Volume 12 Number 1 1984 Nucleic Acids Research

Fast optimal alignment

James W.Fickett

Theoretical Division, Los Alamos National Laboratory, University of California, Los Alamos, NM 87545, USA

Received 15 August 1983

ABSTRACT

We show how to speed up sequence alignment algorithms of the type introduced by Needleman and Wunsch (and generalized by Sellers and others). Faster alignment algorithms have been introduced, but always at the cost of possibly getting sub-optimal alignments. Our modification results in the optimal alignment still being found, often in 1/10 the usual time. What we do is reorder the computation of the usual alignment matrix so that the optimal alignment is ordinarily found when only a small fraction of the matrix is filled. The number of matrix elements which have to be computed is related to the distance between the sequences being aligned; the better the optimal alignment, the faster the algorithm runs.

INTRODUCTION

DEFINITION OF TERMS. The type of algorithm we consider was introduced by S.B.Needleman and C.D.Wunsch (1). Since then many variations have been introduced; see the review by J.B.Kruskal (2). The basic idea of this family of algorithms is very simple. For any possible alignment of two sequences we assign a distance corresponding to that alignment: it is the number of gaps in the alignment multiplied by a number called the gap penalty, plus the number of mismatches in the alignment multiplied by a number called the mismatch penalty. (The number of gaps in an alignment is the number of elements of either sequence not aligned with any element of the other.) Among all alignments a given one is optimal if its corresponding distance is as small as possible. The distance between the two sequences is then that distance which corresponds to the optimal alignment. The idea of Needleman and Wunsch was to find the optimal alignment recursively, by aligning successively longer initial segments of the two sequences.

Let the two sequences to align be $a_1,\ldots,a_m$ and $b_1,\ldots,b_n$. Let d_{ij} be the distance corresponding to the optimal alignment of $a_1,\ldots,a_i$ with $b_1,\ldots,b_j$, where i runs from 0 to m and j runs from 0 to n. Then with initial values

$$d_{i,0} = i*g \qquad d_{0,j} = j*g$$

d_{ij} is defined recursively by

$$(1) \qquad d_{i,j} = \min \begin{cases} d_{i-1,j} + g \\ d_{i-1,j-1} + x \\ d_{i,j-1} + g \end{cases},$$

where g is the gap penalty and x is either 0, if $a_i = b_j$, or the mismatch penalty, if $a_i \neq b_j$. Intuitively this says the following. One of three things can happen in the optimal alignment of $a_1,\ldots,a_i$ with $b_1,\ldots,b_j$. The first possibility (first line of the equation) is that b_j could be aligned with some a_k with $k < i$. Then the distance of $a_1,\ldots,a_i$ to $b_1,\ldots,b_j$ is the same as the distance from $a_1,\ldots,a_{i-1}$ to $b_1,\ldots,b_j$ plus the cost of the gap across from a_i. The second line of the equation corresponds to the case where a_i is aligned with b_j. The distance here is the same as the distance from $a_1,\ldots,a_{i-1}$ to $b_1,\ldots,b_{j-1}$, plus whatever the alignment of a_i with b_j adds. The third line of the equation corresponds to the case where a_i is aligned with some b_k where $k < j$.

In the usual implementation of this algorithm one calculates all d_{ij} by initializing $d_{0,j}$ and $d_{i,0}$, and then filling the rest of the matrix row by row, left to right. However it is possible to fill the matrix in many different orders, the only restriction being that the calculation of any given d_{ij} depends on already having the values of the three elements up and to the left of it.

While calculating the distance matrix (with entries d_{ij}) one also calculates a path matrix, which records for each d_{ij} which line of equation (1) was used in its evaluation. When these two matrices are done one looks on the lower and right hand boundaries of the distance matrix to find the smallest d_{ij} satisfying i=m or j=n. This gives the distance between the sequences, and moving back from this point to (0,0) through the path matrix gives the alignment.

This basic algorithm has been generalized in many ways, such as for example allowing the gap penalty to depend on whether a gap is isolated or flanked by another gap. We will discuss our improvement solely in terms of the basic algorithm in order to keep the discussion simple. It will be clear that the ideas we present apply to most generalizations, as long as d_{ij} is a positive function.

NEED FOR IMPROVEMENTS. The type of algorithm specified above will always find the optimal alignment of two sequences. However in the process of doing

so it considers every possible alignment of the two sequences and calculates a distance for each. Not surprisingly it takes a long time, and even when programmed carefully can be prohibitively expensive if done often or on long sequences. Several authors have proposed faster algorithms which give good, though not necessarily optimal alignments. The most practical is probably that of W.J.Wilbur and D.J.Lipman (3,4).

Thus at present one must choose between an algorithm which gives the best alignment but is expensive, and an algorithm which is fast but may not give the best alignment. In this paper we narrow the gap between these choices by showing how to get the optimal alignment much more quickly than before.

MAIN IDEA OF THIS IMPROVEMENT. If, as in the usual implementation of Needleman-Wunsch type algorithms, all m*n elements d_{ij} are calculated, one is considering every possible alignment of the two sequences before the best alignment is identified. This is wasteful, especially if the best alignment is relatively good. The optimal alignment of the two sequences is built from alignments of initial subsequences. But the only alignments of subsequences which are relevant are ones at least as good (distance at least as small) as the overall one. I.e. one really only needs those d_{ij} which are below a fixed bound. We show how to reorder the computation of the distance matrix so that all d_{ij} less than or equal to a given bound are calculated while most d_{ij} greater than that bound are not.

USE. There are three ways to put this idea to practical use. One is to use one of the fast-but-suboptimal alignment algorithms to get a good upper bound D on the distance between two sequences, and then calculate all $d_{ij} \leq D$ to find the optimal alignment. In many cases one will have a good a priori notion of the distance between the sequences, or will only be interested in the optimal alignment if the distance is relatively small. So a second possibility is to use an a priori bound D. Third, one can fill as much of the distance matrix as needed to get the optimal alignment, but in an order which avoids wasted computation. Namely, choose an initial D_0, and compute all $d_{ij} \leq D_0$. If this fails to find an alignment, choose $D_1 > D_0$ and fill in more of the matrix to get all $d_{ij} \leq D_1$. Continue until an alignment is found.

DESCRIPTION OF THE ALGORITHM

FILLING PART OF THE MATRIX. As above, assume we are aligning two sequences $a_1,\ldots,a_m$ and $b_1,\ldots,b_n$, and let d_{ij} be the distance between $a_1,\ldots,a_i$ and $b_1,\ldots,b_j$. We first need to show how to calculate all the matrix entries d_{ij} satisfying $d_{ij} \leq D$ for some arbitrary bound D, without

having to calculate all the $d_{ij} > D$. The calculation is essentially as usual, the only difference being in the order in which the entries are made. We work row by row. Calculate first $d_{1,1},\ldots,d_{1,l_1}$, where l_1 is minimal with $d_{1,l_1} \geq D$. Note that $d_{1,j} > D$ for $j > l_1$. Next calculate $d_{2,1},\ldots,d_{2,l_2}$, where l_2 is minimal such that $d_{2,l_2} \geq D$ and $l_2 > l_1$. Thus we can be sure that $d_{2,j} > D$ for $j > l_2$. We need to note two things here. First, as long as $d_{2,j} \leq D$ its value can be calculated exactly; if $j > l_1$ its value will depend only on the value of $d_{2,j-1}$. Second, for those $d_{2,j} > D$, we may not be able to calculate the exact value, but only the fact that it is greater than D. We can use a special symbol to record this fact.

Continue calculating one partial row at a time in this way. Eventually a row will start with a value > D. From then on the initial segments, as well as the terminal segments of the rows can be skipped. In general, if the elements $d_{i,k_i},\ldots,d_{i,l_i}$ of the i^{th} row have been calculated, then we start the $i+1^{st}$ row by calculating $d_{i+1,k_{i+1}}$, where k_{i+1} is minimal with $d_{i,k_{i+1}} < D$. In this way we calculate all $d_{ij} \leq D$, and possibly a few (on the end of each row) > D.

FINDING AN ALIGNMENT. If one has a bound D to start with, so that one is only interested in alignments with distance no greater than D, the programming modifications to the usual algorithm are trivial. Just calculate those entries d_{ij} and p_{ij} of the distance and path matrices where i and j satisfy $d_{ij} \leq D$. This will include all the path matrix entries necessary to find any alignment with distance no greater than D. If one has no bound D to work with, but plans to fill as much of the matrices as necessary, things are slightly more complicated. We pick an initial bound D_0 and calculate all d_{ij} and p_{ij} with $d_{ij} \leq D_0$. Say that in the i^{th} row we end up calculating $d_{i,k_i},\ldots,d_{i,l_i}$. In order to be know where to start in filling more of the matrix later we save k_i and l_i. If it turns out that there is no alignment with distance $\leq D_0$ we choose a larger bound D_1. Then we just go through and calculate row by row, left to right, using the k_i and l_i to skip over the values already calculated.

ANALYSIS OF TIME SAVED

Now we give an estimate of the computation saved by using this method. Say we are comparing two sequences a distance d apart, and we do so by computing all $d_{ij} \leq D$, a bound greater than, but not too much greater than d.

We now show that all the d_{ij} we compute are in a strip centered on the main diagonal of the distance matrix, defined by $|i-j| < D/g + 1$. The main

fact we need to note is that any d_{ij} with $|i-j| \geq k$ corresponds to an alignment with at least k gaps, and hence is $\geq k*g$. So if $i-j \geq D/g$ $d_{ij} \geq D$. We will show that no d_{ij} with $|i-j| \geq d/g + 1$ is calculated.

If $i-j \geq D/g + 1$ then we have $(i-1)-(j-k) \geq D/g$ for $k \geq 0$, so that d_{ij} is before the beginning of the elements of the i^{th} row which are calculated. If $j-i \geq D/g + 1$ then we have $(j-1)-i \geq D/g$ (so that $d_{i,j-1}$ is at least D) and for $k \leq 2$ $(j-k)-(i-1) \geq D/g$ (so that $d_{i-1,j-k}$ is at least D for $k \leq 2$). So d_{ij} is after all the elements of the i^{th} row which are calculated.

Thus we will calculate at most $2*\min(m,n)*(D/g + 1)$ matrix entries. This estimator of the number of matrix entries which have to be computed is proportional to the length of the shorter sequence times the distance between the sequences. The number total number of matrix entries is of course the product of the length of the two sequences.

We have only used the gap penalty in estimating how much the computation is pruned. In reality of course, the mismatch penalty will result in further pruning. Yet even with this simple estimate, we see impressive savings. Suppose that we are aligning two sequences of equal length, with gap penalty at least as large as mismatch penalty. Then our estimate of the number of entries which need to be computed is about $2*n*d/g$. d is less than g*(# of gaps plus mismatches), so 2*n*(# of gaps plus mismatches) is probably considerably greater than the number of elements to be computed. If the number of gaps and mismatches is 5% of n, we will almost certainly end up filling less than 10% of the matrices.

When aligning nucleic acid sequences one expects to always find some alignment with distance considerably less than the maximum possible, because one out of four bases will match just by chance. Thus in this case one is essentially always able to avoid filling a significant fraction of the matrix.

ACKNOWLEDGEMENTS

I would like to thank J.Wilbur, W.Goad and M.Kanehisa for helpful comments. This work was done under the auspices of the Department of Energy, and was begun at the Aspen Center for Physics.

REFERENCES

1. Needleman,S.B. and Wunsch,C.D. (1970) J.Mol.Biol 48, 443-453.
2. Kruskal,J.B. (1983) Siam Review 25, 201-237.
3. Wilbur,W.J. and Lipman,D.J. (1983) Proc. Natl. Acad. Sci. USA 80, 726-730.
4. Wilbur,W.J. and Lipman,D.J. (1983) SIAM J. Appl. Math. (to appear)