# A Class Note on Global Alignment

Kun-Mao Chao[1,2,3]

[1]Graduate Institute of Biomedical Electronics and Bioinformatics
[2]Department of Computer Science and Information Engineering
[3]Graduate Institute of Networking and Multimedia
National Taiwan University, Taipei, Taiwan 106

September 30, 2008

In nature, even a single amino acid sequence contains all the information necessary to determine the fold of the protein. However, the folding process is still mysterious to us, and some valuable information can be revealed by sequence comparison. Take a look at the following sequence:

```
THETR UTHIS MOREI MPORT ANTTH ANTHE FACTS
```

What did you see in the above sequence? By comparing it with the words in the dictionary, we find the tokens "`FACTS`," "`IMPORTANT`," "`IS`," "`MORE`," "`THAN`," "`THE`," and "`TRUTH`." Then we figure out the above is the sentence "The truth is more important than the facts."

Even though we have not yet decoded the DNA and protein languages, the emerging flood of sequence data has provided us with a golden opportunity of investigating the evolution and function of biomolecular sequences. We are in a stage of compiling dictionaries for DNA, proteins, and so forth. Sequence comparison plays a major role in this line of research and thus becomes the most basic tool of bioinformatics.

Sequence comparison has wide applications to molecular biology, computer science, speech processing, and so on. In molecular biology, it is often used to reveal similarities among sequences, determine the residue-residue correspondences, locate patterns of conservation, study gene regulation, and infer evolu-

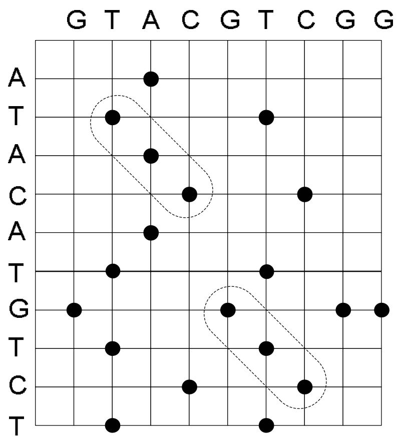

Figure 1: A dot matrix of the two sequences ATACATGTCT and GTACGTCGG.

tionary relationships. It helps us to fish for related sequences in databanks, such as the GenBank database. It can also be used for the annotation of genomes.

## 1 Dot Matrix

A dot matrix is a two-dimensional array of dots used to highlight the exact matches between two sequences. Given are two sequences $A = \langle a_1, a_2, \ldots, a_m \rangle$ (or $A = a_1 a_2 \ldots a_m$ in short), and $B = \langle b_1, b_2, \ldots, b_n \rangle$. A dot is plotted on the $(i, j)$ entry of the matrix if $a_i = b_j$. Users can easily identify similar regions between the two sequences by locating those contiguous dots along the same diagonal. Figure 1 gives a dot matrix of the two sequences ATACATGTCT and GTACGTCGG. Dashed lines circle those regions with at lease three contiguous matches on the same diagonal.

A dot matrix allows the users to quickly visualize the similar regions of two sequences. However, as the sequences get longer, it becomes more involved to determine their most similar regions, which can no longer be answered by merely looking at a dot matrix. It would be more desirable to automatically identify those similar regions and rank them by their "similarity scores." This leads to the development of sequence alignment.

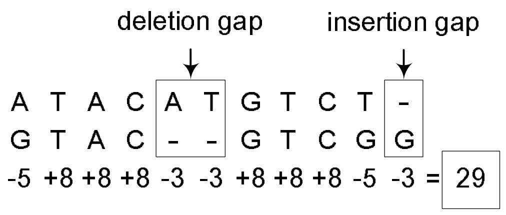

Figure 2: An alignment of the two sequences `ATACATGTCT` and `GTACGTCGG`.

## 2 Global Alignment

Following its introduction by Needleman and Wunsch in 1970, dynamic programming has become a major algorithmic strategy for many optimization problems in sequence comparison. This strategy is guaranteed to produce an alignment of two given sequences having the highest score for a number of useful alignment-scoring schemes.

Given two sequences $A = a_1a_2\ldots a_m$, and $B = b_1b_2\ldots b_n$, an *alignment* of $A$ and $B$ is obtained by introducing dashes into the two sequences such that the lengths of the two resulting sequences are identical and no column contains two dashes. Let $\Sigma$ denote the alphabet over which $A$ and $B$ are defined. To simplify the presentation, we employ a very simple scoring scheme as follows. A score $\sigma(a,b)$ is defined for each $(a,b) \in \Sigma \times \Sigma$. Each indel, *i.e.*, a column with a space, is penalized by a constant $\beta$. The score of an alignment is the sum of $\sigma$ scores of all columns with no dashes minus the penalties of the gaps. An *optimal global alignment* is an alignment that maximizes the score. By global alignment, we mean that both sequences are aligned globally, i.e., from their first symbols to their last.

Figure 2 gives an alignment of sequences `ATACATGTCT` and `GTACGTCGG` and its score. In this and the next sections, we assume the following simple scoring scheme. A match is given a bonus score 8, a mismatch is penalized by assigning score $-5$, and the gap penalty for each indel is $-3$. In other words, $\sigma(a,b) = 8$ if $a$ and $b$ are the same, $\sigma(a,b) = -5$ if $a$ and $b$ are different, and $\beta = -3$.

It is quite helpful to recast the problem of aligning two sequences as an equivalent problem of finding a maximum-scoring path in the alignment graph as has been observed by a number of researchers. Recall that the alignment graph of $A$ and $B$ is a directed acyclic graph whose vertices are the pairs $(i,j)$ where $i \in \{0,1,2,\ldots,m\}$ and $j \in \{0,1,2,\ldots,n\}$. These vertices are arrayed in $m+1$ rows and $n+1$ columns. The edge set consists of three types of edges. The substitution aligned pairs, insertion aligned pairs, and deletion aligned pairs correspond to the diagonal edges, horizontal edges, and vertical edges, respectively. Specifically, a vertical edge from $(i-1, j)$ to $(i, j)$, which corresponds to a deletion of $a_i$, is drawn for $i \in \{1,2,\ldots,m\}$ and $j \in \{0,1,2,\ldots,n\}$. A horizontal edge from $(i, j-1)$ to $(i, j)$ , which corresponds to an insertion of $b_j$, is drawn for $i \in \{0,1,2,\ldots,m\}$ and $j \in \{1,2,\ldots,n\}$. A diagonal edge from $(i-1, j-1)$ to $(i, j)$ , which corresponds to a substitution of $a_i$ with $b_j$, is drawn for $i \in \{1,2,\ldots,m\}$ and $j \in \{1,2,\ldots,n\}$.

It has been shown that an alignment corresponds to a path from the leftmost cell of the first row to the rightmost cell of the last row in the alignment graph. Figure 3 gives another example of this correspondence.

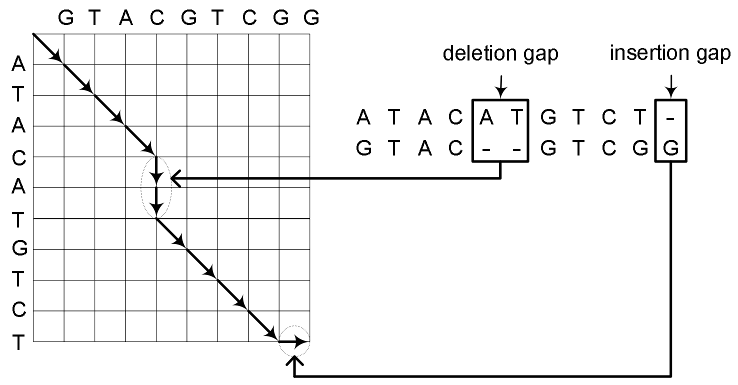

Figure 3: A path in an alignment graph of the two sequences `ATACATGTCT` and `GTACGTCGG`.

Let $S[i, j]$ denote the score of an optimal alignment between $a_1a_2\ldots a_i$, and $b_1b_2\ldots b_j$. By definition, we have $S[0,0] = 0$, $S[i,0] = -\beta \times i$, and $S[0, j] = -\beta \times j$. With these initializations, $S[i, j]$ for $i \in \{1,2,\ldots,m\}$ and $j \in \{1,2,\ldots,n\}$ can be computed by the following recurrence.

$$S[i,j] = \max \begin{cases} S[i-1,j]-\beta, \\ S[i,j-1]-\beta, \\ S[i-1,j-1]+\sigma(a_i,b_j). \end{cases}$$

Figure 4 explains the recurrence by showing that there are three possible ways entering into the grid point $(i, j)$, and we take the maximum of their path weights. The weight of the maximum-scoring path entering $(i, j)$ from $(i-1, j)$ *vertically* is the weight of the maximum-scoring path entering $(i-1,j)$ plus the weight of edge $(i-1,j) \to (i,j)$. That is, the weight of the maximum-scoring path entering $(i,j)$ with a deletion gap symbol at the end is $S[i-1,j]-\beta$. Similarly, the weight of the maximum-scoring path entering $(i,j)$ from $(i,j-1)$ *horizontally* is $S[i,j-1]-\beta$ and the weight of the maximum-scoring path entering $(i,j)$ from $(i-1,j-1)$ *diagonally* is $S[i-1,j-1]+\sigma(a_i,b_j)$. To compute $S[i,j]$, we simply take the maximum value of these three choices. The value $S[m,n]$ is the score of an optimal global alignment between sequences $A$ and $B$.

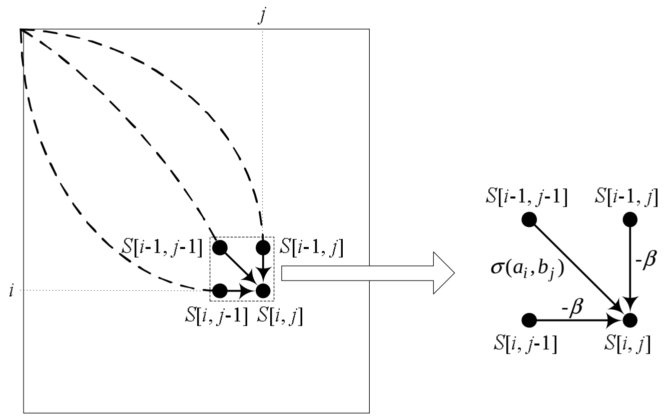

Figure 4: There are three ways entering the grid point $(i,j)$.

Figure 5 gives the pseudo-code for computing the score of an optimal global alignment. Whenever there is a tie, any one of them will work. Since there are $O(mn)$ entries and the time spent for each entry is $O(1)$, the total running time of algorithm GLOBAL_ALIGNMENT_SCORE is $O(mn)$.

Now let us use an example to illustrate the tabular computation. Figure 6 computes the score of an optimal alignment of the two sequences ATACATGTCT and GTACGTCGG, where a match is given a bonus score 8, a mismatch is penalized by a score $-5$, and the gap penalty for each gap symbol is $-3$. The first row and column of the table are initialized with proper penalties. Other entries are computed in order. Take the entry $(5,3)$ for example. Upon computing the value of this entry, the following values are ready: $S[4,2]=-3$, $S[4,3]=8$, and $S[5,2]=-6$. Since the edge weight of $(4,2) \to (5,3)$ is 8 (a match symbol "A"), the maximum score from $(4,2)$ to $(5,3)$ is $-3+8=5$. The maximum score from $(4,3)$ is $8-3=5$, and the maximum score from $(5,2)$ is $-6-3=-9$. Taking the maximum of them, we have $S[5,3]=5$. Once the table has been computed,

```
Algorithm GLOBAL_ALIGNMENT_SCORE($A = a_1 a_2 \ldots a_m$, $B = b_1 b_2 \ldots b_n$)
begin
    $S[0,0] \leftarrow 0$
    for $j \leftarrow 1$ to $n$ do $S[0,j] \leftarrow -\beta \times j$
    for $i \leftarrow 1$ to $m$ do
        $S[i,0] \leftarrow -\beta \times i$
        for $j \leftarrow 1$ to $n$ do
            $S[i,j] \leftarrow \max \begin{cases} S[i-1,j] - \beta \\ S[i,j-1] - \beta \\ S[i-1,j-1] + \sigma(a_i, b_j) \end{cases}$
    Output $S[m,n]$ as the score of an optimal alignment.
end
```

Figure 5: Computation of the score of an optimal global alignment.

the value in the rightmost cell of the last row, *i.e.*, $S[10,9] = 29$, is the score of an optimal global alignment.

In Section **??**, we have shown that if a backtracking information is saved for each entry while we compute the dynamic-programming matrix, an optimal solution can be derived following the backtracking pointers. Here we show that even if we don't save those backtracking pointers, we can still reconstruct an optimal solution by examining the values of an entry's possible contributors. Figure 7 lists the pseudo-code for delivering an optimal global alignment, where an initial call GLOBAL_ALIGNMENT_OUTPUT($A$, $B$, $S$, $m$, $n$) is made to deliver an optimal global alignment. Specifically, we trace back the dynamic-programming matrix from the entry $(m,n)$ recursively according to the following rules. Let $(i,j)$ be the entry under consideration. If $i = 0$ or $j = 0$, we simply output all the insertion pairs or deletion pairs in these boundary conditions. Otherwise, consider the following three cases. If $S[i,j] = S[i-1,j-1] + \sigma(a_i, b_j)$, we make a diagonal move and output a substitution pair $\begin{pmatrix} a_i \\ b_j \end{pmatrix}$. If $S[i,j] = S[i-1,j] - \beta$, then we make a vertical move and output a deletion pair $\begin{pmatrix} a_i \\ - \end{pmatrix}$. Otherwise, it must be the case where $S[i,j] = S[i,j-1] - \beta$. We simply make a horizontal move and output an insertion pair $\begin{pmatrix} - \\ b_j \end{pmatrix}$. Algorithm GLOBAL_ALIGNMENT_OUTPUT takes

|  |  | G | T | A | C | G | T | C | G | G |
|---|---|---|---|---|---|---|---|---|---|---|
|  | 0 | -3 | -6 | -9 | -12 | -15 | -18 | -21 | -24 | -27 |
| A | -3 | -5 | -8 | 2 | -1 | -4 | -7 | -10 | -13 | -16 |
| T | -6 | -8 | 3 | 0 | -3 | -6 | 4 | 1 | -2 | -5 |
| A | -9 | -11 | 0 | 11 | 8 | 5 | 2 | -1 | -4 | -7 |
| C | -12 | -14 | -3 | 8 | 19 | 16 | 13 | 10 | 7 | 4 |
| A | -15 | -17 | -6 | 5 | 16 | 14 | 11 | 8 | 5 | 2 |
| T | -18 | -20 | -9 | 2 | 13 | 11 | 22 | 19 | 16 | 13 |
| G | -21 | -10 | -12 | -1 | 10 | 21 | 19 | 17 | 27 | 24 |
| T | -24 | -13 | -2 | -4 | 7 | 18 | 29 | 26 | 24 | 22 |
| C | -27 | -16 | -5 | -7 | 4 | 15 | 26 | 37 | 34 | 31 |
| T | -30 | -19 | -8 | -10 | 1 | 12 | 23 | 34 | 32 | 29 |

Figure 6: The score of an optimal global alignment of the two sequences ATACATGTCT and GTACGTCGG, where a match is given a bonus score 8, a mismatch is penalized by a score $-5$, and the gap penalty for each gap symbol is $-3$.

$O(m+n)$ time in total since each recursive call reduces $i$ and/or $j$ by one. The total space complexity is $O(mn)$ since the size of the dynamic-programming matrix is $O(mn)$. In Section **??**, we shall show that an optimal global alignment can be recovered even if we don't save the whole matrix.

Figure 8 delivers an optimal global alignment by backtracking from the rightmost cell of the last row of the dynamic-programming matrix computed in Figure 6. We start from the entry $(10,9)$ where $S[10,9]=29$. We have a tie there because both $S[10,8]-3$ and $S[9,8]-5$ equal to 29. In this illustration, the horizontal move to the entry $(10,8)$ is chosen. Interested readers are encouraged to try the diagonal move to the entry $(9,8)$ for an alternative optimal global alignment, which is actually chosen by GLOBAL_ALIGNMENT_OUTPUT. Continue this process until the entry $(0,0)$ is reached. The shaded area depicts the backtracking path whose corresponding alignment is given on the right-hand side of the figure.

It should be noted that during the backtracking procedure, we derive the aligned pairs in a reverse order of the alignment. That's why we make a recursive call before actually printing out the pair in Figure 7. Another approach is to compute the

```
**Algorithm** GLOBAL_ALIGNMENT_OUTPUT($A = a_1a_2 \dots a_m$, $B = b_1b_2 \dots b_n$, $S$, $i$, $j$)
**begin**
    **if** $i = 0$ or $j = 0$ **then**
        **if** $i > 0$ **then for** $i' \leftarrow 1$ **to** $i$ **do print** $\begin{pmatrix} a_{i'} \\ - \end{pmatrix}$
        **if** $j > 0$ **then for** $j' \leftarrow 1$ **to** $j$ **do print** $\begin{pmatrix} - \\ b_{j'} \end{pmatrix}$
        **return**
    **if** $S[i,j] = S[i-1,j-1] + \sigma(a_i, b_j)$ **then**
        GLOBAL_ALIGNMENT_OUTPUT($A$, $B$, $S$, $i-1$, $j-1$)
        **print** $\begin{pmatrix} a_i \\ b_j \end{pmatrix}$
    **else if** $S[i,j] = S[i-1,j] - \beta$ **then**
        GLOBAL_ALIGNMENT_OUTPUT($A$, $B$, $S$, $i-1$, $j$)
        **print** $\begin{pmatrix} a_i \\ - \end{pmatrix}$
    **else**
        GLOBAL_ALIGNMENT_OUTPUT($A$, $B$, $S$, $i$, $j-1$)
        **print** $\begin{pmatrix} - \\ b_j \end{pmatrix}$
**end**
```

Figure 7: Backtracking procedure for delivering an optimal global alignment.

dynamic-programming matrix backward from the rightmost cell of the last row to the leftmost cell of the first row. Then when we trace back from the leftmost cell of the first row toward the rightmost cell of the last row, the aligned pairs are derived in the same order as in the alignment. This approach could avoid the overhead of reversing an alignment.

|   |   | G | T | A | C | G | T | C | G | G |
|---|---|---|---|---|---|---|---|---|---|---|
|   | **0** | -3 | -6 | -9 | -12 | -15 | -18 | -21 | -24 | -27 |
| A | -3 | **-5** | -8 | 2 | -1 | -4 | -7 | -10 | -13 | -16 |
| T | -6 | -8 | **3** | 0 | -3 | -6 | 4 | 1 | -2 | -5 |
| A | -9 | -11 | 0 | **11** | 8 | 5 | 2 | -1 | -4 | -7 |
| C | -12 | -14 | -3 | 8 | **19** | 16 | 13 | 10 | 7 | 4 |
| A | -15 | -17 | -6 | 5 | **16** | 14 | 11 | 8 | 5 | 2 |
| T | -18 | -20 | -9 | 2 | **13** | 11 | 22 | 19 | 16 | 13 |
| G | -21 | -10 | -12 | -1 | 10 | **21** | 19 | 17 | 27 | 24 |
| T | -24 | -13 | -2 | -4 | 7 | 18 | **29** | 26 | 24 | 22 |
| C | -27 | -16 | -5 | -7 | 4 | 15 | 26 | **37** | 34 | 31 |
| T | -30 | -19 | -8 | -10 | 1 | 12 | 23 | 34 | **32** | **29** |

|   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | T | A | C | A | T | G | T | C | T | - |   |
| G | T | A | C | - | - | G | T | C | G | G |   |
| -5 | +8 | +8 | +8 | -3 | -3 | +8 | +8 | +8 | -5 | -3 | = $\boxed{29}$ |

Figure 8: Computation of an optimal global alignment of sequences `ATACATGTCT` and `GTACGTCGG`, where a match is given a bonus score 8, a mismatch is penalized by a score $-5$, and the gap penalty for each gap symbol is $-3$.