J. theor. Biol. (1994) **167**, 169–174

An Algorithm Combining DNA and Protein Alignment

JOTUN HEIN

Institute of Biological Sciences, Aarhus University, DK-8000 Aarhus, Denmark

(*Received on* 6 *April* 1993, *Accepted in revised form on* 31 *August* 1993)

An algorithm is presented that aligns two DNA sequences minimizing the overall amount of evolution that the associated proteins have experienced. It is generalized to minimizing a weighted average of protein and DNA evolution.

1. Introduction

As protein evolves slower than its coding DNA it will be more reliable to align the protein than the t'underlying DNA and an algorithm that compares coding DNA should incorporate the information from the protein. Presently, this problem is solved by aligning the sequence at the protein level and then backtranslating to DNA. This approach falls short on three accounts. First, it forces all insertions/deletions (abbreviated as indels) to occur at codon boundaries. Second, it ignores homologies at the DNA level. Third, it is not possible to backtranslate in cases where there are overlapping reading frames.

The result of a mixed alignment algorithm is an alignment of all the DNA with its coding regions. However, it does not define a protein alignment without additional assumptions, since one codon in one sequence can be matched to several codons in the other sequence. If the rule is introduced that an amino acid is associated the middle nucleotide of its codon, then the DNA alignment can be translated unambiguously into a protein alignment.

2. The Traditional Algorithm

Let $s1$ and $s2$ be two sequences of length 11 and 12, respectively. The substring consisting of the first i elements of sk ($k = 1,2$) is denoted sk_i and $sk[i]$ refers to the i-th element of sk. Let g be the cost of a gap and $d(\,,\,)$ a distance function on the set of nucleotides (proteins) in the string. $D_{i,j}$ refers to the distance between $s1_i$ and $s2_j$.

Sellers (1974) and Sankoff (1972) presented an algorithm that allowed calculation of the distance between $s1$ and $s2$ and the corresponding alignment. It is encapsulated in the following recursion:

$$D_{i,j} = \min\{D_{i-1,j-1} + d(s1[i], s2[j]),\; D_{i,j-1} + g, D_{i-1,j} + g\} \quad (1)$$

subject to the initial condition $D_{0,0} = 0$ and all $D_{i,j}$'s with negative arguments are defined to be infinity (Fig. 1).

Aligning two sequences corresponds to finding the cheapest path in a graph with nodes indicated by the grid $\{0, \ldots, 11\}^*\{0, \ldots, 12\}$. Each node (i,j) has three predecessors $(i-1, j-1)$, $(i, j-1)$ and $(i-1, j)$. The latter edges are weighted g, since they correspond to insertion-deletions. The first edge is weighted as $d(s1[i], s2[j])$. One alignment of two sequences correspond to a set of histories of the two sequences. If the alignment postulates n differences, then there will be $n!$ possible orders to apply these to $s1$ to obtain $s2$, as the alignment makes no restriction on their order.

Transition weight: 2
Transversion weight: 5
Single indel: 10

	50	42	32	27	17	**9**	**19**
G							
	40	32	22	17	**9**	**17**	27
T							
	30	22	12	**4**	**12**	22	32
G							
	20	12	**2**	**12**	22	32	42
T							
	10	**2**	10	20	30	40	50
T							
	0	10	20	30	40	50	60
		C	T	A	G	G	A

One minimal alignment CTAGGA
TT–GTG

FIG. 1. Distance matrix for two short sequences. Two short sequences are aligned using eqn (1). The weights used are 2 for a transition, 5 for transversions and 10 for an indel of length 1. The distance between the two sequences are found in the upper right corner (19). The alignment(s) corresponding to this minimal value is found by backtracking. The 19 found at position (6,5) in a matrix [0:6][0:5], could come from the value 17 at (5,4), correspond to a match in of the last nucleotides in the sequences: A and G, which differ by a transition of weight 2. By continuing from the upper right corner to the lower left corner, at each step determining how the value was obtained, the complete alignment can be reconstructed.

An Algorithm Aligning DNA, Minimizing Change at the Protein Level

Evolution of a coding DNA string will be idealized to only involve the following events:

(i) Substitutions and amino acid replacements. The cost of a protein substitution will be c_P and of a DNA change, c_D.

(ii) Insertion-deletions (abbreviated as indels). It will be assumed that all insertion-deletions in coding regions are a multiple of three nucleotides. This assumption is central to the formulation of the algorithm. An indel with a length divisible by 3 will change the amino acids coded for within the codon where it is positioned, since it cannot change the reading frame. Were it not divisible by 3 it could change all of the amino acids coded for 3′-ward to the insertion, which would be very hard to analyze.

The gap penalty function, g_k (where k is the length of the insertion-deletion), will be strictly positive and obey:

$$g_i + g_j > g_{i+j}. \tag{2}$$

This implies that it is always cheaper to explain contiguous stretches of gap signs in the alignment as one indel only, making the evolutionary interpretation of an alignment unambiguous.

In the solution of this problem several new features appear. To illustrate the problems consult the hypothetical alignment of two DNA pieces coding for proteins [Fig. 2(a)]. What is the overall amount of change in the coded protein? In the traditional alignment problem this would be trivial: Simply traverse the graph from lower left to upper right, horizontal and parallel runs of k edges adds g_k to the weight and matching edges that matches different nucleotides adds c_N. The alignment would represent a set of events that would transform $s1$ into $s2$ and the order in which these events were applied to the sequences, would not influence the total amount of evolution in their history. Not so when the focus is at the evolution at the protein level (as first observed by Miyata & Yasanunga, 1980). To see this, focus on the two intact matched codons [Fig. 2(b)]. They differ in two nucleotides. Two implied substitutions can have occurred in two orders and they will have different ancestral amino acids. The path Asp-Lys-Thr changes amino acid twice, but Asp-Thr-Thr only once.

The weight of the proposed alignment can be evaluated by decomposing the graph into pieces, by cutting it where it intersects codon boundaries in both the first and the second sequence [Fig. 2(a)]. The amount of change within each piece can then be found by trying all orders of events and choosing the most parsimonious.

By trying all alignments obeying the restriction of indel lengths only being multiples of 3, the minimal alignment could be found. However, as the number of possible alignments for real sequences is astronomical, this is not a practical algorithm. To design a realistic algorithm it would be advantageous to mimic the reasoning in the traditional dynamical algorithm and calculate the distances between substrings. The main obstacle is that it is not possible to evaluate the weight of substitution (or indel) in one part of a codon without knowing how the rest of the codon is aligned.

To formulate the algorithm a few concepts must be introduced (Fig. 3). The smallest piece of alignment connecting two nodes, that correspond to codon boundaries in both sequences, is called a *codon alignment*. Focus on block with an asterisk in Fig. 2(a), the first nucleotides in both sequences are matched, then an indel of length 3 and then two pairs of nucleotides are matched. If the nucleotide pairs in this building block match different nucleotides, this building block corresponds to four events: One indel and three substitutions. By trying all 4! = 24 possible orders the cheapest interpretation is chosen as the weight. This would minimize change at the protein

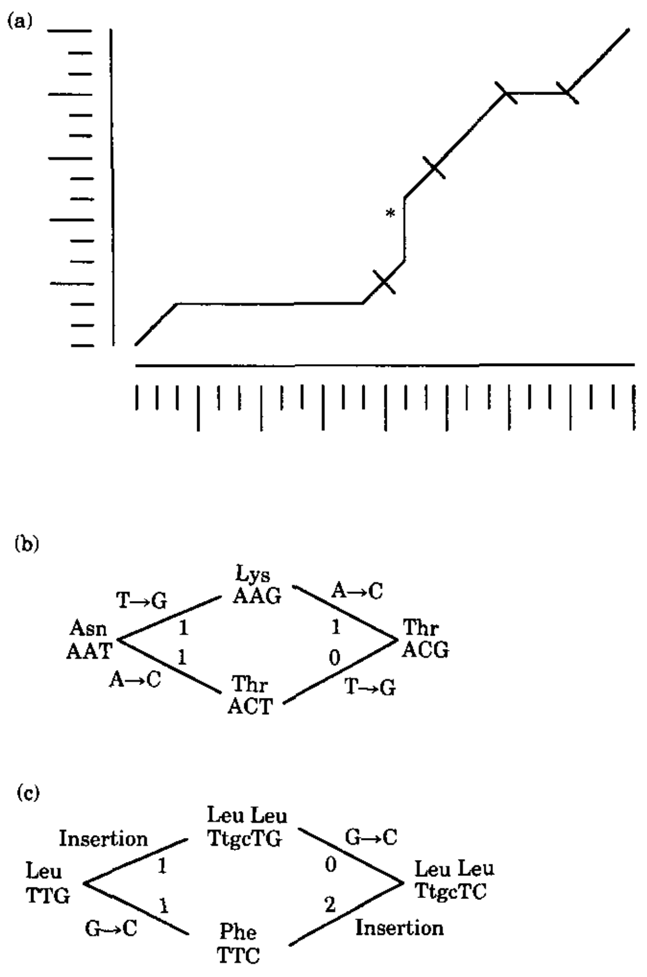

FIG. 2. An alignment of two sequences summarizes a set of transformations of the first sequence into the second sequence. If the alignment indicates n events, then there is $n!$ ways to order these events from a traditional alignment. Panel (a) shows an alignment of eight codons with five codons without showing the actual nucleotides. The alignment graph can be cut every time it passes through a node that corresponds to a codon boundary in both sequences. So the example shown can be decomposed into five blocks, whose weight can be evaluated independently. When only two codons are matched without gaps there is at most 3! possible paths between them and they can all be investigated to find the cheapest interpretation. However, when the alignment indicates n events it quickly becomes impossible to investigate all paths from the first sequence to the second sequence and it is then of interest to decompose the alignment into blocks that can be evaluated independently. It is not straightforward to deduce a protein alignment from a DNA alignment. The second block aligns the fifth codon with codon number 2 and 3. As two of the matchings occur between codon 5 and 3, it might be natural to align the fifth amino acid with the third amino acid, but this is a rule that has to be added to allow conversion from a DNA alignment to a protein alignment and it still does not define such a conversion fully. Panel (b) shows an alignment of two codons different at two positions that could have two ways to transform the first codon into the second codon. However, as is seen in this small example, these ways might have different total overall amount of amino acid change. So when alignments match coding sequences, there can be restrictions on the order in which events can occur. Panel (c) shows two possible paths from a trinucleotide to a hexanucleotide. The upper path has a total of one amino acid change, while the lower has three amino acid replacements. The small letters are the inserted/deleted nucleotides.

level, but a cost for the changes at the DNA level could easily be incorporated.

There will be 11 basic types of codon alignments (Fig. 3). The indel lengths show are all of length 3, but could be any multiple of 3.

The considerations above allow evaluation of the weight of a given alignment, but do not specify how to find the most parsimonious alignment of unaligned sequences. This can be formulated analogously to the simple dynamical programming algorithm.

Let the cheapest alignment of the first i codons of the first sequence against the first j codons from the second sequence be $D_{3i,3j}$ [Fig. 4 (top)] and assume that the costs of aligning all sequences shorter than the first i codons of the first sequence against the first j codons from the second sequence are known. It is then possible to calculate $D_{3i,3j}$ by considering all $D_{3k,3l}$, with $k < = i$ and $l < = j$ and which codon alignments could connect $(3k, 3l)$ with $(3i, 3j)$. $D_{3i,3j}$ would obey following recursion:

$$D_{3i,3j} = \min\langle D_{3k,3l} + \min[w(c.a\,\{(3k,3l),(3i,3j)\})]\rangle, \quad (3)$$

with the initial condition:

$$D_{0,0} = 0$$

The first minimum is taken over all $(3k, 3l)$ that can have a path leading to $(3i, 3j)$. The second minimum is taken over the weights of different codon alignments leading from $(3k, 3l)$ to $3i, 3j)$. The abbreviations, *c.a.* and *w* are for codon alignment and weight, respectively.

The algorithm can be implemented by two for loops going from lower to higher indices, one for rows and one for columns. The running time for the traditional algorithm is $O(n^2)$ because each entry had a constant number of predecessors (3). However in this case the number of predecessors grow as (i, j) gets bigger. Counting (see Fig. 4) the number of predecessors and the number of paths leading from them, reveals that this grows as $O(\max(i, j)^2)$. This leads to an overall $O(n^4)$ algorithm.

If two indels in the same codon alignment were precluded, the algorithm could be sped up to a $O(n^3)$ algorithm (Fig. 5). If g_k obeys a stronger inequality:

$$g_i + g_j > g_{i+j} + 2^*c_P$$

two indels in the same codon alignment would not occur, because it will not pay to introduce an additional indel instead of postulating a series of mutations.

If g_k was of the form $a + b^*k$, it would most likely be possible to formulate a $O(n^2)$ algorithm, by using

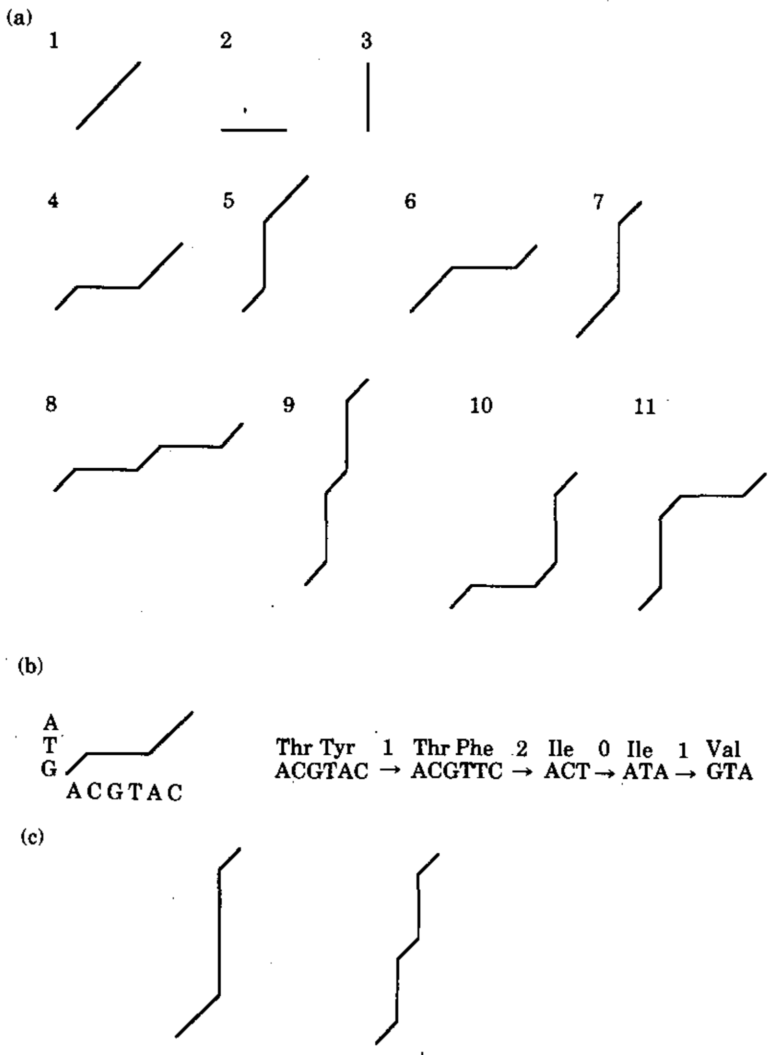
(a)
1
2
3
4
5
6
7
8
9
10
11
(b)
A
T
G
ACGTAC
Thr Tyr 1 Thr Phe 2 Ile 0 Ile 1 Val
ACGTAC → ACGTTC → ACT → ATA → GTA
(c)

FIG. 3. The condition that all insertion-deletions have lengths that are multiples of 3, implies that the alignment graph will intersect both of the codon boundaries for every three matching edges. If the length of the insertion-deletion is ignored then there are 11 basic building blocks. Ignoring the two that do not match any nucleotides (2 and 3), then the nine types correspond to the two choices between three possibilities: Is there an insertion between the first and second nucleotide in the first sequence, the second sequence or in neither? The same choice has to be made between the second and third nucleotide. Panel (b) shows a codon alignment of type 4 above and one (of 24) paths leading from the first sequence (one codon) to the second sequence (two codons). The weight of the codon alignment is the weight of the cheapest of the 24 paths. If the alignment of the sequences should also penalize events at the DNA level, it would just be included in the lengths of the paths. Panel (c) a stronger inequality on g_k would always make a codon alignment with one indel cheaper than codon alignments with two indels.

a trick highly analogous to Gotoh's (Gotoh, 1981) algorithm. The distance $D_{i,j}$ is split into many $Dt_{i,j}$, where t would anticipate what could happen in the rest of the codon alignment.

Programs implementing the sketched algorithms could be programmed, but as formulated could not handle overlapping reading frames, although they could most likely be generalized to do so. This is a serious shortcoming, as large scale DNA would contain a mix of coding, non-coding and possibly overlapping reading frames.

An additional shortcoming would be that a DNA alignment does not unambiguously define a protein alignment. If one codon is aligned to two codons in the other strand, then it does not follow naturally how to match the amino acids. A rule could be introduced, that the amino acid should always follow the middle nucleotide.

Discussion

The basic idea is to combine the protein alignment problem with the DNA alignment problem and then solve them simultaneously. The result is a DNA

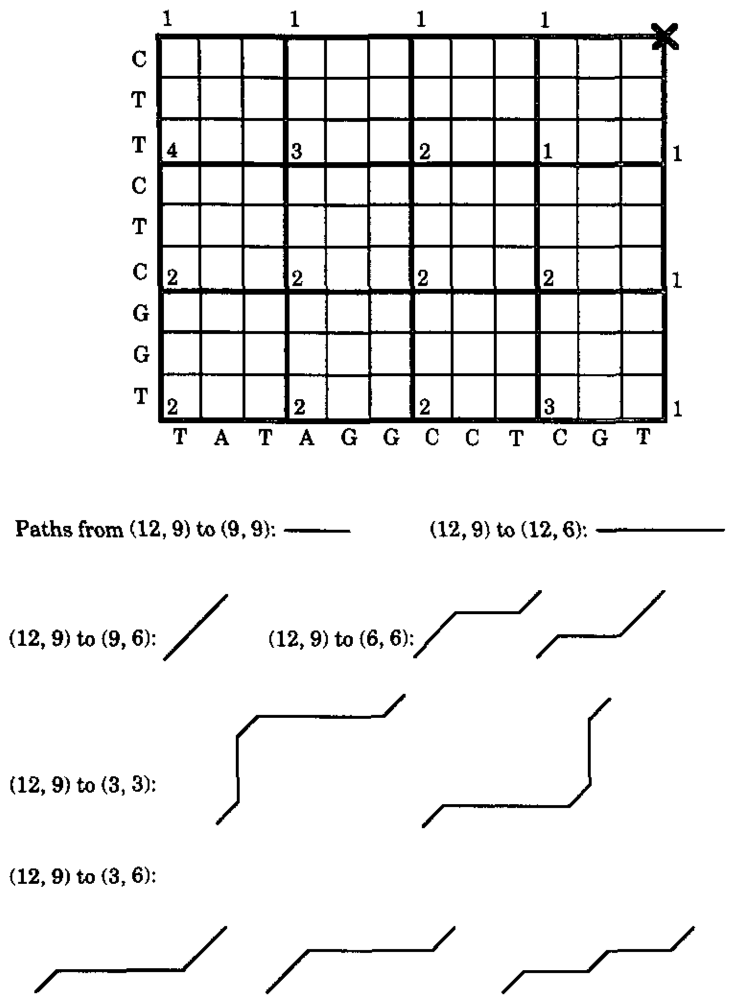

FIG. 4. If the distance, $D_{12,9}$, between $s1[1:12]$ and $s2[1:9]$ (entry (12,9) in the matrix) is to be calculated, then the values at the nodes with numbers would have to be considered. Naturally, it would not be necessary to store values in the matrix that are not calculated and the matrix used in a program could be comressed 3*3 times compared to the one shown here. The numbers at the nodes are the number paths (codon alignments) leading from that node to (12,9). Some of these are illustrated below.

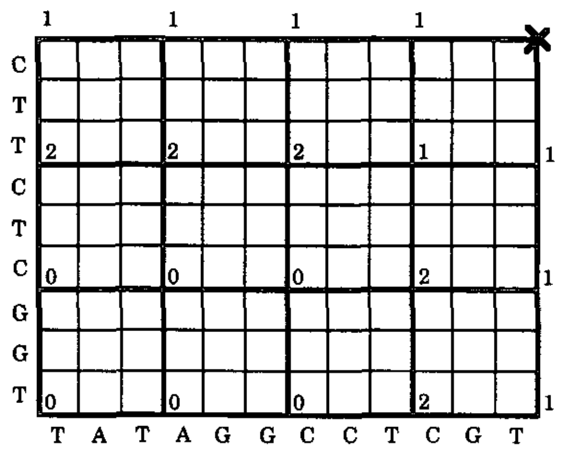

FIG. 5. If no codon alignment has two indels there will be much fewer relevant paths and predecessors. The total number of paths leading to (i, j) would grow as $O(\max(i, j))$.

alignment, that is guided by both DNA and protein similarities.

This algorithm could undoubtedly be generalized to align DNA with many coding frames in it. However, this would be very complicated, but highly practical as this could align genomic structures well. A heuristic method accomplishing this was devised and will be published elsewhere.

J.H. was supported by Japanese Society for the Promotion of Science, the Carlsberg Foundation, The Danish Research Council, grants 11-8916-1 and 11-9639-1. Jens Støvlbæk and Bernt Guldbrandtsen are thanked for comments on the manuscript.

REFERENCES

GOTOH, O. (1981) An improved algorithm for matching biological sequences. *J. molec. Biol.* **162,** 705–708.

MYIATA, T. & YASANUNGA, T. (1980). Molecular evolution of mRNA: A method for estimating evolutionary rates of synonymous and amino acid substitutions from nucleotide sequences and its applications. *J. molec. Evol.* **16,** 23–36.

NEEDLEMAN, S. B. & WUNSCH, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. molec. Biol.* **48,** 444–453.

SANKOFF, D. (1972). Matching sequences under deletion insertion constraints. *Proc. natn. Acad. Sci. U.S.A.* **69,** 4–6.

SELLERS, P. (1974). An algorithm for calculating the distance between two finite sequences. *J. Comb. Theory* **16,** 253–258.