TR-730

Multiple sequence alignment by parallel simulated annealing

by
M. Ishikawa, T. Toya, M. Hoshida, K. Nitta,
A. Ogiwara & M. Kanchisa

January, 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

Multiple sequence alignment by parallel simulated annealing

Masato Ishikawa, Tomoyuki Toya, Masaki Hoshida, Katsumi Nitta,
(Institute for New Generation Computer Technology (ICOT),
1-4-28 Mita, Minato-ku, Tokyo 108, Japan)

Atsushi Ogiwara and Minoru Kanehisa
(Institute for Chemical Research, Kyoto University,
Gokashou, Ujishi, Kyoto 611, Japan)

Key Words: Multiple Alignment, Simulated Annealing, Parallelism and Protein Sequence.

Abstract

We have applied simulated annealing algorithms to solve problems of multiple alignment. A temperature parallel algorithm, a simple parallel algorithm and a sequential algorithm were tested on a problem. The results were compared with the result obtained by the conventional tree-based method. As a result, every annealing algorithm has produced a better energy than the tree-based method. Particularly, either of the parallel annealing algorithm has reached the best energy, probably the optimal solution. We consider the temperature parallel algorithm of simulated annealing to be the most suitable for finding the optimal multiple alignment, because the algorithm does not require any re-scheduling for further optimization. The algorithm is also suitable for refinement of a multiple alignment obtained by other heuristic methods.

1 Introduction

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. Computers partly solve the problem of multiple alignment automatically, instead of relying on the hands and eyes of experts. However the results obtained by computers have not been as satisfying as those by human experts. That is because multiple alignment is one of the most time and space consuming problems. Theoretically, dynamic programming algorithms provide an optimal solution to multiple alignment (Needleman and Wunsch, 1970; Smith and Waterman, 1981). But, this 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 the unnecessary computation of the dynamic programming algorithm (Carrillo and Lipman, 1988), it still needs too much computation to solve a practical alignment problem. In order to obtain approximate solutions for multiple alignment problems within a practical time, a number of heuristic algorithms have been devised (Barton, 1990; Johnson and Doolittle). Most of these are based on pairwise dynamic programming combinations. These are likely to promote so-called local optimals.

In this paper, we propose a parallel simulated annealing algorithm that can be applied to be solving multiple alignment problems. Simulated annealing is a stochastic algorithm used to solve complex combinatorial optimization problems (Kirkpatrick *et al.*, 1983). When the algorithm is applied to multiple alignment, it can evaluate all sequences simultaneously. The annealing process often gives an optimal or a semi-optimal solution in a reasonable time. We have adopted two parallel algorithms for simulated annealing to solve the problem efficiently. One is a simple parallel algorithm while the other is a temperature parallel or *time-homogeneous* algorithm (Kimura and Taki, 1990).

The organization of the rest of this paper is as follows. We present the simulated annealing algorithm in Section 2 and its parallelization mechanism in Section 3. After that, we formulate the multiple alignment as a combinatorial optimization problem in Section 4. The results of experiments and comparison with other methods are discussed in Section 5. Finally, conclusions are given in Section 6.

2 Simulated annealing algorithm

An outline of the simulated annealing (SA) algorithm is as follows.

Let X be a solution space and $E : X \to \mathbb{R}$ be the objective function which we want to minimize. Given an arbitrary initial solution $x_0 \in X$, the algorithm generates a sequence of solutions $\{x_n\}_{n=0,1,2,\ldots}$ iteratively as follows, and finally outputs x_n for a large enough n.

(i) Modify the current solution x_n randomly and get a candidate for the next solution x'_n.

(ii) Calculate the change in the objective function: $\Delta E = E(x'_n) - E(x_n)$.

(iii) When $\Delta E \leq 0$, accept the candidate: $x_{n+1} = x'_n$. When $\Delta E > 0$, accept the candidate with probability $p = \exp(-\Delta E/T_n)$, but reject it if $x_{n+1} = x_n$ where $T_n > 0$ is a control parameter decreasing with n.

When $T_n \equiv +\infty$, SA is reduced to a blind random search; and when $T_n \equiv +0$, it is reduced to a greedy algorithm (iterative improvement) converging to one of the local optimals at hand. For general $0 < T < +\infty$, it behaves in a manner between these two extreme cases. In the following, we refer to an operation (i) as a *move* and a sequence of operations (i)~(iii) as a *step*.

SA is based on an analogy between combinatorial optimization and statistical mechanics. The objective function E is referred to as *energy* and T_n is referred to as *temperature*. We call $\{T_n\}_{n=0,1,\ldots}$ a *cooling schedule*. When the temperature is constant ($T_n \equiv T$), SA simulates the equilibrium states of an imaginary physical system at temperature T, which corresponds to a combinatorial optimization problem. Hence, the solution x_n is distributed according to the Boltzmann distribution at temperature T. This Boltzmann distribution converges to the lowest energy state (optimal solution) as the temperature decreases to zero. Thus, one might expect SA to be capable of providing the optimal solution, in principle.

It is well-known that the cooling schedule has a great influence on SA performance; a poor schedule may lead to a poor solution. Here arises the *cooling schedule problem*: how slowly should we decrease the temperature so as to get the best possible solution within a given number of annealing steps (within a given amount of computation time)? However, characterizing the ideal cooling schedule is also a difficult stochastic control problem.

3 Parallel algorithms of SA

SA has received much attention since it can provide much better solutions than conventional heuristics, although it often requires lengthy execution time. In order to accelerate its execution, the parallelization of SA has been studied extensively. We applied the following two parallel algorithms to the problem of multiple alignment.

3.1 Simple parallel SA

The simple parallel algorithm is a naive combination of sequential SAs: every available processing element (PE) has one solution and anneals it sequentially using a *distinct* sequence of random numbers. All solutions are compared with each other and the best one, the one with the minimum energy value, is selected as a solution for the parallel algorithm. In each PE, such a simple cooling schedule as:

$$T_n = \alpha^{\lfloor n/K \rfloor} \cdot T_0, \qquad n = 1, 2, \ldots, N$$
$$0 < \alpha < 1, \qquad K \gg 1$$

is often used (Kirkpatrick *et al.*, 1983). When the initial temperature T_0 and the final temperature T_N are chosen properly and both K and N/K (the number of so-called inner loops and outer loops) are large enough, such a cooling schedule has been known to work well in many applications.

3.2 Temperature parallel SA

The basic idea behind the algorithm is to use parallelism in temperature, to perform annealing processes concurrently at various temperatures (Kimura and Taki, 1990). The algorithm automatically constructs an appropriate cooling schedule from a given set of temperatures. Hence it partly solves the cooling schedule problem.

The outline of the algorithm is as follows. Each PE maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs between PEs. After every k annealing

steps, each pair of PEs with adjacent temperatures performs a *probabilistic exchange of solutions*. Let $p(T, E, T', E')$ denote the probability of the exchange between two solutions: one with energy E at temperature T and the other with energy E' at temperature T'. It is defined as follows:

$$p(T,E,T',E') = \begin{cases} 1 & \text{if } \Delta T \cdot \Delta E < 0 \\ \exp(-\frac{\Delta T \cdot \Delta E}{TT'}) & \text{otherwise} \end{cases}$$

$$\text{where} \qquad \Delta T = T - T', \qquad \Delta E = E - E'.$$

In order to avoid the possible collisions between the pairwise exchanges, half of them perform $k/2$ steps after the other half (Fig.1). The algorithm can be stopped at any time after a large number of steps and we will find a well-optimized solution on the PE that has the lowest temperature.

Since exchanging the solutions between processors with different temperatures is merely changing the temperature for each participant solution, each solution will select its appropriate cooling schedule dynamically through successive competitions with others for lower temperatures. The probability of exchange has been defined such that it is advantageous to solutions having low energies. Hence, the solution having the lowest temperature, the winner of the competition, is expected to be the best solution so far. The cooling schedule adopted by this winner, which is dynamically and stochastically determined, is supposed to be an efficient cooling schedule and is invisibly embedded in the parallel execution (Fig.1).

The temperature parallel algorithm is advantageous when we want to continue the execution until a satisfactory solution is found. We can stop the execution at any time and examine whether a satisfactory solution has already been obtained. If a solution has yet to be reached, we can resume the execution *without any re-scheduling* to obtain a better solution. In contrast, in the simple parallel algorithm, when an obtained solution is not satisfactory, we have to reconsider the cooling schedule and repeat the time-consuming annealing process.

It was reported that the temperature parallel algorithm gave better results on a graph-partitioning problem than the simple parallel algorithm (Kimura and Taki, 1990).

4 Application to multiple alignment

We must formulate multiple alignment as a combinatorial optimization problem before applying the SA algorithm to it. Namely, a move to modify the current solution and an energy function to evaluate a solution should be defined.

4.1 Move definition

A multiple alignment solution has a number of gaps (represented by "-"). The number of gaps varies so much between problems that managing gaps is important for move definition. We add sufficient gaps to both the head and tail of each sequence. These are aligned to prevent the number of gaps changing (Kanehisa, 1989). We refer to the special gaps as *outgaps*. For example, we start annealing from the initial solution as Fig.2 (a).

To modify the solution, we focus on one sequence in the alignment and select a gap and a column (horizontal location) randomly in that sequence. For the sake of efficiency, we don't select any outgaps except those adjacent to letters and any columns where outgaps are located. When the sequence RSV is focused and the gap at column 63 and column 37 are selected, the gap moves to column 37 to give solution Fig.2 (b).

We have employed not only a move depending on a single gap but also a move depending on a cluster of gaps, which we refer to as a *block operation* (Ogiwara, 1990). Since a good multiple alignment has some clusters of gaps in general, the block operation accelerates convergence of the annealing process. The operation has three modes: the vertical cluster mode, the horizontal cluster mode and the block cluster mode.

Let's apply the operations successively to solution Fig.3 (a). In the vertical mode, gaps at the same column move together to another selected column. When the number of gaps is equal to the number of sequences, they are forced to move to the head or tail of the sequences and become outgaps. If a gap at column 21 and column 15 are selected, all five gaps at column 21 move to column 15. That modifies solution 100 making it solution Fig.3 (b).

In the horizontal mode, a row of gaps on a sequence move to another selected column on the sequence. If a gap at column 38 and column 21 are selected, a row of three gaps at column 38 moves to column 21. That modifies solution 101 making it solution Fig.3 (c).

In the block mode, rows of gaps at the same column move together to another place selected by column number. If a gap at column 57 and column 63 are selected, six rows of two gaps at column 57 move to column 63. That modifies the solution 102 making it solution Fig.3 (d).

4.2 Definition of energy function

The SA algorithm makes it possible to evaluate all sequences simultaneously, thus solving problems of multiple alignment. An energy function gives energy as the evaluation value. We have defined the energy function as follows. The function sums up the similarity scores of every pair of aligned sequences. Each similarity score is derived by summing up the similarity values of every character pair in the column. Each similarity value is given by *odds matrix*. A *gap penalty* corresponding to each row of gaps in the two sequences is added to the similarity score.

We use $PAM250$ (Dayhoff *et al.*, 1978) as the odds matrix. Each value of the matrix is a logarithm of mutation probability of a character pair so that zero is the neutral value. We have reversed the sign of each value of the matrix according to the habit of SA; the lower the value the better it is. So the most similar character pair, `W` *vs.* `W`, is assigned to the lowest value, -17, and the least similar one, `W` *vs.* `C`, is assigned to the highest value, 8.

The gap penalty imposed on a row of n gaps is a linear relation; $a+bn$ where a and b are parameters. We set $a = 4$ and $b = 1$ as default values. The linear relation is suitable and popular for alignment done by dynamic programming algorithm (Gotoh, 1982). Character pairs *gap vs. gap* and *outgap vs. any character* are ignored; they are assigned the neutral value zero.

5 Experimental results

We have implemented the two parallel algorithms for multiple alignment on the Multi-PSI/V2 (Nakajima *et al.*, 1989), an MIMD parallel machine having 64 processing elements (PEs). The initial solution, Fig.2 (a) was annealed by the two parallel algorithms using 63 PEs; one PE was used for monitoring. We compared the histories of the energy given by the parallel algorithms with those given by other methods (Fig.4).

(a) Temperature parallel SA: Each of the 63 PEs perform 20,000 annealing steps at a distinct constant temperature. The highest and lowest temperatures are determined empirically, and the other temperatures are determined so that adjacent ones have the same ratio. We set 1/400 as the frequency of exchange f. Each point represents the average of two runs with different sequences of random numbers.

(b) Simple parallel SA: Each of the 63 PEs executes the sequential annealing of 10,000 steps using a distinct sequence of random numbers. The cooling schedule consists of exactly the same sequence of 63 temperatures as those in (a). A history of the energy of the best solution among those held by PEs is indicated.

(c) Sequential SA: On a single PE, 20,000 annealing steps are performed with the same type of cooling schedule as that held by each PE in (b). Each point represents an average over 60 runs with different sequences of random numbers.

(d) Tree-based algorithm: The tree-based algorithm is a conventional method to rapidly produce a practical multiple alignment (Barton, 1990). The algorithm aligns sequences one after another by pairwise dynamic programming. The order of aligning sequences depends on a tree-like representation that was previously determined by distance analysis of every pair in the sequences. Our implementation of the algorithm solves the problem within a minute. The energy of the solution, -1694 is indicated by a horizontal line.

We made the following observations from these results.

1. (a) and (b) show that either of the parallel algorithms gives the best solution within 30 minutes (Fig.5). The energy of this result might be a global minimum, because another 30-minute annealing on (a) didn't improve it further.

2. Although the conventional tree-based algorithm (d) is fast, it sometimes provides a result that has an energy separate from the global minimum, as in this case.

3. The sequential algorithm (c) is unable to find the optimal solution even within an hour. But the final energy obtained by (c) is better than (d).

6 Conclusions

We have applied simulated annealing algorithms to the problems of multiple alignment. Parallel annealing algorithms could find an optimal solution for a problem in a reasonable time. The result had a better energy value than that obtained by the conventional tree-based method. Sequential annealing algorithm could also find a better result than the conventional method, but its energy value didn't reach the best one in a so short time as the parallel algorithms.

Although the two parallel algorithms, simple and temperature parallel, gave the best solution to the problem, the temperature parallel algorithm is advantageous for multiple alignment, since it gives better solutions earlier and it does not require any re-scheduling for further optimization.

The alignment problem we deal with in this paper has only 385 residues, whereas a typical practical problem has about 4000 residues, which is about ten times more. And the number of gaps in a solution of the practical problem is supposed to be also ten times more. We can roughly estimate that the practical problem would need a hundred (ten times ten) times more annealing steps to solve than the problem shown in this paper.

When we don't have enough time to execute whole annealing steps for a practical alignment problem, it is useful to regard the simulated annealing algorithm as a refinement tool for a multiple alignment. After we make an initial alignment by a heuristic method like the tree-based algorithm, we can often anneal up the initial alignment in a reasonable time. Actually, we could have the best solution, Fig.5 within fifteen minutes annealing from the initial alignment obtained by the conventional tree-based method. In this case, the temperature parallel algorithm is also advantageous, because it works without designing a special cooling schedule for annealing from a half-aligned solution.

Acknowledgments

We would like to thank Kouichi Kimura, the founder of temperature parallel SA, for valuable discussions.

References

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

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

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

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

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

[6] Kanehisa,M. (1989) *Bunshiseibutsugakkai-nenkai,* (in Japanese).

[7] Kimura,K. and Taki,K. (1990) Time-homogeneous parallel annealing algorithm. *Proc. Comp. Appl. Math. (IMACS'91),* **13**, 827–828.

[8] Kirkpatrick,S. Gelatt,C.D. and Vecci,M.P. (1983) Optimization by simulated annealing. *Science, vol.220, no.4598.*

[9] Nakajima,K., Inamura,Y., Ichiyoshi,N., Rokusawa,K. and Chikayama,T. (1989) Distributed implementation of KL1 on the Multi-PSI/V2. *Proc. 6th Int. Conf. on Logic Programming.*

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

[11] Ogiwara,A. (1990) *Nihonseibutsubutsurigakkai-nenkai,* (in Japanese).

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

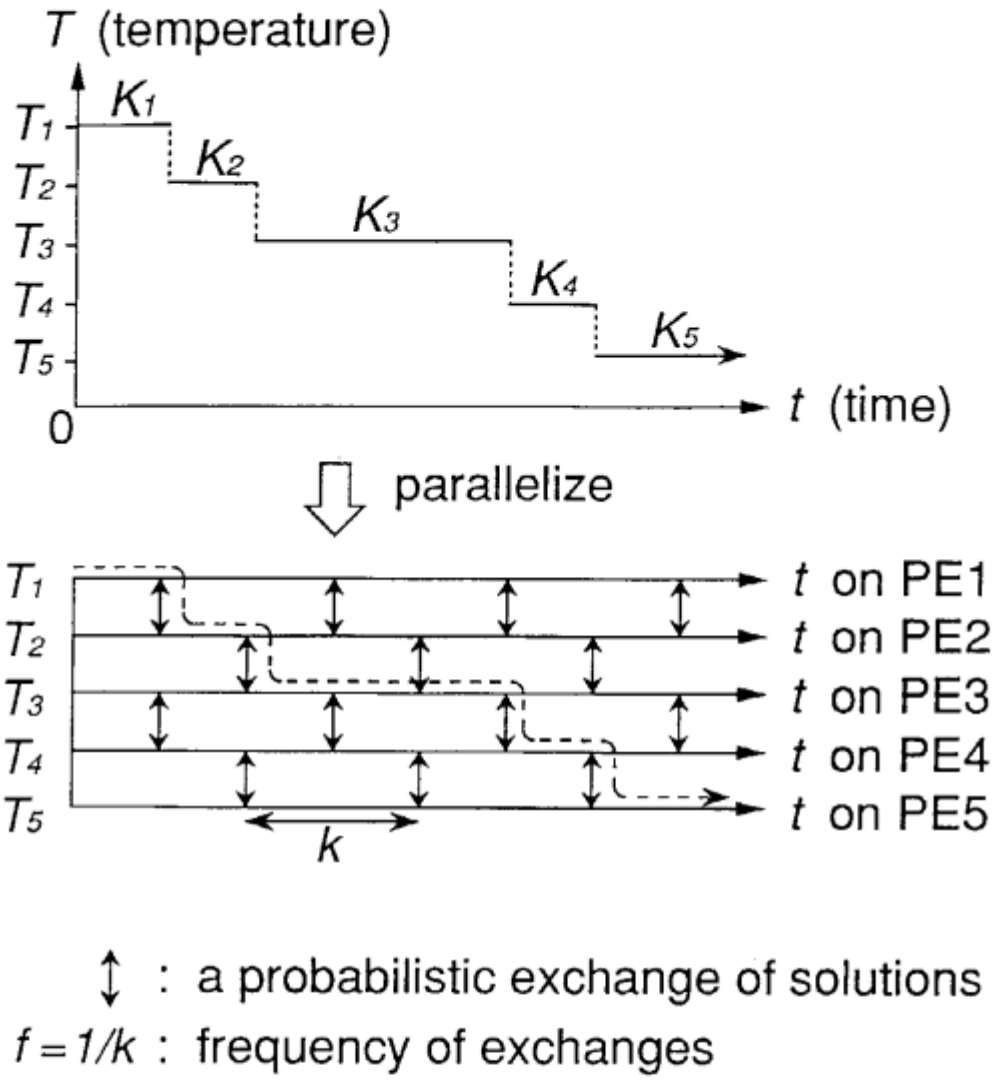

Fig. 1. Parallelization of sequential simulated annealing in temperature (Kimura and Taki, 1990). A cooling schedule for a sequential simulated annealing (upper part) is embedded in the parallel execution of the processing elements, PEs (lower part). An appropriate cooling schedule is dynamically constructed from a given set of temperatures assigned to PEs, but it never manifests itself explicitly.

```
(a)                1         2         3         4         5         6
          12345678901234567890123456789012345678901234567890123456789012345678
   SMRV   :-------GFILATPQTGEASKNVISHVIHCLATIGKPHTIKTDNGPGYTGKNFQDFCQKLQI-------
   MMTV   :-------YSHFTFATARTGEATKDVLQHLAQSFAYMGIPQKIKTDNAPAYVSRSIQEFLARW-------
   IAP    :-------GVMFATTLTGEKASYVIQHCLEAWSAWGKPRIKTDNGPAYTSQKFRQFCRQMDVT-------
   RSV    :-------IVVTQHGRVTSVAVQHHWATAIAVLGRPKAIKTDNGSCFTSKSTREWLARWGIAH-------
   HTLV-1:-------SGAISATQKRKETSSEAISSLLQAIAHLGKPSYINTDNGRAYISQDFLNMCTSLA-------
   HTLV-2:-------DTFSGAVSVSCKKKETSCETISAVLQAISLLGKPLHINTDNGPAFLSQEFQEFCT-------
   BLV    :-------HASAKRGLTTQTTIEGLLEAIVHLGRPKKLNTDQGANYTSKTFVRFCQQFGVSLS-------
   (energy = 877)

(b)                1         2         3         4         5         6
          12345678901234567890123456789012345678901234567890123456789012345678
   SMRV   :-------GFILATPQTGEASKNVISHVIHCLATIGKPHTIKTDNGPGYTGKNFQDFCQKLQI-------
   MMTV   :-------YSHFTFATARTGEATKDVLQHLAQSFAYMGIPQKIKTDNAPAYVSRSIQEFLARW-------
   IAP    :-------GVMFATTLTGEKASYVIQHCLEAWSAWGKPRIKTDNGPAYTSQKFRQFCRQMDVT-------
   RSV    :-------IVVTQHGRVTSVAVQHHWATAIAVLGRPK-AIKTDNGSCFTSKSTREWLARWGIAH------
   HTLV-1:-------SGAISATQKRKETSSEAISSLLQAIAHLGKPSYINTDNGRAYISQDFLNMCTSLA-------
   HTLV-2:-------DTFSGAVSVSCKKKETSCETISAVLQAISLLGKPLHINTDNGPAFLSQEFQEFCT-------
   BLV    :-------HASAKRGLTTQTTIEGLLEAIVHLGRPKKLNTDQGANYTSKTFVRFCQQFGVSLS-------
   (energy = 866)
```

Fig. 2. Initial and firstly modified solution of simulated annealing. In the initial solution (a), a number of gaps are added to both head and tail of each sequence. The first move of a gap, for instance, modifies (a) to (b). When a move reduces the energy of the solution, it will be accepted.

Fig. 3. Modification by block operations. A move of a vertical cluster of gaps modifies (a) to (b), a move of a horizontal cluster modifies (b) to (c), and a move of a block of gaps does (c) to (d).

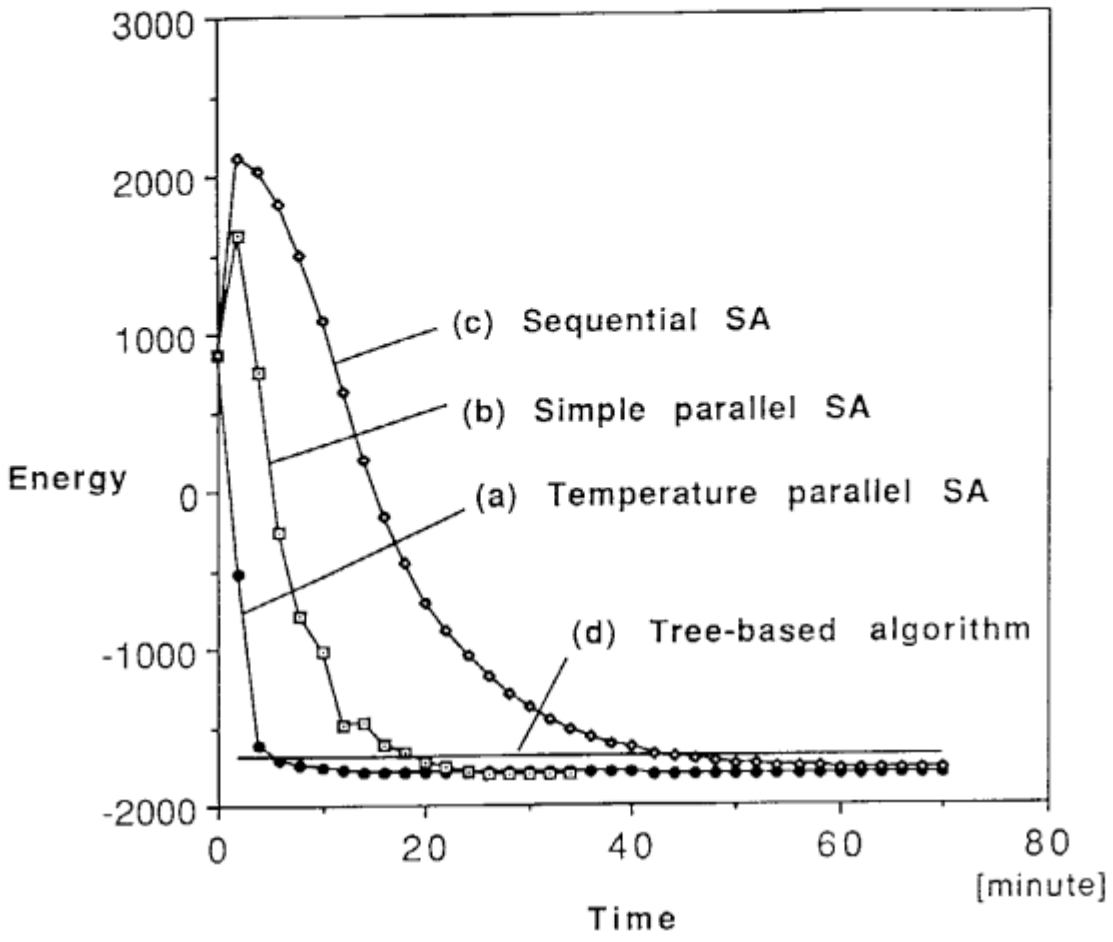

Fig. 4. Comparing energy histories of the simulated annealing algorithms. For the temperature parallel SA algorithm (a), each of 63 PEs performed 20,000 annealing steps at a distinct constant temperature and the energy history at the lowest temperature is displayed. For the simple parallel SA algorithm (b), each of 63 PEs executed the sequential annealing of 10,000 steps and the best energy history obtained from them is displayed. The sequential SA algorithm (c) shows a history of an average energy along 20,000 annealing steps using a single PE. The horizontal line (d) shows the energy level of the solution obtained by a conventional tree-based algorithm.

```
SMRV   :-------GFILATPQTGEASKNVISHV-IHCLATIGKPHTIKTDNGPGYTGKNFQDFCQKLQI--------
MMTV   :----YSHFTFATARTGEATKDVLQHL-AQSFAYMGIPQKIKTDNAPAYVSRSIQEFLARW----------
IAP    :------GVMFATTLTGEKASYVIQHC-LEAWSAWGKPR-IKTDNGPAYTSQKFRQFCRQMDVT-------
RSV    :---------IVVTQHGRVTSVAVQHHWATAIAVLGRPKAIKTDNGSCFTSKSTREWLARWGIAH------
HTLV-1:-----SGAISATQKRKETSSEAISSL-LQAIAHLGKPSYINTDNGPAYISQDFLNMCTSLA---------
HTLV-2:--DTFSGAVSVSCKKKETSCETISAV-LQAISLLGKPLHINTDNGPAFLSQEFQEFCT-----------
BLV    :----------HASAKRGLTTQTTIEGL-LEAIVHLGRPKKLNTDQGANYTSKTFVRFCQQFGVSLS-----
(energy = -1800)
```

Fig. 5. The best energy solution. This solution was generated by the parallel simulated annealing algorithms within fifteen minutes. It might be the optimal alignment.