = 7$ in Figure 1. Since the score function is defined to be the sum of scores of all pairs, the A* approach can be directly extended to this k-group alignment problem by virtue of the principle of optimality.

There may be considered two types of A* search algorithm for k-group alignment.

- (group-based precomputing strategy) For each pair of groups, compute the score of 2-group alignment between the two groups, and make the summation of those scores as a lower-bound estimator in solving the k-group alignment problem.
- (sequence-based preprocessing strategy) In the initial stage of the algorithm, solve 2-dimensional alignment between every pair of sequences as preprocessing. Then, in solving each k-group alignment problem, compute the summation of scores of all pairs of sequences contained in different groups.

The former A* estimate is stronger than the latter. On the other hand, to obtain the former estimate we may have to solve $\binom{k}{2}$ 2-group alignment problems, while in the latter only preprocessing in the beginning is sufficient. Even when solving 2-group alignment problems between general two groups, we can make use of the scores of all-pairs of sequences computed in the preprocessing stage to solve it by the A* search by the sequence-based strategy.

3.2 Iterative improvement

The standard randomized iterative improvement method proposed by Berger and Munson [2] works as follows:

1. Construct an initial alignment by some method;
2. Divide d sequences into two groups randomly;

3. Remove trivial gaps in each group;
4. Solve the 2-group alignment to obtain a new alignment;
5. If the score decreases, update the current alignment to the new one;
6. If a stopping condition is met, stop; otherwise return to step 2;

In the step 2 above, all the sequences are divided into two groups randomly. There is a method of dividing them into a group of one sequence and a group consisting of the other $d-1$ sequences. This partition is called a restricted partition in [6]. Also in that case, instead of using randomization, one sequence for a group of a single element may be changed in a round-robin fashion. There are many other methods (see [6]).

It is rather natural to extend the iterative algorithm in a way that in step 2 it divides the sequences into k groups for $k \geq 2$. We call such iterative algorithm the k-group iterative algorithm. The A* algorithm for group alignments can be utilized in the k-group algorithm. There can be considered two methods of dividing d sequences into k groups, by directly generalizing the above-mentioned existing methods for $k = 2$.

- (k-random-grouping; simply called *random*) This method divides d sequences randomly into k groups so that no group becomes empty. This will be denoted by RA(k) in the computational results below.

- (k-restricted-grouping; simply called *restricted*) This method divides them into $k-1$ groups consisting of a single sequence and another group consisting of the other sequences where each sequence in the former $k-1$ groups is chosen randomly. This will be denoted by RI(k) in the next section.

In [5], the latter method with $k = 2$ was used to derive a best-first iterative improvement algorithm, and it was observed that the restricted-grouping strategy produces favorably nice solutions compared with the random-grouping method.

When the A* is used in the iterative improvement algorithm with the restricted-grouping strategy, solving $\binom{k-1}{2}$ subproblems out of $\binom{k}{2}$ can be dispensed with if we do the same preprocessing as in sequence-based preprocessing strategy in the preceding subsection.

4 Computational Results

In order to investigate the actual efficiency of this approach, experiments aligning actual sequences of proteins have been performed. Our implementation is designed to evaluate several strategies in a system, and is coded in such a general setting. This causes in some places redundant computation which can be avoided by cleverly using the information obtained in the preprocessing stage. This point should be remarked especially in observing timing results in the computational results. For example, the following points can be improved further from the current code.

- When the A* is used in the iterative improvement algorithm with the restricted-grouping, as described in the preceding section, solving $\binom{k-1}{2}$ subproblems out of $\binom{k}{2}$ can be dispensed with. However, in the current implementation, these subproblems are solved from scratch every time.

- Even when solving 2-group alignment problems between general two groups, we can make use of the scores of all-pairs of sequences computed in the preprocessing stage to solve it by A* search. However, we do not incorporate this in the code. Instead, a kind of lazy 2-dimensional DP algorithm is implemented. Here, "lazy" execution may be done in many ways. For example, in the beginning only a fraction of DP table is computed, say in a constant-bandwidth region, and values of other elements are computed when necessary.

- In the experiment, the linear gap system is used instead of the affine gap system (see below). To obtain more meaningful alignments, the affine gap system is considered to be better, and performing the experiment with the affine gap system is important. For d-dimensional DP alignments, it is known that with the affine gap system it takes more time compared with the case of the linear gap system as d increases. From this, one might think that k-group alignment method by A* with the affine gap system performs much worse than that with the linear gap system as k becomes larger, but we suspect that for $k = 3$ (and maybe 4,5) the slow-down would be a small constant factor since by A* the search space is drastically cut off. Of course, this point should be actually tested, which is left as future work. Concerning the performance of our A* approach, we suspect that changing the alignment cost system from the simple pairwise sum of 2-alignments to the weighted sum may affect more.

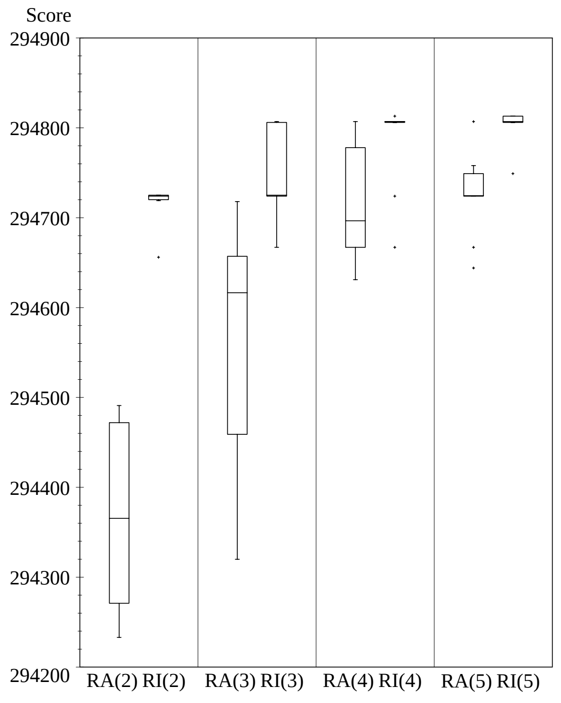
Concerning the score matrix, the PAM-250 matrix has been used in assigning edge length with each sign of score reversed. The linear gap system ax for the gap of length x is used (extending the current system to the affine gap system $ax + b$ for the gap of length x would be practically important work). With regard to the gap penalty, the minimum value in the PAM-250 matrix, $a = -8$, has been adopted. All the experiments were done on SPARCStation 20 with 128 megabytes memory.

4.1 Case with Higher Similarity

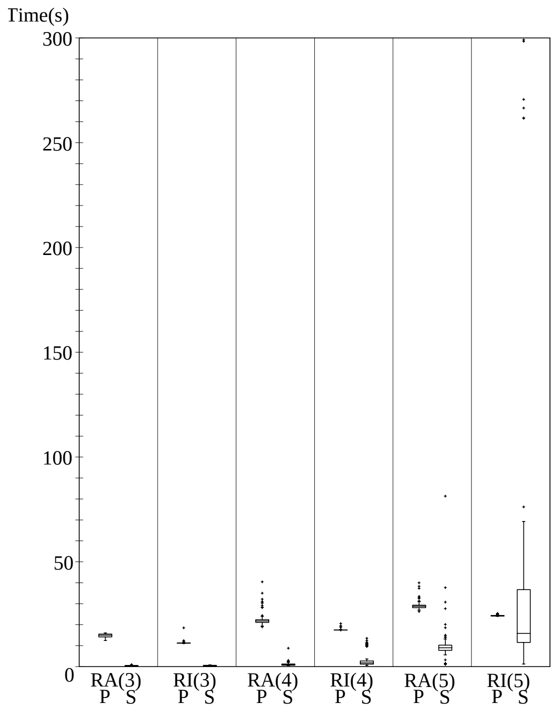
In this experiment, elongation factor TU (EF-TU) and elongation factor 1α (EF-1α) are used as in [8]. The number d of sequences is 21, and the length n of each sequence is about 450. The cost of the best alignment found by the experiment is 294813 with length 482. The average score per amino pair is $\frac{294813}{482 \cdot \binom{21}{2}} \approx 2.91$, and is higher than in the experiment in the next subsection.

As an initial alignment, we adopted a solution of the A algorithm in [7, 8] with parameter 81/80. The score of this initial alignment is 294201. By using a tree-based DP, better initial solutions can be obtained, but those solutions are processed by group DP, while the solution by the A algorithm is not. Starting with the solution by the A algorithm, the alignment score is improved fast initially.

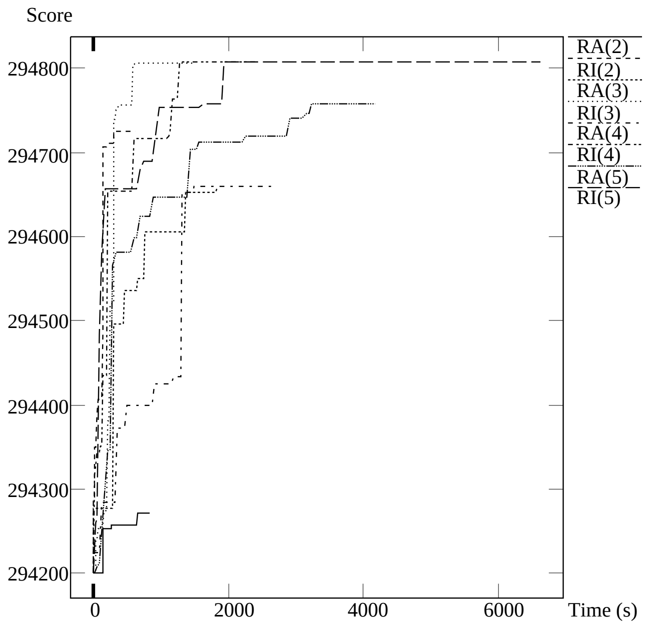
Starting from this initial solution, we tested 10 series of 100 iterations with using different random numbers. Both of the k-random-grouping RA(k) and k-restricted-grouping RI(k) are examined. Some of computational results are given in Figure 2. Box plots are used, where a box plot comprises these elements: 1) a box with 1a) a central line showing the median, 1b) a lower line showing the first quartile, 1c) an upper line showing the third quartile; 2) 2 lines extending from the central box of maximal length 3/2 the interquartile range but not extending past the range of the data; 3) outliers, points that lie outside the extent of the previous elements.



(a)



(b)



(c)

Figure 2: Computational results for EF-1α: (a) A box plot of final scores after 100 iterations (10 series), (b) A box plot of running times for the estimator precomputation ("P" in short; in the left side) and A* search ("S" and in the right side) per step in the computation of typical series in (c), (c) Score improvement processes of typical series such that its final score attains the median score among the first three trials for each k and strategy.

For smaller k such as 2 and 3, the k-random-grouping method RA(k) tends to produce worse solutions than the k-restricted-grouping method RI(k). For large k such as 4 and 5, this tendency becomes less clear. In these experiments, the 3-restricted grouping method may be said to be the best one in regard to both computation time and solution quality. About the solution quality, the 3-restricted grouping produces better solutions than 2-restricted grouping, as seen in Figure 2(a) and even if we use the scores of the 2-restricted grouping with 200 iterations, results are almost same. Here, it should be stressed that this speed of 3-group alignment is achieved by the use of A*. It is also observed that, even for $k = 4, 5$, k-alignment problem can be solved in a reasonable time even for this rather large-scale problem. Concerning the effect of A*, for $k = 3$ (and 4), its estimates are nice enough to reduce the search space drastically. In fact, the A* search takes much less time than the precomputation of estimator in this case (Figure 2(b)). For $k = 5$, the estimates become less effective, and sometimes the A* search space becomes large, although it is still manageable.

4.2 Case with Lower Similarity

In the next experiment, $d = 9$ sequences of Chitin Synthase of lengths 710, 717, 723, 728, 730, 732, 756, 760, 762 are investigated. The best alignment score found by repeated applications of RI(4) is 59606 with length 811. The average score per amino pair is $\frac{59606}{811 \cdot \binom{9}{2}} \approx 2.04$, and is lower than the number 2.91 for EF-1α above. In this respect, the similarity is less compared with EF-1α, but each sequence is about twice longer.

We adopted a left-aligned alignment as an initial one. Since the initial alignment is simply left-aligned, the initial score is bad and -25992. Even for this bad initial alignment, 3-group alignments with A* can be performed efficiently, and yet 4-group alignment for such a bad alignment is hard to execute with A*. The score improvement processes for 2- and 3-group alignments are shown in Figure 3(c).

To see the effectiveness of 4-group alignments, we adopted another initial alignment for $k = 4$ obtained by a series of 3-restricted-grouping iterative improvement alignment after 50 iterations starting with the above-mentioned bad alignment. This alignment has score 59409. Note that there are many ways of obtaining rather good initial solutions, and in this regard this kind of initial alignment for $k = 4$ is easily obtained. After 50 iterations from this initial solution for $k = 4$, the 3-restricted-grouping method finds an alingment of score 59499 (the best score among thee trials of 100 iterations using different random numbers for $k = 2, 3$). Starting with this initial solution for $k = 4$, 4-group alignments RI(4) are used for the iterative algorithm with 20 iterations. Some of computational results are shown in Figure 3(a) and (b) where box plots are again used (see the explanation in the previous subsection).

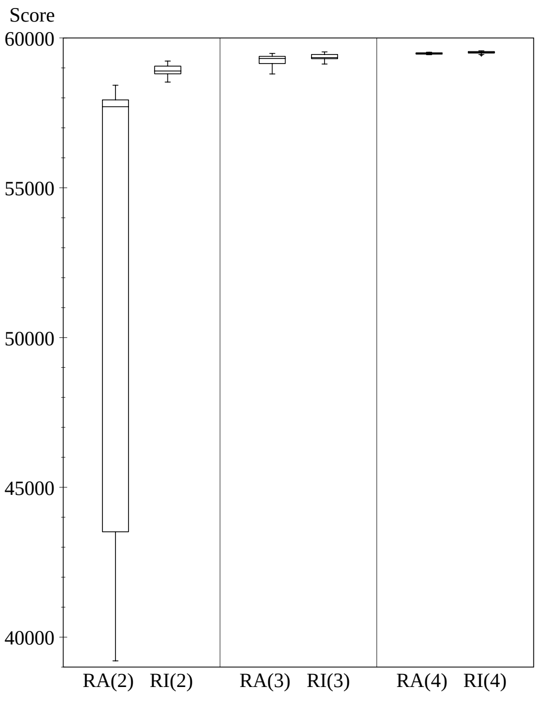
It would be rather striking that even for this set of long sequences with lower similarity, the A* method solves the 3-alignment problems very well, as in the above experiments. Again, the actual A* search time is less than the precomputation time for its estimator. Also, 4-group alignment could be run fast enough to improve nice solutions more. Since the similarity is relatively low, the A* search often requires more time than the estimator precomputation time for $k = 4$ here, unlike it start to occur for $k = 5$ in the preceding case of high similarity.

4.3 Observations on Computational Results

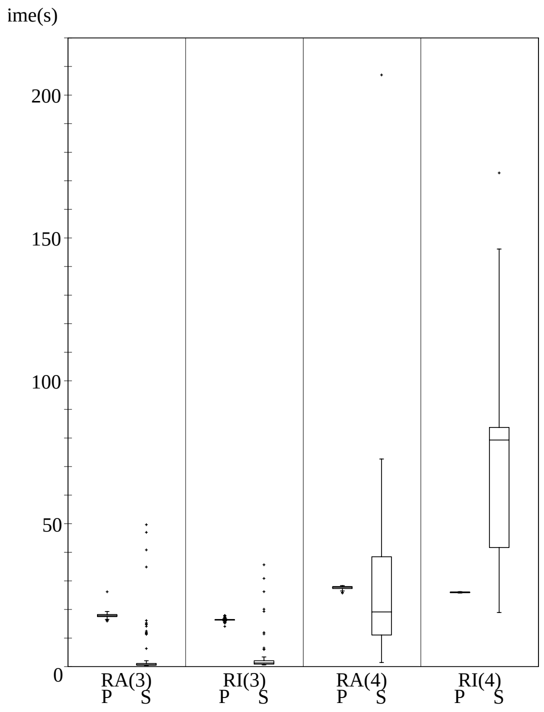
Although the above-mentioned results are still preliminary ones, the following may be observed.

- We have tested two methods of dividing d sequences into k groups. It is observed that the k-restricted-grouping method RI(k) tends to produce better-score alignments than the k-random-grouping method RA(k) on the average.

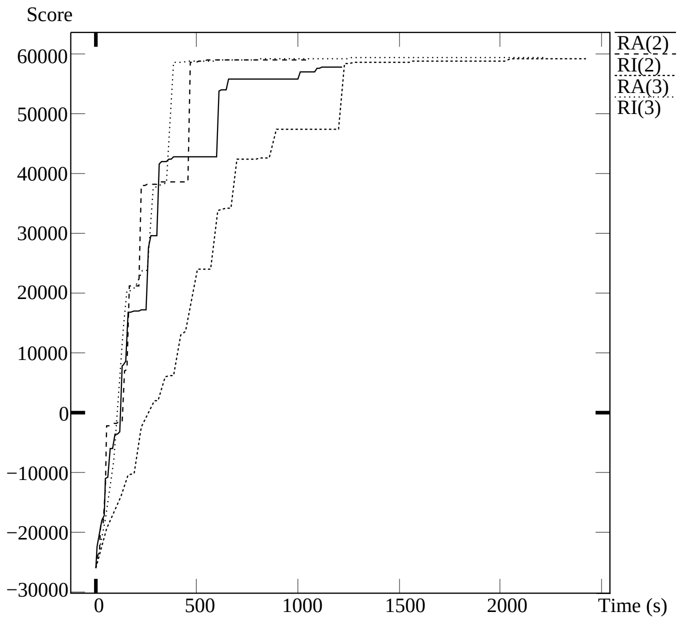
- As k becomes larger, the k-grouping takes more time, but produces on the average better alignment results. Although for $k = 4, 5$ it takes definitely larger time compared with 2-



(a)



(b)



(c)

Figure 3: Computational results for 9 sequences of Chitin Synthase: (a) A box plot of final scores (100 iterations from the left-aligned initial solution for $k = 2, 3$; 20 iterations from the intermediate one for $k = 4$; 10 series for each), (b) A box plot of running times for estimate precomputation ("P"; left) and A* search ("S"; right) per step in the computation of typical series in (a,c), (c) Score improvement processes of typical series from the left-aligned initial alignment such that its final score attains the median score among the first three trials for each k and strategy,

and 3-group alignments, by these results it is verified that 4- (and 5- with high similarity) group alignments can be practically used to polish up an obtained alignment further.

- The A* search greatly reduces the running time required by DP. In fact, in these experiments, 3-group alignments can be solved in time within a small constant factor of the running time of standard two-dimensional DP. Furthermore, their solution quality is definitely better than that of 2-group methods.

- Related to a widely known fact concerning k-opt local search algorithms for combinatorial problems, one should realize that there are so many local optima in such combinatorial problems, and there do exist many local optima for 2-group alignments which are not local optima for 3- and higher order group alignments. By increasing k, one might think it takes more time to check the local optimality of current solution which makes this approach less practical, but this is not a problem at all since 3- and higher order group alignments can produce better solution than 2-grouping method within reasonable additional time and checking the local optimality is less important in this respect.

Thus, this paper proposed the use of k-group alignment for $k \geq 3$, and showed its power through computational experiments. There are still many points which can be improved further in the current code, and developing an refined system would be very interesting as future work.

Acknowledgement

This work was supported in part by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Informatics" from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] S. Araki, M. Goshima, S. Mori, H. Nakashima, S. Tomita, Y. Akiyama, and M. Kanehisa, "Application of Parallelized DP and A* Algorithm to Multiple Sequence Alignment," *Proc. Genome Informatics Workshop IV,* 1993, pp.94–102.

[2] M. P. Berger and P. J. Munson, "A Novel Randomized Iterative Strategy for Aligning Multiple Protein Sequences," *Comput. Applic. Biosci.*, Vol.7 (1991), pp.479–484.

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

[4] O. Gotoh, "Optimal Alignment between Groups of Sequences and its Application to Multiple Sequence Alignment," *Comput. Applic. Biosci.*, Vol.9 (1993), pp.361–370.

[5] M. Hirosawa, M. Hoshida, M. Ishikawa and T. Toya, "MASCOT: Multiple Alignment System for Protein Sequence Based on Three-way Dynamic Programming," *Comput. Applic. Biosci.*, Vol.9 (1993), pp.161–167.

[6] M. Hirosawa, Y. Totoki, M. Hoshida and M. Ishikawa, "Comprehensive Study on Iterative Algorithms of Multiple Sequence Alignment," *Comput. Applic. Biosci.*, Vol.11, No.1 (1995), pp.13–18.

[7] T. Ikeda: Applications of the A* Algorithm to Better Routes Finding and Multiple Sequence Alignment. Master's Thesis, Department of Information Science, University of Tokyo, 1995.

[8] T. Ikeda and H. Imai, "Fast A* Algorithms for Multiple Sequence Alignment," *Proceedings of the Genome Informatics Workshop 1994*, 1994, pp.90–99.

[9] J. L Spouge, "Speeding Up Dynamic Programming Algorithms for Finding Optimal Lattice Paths," *SIAM J. Appl. Math.*, Vol.49 (1989), pp.1552–1566.