The Better Alignment Among Output Alignments

Kuo-Tsung Tseng, Chang-Biau Yang and Kuo-Si Huang
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
cbyang@cse.nsysu.edu.tw

Key words: computational biology, longest common sequence, sequence alignment, dynamic programming

***Abstract*—**

The sequence alignment is the one of the most fundamental technique of nowaday molecular biology. Computer scientists mapped it into the longest common sequence (LCS) problem which is a well-studied problem in algorithms. The optimal alignment score of LCS can be found in $O(n_1n_2)$ time with the dynamic programming technique, where n_1 and n_2 is the lengths of the two sequences. Scientists presented many scoring functions to measure the goodness of the alignments in different criteria, such as 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, this optimal alignment is not unique in most time and the biologically meaningful alignment may not be optimal alignment. In this paper, we present some new definitions to measure the best alignment of the optimal alignments and illustrate the algorithms to solve them. The proposed algorithms give not only alignment of the optimal score but also more biologically meaningful without increasing the computing complexity of the original algorithm. Additionally new definitions can be used to find the better template when predicting the 3D protein structures. They are also interesting problems even if we do not consider the practical use of them.

I. Introduction

One of the most important problems that biologists desire to solve is the *sequence alignment* problem, on which there is plenty of research [1, 8–10, 17]. The problem is given two biological sequences and its goal is to obtain the optimal score based on some predefined scoring criteria. A biological sequence is a string consisting of characters chosen from a set of alphabets $\sum$, where $\sum$ 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. The gap in alignment is represented by the dash symbol (-). The sequence alignment problem was mapped into the longest common subsequence problem which is well studied problem in algorithms [4, 5, 12, 13, 19] by computer scientists.

The sequence alignment judges similarities and differences between two biosequences that are usually taken as the selecting criteria of the prediction templates when 3D protein structures are to be predicted. It is important to select good templates to predict protein 3D structures in homology modeling [6, 11, 15]. The better templates produce the better prediction results.

The sequence 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 sequences. The argument really is the definition of the optimum between biologists and computer scientists. The algorithms designed by computer scientists usually result in a mathematically optimal score, but what biologists wish is a biologically optimal score. Though many scoring criteria were proposed [3, 7, 18], those are still actually mathematically 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 optimal one, so that they preasented the near-optimal alignments to provide more possible alignments for biologists to choose [16]. In fact, more alignments increase the possibility of finding the coreect alignment, but too many alignments may confuse the biologists. Thus, we do need some other biologically filtering criteria to help us to choose the correct 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 we know that the alignment of the optimal score is not unique in most time, we should choose the most biologically meaningful one in those mathematically optimal alignments.

In this paper, we present some mathematical scoring criteria to define so called the most biologically meaningful alignment when the alignment of the optimal score is not unique. (In [16], we know that the correct alignment may not be the optimal one. We leave it out of consideration in this paper since it is easy to apply our concepts in near-optimal alignments.) The definitions do not guarantee the unique result, but it can be conquered by more criteria. Our algorithms with the new problem definitions can choose the best alignment from a set of optimal alignments without increasing time complexity of the original algorithm. Even if we do not consider the practical use of the new problem definitions, they are themselves interesting problems.

The rest of this paper is organized as follows. In Section II, we first introduce some scoring criteria that are biologically meaningful in optimal alignments. Next, we illustrate our

	−	c	a	a	g	t
−	0	−1	−2	−3	−4	−5
c	−1	1	0	−1	−2	−3
a	−2	0	2	1	0	−1
a	−3	−1	1	3	2	1
a	−4	−2	0	2	1	0

Fig. 1. The alignment lattice M and possible optimal paths of sequences caagt and caaa.

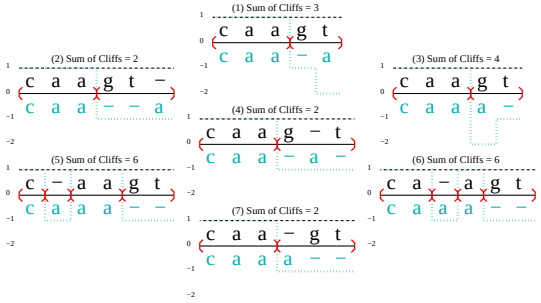

Fig. 2. The seven optimal alignments of sequences caagt and caaa.

algorithms to solve the problem in Sections III and IV. Finally, some discussions and conclusions are given in Section V.

II. Definitions

A. 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 sequences $S = \texttt{caagt}$ and $T = \texttt{caaa}$. The cost function is defined as follows.

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

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

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

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

The cost (scoring) function can be replaced by any other scoring functions such as affine gap penalty [2, 14] or score matrices [7, 18] like PAMs, Blosums, Gonnets. We do not explain how traditional dynamic programming works in the sequence alignment problem here. It can be found almost in every book talking about algorithms.

Figure 1 shows the alignment lattice M of sequences 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 sequences are short, the optimal alignment is not unique. The number of the optimal alignments grows larger and larger if the given sequences become longer. Thus we do need some other criteria to tell us which alignment is the best of the bests.

In Figure 2, we show all seven possible optimal alignments in the example. It is easy to see that the ideal alignment curve of each is 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 sum of cliffs of case (1) in Figure 2 is $|1-(-1)|$ (first cliff, between $\frac{\text{c a a}}{\text{c a a}}$ and $\frac{\text{g t}}{\text{- a}}$) $+ |-1-(-2)|$ (second cliff, between $\frac{\text{c a a g}}{\text{c a a -}}$ and $\frac{\text{t}}{\text{a}}$) $= 2+1 = 3$, and case (5) is $|1-(-1)|$ (first cliff, between $\frac{\text{c}}{\text{c}}$ and $\frac{\text{- a a g t}}{\text{a a a - -}}$) $+ |-1-1|$ (second cliff, between $\frac{\text{c -}}{\text{c a}}$ and $\frac{\text{a a g t}}{\text{a a - -}}$) $+ |1-(-1)|$ (third cliff, between $\frac{\text{c - a a}}{\text{c a a a}}$ and $\frac{\text{g t}}{\text{- -}}$) $= 2+2+2 = 6$.

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

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

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

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

The $p_i^k - p_{i-1}^k$ 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. Let us illustrate the example for clarity in Table I.

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

It is easy to see that the smoothest (minimum sum of cliffs ζ) optimal alignment here is similar to affine gap penalty. The main idea of 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 gap. The difference is that affine gap penalty adjusts the result during sequence alignment, but we fine-tune the result after affine gap penalty has adjusted it.

B. The Most Conserved Optimal Alignment

Figure 3 shows two possible templates of sequence aaggcct. They use the same score function as in Section II-A. Both templates get the same optimal alignment score = 3, and even the same ζ = 4. Which one is better? The answer is case (2) agcct if we add the criteria *conserved* ω in consideration.

The concept of ω actually comes directly from the homology modeling [6], which is one of the most famous methods applied to prediction of protein structures. Its main idea is to search for a similar protein (template) with known 3D structure at sequence level. We can roughly determine the

TABLE I

AN EXAMPLE FOR ILLUSTRATING THE SMOOTHEST OPTIMAL ALIGNMENT FOR SEQUENCES CAAGT AND CAAA

	A_k	ζ
(1)	{0,1,2,3,2,0}	$\|0+2-2\times1\|+\|1+3-2\times2\|+\|2+2-2\times3\|+\|3+0-2\times2\| = 3$
(2)	{0,1,2,3,2,1,0}	$\|0+2-2\times1\|+\|1+3-2\times2\|+\|2+2-2\times3\|+\|3+1-2\times2\|+\|2+0-2\times1\| = 2$
(3)	{0,1,2,3,1,0}	$\|0+2-2\times1\|+\|1+3-2\times2\|+\|2+1-2\times3\|+\|3+0-2\times1\| = 4$
(4)	{0,1,2,3,2,1,0}	$\|0+2-2\times1\|+\|1+3-2\times2\|+\|2+2-2\times3\|+\|3+1-2\times2\|+\|2+0-2\times1\| = 2$
(5)	{0,1,0,1,2,1,0}	$\|0+0-2\times1\|+\|1+1-2\times0\|+\|0+2-2\times1\|+\|1+1-2\times2\|+\|2+0-2\times1\| = 6$
(6)	{0,1,2,1,2,1,0}	$\|0+2-2\times1\|+\|1+1-2\times2\|+\|2+2-2\times1\|+\|1+1-2\times2\|+\|2+0-2\times1\| = 6$
(7)	{0,1,2,3,2,1,0}	$\|0+2-2\times1\|+\|1+3-2\times2\|+\|2+2-2\times3\|+\|3+1-2\times2\|+\|2+0-2\times1\| = 2$

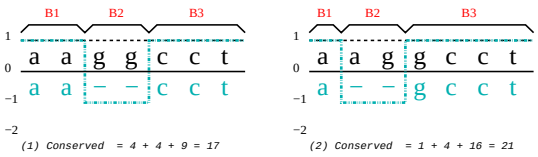

Fig. 3. Two possible templates of sequence aaggcct.

structure of the query protein by the template. Then perform the global sequence 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 threshold τ in each position of the region. In other words, longer such regions are better.

The definition here is the same as that in Section II-A. Additionally we introduce the concept of blocks. As shown in Figure 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 viedwed as structurally conversed regions, and we should choose the alignment with the longest positive block . Thus the definition is given as follows.

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

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 II.

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 various criteria will be listed in Section II-C.

TABLE II

AN EXAMPLE FOR ILLUSTRATING THE MOST CONSERVED OPTIMAL ALIGNMENT FOR SEQUENCES CAAGT AND CAAA, USED IN FIGURE 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}	$3^2+3^2 = 9+9 = 18$

C. The Miscellaneous Reasonable Optimal Alignments

There are some other reasonable criteria to get the biologically meaningful optimal alignments. Since they can be solved similarly with our algorithms or trivial to solve, 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 II-A. Here we modify it that the longest vertical line (cliff) of the optimal alignment should be 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 Figure 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. The cases (1) and (3) in Figure 2 are what we are looking for here.

Largest Block Optimal Alignment:
In Section II-B, we summarize the square 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 II-B 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 Figure 3 has length 4.

Positive Blocks Only Optimal Alignment:

With minor modification from Section II-B, we summarize the square 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 II-B.

$$\omega_p(A_k) = \sum_{1 \le i \le b_k} S_i (B_i^k)^2$$

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 [20]. Its spirit is to align the sequences 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.

$$\text{Loser Region } LR_k = minimum_{1 \le i \le j \le l_k}(p_j^k - p_i^k)$$

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 Figure 2 are -2, which are maximum, since all the other loser region scores are -3.

III. 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 II-A. The problem is to find the optimal alignment which is with the minimum sum of cliffs ζ. Let us take the sequences `caagt` and `caaa` as our example in Figure 4. The number in each circle means 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 (weight) when the edge is included.

Our algorithm is 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 sequences. In our algorithm, $C_{i,j}$[V,H,D] means the added score (edge weight) from prior V-vertical, H-horizontal, and D-diagonal position to position (i, j), and $\zeta_{i,j}\{$V,H,D$\}$ means the minimum sum of cliffs from position (n_1,n_2) across prior V-vertical, H-horizontal, and D-diagonal position to position (i, j). For example, $C_{2,2} = [-1, \infty, 1]$ means that the vertical edge going to position (2,2) is of weight -1. There is no horizontal edge going to position (2,2), so its weight is ∞. And the diagonal edge is of weight 1. $\zeta_{2,2} = \{4, \infty, 2\}$ means that the minimum accumulated cliffs from position (n_1,n_2) across vertical edge to position (2,2) is 4. There is no passible 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 is 2.

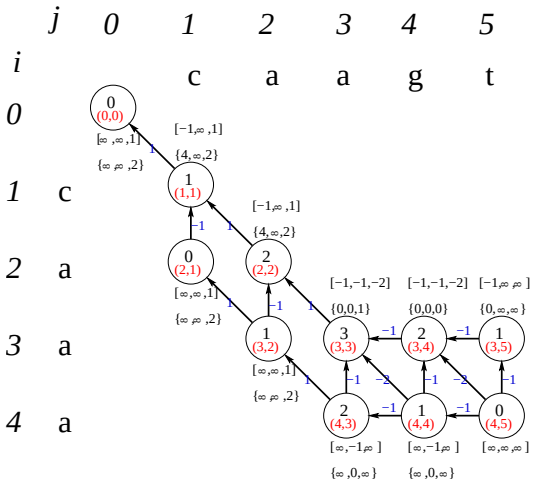

Fig. 4. The final result graph after algorithm SOA is performed on sequences `caagt` and `caaa`. [ ] represents the values in $C_{i,j}[V,H,D]$ and { } represents the values in $\zeta_{i,j}\{$V,H,D$\}$.

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 Figure 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 possible optimal alignment paths. The grey positions in Figure 1 are in tracing. It is a little programming skill and this will be easier for us to explain our algorithm.

Algorithm SOA:

Smoothest Optimal Alignment

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]=[∞,∞,∞], 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 $0 \le i \le n_1$, $0 \le j \le n_2$.

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

Step 3:

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

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

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

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

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 Figure 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 sequences.

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 $0\le j\le n_2-1$.

Figure 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 $O(n^2)$.

IV. An Algorithm for the Most Conserved Optimal Aligmnet

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

In our algorithm, n_1 and n_2 denotes the lengths of two previous given sequences. 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 position to position (i,j), and $\omega_{i,j}\{$V,H,D$\}$ means the maximum ω from position (n_1,n_2) across prior V-vertical, H-horizontal, and D-diagonal position

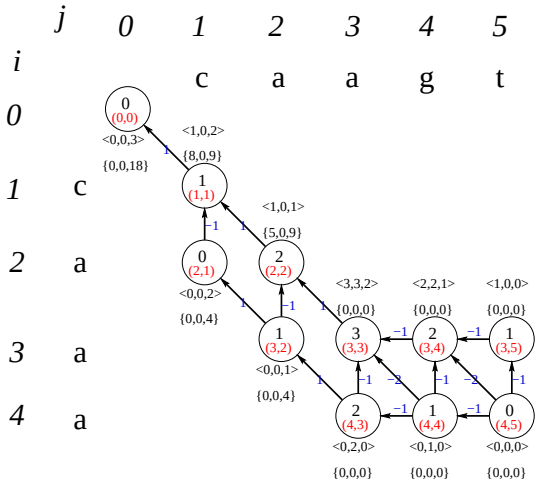

Fig. 5. The final result graph after algorithm MCOA is performed on sequences caagt and caaa. < > represents the values in $L_{i,j}<V,H,D>$ and { } represents the values in $\omega_{i,j}\{$V,H,D$\}$.

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 MCOA:

Most Conserved Optimal Alignment

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$\}=\{0,0,0\}$, $0\le i\le n_1$,$0\le j\le n_2$.

If (i,j) is not in tracing, $C_{i,j}$[V,H,D]=[∞,∞,∞], 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 $0\le i\le n_1-1$, $0\le j\le n_2-1$.

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 sequences.

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

Function: Choose(i, j, W)

Input:

i,j,W, where i,j means the coordinates and W (may beV or H or D) means the direction that it came from.

Output:

The correct value of $\omega_{i,j}\{\text{W}\}$, and change the value of $L_{i,j}$ <W> to correct one.

Step 1:

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

Step 2:

Is the phase changed from $(x,y).W'$ to $(i,j).W$?

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

where 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 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:

Suppose Temp$\omega\{\text{Z}\}$ 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 $Temp\omega\{\text{V,H,D}\}$, 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 III. Figure 5 shows the full result after MCOA is performed. The numbers in $<\ >$ are $L_{i,j}$<V,H,D>, and the numbers in $\{\ \}$ are $\omega_{i,j}\{\text{V,H,D}\}$. Since function $Choose(i,j,W)$ can be achieved in constant time, the time complexity of MCOA is clearly $O(n^2)$.

V. Conclusions

In this paper, we propose two algorithms to refine the traditional optimal sequence 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 best of the bests are reasonable and biologically meaningful. Though we may not decide the unique best alignment, this is still a whole novel concept in improving the biological knowledge and a brand new way to research about what the real optimal means.

The refined optimal alignment of our algorithm is both mathematically optimal and biologically meaningful. It helps any methods based on homology modeling in predicting protein 3D structure to find a better template, then to get a better predicting result.

What are criteria that we need to find the best of the bests? The problem can be open to discuss. One thing we should notice is that the criteria to measure if an optimal alignment is better than the other one may be meaningless in traditional sequence alignment technique. Taking definitions in Section II for example, any two sequences of the alignment below may get the best score in some of our definitions.

$$\begin{array}{l} a_1 a_2 \ \cdots \ a_m - - - - - - \\ - - - - - - b_1 b_2 \ \cdots \ b_n \end{array}$$

It is a very bad alignment. Thus we can not apply those criteria to traditional sequence alignment problem. How to solve the problem with some different fantastic criteria are interesting problems to study, 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 in the same time with two or more criteria, or we have to do the sequence alignments over and over again to extract the best of the bests. Moreover, can we do any good about this refining concept in multiple sequence alignment? Those problems may be worth to study.

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

[3] 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.

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

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

[6] 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.
[7] M. O. Dayhoff., *Atlas of Protein Sequence and Structure.* National Biomedical Research Foundation, Washington, DC, 1978.
[8] 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.
[9] O. Gotoh, “An improved algorithm for matching biological sequences,” *Journal of Molecular Biology*, Vol. 162, pp. 705–708, 1982.
[10] O. Gotoh, “Optimal sequence alignment allowing for long gaps,” *Bulletin of Mathematical Biology*, Vol. 52, pp. 359–373, 1990.
[11] M. Hilbert, G. Bohm, and R. Jaenicke, “Structural relationships of homologous proteins as a fundamen tal principle in homology modeling,” *Proteins*, Vol. 17, No. 2, pp. 138–151, 1993.
[12] D. S. Hirschberg, “Algorithms for the longest common subsequence problem,” *Journal of the ACM*, Vol. 24, No. 4, pp. 664–675, 1977.
[13] 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.
[14] A. M. Lesk, M. Levitt, and C. Chothia, “Alignment of the amino acid sequences of distantly related protei ns using variable gap penalties,” *Protein Engineering*, Vol. 1, pp. 77–78, 1986.
[15] R. Lewin, “When does homology mean something else?,” *Science*, Vol. 237, p. 1570, 1987.
[16] D. Naor and D. L. Brutlag, “On near-optimal alignments of biological sequences,” *Journal of Computing Biology*, Vol. 4, pp. 349–366, 1994.
[17] W. Pearson and W. Miller, “Dynamic programming algorithms for biological sequence comparison,” *Methods in Enzymology*, Vol. 210, pp. 575–601, 1992.
[18] R. M. Schwartz and M. O. Dayhoff., *Matrices for detecting distant relationships.* National Biomedical Research Foundation, Washington, DC, 1979.
[19] 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.
[20] Z. Zhang, P. Berman, and W. Miller, “Alignments without low-scoring regions,” *Research in Computational Molecular Biology*, Vol. 5, pp. 294–301, 1998.