A Class Note on Space-Saving Strategies

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

November 5, 2008

In this note, we shall briefly discuss how to modify the dynamic programming methods to copy with three scoring schemes that are frequently used in biological sequence analysis.

1 Hirschberg's approach

Straightforward implementation of the dynamic-programming algorithms utilizes quadratic space to produce an optimal global or local alignment. For analysis of long DNA sequences, this space restriction is more crucial than the time constraint. Because of this, different methods have been proposed to reduce the space used for aligning globally or locally two sequences.

We first describe a space-saving strategy proposed by Hirschberg in 1975. It uses only "linear space," *i.e.*, space proportional to the sum of the sequences' lengths. The original formulation was for the longest common subsequence problem. But the basic idea is quite robust and works readily for aligning globally two sequences with affine gap costs as shown by Myers and Miller in 1988. Remarkably, this space-saving strategy has the same time complexity as the original dynamic-programming method.

To introduce Hirschberg's approach, let us first review the original algorithm for aligning two sequences of lengths m and n. It is apparent that the scores in row

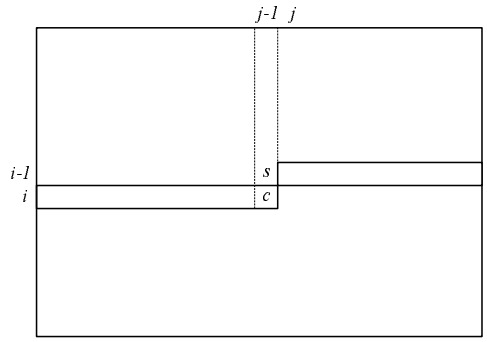

Figure 1: Entry locations of S^- just before the entry value is evaluated at (i, j).

i of dynamic programming matrix S are calculated from those in row $i-1$. Thus, after the scores in row i of S are calculated, the entries in row $i-1$ of S will no longer be used and hence the space used for storing these entries can be recycled to calculate and store the entries in row $i+1$. In other words, we can get by with space for two rows, since all that we ultimately want is the single entry $S[m,n]$ in the rightmost cell of the last row.

In fact, a single array S^- of size n, together with two extra variables, is adequate. $S^-[j]$ holds the most recently computed value for each $1 \leq j \leq n$, so that as soon as the value of the jth entry of S^- is computed, the old value at the entry is overwrited. There is a slight conflict in this strategy since we need the old value of an entry to compute a new value of the entry. To avoid this conflict, two additional variables, say s and c, are introduced to hold the new and old values of the entry, respectively. Figure 1 shows the locations of the scores kept in S^- and in variables s and c. When $S^-[j]$ is updated, $S^-[j']$ holds the score in the entry (i, j') in row i for each $j' < j$, and it holds the score in the entry $(i-1, j')$ for any $j' \geq j$. Figure 2 gives the pseudo-code for computing the score of an optimal global alignment in linear space.

In the dynamic programming matrix S of aligning sequences $A = a_1 a_2 \ldots a_m$ and $B = b_1 b_2 \ldots b_n$, $S[i, j]$ denotes the optimal score of aligning $a_1 a_2 \ldots a_i$ and $b_1 b_2 \ldots b_j$ or, equivalently, the maximum score of a path from $(0,0)$ to the cell (i, j) in the alignment graph. By symmetry, the optimal score of aligning $a_{i+1} a_{i+2} \ldots a_m$ and $b_{j+1} b_{j+2} \ldots b_n$ or the maximum score of a path from (i, j) to (m, n) in the alignment graph can be calculated in linear space in a backward manner. Figure 3 gives the pseudo-code for computing the score of an optimal global alignment in a backward manner in linear space.

Algorithm FORWARD_SCORE($A = a_1a_2\ldots a_m$, $B = b_1b_2\ldots b_n$)
begin
  $S^-[0] \leftarrow 0$
  for $j \leftarrow 1$ **to** n **do** $S^-[j] \leftarrow S^-[j-1] - \beta$
  for $i \leftarrow 1$ **to** m **do**
    $s \leftarrow S^-[0]$
    $c \leftarrow S^-[0] - \beta$
    $S^-[0] \leftarrow c$
    for $j \leftarrow 1$ **to** n **do**

$$c \leftarrow \max \begin{cases} S^-[j] - \beta \\ c - \beta \\ s + \sigma(a_i, b_j) \end{cases}$$

      $s \leftarrow S^-[j]$
      $S^-[j] \leftarrow c$
  Output $S^-[n]$ as the score of an optimal alignment.
end

Figure 2: Computation of the optimal score of aligning sequences of lengths m and n in linear space $O(n)$.

In what follows, we use $S^-[i,j]$ and $S^+[i,j]$ to denote the maximum score of a path from $(0,0)$ to (i,j) and that from (i,j) to (m,n) in the alignment graph, respectively. Without loss of generality, we assume that m is a power of 2. Obviously, for each j, $S^-[m/2,j] + S^+[m/2,j]$ is the maximum score of a path from $(0,0)$ to (m,n) through $(m/2,j)$ in the alignment graph. Choose j_{mid} such that

$$S^-[m/2, j_{mid}] + S^+[m/2, j_{mid}] = \max_{1 \le j \le n} S^-[m/2, j] + S^+[m/2, j].$$

Then, $S^-[m/2, j_{mid}] + S^+[m/2, j_{mid}]$ is the optimal alignment score of A and B and there is a path having such a score from $(0,0)$ to (m,n) through $(m/2, j_{mid})$ in the alignment graph.

Hirschberg's linear-space approach is first to compute $S^-[m/2,j]$ for $1 \le j \le n$ by a forward pass, stopping at row $m/2$ and to compute $S^+[m/2,j]$ for $1 \le j \le n$ by a backward pass and then to find j_{mid}. After j_{mid} is found, recursively compute an optimal path from $(0,0)$ to $(m/2, j_{mid})$ and an optimal path from $(m/2, j_{mid})$ to (m,n).

As the problem is partitioned further, there is a need to have an algorithm that is capable of delivering an optimal path for any specified two ends. In Figure 4, algorithm LINEAR_ALIGN is a recursive procedure that delivers a maximum-scoring path from (i_1, j_1) to (i_2, j_2). To deliver the whole optimal alignment, the two ends are initially specified as $(0,0)$ and (m,n).

Now let us analyze the time and space taken by Hirschberg's approach. Using the algorithms given in Figures 2 and 3, both the forward and backward pass take $O(nm/2)$-time and $O(n)$-spaces. Hence, it takes $O(mn)$-time and $O(n)$-spaces to find j_{mid}. Set $T = mn$ and call it the size of the problem of aligning A and B. At each recursive step, a problem is divided into two subproblems. However, regardless of where the optimal path crosses the middle row $m/2$, the total size of the two resulting subproblems is exactly half the size of the problem that we have at the recursive step (see Figure 5). It follows that the total size of all problems, at all levels of recursion, is at most $T + T/2 + T/4 + \cdots = 2T$. Because computation time is directly proportional to the problem size, Hirschberg's approach will deliver an optimal alignment using $O(2T) = O(T)$ time. In other words, it yields an $O(mn)$-time, $O(n)$-space global alignment algorithm.

Hirschberg's original method, and the above discussion, apply to the case where the penalty for a gap is merely proportional to the gap's length, i.e., $k \times \beta$ for a k-symbol gap. For applications in molecular biology, one wants penalties of the form $\alpha + k \times \beta$, i.e., each gap is assessed an additional "gap-open" penalty α. Actually, one can be slightly more general and substitute residue-dependent penalties for β. In our previous note, we have shown that the relevant alignment graph is more complicated. Now at each grid point (i, j) there are three nodes, denoted $(i, j)_S$, $(i, j)_D$, and $(i, j)_I$, and generally seven entering edges. The alignment problem is to compute a highest-score path from $(0,0)_S$ to $(m,n)_S$. Fortunately, Hirschberg's strategy extends readily to this more general class of alignment scores. In essence, the main additional complication is that for each defining corner of a subproblem, we need to specify one of the grid point's three nodes.

Another issue is how to deliver an optimal *local* alignment in linear space. Recall that in the local alignment problem, one seeks a highest-scoring alignment where the end nodes can be arbitrary, *i.e.*, they are not restricted to $(0,0)_S$ and $(m,n)_S$. In fact, it can be reduced to a global alignment problem by performing a linear-space score-only pass over the dynamic-programming matrix to locate the first and last nodes of an optimal local alignment, then delivering a global alignment between these two nodes by applying Hirschberg's approach.

Algorithm BACKWARD_SCORE($A = a_1a_2\ldots a_m$, $B = b_1b_2\ldots b_n$)
begin
  $S^+[n] \leftarrow 0$
  for $j \leftarrow n-1$ **down to** 0 **do** $S^+[j] \leftarrow S^+[j+1] - \beta$
  for $i \leftarrow m-1$ **down to** 0 **do**
    $s \leftarrow S^+[n]$
    $c \leftarrow S^+[n] - \beta$
    $S^+[n] \leftarrow c$
    for $j \leftarrow n-1$ **down to** 0 **do**
      $$c \leftarrow \max \begin{cases} S^+[j] - \beta \\ c - \beta \\ s + \sigma(a_{i+1}, b_{j+1}) \end{cases}$$
      $s \leftarrow S^+[j]$
      $S^+[j] \leftarrow c$
  Output $S^+[0]$ as the score of an optimal alignment.
end

Figure 3: Backward computation of the score of an optimal global alignment in linear space.

2 Constrained Sequence Alignment

Rigorous sequence alignment algorithms compare each residue of one sequence to every residue of the other. This requires computational time proportional to the product of the lengths of the given sequences. Biologically relevant sequence alignments, however, usually extend from the beginning of both sequences to the end of both sequences, and thus the rigorous approach is unnecessarily time consuming; significant sequence similarities are rarely found by aligning the end of one sequence with the beginning of the other.

As a result of the biological constraint, it is frequently possible to calculate an optimal alignment between two sequences by considering only those residues that are within a diagonal band in which each row has only w cells. With sequences $A = a_1a_2\ldots a_m$ and $B = b_1b_2\ldots b_n$, one can specify constants $\ell \leq u$ such that aligning a_i with b_j is permitted only if $\ell \leq j - i \leq u$. For example, it rarely takes a dozen insertions or deletions to align any two members of the globin superfamily; thus, an optimal alignment of two globin sequences can be calculated in $O(nw)$

time that is identical to the rigorous alignment that requires $O(nm)$ time.

Alignment within a band is used in the final stage of the FASTA program for rapid searching of protein and DNA sequence databases (Pearson and Lipman, 1988; Pearson, 1990). For optimization in a band, the requirement to "start at the beginning, end at the end" is reflected in the $\ell \leq \min\{0, n-m\}$ and $u \geq \max\{0, n-m\}$ constraints. "Local" sequence alignments do not require that the beginning and end of the alignment correspond to the beginning and end of the sequence, *i.e.*, the aligned sequences can be arbitrary substrings of the given sequences, A and B; they simply require that the alignment have the highest similarity score. For a "local" alignment in a band, it is natural to relax the requirement to $\ell \leq u$. Algorithms for computing an optimal local alignment can utilize a global alignment procedure to perform subcomputations: once locally optimal substrings A' of A and B' of B are found, which can be done by any of several available methods, a global alignment procedure is called to align A' and B'. Appropriate values of ℓ' and u' for the global problem are inferred from the ℓ and u of the local problems. In other situations, a method to find unconstrained local alignments, *i.e.*, without band limits, might determine appropriate values of ℓ and u before invoking a global alignment procedure within a band.

Although the application of rigorous alignment algorithms to long sequences can be quite time-consuming, it is often the space requirement that is limiting in practice. Hirschberg's approach can be easily modified to find a solution locating in a band. Unfortunately, the resulting time required to produce the alignment can exceed that of the score-only calculation by a substantial factor. If T denotes the number of entries in the band of the dynamic programming matrix, then $T = O(nw)$. Producing an alignment involves computing as many as $T \times \log_2 n$ entries (including recomputations of entries evaluated at earlier steps). Thus, the time to deliver an alignment exceeds that for computing its score in a band by a log factor.

To avoid the log factor, we need a new way to subdivide the problem that limits the subproblems to some fraction, $\alpha < 1$, of the band. Figure 6 illustrates the idea. The score-only backward pass is augmented so that at each point it computes the next place where an optimal path crosses the mid-diagonal, *i.e.*, diagonal $(\ell + u)/2$. Using only linear space, we can save this information at every point on the "current row" or on the mid-diagonal. When this pass is completed, we can use the retained information to find the sequence of points where an optimal solution crosses the mid-diagonal, which splits the problem into some number of subproblems. The total area of these subproblems is no more than half of the original area for a narrow band with widely spaced crossing points; in other cases it is even less.

It should be noted that this band-aligning algorithm could be considered as a generalization of Hirschberg's approach by rotating the matrix partition line. The idea of partition line rotation has been exploited in devising parallel sequence comparison algorithms. Nevertheless, the dividing technique proposed in this section, which produces more than two subproblems, reveals a new paradigm for space-saving strategies.

Another extension is to consider the situations where the ith entry of the first sequence can be aligned to the jth entry of the second sequence only if $L[i] \leq j \leq U[i]$, for given left and right bounds L and U. As in the band alignment problem, we can apply the idea of defining a midpoint *partition line* that bisects the region into two nearly equal parts. Here we introduce a more general approach that can be easily utilized by other relevant problems.

Given a narrow region R with two boundary lines L and U, we can proceed as follows. We assume that L and U are non-decreasing since if, e.g., $L[i]$ were larger than $L[i+1]$, we could set $L[i+1]$ to equal $L[i]$ without affecting the set of constrained alignments. Enclose as many rows as possible from the top of the region in an upright rectangle, subject to the condition that the rectangle's area at most doubles the area of its intersection with R. Then starting with the first row of R not in the rectangle, we cover additional rows of R with a second such rectangle, and so on.

A score-only backward pass is made over R, computing S^+. Values of S^+ are retained for the top line in every rectangle (the top rectangle can be skipped). It can be shown that the total length of these pieces cannot exceed three times the total number of columns, as required for a linear space bound. Next, perform a score-only forward pass, stopping at the last row in the first rectangle. A sweep along the boundary between the first and second rectangles locates a crossing edge on an optimal path through R. That is, we can find a point p on the last row of the first rectangle and a point q on the first row of the second rectangle such that there is a vertical or diagonal edge e from p to q, and e is on an optimal path. Such an optimal path can be found by applying Hirschberg's strategy to R's intersection with the first rectangle (omitting columns following p) and recursively computing a path from q through the remainder of R. This process inspects a grid point at most once during the backward pass, once in a forward pass computing p and q, and an average of four times for applying Hirschberg's method to R's intersection with a rectangle.

3 Suboptimal Alignment

Molecular biology is rapidly becoming a data-rich science with extensive computational needs. More and more computer scientists are working together on developing efficient software tools for molecular biologists. One major area of potential interaction between computer scientists and molecular biologists arises from the need for analyzing biological information. In particular, optimal alignments mentioned in previous sections have been used to reveal similarities among biological sequences, to study gene regulation, and even to infer evolutionary trees.

However, biologically significant alignments are not necessarily mathematically optimized. It has been shown that sometimes the neighborhood of an optimal alignment reveals additional interesting biological features. Besides, the most strongly conserved regions can be effectively located by inspecting the range of variation of suboptimal alignments. Although rigorous statistical analysis for the mean and variance of optimal global alignment scores is not yet available, suboptimal alignments have been successfully used to informally estimate the significance of an optimal alignment.

For most applications, it is impractical to enumerate all suboptimal alignments since the number could be enormous. Therefore, a more compact representation of all suboptimal alignments is indispensable. A 0-1 matrix can be used to indicate if a pair of positions is in some suboptimal alignment or not. However, this approach misses some connectivity information among those pairs of positions. An alternative is to use a set of "canonical" suboptimal alignments to represent all suboptimal alignments. The kernel of that representation is a minimal directed acyclic graph (DAG) containing all suboptimal alignments.

Suppose we are given a threshold score that does not exceed the optimal alignment score. An alignment is suboptimal if its score is at least as large as the threshold score. Here we briefly describe a linear-space method that finds all edges that are contained in at least one path whose score exceeds a given threshold τ. Again, a recursive subproblem will consist of applying the alignment algorithm over a rectangular portion of the original dynamic-programming matrix, but now it is necessary that we continue to work with values S^- and S^+ that are defined relative to the original problem. To accomplish this, each problem to be solved is defined by specifying values of S^- for nodes on the upper and left borders of the defining rectangle, and values of S^+ for the lower and right borders.

To divide a problem of this form, a forward pass propagates values of S^- to nodes in the middle row and the middle column, and a backward pass propagates values S^+ to those nodes. This information allows us to determine all edges start-

ing in the middle row or middle column that are contained in a path of score at least τ. The data determining any one of the four subproblems, *i.e.*, the arrays of S values on its borders, is then at most half the size of the set of data defining the parent problem. The maximum total space requirement is realized when recursion reaches a directly solvable problem where there is only the leftmost cell of the first row of the original grid left; at that time there are essentially $2(m+n)$ S-values saved for borders of the original problem, $m+n$ values on the middle row and column of the original problem, $(m+n)/2$ values for the upper left subproblem, $(m+n)/4$ values for the upper-left-most subsubproblem, etc., giving a total of about $4(m+n)$ retained S-values.

4 Robustness Measurement

The utility of information about the reliability of different regions within an alignment is widely appreciated. One approach to obtaining such information is to determine suboptimal alignments, *i.e.*, some or all alignments that come within a specified tolerance of the optimum score, as discussed in Section 3. However, the number of suboptimal alignments, or even alternative optimal alignments, can easily be so large as to preclude an exhaustive enumeration.

Sequence conservation has proved to be a reliable indicator of at least one class of regulatory elements. Specifically, regions of six or more consecutive nucleotides that identical across a range of mammalian sequences, called "phylogenetic footprints," frequently correspond to binding sits for sequence-specific nuclear proteins. It is also interesting to look for longer, imperfectly conserved (but stronger matching) regions, which may indicate other sorts of regulatory elements, such as a region that binds to a nuclear matrix or assumes some altered chromatin structure.

In the following, we briefly describe some interesting measurements of the robustness of each aligned pair of a pairwise alignment. The first method computes, for each position i of the first sequence, the lower and upper limits of the positions in the second sequence to which it can be aligned and still come within a specified tolerance of the optimum alignment score. Delimiting suboptimal alignments this way, rather than enumerating all of them, allows the computation to run in only a small constant factor more time than the computation of a single optimal alignment.

Another method determines, for each aligned pair of an optimal alignment, the amount by which the optimum score must be lowered before reaching an align-

ment not containing that pair. In other words, if the optimum alignment score is s and the aligned pair is assigned the robustness-measuring number r, then any alignment scoring strictly greater than $s-r$ aligns those two sequence positions, whereas some alignment of score $s-r$ does not align them. As a special case, this value tells whether the pair is in all optimal alignments (namely, the pair is in all optimal alignments if and only if its associated value is non-zero). These computations are performed using dynamic-programming methods that require only space proportional to the sum of the two sequence lengths. It has also been shown on how to efficiently handle the case where alignments are constrained so that each position, say position i, of the first sequence can be aligned only to positions on a certain range of the second sequence.

To deliver an optimal alignment, Hirschberg's approach applies forward and backward passes in the first nondegenerate rectangle along the optimal path being generated. Within a subproblem (i.e., rectangle) the scores of paths can be taken relative to the "start node" at the rectangle's upper left and the "end node" at the rightmost cell of the last row. This means that a subproblem is completely specified by giving the coordinates of those two nodes. In contrast, methods for the robustness measurement must maintain more information about each pending subproblem. Fortunately, it can be done in linear space by observing that the total number of the boundary entries of all pending subproblems of Hirschberg's approach is bounded by $O(m+n)$ (see Figure 7).

Algorithm LINEAR_ALIGN($A = a_1 a_2 \ldots a_m$, $B = b_1 b_2 \ldots b_n$, i_1, j_1, i_2, j_2)
begin
  if $i_1 + 1 \geq i_2$ **or** $j_1 + 1 \geq j_2$ **then**
    Output the aligned pairs for the maximum-score path from (i_1, j_1) to (i_2, j_2)
  else
    $i_{mid} \leftarrow \lfloor (i_1 + i_2)/2 \rfloor$
    // Find the maximum scores from (i_1, j_1)
    $S^-[j_1] \leftarrow 0$
    for $j \leftarrow j_1 + 1$ **to** j_2 **do** $S^-[j] \leftarrow S^-[j-1] - \beta$
    for $i \leftarrow i_1 + 1$ **to** i_{mid} **do**
      $s \leftarrow S^-[j_1]$
      $c \leftarrow S^-[j_1] - \beta$
      $S^-[j_1] \leftarrow c$
      for $j \leftarrow j_1 + 1$ **to** j_2 **do**
        $$c \leftarrow \max \begin{cases} S^-[j] - \beta \\ c - \beta \\ s + \sigma(a_i, b_j) \end{cases}$$
        $s \leftarrow S^-[j]$
        $S^-[j] \leftarrow c$
    // Find the maximum scores to (i_2, j_2)
    $S^+[j_2] \leftarrow 0$
    for $j \leftarrow j_2 - 1$ **down to** j_1 **do** $S^+[j] \leftarrow S^+[j+1] - \beta$
    for $i \leftarrow i_2 - 1$ **down to** i_{mid} **do**
      $s \leftarrow S^+[j_2]$
      $c \leftarrow S^+[j_2] - \beta$
      $S^+[j_2] \leftarrow c$
      for $j \leftarrow j_2 - 1$ **down to** j_1 **do**
        $$c \leftarrow \max \begin{cases} S^+[j] - \beta \\ c - \beta \\ s + \sigma(a_{i+1}, b_{j+1}) \end{cases}$$
        $s \leftarrow S^+[j]$
        $S^+[j] \leftarrow c$
    // Find where maximum-score path crosses row i_{mid}
    $j_{mid} \leftarrow$ value $j \in [j_1, j_2]$ that maximizes $S^-[j] + S^+[j]$
    LINEAR_ALIGN(A,B, i_1, j_1, i_{mid}, j_{mid})
    LINEAR_ALIGN(A,B, i_{mid}, j_{mid}, i_2, j_2)
end

Figure 4: Computation of an optimal global alignment in linear space.

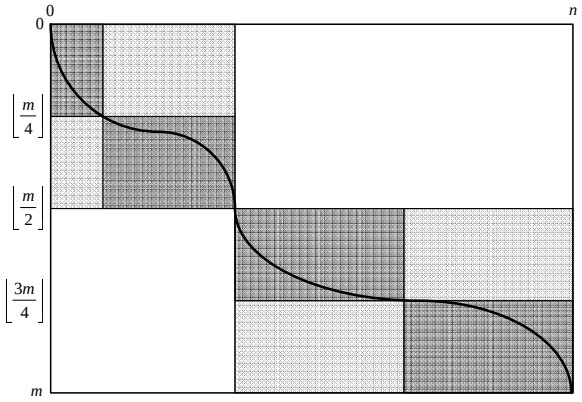

Figure 5: Hirschberg's linear-space approach.

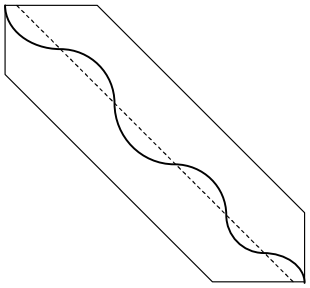

Figure 6: Dividing a band by its middle diagonal.

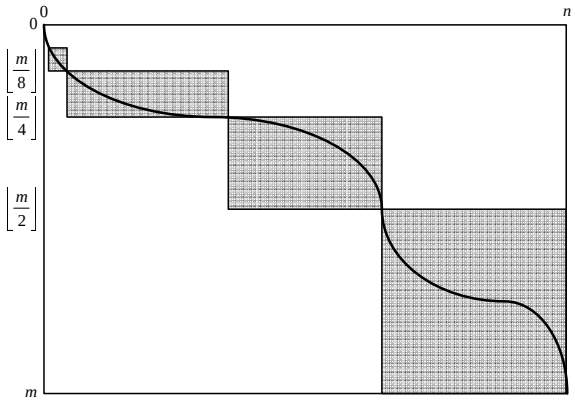

Figure 7: The total number of the boundary entries in the active subproblems is $O(m+n)$.