The Better Alignment among Output Alignments

Kuo-Tsung Tseng, Chang-Biau Yang*, Kuo-Si Huang

Department of Computer Science and Engineering,
National Sun Yat-sen University
Kaohsiung 80424, Taiwan
cbyang@cse.nsysu.edu.tw

Received 24 August 2007; Accepted 22 September 2007

Abstract. In nowaday molecular biology, the biosequence alignment is one of the most fundamental techniques. It can be mapped into the longest common subsequence problem, which can be solved in $O(n_1n_2)$ time with the dynamic programming technique, where n_1 and n_2 are the lengths of the two biosequences. In fact, the reasonability of an alignment of two biosequences depends on the scoring function used by the algorithm. Scientists have presented many scoring functions to measure the goodness of the alignments in different criteria, such as the affine gap penalty, and score matrices like PAMs, Blosums, Gonnets. All of these scoring functions are based on the same core, the dynamic programming. Once the optimal alignment score is found, tracing back the alignment lattice, which is produced during the dynamic programming, will obtain the alignment of the optimal score. Unfortunately, the optimal alignment may not be unique in most cases and the most biologically meaningful alignment may not be an optimal alignment. In this paper, we present some mathematical scoring criteria that should help in finding the better, according to biological considerations, alignment among output (optimal) alignments of the original LCS algorithm and illustrate our algorithms to solve them. Our algorithms give not only the alignment of the optimal score but also more biologically meaningful without increasing the computing complexity of the original algorithm.

Keywords: Bioinformatics, Computational Biology, Longest Common Subsequence, Biosequence Alignment,

1 Introduction

One of the most important problems that biologists desire to solve is the biosequence alignment problem, on which there is plenty of research [1-5]. The problem is given two biosequences and its goal is to obtain the optimal score based on some predefined scoring criteria. A biosequence is a string consisting of characters chosen from a set of alphabets Σ, where Σ contains the 4 nucleotides {A, T, C, G} in DNA sequences, {A, U, C, G} in RNA sequences, and the 20 amino acids {A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V} in protein sequences. Gaps in an alignment are represented by the dash symbols (-). The biosequence alignment problem was mapped into the longest common subsequence (LCS) problem, which is a well-studied problem in algorithms [6-10] by computer scientists.

The biosequence alignment problem can be used to judge similarities and differences between two biosequences that are usually taken as the selecting criteria of the prediction templates when 3D protein structures (tertiary structures) are to be predicted. It is important to select good templates to predict 3D protein structures in homology modeling [11-13]. The better templates produce the more accurate prediction results.

The biosequence alignment problem can be done efficiently in $O(n_1n_2)$ time with the dynamic programming technique, where n_1 and n_2 denote the lengths of the two given biosequences. An argument arises due to the different definitions of the optimum between biologists and computer scientists. The algorithms designed by computer scientists usually result in a mathematically optimal score by then biologists wish to get a biologically optimal score. Though many scoring criteria have been proposed [14-16], those are still actually mathematically optimal, while they may not be biologically optimal. To be mathematical or to be biological, that is the question.

Naor and Brutlag demonstrated that the biologically meaningful alignment is not always the mathematically optimal one, so that they presented the near-optimal alignments to provide more possible alignments to be selected by biologists [17]. In fact, more alignments increase the possibility of finding the suitable alignment, but too many alignments may confuse the biologists. Thus, we do need some other biologically filtering criteria to help us to choose the suitable alignment.

Both biological and mathematical optimization may not be satisfied simultaneously, but it is possible to get an alignment of biologically optimal score from mathematically optimal results. Since the alignment of the optimal

* Correspondence author

score may not be unique in most cases, we should choose the most biologically meaningful one from those mathematically optimal alignments or from the output alignments if near-optimal alignments are applied.

In this paper, we present some mathematical scoring criteria to define the most biologically meaningful alignment when the alignment of the optimal score is not unique. Naor and Brutlag [17] showed that the biologically meaningful alignment may not be mathematically optimal. We leave it out of consideration in this paper since we can also apply our concepts in near-optimal alignments.

The definitions we propose do not guarantee the unique result, but it can be conquered by more criteria. Our algorithms with the new problem definitions can choose the better alignment from a set of (optimal) output alignments without increasing time complexity of the original algorithm. They are themselves interesting problems, even if we do not consider the practical use of the new problem definitions.

The rest of this paper is organized as follows. In Section 2, we introduce some scoring criteria that are biologically meaningful in optimal alignments. Next, we propose some algorithms for solving the problems in Sections 3 and 4. Finally, some discussions and conclusions are given in Section 5.

2 Definitions

2.1 The Smoothest Optimal Alignment

Here we use the simplest cost function for DNA sequences as our example to explain the idea of the *smoothest optimal alignment* of biosequences S = `caagt` and T = `caaa`. Suppose cost function is defined as follows.

$$\delta(x,y)=\begin{cases}1 & x=y\\ -2 & \text{if } x\neq y\\ -1 & x=gap \text{ or } y=gap\end{cases}$$

The dynamic programming function to align two biosequences with the above cost function is given as follows.

$$M(i,j)=\begin{cases}-\max(i,j) & i=0 \text{ or } j=0\\ \max\begin{cases}M(i-1,j-1)-2\\ M(i-1,j)-1\\ M(i,j-1)-1\end{cases} & \text{if } S_i\neq T_j \text{ and } i,j\geq 1\\ M(i-1,j-1)+1 & S_i=T_j \text{ and } i,j\geq 1\end{cases}$$

where $0 \leq i \leq n_1$, $0 \leq j \leq n_2$, n_1 and n_2 are the lengths of biosequences S and T, respectively.

The cost (scoring) function can be replaced by any other scoring functions such as the affine gap penalty [19,20] or score matrices [15,16] like PAMs, Blosums, Gonnets. We do not explain in detail how the traditional dynamic programming works in the biosequence alignment problem here.

Fig. 1 shows the alignment lattice M of biosequences `caagt` and `caaa` with dynamic programming. There are seven possible paths from the lower right corner to the upper left corner in the alignment lattice, and each of them is of optimal score 0. Though our example biosequences are short, the optimal alignment is not unique. The number of the optimal alignments grows rapidly if the given biosequences become longer. Thus we need some other criteria to tell us which alignment is the better among the output alignments.

In Fig. 2, we show all seven possible optimal alignments in the example. It can be seen that each ideal alignment curve is shown by the horizontal dashed line (each alignment position gets the greatest score 1), and the dotted line means the real alignment curve. To transform the dotted line into the horizontal dashed line (ideal alignment curve, the smoothest), one should eliminate the vertical parts (cliffs) of the dotted line. We define the *sum of the cliffs* ζ of each optimal alignment to be the total lengths of the vertical parts of the dotted line. For example, the ζ of case (1) in Fig. 2 is $|1-(-1)|$ (first cliff, between $\frac{\texttt{caa}}{\texttt{caa}}$ and $\frac{\texttt{gt}}{\texttt{-a}}$)+$|-1-(-2)|$ (second cliff, between $\frac{\texttt{caag}}{\texttt{caa-}}$ and $\frac{\texttt{t}}{\texttt{a}}$)=2+1=3, and case (5) is $|1-(-1)|$ (first cliff, between $\frac{\texttt{c}}{\texttt{c}}$ and $\frac{\texttt{-aagt}}{\texttt{aaa--}}$) + $|-1-1|$ (second cliff, between $\frac{\texttt{c-}}{\texttt{ca}}$ and $\frac{\texttt{aagt}}{\texttt{aa--}}$) + $|1-(-1)|$ (third cliff, between $\frac{\texttt{c-aa}}{\texttt{caaa}}$ and $\frac{\texttt{gt}}{\texttt{--}}$) = 2+2+2 = 6.

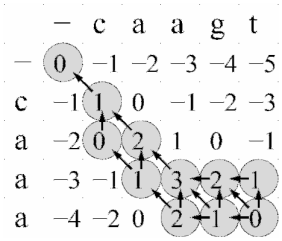

Fig. 1. The alignment lattice *M* and possible optimal paths of biosequences `caagt` and `caaa`.

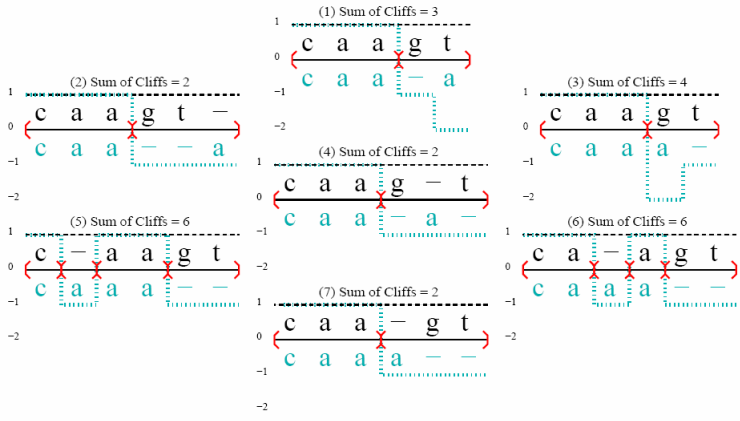

Fig. 2. The seven optimal alignments of biosequences `caagt` and `caaa`.

Table 1. An example for illustrating the smoothest optimal alignment for biosequences `caagt` and `caaa`.

	A_k	ζ
(1)	{0,1,2,3,2,0}	$\|0+2-\ 2\times 1\|+\|1+3-\ 2\times 2\|+\|2+2-\ 2\times 3\|+\|3+0-\ 2\times 2\| = 3$
(2)	{0,1,2,3,2,1,0}	$\|0+2-\ 2\times 1\|+\|1+3-\ 2\times 2\|+\|2+2-\ 2\times 3\|+\|3+1-\ 2\times 2\|+\|2+0-\ 2\times 1\| = 2$
(3)	{0,1,2,3,1,0}	$\|0+2-\ 2\times 1\|+\|1+3-\ 2\times 2\|+\|2+1-\ 2\times 3\|+\|3+0-\ 2\times 1\| = 4$
(4)	{0,1,2,3,2,1,0}	$\|0+2-\ 2\times 1\|+\|1+3-\ 2\times 2\|+\|2+2-\ 2\times 3\|+\|3+1-\ 2\times 2\|+\|2+0-\ 2\times 1\| = 2$
(5)	{0,1,0,1,2,1,0}	$\|0+0-\ 2\times 1\|+\|1+1-\ 2\times 0\|+\|0+2-\ 2\times 1\|+\|1+1-\ 2\times 2\|+\|2+0-\ 2\times 1\| = 6$
(6)	{0,1,2,1,2,1,0}	$\|0+2-\ 2\times 1\|+\|1+1-\ 2\times 2\|+\|2+2-\ 2\times 1\|+\|1+1-\ 2\times 2\|+\|2+0-\ 2\times 1\| = 6$
(7)	{0,1,2,3,2,1,0}	$\|0+2-\ 2\times 1\|+\|1+3-\ 2\times 2\|+\|2+2-\ 2\times 3\|+\|3+1-\ 2\times 2\|+\|2+0-\ 2\times 1\| = 2$

For formal definition, let A_k denote the path of the optimal alignment k and l_k denote the path length.

$$A_k = \{p_0^k, p_1^k, p_2^k, ..., p_{l_k}^k\} \text{ (from the upper left corner to the lower right corner),}$$

where $p_0^k = 0$ and p_i^k corresponds to the accumulated score of the optimal alignment k from path position 1 through path position i,

$$\begin{aligned} \zeta(A_k) &= \sum_{1\le k\le l_k-1} |\,(p_{i+1}^k - p_i^k) - (p_i^k - p_{i-1}^k)\,| \\ &= \sum_{1\le k\le l_k-1} |\,p_{i-1}^k + p_{i+1}^k - 2p_i^k\,| \end{aligned}$$

In the above formula, $p_i^k - p_{i-1}^k$ calculates the height (added score) in position i of alignment k. So that the cliff between positions i+1 and i is $|\,(p_{i+1}^k - p_i^k) - (p_i^k - p_{i-1}^k)\,|$. The smoothest optimal alignment will be A_k if $\zeta(A_k)$ is minimum. We illustrate the example for clarity in Table 1.

In Fig. 2, cases (2), (4) and (7) are three smoothest optimal alignment cases.

It can be seen that the smoothest (minimum sum of cliffs ζ) optimal alignment here is similar to the affine gap penalty. The main idea of the affine gap penalty is to add an extra penalty when a new gap starts. In our definition, each new gap (i.e. from a match or mismatch to gap) causes the cliff (penalty) and there is no cliff if the position is not a new starting gap. The difference is that the affine gap penalty adjusts the result during biosequence alignment, but we fine-tune the result after the affine gap penalty has been adjusted.

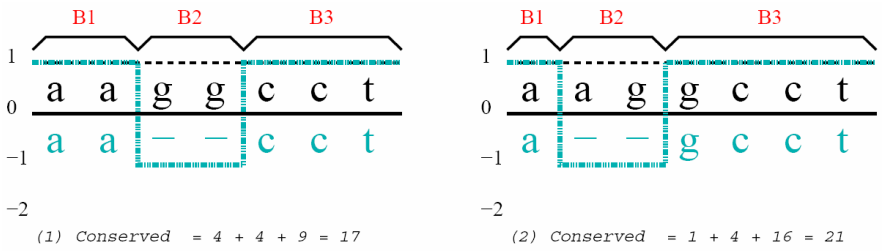

Fig. 3. Two possible templates of biosequence `aaggcct`.

2.2 The Most Conserved Optimal Alignment

Fig. 3 shows two possible templates of biosequence `aaggcct`. The same scoring function in Section 2.1 is applied to them. Both templates get the same optimal alignment score = 3, and even the same $\zeta = 4$. Which one is better? The answer is case (2) `agcct` if we add one more criterion - *conserved* ω.

Table 2. An example for illustrating the most conserved optimal alignment for biosequences `caagt` and `caaa`, used in Fig. 2, where $\alpha = 2$.

	A_k	ω
(1)	{3,2}	$3^2+2^2=9+4=13$
(2)	{3,3}	$3^2+3^2=9+9=18$
(3)	{3,2}	$3^2+2^2=9+4=13$
(4)	{3,3}	$3^2+3^2=9+9=18$
(5)	{1,1,2,2}	$1^2+1^2+2^2+2^2=1+1+4+4=10$
(6)	{2,1,1,2}	$2^2+1^2+1^2+2^2=4+1+1+4=10$
(7)	{3,3}	$3^2+3^2=9+9=18$

The concept of ω actually comes directly from the homology modeling [11], which is one of the most popular methods applied to prediction of protein structures. Its main idea is to search for a similar protein (template) with known 3D structure at the sequence level. We can roughly determine the structure of the target protein by the template. Then perform the global biosequence alignment to get the structurally conserved regions, and copy the structure of those regions as a part of the predicting structure. A region is called *structurally conserved* if the alignment scores are greater than a predefined threshold τ in each position of the region. In other words, longer such region is better.

The definition here is the same as that in Section 2.1. Additionally we introduce the concept of blocks. As shown in Fig. 3, a *block* is an area with either continuously positive scores or continuously negative scores if threshold τ is defined as 0. The positive blocks can be viewed as structurally conversed regions, and we should choose the alignment with the longest positive block. Thus the definition is given as follows.

$$A_k = \{B_0^k, B_1^k, B_2^k, ..., B_{b_k}^k\},$$

where B_i^k denotes the length of block i in optimal alignment k, and b_k is the number of blocks in A_k.

$$\omega(A_k) = \sum_{1 \le i \le b_k} (B_i^k)^\alpha ,$$

where α is a natural number, a parameter to control the weight of the longest block.

The most conserved optimal alignment will be A_k if $\omega(A_k)$ is maximum. An example is illustrated in Table 2.

Though the definition of ω does not choose the alignment with the largest positive block, we think it is more biologically reasonable. The most conserved optimal alignment (i.e. ω is the maximum) breaks the template sequence into two kinds of regions, the similar regions and dissimilar regions. This helps a lot when the homology modeling based methods are applied. The definition of criteria can be modified to fit our requirements if necessary. Some other various criteria will be listed in Section 2.3.

2.3 The Miscellaneous Reasonable Optimal Alignments

There are some other reasonable criteria to get the biologically meaningful optimal alignments. Since they can be solved trivially or similarly with our algorithms, we only list these criteria here, and ignore the discussion about how to solve them.

Minimum Highest Cliff Optimal Alignment We have mentioned about the sum of the cliffs ζ in Section 2.1. Here we modify it that the longest vertical line (cliff) of the optimal alignment should be the minimum among all optimal alignments. That is, the highest cliff of the alignment is the minimum among all the highest cliffs in all optimal alignments. We call it the *minimum highest cliff optimal alignment*. In other words, it becomes a mini-max problem or a bottleneck problem. Let HC_k be the highest cliff of optimal alignment A_k, then the minimum highest cliff optimal alignment will be A_k if HC_k is minimum. As an example, the minimum highest cliff optimal alignments in Fig. 2 are cases (1), (2), and (4)-(7). The highest cliffs of them are all 2.

Shortest Path Optimal Alignment It is widely believed that the biosequences should be compact. Since the scores of those optimal alignments are the same, the shortest one (l_k is minimum) is the most compact to meet our wish. Cases (1) and (3) in Fig. 2 are what we are looking for here.

Largest Block Optimal Alignment In Section 2.2, we summarize the α power of all blocks in optimal alignments, and find which one is the maximum. Here we try to select the optimal alignment with the largest block. This usually means the same alignment as Section 2.2 mentioned. Let LB_k be the largest block of optimal alignment A_k. The largest block optimal alignment will be A_k if LB_k is maximum. The largest block of case (2) in Fig. 3 has length 4.

Positive Blocks Only Optimal Alignment With slight modification from Section 2.2, we summarize the α power of positive blocks only and ignore the negative blocks in the optimal alignment. The reason is that the positive blocks are more meaningful in conserved regions. We use the same definitions in Section 2.2.

$$\omega_p(A_k) = \sum_{1 \le i \le b_k} S_i (B_i^k)^\alpha ,$$

where S_i is 1 if block i is positive and 0 if otherwise. The positive block only optimal alignment will be A_k if $\omega_p(A_k)$ is maximum.

Maximum Loser Region Optimal Alignment The idea comes from Zhang's alignment algorithm [20]. Its spirit is to align the biosequences without low-scoring regions. It is an opposite point of view with the other criteria we mentioned above. A *region* is any continuous part of the alignment and a *loser region* is the region with the minimum score in the alignment, defined as follows.

$$\text{Loser Region LR}_k = \min_{1 \le i \le j \le l_k} (p_j^k - p_i^k) ,$$

where LR_k is the loser region of optimal alignment A_k.

The maximum loser region optimal alignment will be A_k if LR_k is maximum. For example, the loser region scores of cases (5) and (6) in Fig. 2 are -2, which are maximum, since all other loser regions have score -3.

3 An Algorithm for the Smoothest Optimal Alignment

In this section, we shall propose an algorithm to solve the smoothest optimal alignment (SOA) problem defined in Section 2.1. The problem is to find the optimal alignment which is with the minimum sum of cliffs ζ. Let us take the biosequences `caagt` and `caaa` as our example in Fig. 4. The number in each circle corresponds to the total score from the starting position, which is (0,0), to the position, and the pair of numbers (i, j) in circle mean the position indexes in the alignment lattice, and the number beside each edge represents the score (cost) when the edge is included.

Our algorithm is given as follows. The alignment lattice M is of size $(n_1+1)\times(n_2+1)$, where n_1 and n_2 are the lengths of the two given biosequences. In our algorithm, $C_{i,j}$[V,H,D] corresponds to the added score (edge cost) from prior V-vertical, H-horizontal, and D-diagonal position to positions (i, j), and $\zeta_{i,j}${V,H,D} corresponds to the minimum sum of cliffs from position (n_1,n_2) across prior V-vertical, H-horizontal, and D-diagonal positions to position (i, j). For example, $C_{2,2}$ = [− 1,∞,1] means that the vertical edge going to position (2,2) is of cost -1. There is no horizontal edge going to position (2,2), so its cost is ∞. And the diagonal edge is of cost 1. $\zeta_{2,2}$= {4,∞,2} means that the minimum accumulated cliffs from position (n_1,n_2) across vertical edge to position (2,2) is 4. There is no possible path from position (n_1,n_2) across horizontal edge to position (2,2), so the minimum accumulated cliffs is ∞. And the minimum accumulated cliffs from diagonal edge are 2.

The *tracings* in our algorithm are defined as the construction of all possible optimal alignment paths. For example, $Tracings_{3,2}$ =<*V*,*H*,*D* > in Fig. 4 is < *True*,*False*,*True* >. This means that the accumulated score of position (3,2) comes from position (2,2) (vertical edge) or position (2,1) (diagonal edge). We say that a position (i, j) is in tracing if (i, j) is in one of optimal alignment paths. The grey positions in Fig. 1 are in tracing. It is a little programming skill and this will be easier for us to explain our algorithm.

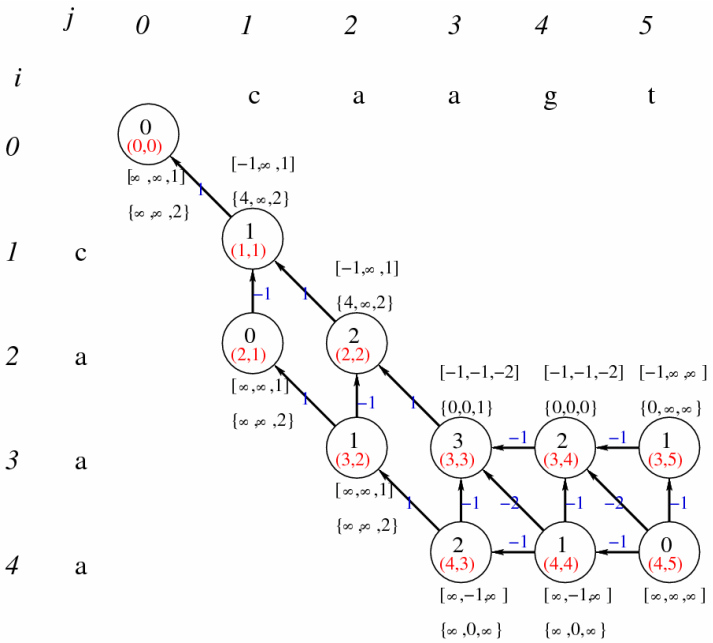

Fig. 4. The final result graph after algorithm SOA is performed on biosequences `caagt` and `caaa`. [ ] represents the values in $C_{i,j}[V,H,D]$ and { } represents the values in $\zeta_{i,j}${V,H,D}.

Algorithm: Smoothest Optimal Alignment (SOA)
Input: Alignment lattice M with tracings.
Output: Minimum sum of cliffs ζ among all optimal alignments.
Step 1: $\zeta_{i,n_2}\{V,H,D\}=\{0,\infty,\infty\}, 0\le i\le n_1-1; \zeta_{n_1,j}\{V,H,D\}=\{\infty,0,\infty\}, 0\le j\le n_2-1.$

If (i,j) is not in tracing, $\zeta_{i,j}\{V,H,D\}=\{\infty,\infty,\infty\}$ and $C_{i,j}[V,H,D]=[\infty,\infty,\infty]$, where 0≤ i≤ n_1, 0≤ j≤ n_2.

Step 2: Compute the following if (i,j) is in tracing. Otherwise do nothing.

$$C_{i,j}=\begin{cases}[V] = \begin{cases}M(i+1,j)-M(i,j) & \text{if there is a tracing from } (i+1,j) \text{ to } (i,j),\\ \infty & \text{otherwise,}\end{cases}\\ [H] = \begin{cases}M(i,j+1)-M(i,j) & \text{if there is a tracing from } (i,j+1) \text{ to } (i,j),\\ \infty & \text{otherwise,}\end{cases}\\ [D] = \begin{cases}M(i+1,j+1)-M(i,j) & \text{if there is a tracing from } (i+1,j+1) \text{ to } (i,j),\\ \infty & \text{otherwise,}\end{cases}\end{cases}$$

where 0≤ i ≤ n_1, 0≤ j≤ n_2.

The goal of this step is to calculate the 3-way (vertical, horizontal, diagonal) added scores (edge costs) of each position.

Step 3: Compute the following if (i,j) is in tracing. Otherwise do nothing.

$$\zeta_{n_1-1,n_2-1}\{V,H,D\}=\{|C_{n_1-1,n_2-1}[V]-C_{n_1,n_2-1}[H]|,|C_{n_1-1,n_2-1}[H]-C_{n_1-1,n_2}[V]|,0\}$$

$$\zeta_{i,j}=\begin{cases}\{V\}=\min\begin{cases}\zeta_{i+1,j}\{V\}+|C_{i,j}[V]-C_{i+1,j}[V]|\\ \zeta_{i+1,j}\{H\}+|C_{i,j}[V]-C_{i+1,j}[H]|\\ \zeta_{i+1,j}\{D\}+|C_{i,j}[V]-C_{i+1,j}[D]|\end{cases}\\ \{H\}=\min\begin{cases}\zeta_{i,j+1}\{V\}+|C_{i,j}[H]-C_{i,j+1}[V]|\\ \zeta_{i,j+1}\{H\}+|C_{i,j}[H]-C_{i,j+1}[H]|\\ \zeta_{i,j+1}\{D\}+|C_{i,j}[H]-C_{i,j+1}[D]|\end{cases}\\ \{D\}=\min\begin{cases}\zeta_{i+1,j+1}\{V\}+|C_{i,j}[D]-C_{i+1,j+1}[V]|\\ \zeta_{i+1,j+1}\{H\}+|C_{i,j}[D]-C_{i+1,j+1}[H]|\\ \zeta_{i+1,j+1}\{D\}+|C_{i,j}[D]-C_{i+1,j+1}[D]|\end{cases}\end{cases}$$

where 0≤ i≤ n_1 − 1, 0≤ j≤ n_2 − 1.

In this step, we calculate the minimum accumulated cliffs from position (n_1,n_2) to position (i, j). For example, position (2,2) in Fig. 4 has two possible ways, vertical and diagonal, to it. In the diagonal way, there are three ways to position (3,3). i.e., the minimum accumulated cliffs across (3,3) to (2,2) is $min(0+|1-(-1)|,0+|1-(-1)|,1+|1-(-2)|) = min(2,2,4) = 2$. There exists no horizontal way to position (2,2), so that $\zeta_{2,2}\{H\} = \infty$. The only possible path across (3,2) to (2,2) has minimum accumulated cliffs $2+|-1-1| = 4$. Finally, we get $\zeta_{2,2}\{V,H,D\} = \{4,\infty,2\}$.

Step 4: After $\zeta_{0,0}\{V,H,D\}$ has been found, we can trace back to find the smoothest optimal alignment of given biosequences.

It is very important to compute elements in order. Here the only condition is that (i+1, j), (i, j+1) and (i+1, j+1) have to be computed before we compute (i, j), $0 \le i \le n_1 - 1$ and $0 \le j \le n_2 - 1$.

Fig. 4 shows the full result after SOA is performed. The numbers in [ ] are $C_{i,j}$ [V,H,D], and the numbers in { } are $\zeta_{i,j}${V,H,D}. It is trivial that the time complexity of Algorithm SOA is $O(n^2)$.

4 An Algorithm for the Most Conserved Optimal Alignment

In this section, we shall find the most conserved optimal alignment which is with the maximum ω, defined in Section 2.2. It can be done by the same technique as we used in Section 3, dynamic programming. We illustrate it with Fig. 5. The meanings of the numbers in Fig. 5 are the same as those in Section 3, except those numbers in { }. And < > are newly defined, we shall explain their meanings later.

In our algorithm, n_1 and n_2 denote the lengths of the two given biosequences. The alignment lattice M here is of size $(n_1+1)\times(n_2+1)$. Here we use $\alpha = 2$ to control the weight of the longest block. In our algorithm, $L_{i,j}$ <V,H,D> means the length of the current block till now from prior V-vertical, H-horizontal, and D-diagonal positions to position (i, j), and $\omega_{i,j}${V,H,D} corresponds to the maximum ω from position (n_1,n_2) across prior V-vertical, H-horizontal, and D-diagonal positions to position (i, j) without adding $L_{i,j}<V,H,D>^2$. $C_{i,j}$ [V,H,D] is reused as the same meaning in algorithm SOA.

Algorithm: Most Conserved Optimal Alignment (MCOA)
Input: Alignment lattice M with tracings.
Output: Maximum ω among all optimal alignments.
Step 1: $L_{i,j}\{V,H,D\} = \{0,0,0\}, \omega_{i,j}\{V,H,D\} = \{0,0,0\}, 0 \le i \le n_i, 0 \le j \le n_2$.

If (i, j) is not in tracing, $C_{i,j}[V,H,D] = [\infty,\infty,\infty]$, where $0 \le i \le n_1$, $0 \le j \le n_2$.

Step 2: Compute the following if (i, j) is in tracing. Otherwise do nothing.

$$C_{i,j} = \begin{cases} [V] = \begin{cases} M(i+1,j) - M(i,j) & \text{if there is a tracing from } (i+1,j) \text{ to } (i,j), \\ \infty & \text{otherwise,} \end{cases} \\ [H] = \begin{cases} M(i,j+1) - M(i,j) & \text{if there is a tracing from } (i,j+1) \text{ to } (i,j), \\ \infty & \text{otherwise,} \end{cases} \\ [D] = \begin{cases} M(i+1,j+1) - M(i,j) & \text{if there is a tracing from } (i+1,j+1) \text{ to } (i,j), \\ \infty & \text{otherwise,} \end{cases} \end{cases}$$

where $0 \le i \le n_1$, $0 \le j \le n_2$.

Step 3: Compute the following if (i, j) is in tracing. Otherwise do nothing.

$$\omega_{i,j} = \begin{cases} \{V\} = \begin{cases} Choose(i,j,V) & \text{if there is a tracing from } (i+1,j) \text{ to } (i,j), \\ 0 & \text{otherwise,} \end{cases} \\ \{H\} = \begin{cases} Choose(i,j,H) & \text{if there is a tracing from } (i,j+1) \text{ to } (i,j), \\ 0 & \text{otherwise,} \end{cases} \\ \{D\} = \begin{cases} Choose(i,j,D) & \text{if there is a tracing from } (i+1,j+1) \text{ to } (i,j), \\ 0 & \text{otherwise,} \end{cases} \end{cases}$$

where $0 \le i \le n_1 - 1$, $0 \le j \le n_2 - 1$.

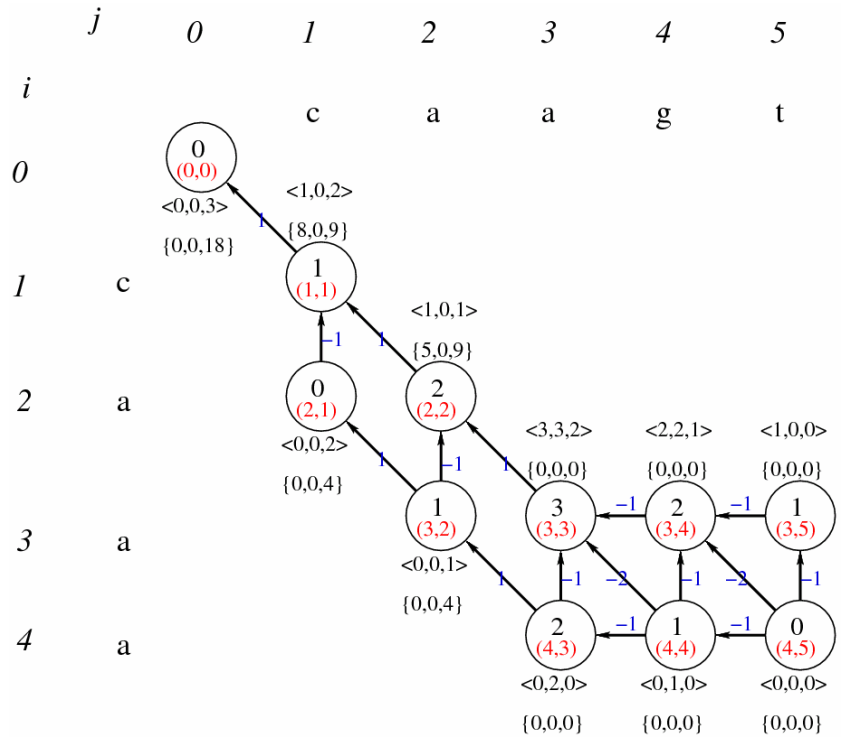

Fig. 5. The final result graph after algorithm MCOA is performed on biosequences `caagt` and `caaa`, which $\alpha = 2$ and $\tau = 0$. < > represents the values in $L_{i,j}$ $<V,H,D>$ and { } represents the values in $\omega_{i,j}\{V,H,D\}$.

Step 4: $\omega_{0,0} = \begin{cases} \{V\} & = & \omega_{0,0}\{V\} + L_{0,0} < V >^2 \\ \{H\} & = & \omega_{0,0}\{H\} + L_{0,0} < H >^2 \\ \{D\} & = & \omega_{0,0}\{D\} + L_{0,0} < D >^2 \end{cases}$.

Step 5: After $\omega_{0,0}\{V,H,D\}$ has been found, we can trace back to find the most conserved optimal alignment of given biosequences.

Since it is complicated to decide the value of $\omega_{i,j}\{V,H,D\}$, we use the function *Choose*(i, j, W) to decide the correct value, where i and j mean the coordinates and W (may be V or H or D) corresponds to the direction that it came from. We show the function *Choose*(i, j, W) as follows.

Function: Choose(i, j,W)
Input: i, j and W, where i, j mean the coordinates and W (may be V or H or D) corresponds to the direction that it came from.

Output: The correct value of $\omega_{i,j}\{W\}$, and the value of $L_{i,j}$ <W> which is updated to a correct one.

Step 1: $(x, y) = \begin{cases} (i+1, j) & \text{if} \quad W = V \\ (i, j+1) & \text{if} \quad W = H \\ (i+1, j+1) & \text{if} \quad W = D \end{cases}$

Step 2: Check whether if the phase is changed from $(x,y).W'$ to $(i, j).W$ or not.

($(x,y).W'$ corresponds to the outgoing edge of direction W' at position (x,y).)

A phase is changed if and only if $\begin{cases} C_{x,y}[W'] < \tau & \& \quad C_{i,j}[W] \geq \tau \\ C_{x,y}[W'] > \tau & \& \quad C_{i,j}[W] \leq \tau \\ C_{x,y}[W'] = \tau & \& \quad C_{i,j}[W] \neq \tau, \end{cases}$

where W' is V, H or D, and τ is the threshold to judge if a cost is conserved. Phase-changed means a new block, and we have to reset the length of the current block to 1.

Step 3: Compute the following:

$$TempL = \begin{cases} < V >= \begin{cases} 1 & \text{if phase is changed from } (x, y).V \text{ to } (i,j).W, \\ L_{x,y} < V > +1 & \text{otherwise,} \end{cases} \\ < H >= \begin{cases} 1 & \text{if phase is changed from } (x, y).H \text{ to } (i,j).W, \\ L_{x,y} < H > +1 & \text{otherwise,} \end{cases} \\ < D >= \begin{cases} 1 & \text{if phase is changed from } (x, y).D \text{ to } (i,j).W, \\ L_{x,y} < D > +1 & \text{otherwise,} \end{cases} \end{cases}$$

$$Temp\omega = \begin{cases} \{V\} = \begin{cases} \omega_{x,y}\{V\} + L_{x,y} < V >^2 + TempL < V >^2 & \text{if phase is changed from } (x, y).V \text{ to } (i,j).W, \\ \omega_{x,y}\{V\} + TempL < V >^2 & \text{otherwise,} \end{cases} \\ \{H\} = \begin{cases} \omega_{x,y}\{H\} + L_{x,y} < H >^2 + TempL < H >^2 & \text{if phase is changed from } (x, y).H \text{ to } (i,j).W, \\ \omega_{x,y}\{H\} + TempL < H >^2 & \text{otherwise,} \end{cases} \\ \{D\} = \begin{cases} \omega_{x,y}\{D\} + L_{x,y} < D >^2 + TempL < D >^2 & \text{if phase is changed from } (x, y).D \text{ to } (i,j).W, \\ \omega_{x,y}\{D\} + TempL < D >^2 & \text{otherwise,} \end{cases} \end{cases}$$

Step 4: Without loss of generality, assume Tempω{Z} is greater than the other two. Then,

$L_{i,j}$ <W >= TempL < Z >

$$OK = \begin{cases} \omega_{x,y}\{Z\} + L_{x,y} < Z >^2 & \text{if phase is changed from } (x, y).Z \text{ to } (i,j).W, \\ \omega_{x,y}\{Z\} & \text{otherwise.} \end{cases}$$

Notice that if there are more than one maximum in *Temp*ω{V,H,D}, we should find the most benefit one as our Z.

Step 5: Return(OK).

The computing order we use here is the same as it in Section 3. Fig. 5 shows the full result after MCOA is performed. The numbers in < > are $L_{i,j}$<V,H,D>, and the numbers in { } are $\omega_{i,j}${V,H,D}. Since function *Choose*(*i*, *j*,*W*) can be achieved in constant time, the time complexity of MCOA is clearly $O(n^2)$.

5 Conclusions

In this paper, we propose two algorithms to refine the traditional optimal biosequence alignments in different criteria without increasing the time complexity of the original algorithm. The algorithms are efficient and easy to implement. The criteria to find the better alignment among output alignments are reasonable and biologically meaningful. Though we may not decide the unique alignment, this is still a novel concept in improving the biological knowledge and a brand new way to study what the real optimal means. The refined optimal alignment of our algorithm is both mathematically optimal and biologically meaningful. It helps any method based on homology modeling in predicting 3D protein structure to find a better template, then to get a more accurate prediction result.

What are the criteria that we need to find the better alignment among output alignments? The problem can be open to discuss. One thing we should notice is that the criteria to measure if an output alignment is better than another may be meaningless in traditional biosequence alignment techniques. Taking the definitions in Section 2, for example, any two biosequences with the below alignment may get the best score in some of our definitions.

$a_1a_2\cdots a_m$--------
--------$b_1b_2\cdots b_n$

However, it is a very bad alignment. Thus we cannot apply those criteria to the traditional biosequence alignment problem. How to solve the problem with some different fantastic criteria is interesting, even if we ignore the practical use of the criteria. In the future, we may like to study if it is possible to solve the problem with the same time complexity by two or more criteria, or we have to perform the biosequence alignments over and over again to extract the better alignment among the output alignments. Moreover, does this refining concept do any good with multiple biosequence alignment problems? Those problems may be worth further study.

Acknowledgement

This research work was partially supported by the National Science Council of Taiwan under contract NSC-95-2221-E-110-084.

References

[1] S. Altschul and B.W. Erickson, "Optimal sequence alignment using affine gap costs," *Journal of Molecular Biology*, Vol. 48, pp. 603–616, 1986.

[2] D. F. Feng, M. S. Johnson, and R. F. Doolittle, "Aligning amino acid sequences: comparison of commonly used method s," *Journal of Molecular Evolution*, Vol. 21, pp. 112–125, 1985.

[3] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of Molecular Biology*, Vol. 162, pp. 705–708, 1982.

[4] O. Gotoh, "Optimal sequence alignment allowing for long gaps," *Bulletin of Mathematical Biology*, Vol. 52, pp. 359–373, 1990.

[5] W. Pearson andW.Miller, "Dynamic programming algorithms for biological sequence comparison," *Methods in Enzymology*, Vol. 210, pp. 575–601, 1992.

[6] A. Apostolico and C. Guerra, "The longest common subsequence problem revisited," *Algorithmica*, No. 2, pp. 315–336, 1987.

[7] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," *Seventh International Symposium on String Processing Information Retrieval*, pp. 39–48, 2000.

[8] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM*, Vol. 24, No. 4, pp. 664–675, 1977.

[9] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Communications of the ACM*, Vol. 20, No. 5, pp. 350–353, 1977.

[10] C. B. Yang and R. C. T. Lee, "Systolic algorithms for the longest common subsequence problem," *Journal of the Chinese Institute of Engineers*, Vol. 10, No. 6, pp. 691–699, 1987.

[11] Y. Y. Chen, C. B. Yang, and K. T. Tseng, "Prediction of protein structures based on curve alignment," *Proc. of the 20th Workshop on Combinatorial Mathematics and Computation Theory*, pp. 33–44, 2003.

[12] M. Hilbert, G. Bohm, and R. Jaenicke, "Structural relationships of homologous proteins as a fundamental principle in homology modeling," *Proteins*, Vol. 17, No. 2, pp. 138–151, 1993.

[13] R. Lewin, "When does homology mean something else?" *Science*, Vol. 237, p. 1570, 1987.

[14] S. F. Altschul, W. Gish, W. Miller, E.W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, Vol. 215, pp. 403–410, 1990.

[15] M. O. Dayhoff., *Atlas of Protein Sequence and Structure*, National Biomedical Research Foundation, Washington, DC, 1978.

[16] R.M. Schwartz and M. O. Dayhoff., *Matrices for detecting distant relationships*. National Biomedical Research Foundation, Washington, DC, 1979.

[17] D. Naor and D. L. Brutlag, "On near-optimal alignments of biological sequences," *Journal of Computing Biology*, Vol. 4, pp. 349–366, 1994.

[18] S. F. Altschul, “Gap costs for multiple sequence alignment,” *Journal of Theoretical Biology*, Vol. 138, pp. 297–309, 1989.

[19] A. M. Lesk, M. Levitt, and C. Chothia, “Alignment of the amino acid sequences of distantly related proteins using variable gap penalties,” *Protein Engineering*, Vol. 1, pp. 77–78, 1986.

[20] Z. Zhang, P. Berman, and W. Miller, “Alignments without low-scoring regions,” *Research in Computational Molecular Biology*, Vol. 5, pp. 294–301, 1998.