Comparison of coding DNA

Christian N. S. Pedersen[1⋆], Rune Lyngsø[1], and Jotun Hein[2]

[1] BRICS[⋆⋆], Department of Computer Science, University of Aarhus, Ny Munkegade, DK-8000 Århus C, Denmark. E-mail: {cstorm,rlyngsoe}@brics.dk.

[2] Department of Ecology and Genetics, University of Aarhus, Ny Munkegade, DK-8000 Århus C, Denmark. E-mail: jotun@pop.bio.aau.dk

Abstract. We discuss a model for the evolutionary distance between two coding DNA sequences which specializes to the DNA/protein model proposed in Hein [4]. We discuss the DNA/protein model in details and present a quadratic time algorithm that computes an optimal alignment of two coding DNA sequences in the model under the assumption of affine gap cost. The algorithm solves a conjecture in [4] and we believe that the constant factor of the running time is sufficiently small to make the algorithm feasible in practice.

1 Introduction

A straightforward model of the evolutionary distance between two coding DNA sequences is to ignore the encoded proteins and compute the distance in some evolutionary model of DNA. We say that such a model is a DNA level model. The evolutionary distance between two sequences in a DNA level model can most often be formulated as a classical alignment problem and be efficiently computed using a dynamic programming approach [9, 11–13].

It is well known that proteins evolve slower than its coding DNA, so it is usually more reliable to describe the evolutionary distance based on a comparison of the encoded proteins rather than on a comparison of the coding DNA itself. Hence, most often the evolutionary distance between two coding DNA sequences is modeled in terms of amino acid events, such as substitution of a single amino acid and insertion-deletion of consecutive amino acids, necessary to transform the one encoded protein into the other encoded protein. We say that such a model is a protein level model. The evolutionary distance between two coding DNA sequences in a protein level model can most often be formulated as a classical alignment problem of the two encoded proteins. Even though a protein level model is usually more reliable than a DNA level model, it falls short because it postulates that all insertions and deletions on the underlying DNA occur at codon boundaries and because it ignores similarities on the DNA level.

⋆ Supported by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT).

⋆⋆ Basic Research in Computer Science, Centre of the Danish National Research Foundation.

In this paper we present a model of the evolutionary distance between two coding DNA sequences in which a nucleotide event is penalized by the change it induces on the DNA as well as on the encoded protein. The model is a natural combination of a DNA level model and a protein level model. The DNA/protein model introduced in Hein [4, 6] is a biological reasonable instance of the general model in which the evolution of coding DNA is idealized to involve only substitution of a single nucleotide and insertion-deletion of a multiple of three nucleotides. Hein [4, 6] presents an $O(n^2m^2)$ time algorithm for computing the evolutionary distance in the DNA/protein model between two sequences of length n and m. This algorithm assumes certain properties of the cost function. We discuss these properties and present an $O(nm)$ time algorithm that solves the same problem under the assumption of affine gap cost.

The practicality of an algorithm not only depends on the asymptotic running time but also on the constant factor hidden by the use of O-notation. To determine the distance between two sequences of length n and m our algorithm computes $400nm$ table entries. Each computation involves a few additions, table lookups and comparisons. We believe the constant factor is sufficiently small to make the algorithm feasible in practice.

The problem of comparing coding DNA is also discussed by Arvestad [1] and Hua, Jiang and Wu [8]. The models discussed in these papers are inspired by the DNA/protein model in Hein [4, 6] but differ in the interpretation of gap cost. A heuristic algorithm for solving the alignment problem in the DNA/protein model is described In Hein [5]. A related problem of how to compare a coding DNA sequence with a protein has been discussed in [10, 14].

The rest of this paper is organized as follows: In Sect. 2 we introduce and discuss the DNA/protein model. In Sect. 3 we describe how to determine the cost of an alignment. In Sect. 4 we present the simple alignment algorithm of Hein [4]. In Sect. 5 we present a quadratic time alignment algorithm. Finally, in Sect. 6 we discuss future work.

2 The DNA/protein model

Let $a = a_1a_2a_3 \ldots a_{3n-2}a_{3n-1}a_{3n}$ be a coding sequence of DNA of length $3n$ with a reading frame starting at a_1. We introduce the notation $a_1^ia_2^ia_3^i$ to denote the ith codon $a_{3i-2}a_{3i-1}a_{3i}$ and the notation A_i to describe the amino acid coded by the ith codon. The amino acid sequence $A = A_1A_2 \ldots A_n$ describes the protein coded by a. Let $b = b_1b_2b_3 \ldots b_{3m-2}b_{3m-1}b_{3m}$, $b_1^ib_2^ib_3^i$ and $B = B_1B_2 \ldots B_m$ be defined similarly.

2.1 The general model

An evolutionary event e on the DNA that transforms a to a' will also change the encoded protein from A to A'. As some amino acids are coded by several codons, the proteins A and A' might be identical. The cost of e should reflect the changes on the DNA as well as the changes on the encoded protein.

$$cost(a \xrightarrow{e} a') = cost_d(a \xrightarrow{e} a') + cost_p(A \xrightarrow{e} A') \quad (1)$$

We say that $cost_d(a \xrightarrow{e} a')$ is the DNA level cost of e and that $cost_p(A \xrightarrow{e} A')$ is the protein level cost of e. In this paper we assume that the DNA level cost and the protein level cost are combined by addition but other combination functions $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ could of course also be considered.

The cost of a sequence E of evolutionary events $e_1, e_2, \ldots, e_k$ transforming $a^{(0)}$ to $a^{(k)}$ as $a^{(0)} \xrightarrow{e_1} a^{(1)} \xrightarrow{e_2} a^{(2)} \xrightarrow{e_3} \cdots \xrightarrow{e_k} a^{(k)}$ is defined as some function of the costs of each event. In the rest of this paper we will assume that this function is the sum of the costs of each event.

$$cost(a^{(0)} \xrightarrow{E} a^{(k)}) = \sum_{i=1}^{k} cost(a^{(i-1)} \xrightarrow{e_i} a^{(i)}) \tag{2}$$

We define the distance between two coding sequences of DNA a and b according to the parsimony principle as the minimum cost of a sequence of evolutionary events which transforms a to b.

$$dist(a, b) = \min\{cost(a \xrightarrow{E} b) \mid E \text{ is a sequence of events}\} \tag{3}$$

In order to compute $dist(a, b)$ we have to specify the set of allowed evolutionary events and define the cost of each event on the DNA level as well as on the protein level. The choice of evolutionary events and cost function influences both the biological relevance of the distance measure and the computational complexity of computing the distance.

2.2 The specific model

The DNA/protein model introduced in [4] can be described as an instance of the general model where the evolution of coding DNA is idealized to involve only substitutions of single nucleotide and insertion-deletions of a multiple of three consecutive nucleotides. The DNA level cost of an event is defined in the classical way by specifying a substitution cost and a gap cost. The protein level cost of an event that changes the encoded protein from A to A' is defined to reflect the difference between protein A and protein A'.

- The DNA level cost $cost_d(a \xrightarrow{e} a')$ depends on e. The cost of substituting a nucleotide σ with σ' is $c_d(\sigma, \sigma')$ for some metric c_d on nucleotides. The cost of inserting or deleting $3k$ consecutive nucleotides is $g_d(3k)$ for some subadditive[1] function $g_d : \mathbb{N} \to \mathbb{R}^+$.
- The protein level cost $cost_p(A \xrightarrow{e} A')$ is defined as the distance $dist_p(A, A')$ between A and A'. The distance $dist_p(A, A')$ is the minimum cost of a distance alignment of A and A' where we allow substitution of a single amino acid and insertion-deletion of consecutive amino acids. The substitution cost is given by a metric c_p on amino acids and the gap cost is given by a subadditive function $g_p : \mathbb{N} \to \mathbb{R}^+$. Additional restrictions will be given in Sect. 2.3.

[1] A function is subadditive if $f(i + j) \leq f(i) + f(j)$. A subadditive gap cost function implies that an insertion-deletion of a consecutive block of nucleotides is best explained as a single event.

The reason why gap lengths are restricted to a multiple of three is that an insertion or deletion of length not divisible by three changes the reading frame. This is called a frame shift and it may change the entire remaining amino acid sequence as illustrated in Fig. 1. Frame shifts are believed to be rare biological events, so it is not unreasonable to leave them out of the model.

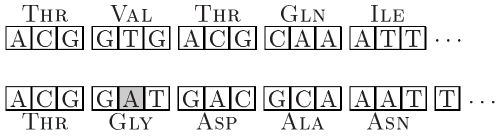


Fig. 1. An insertion-deletion of length not divisible by three changes the reading frame.

Except for the restriction on insertion-deletion length the DNA/protein model allows the traditional set of symbol based nucleotide events. This allows us to use the notion of an alignment. An alignment of two sequences describes a set of substitution or insertion-deletion events necessary to transform one of the sequences into the other sequence. The set of events is usually described by a matrix or a path in a graph as illustrated in Fig. 2. The cost of an alignment is the optimal cost of any sequence of the events described by the alignment. Hence, the evolutionary distance $dist(a, b)$ in the DNA/protein model between two coding DNA sequences a and b is the cost of an optimal alignment of a and b in the DNA/protein model. In the rest of this paper we will address the problem of computing the cost of an optimal alignment in the DNA/protein model.

$$\begin{bmatrix} \mathrm{T\,T\,G\,C\,T} - - - \mathrm{C} \\ \mathrm{T} - - - \mathrm{C\,A\,T\,G\,C} \end{bmatrix}$$

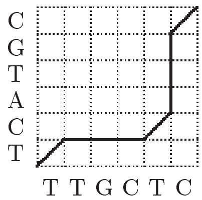


Fig. 2. An alignment can be described by a matrix or a path in the alignment graph. The above alignment describes three matches and two gaps of combined length six.

If the cost of any sequence of events is independent of the order but only depends on the set of events, then an optimal alignment can be computed efficiently using dynamic programming [9, 11–13]. In the DNA/protein model the cost of an event is the sum of the DNA level cost and the protein level cost. We observe that the DNA level cost of a sequence of events is independent of the order but that the protein level cost is not [4, Fig. 2]. This implies that we cannot use a classical alignment algorithm to compute an optimal alignment in the DNA/protein model. In order to formulate an efficient alignment algorithm in the DNA/protein model we must examine the protein level cost further.

2.3 Restrictions on the cost function

A single nucleotide event affects nucleotides in one or more consecutive codons. Since a nucleotide event in the DNA/protein model cannot change the reading frame then only the amino acids encoded by the affected codons are affected by the nucleotide event. A nucleotide event thus changes protein $A = UXV$ to protein $A' = UX'V$ where X and X' are the amino acids affected by nucleotide event. Let us consider the protein level cost $dist_p(A, A')$ of the event.

Hein [4] implicitly assumes that $dist_p(A, A')$ is the cost of a distance alignment of X and X' with the minimum number of insertion-deletions. This assumption implies that $dist_p(A, A')$ is the cost of one of the alignments of A and A' shown in Fig. 3. This assumption is essential to the formulation of our alignment algorithms in Sect. 4 and 5.

If the cost of alignment of A and A' implied by Fig. 3 is not minimal then the assumption conflicts with our previous definition of $dist_p(A, A')$ as being the minimum cost of an alignment of A and A'. Lemma 1 states restrictions on c_p and g_p that prevent this conflict. If we assume an affine gap cost function $g_p(k) = \alpha_p + \beta_p k$ for some $\alpha_p, \beta_p \geq 0$ (and define $g_p(0)$ to be zero), then the restrictions on c_p and g_p becomes $c_p(\sigma, \tau) + \alpha_p + \beta_p k \leq 2\alpha_p + \beta_p(k + 2l)$ for any amino acids σ, τ and all lengths $0 < l \leq n - k$. This simplifies to $c_p(\sigma, \tau) \leq \alpha_p + 2\beta_p$ for all amino acids σ, τ which is a biological reasonable restriction as insertions and deletions are rare events compared to substitutions.

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_{i-1} & A_i & A_{i+1} & \cdots & A_n \\ A_1 & A_2 & \cdots & A_{i-1} & A'_i & A_{i+1} & \cdots & A_n \end{bmatrix}$$

(a) A substitution in the ith codon. The cost is $c_p(A_i, A'_i)$.

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_{i-1} & A_i & A_{i+1} & \cdots & A_{i+k} & A_{i+k+1} & \cdots & A_n \\ A_1 & A_2 & \cdots & A_{i-1} & A_i & - & \cdots & - & A_{i+k+1} & \cdots & A_n \end{bmatrix}$$

(b) An insertion-deletion of $3k$ nucleotides affecting exactly k codons. The cost is $g_p(k)$.

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_{i-1} & A_i & \cdots & A_{j-1} & A_j & A_{j+1} & \cdots & A_{i+k} & A_{i+k+1} & \cdots & A_n \\ A_1 & A_2 & \cdots & A_{i-1} & - & \cdots & - & \upsilon & - & \cdots & - & A_{i+k+1} & \cdots & A_n \end{bmatrix}$$

(c) An insertion-deletion of $3k$ nucleotides affecting $k+1$ codons. The remaining amino acid υ is matched with one of the amino acids affected by the deletion. The cost is $\min_{j=0,1,\ldots,k} \{g_p(j) + c_p(A_{i+j}, \upsilon) + g_p(k - j)\}$.

Fig. 3. The protein level cost of a nucleotide event can be determined by considering only the amino acids affected by the event.

Lemma 1. *Assume a nucleotide event changes $A = UXV$ to $A' = UX'V$. Let $n = |A|$ and $k = ||A| - |A'||$. If there for any amino acids σ, τ and for all $0 < l \leq n-k$ exists $0 \leq j \leq k$ such that $c_p(\sigma,\tau)+g_p(j)+g_p(k-j) \leq g_p(l)+g_p(l+k)$, then $dist_p(A, A')$ is the cost of an alignment describing exactly k insertions or deletions. Furthermore $dist_p(A, A')$ only depends on X and X'.*

Proof. We note that the alignments in Fig. 3 all describe the minimum number of insertion-deletions and that only the sub-alignment of X and X', as illustrated by the shaded parts, contributes to the cost. We will argue that the assumption on c_p and g_p stated in the lemma ensures that $dist_p(A, A')$ is the cost of one of the alignments in Fig. 3. We split the argumentation depending on the event. Since $dist_p(A, A')$ is equal to $dist_p(A', A)$ the cost of an insertion transforming A to A' is equal to the cost of a deletion transforming A' to A. We thus only consider substitutions and deletions.

A substitution of a nucleotide in the ith codon of A transforms A_i to A'_i. The alignment in Fig. 3(a) describes no insertion-deletions and has cost $c_p(A_i, A'_i)$. Any other alignment of A and A' must describe an equal number of insertions and deletions, so by subadditivity of g_p the cost is at least $2g_p(l)$ for some $0 < l \leq n$. The assumption in the lemma implies that $c_p(A_i, A'_i) \leq 2g_p(l)$ for any $0 < l \leq n$. The alignment in Fig. 3(a) is thus optimal and the protein level cost of the substitution is $c_p(A_i, A'_i)$.

A deletion of $3k$ nucleotides affecting k codons transforms $A = A_1A_2 \cdots A_n$ to $A' = A_1A_2 \cdots A_iA_{i+k+1}A_{i+k+2} \cdots A_n$. Any alignment of A and A' must describe l insertions and $l + k$ deletions for some $0 \leq l \leq n - k$, so the cost is at least $g_p(l) + g_p(l + k)$. The alignment in Fig. 3(b) describes k deletions and has cost $g_p(k)$. The assumption in the lemma and the sub-additivity of g_p implies that $g_p(k) \leq g_p(j) + g_p(k - j) \leq g_p(l) + g_p(l + k)$ for all $l > 0$. The alignment in Fig. 3(b) is thus optimal and the protein level cost of the deletion is $g_p(k)$.

A deletion of $3k$ nucleotides affecting $k + 1$ codons, say a deletion of the $3k$ nucleotides $a_3^i a_1^{i+1} a_2^{i+1} a_3^{i+1} \cdots a_1^{i+k} a_2^{i+k}$, transforms $A = A_1A_2 \cdots A_n$ to $A' = A_1A_2 \cdots A_{i-1}\upsilon A_{i+k+1} \cdots A_n$ where υ is the amino acid coded by $a_1^i a_2^i a_3^{i+k}$. We say that υ is the remaining amino acid and $a_1^i a_2^i a_3^{i+k}$ is the remaining codon. Any alignment of A and A' describing exactly k deletions must align υ with A_{i+j} for some $0 \leq j \leq k$, so by subadditivity of g_p the cost is at least $g_p(j)+c_p(A_{i+j}, \upsilon)+g_p(k - j)$. The alignment in Fig. 3(c) illustrates one of the $k + 1$ alignments of A and A' where υ is aligned with an affected amino acid and all non-affected amino acids are aligned. Such an alignment describes exactly k deletions and the cost of the optimal alignment among them has cost

$$\min_{j=0,1,\ldots,k} \{g_p(j) + c_p(A_{i+j}, \upsilon) + g_p(k - j)\}, \tag{4}$$

and is thus optimal for any alignment describing exactly k deletions. Any other alignment of A and A' must describe l insertions and $l + k$ deletions for some $0 < l \leq n - k$, so the cost is at least $g_p(l) + g_p(l + k)$. The assumption in the lemma implies that the cost given by (4) is less than or equal to $g_p(l)+g_p(l+k)$. The protein level cost of the deletion is thus given by (4). □

The assumption in Lemma 1 is sufficient to ensure that we can compute the protein level cost of a nucleotide event efficiently, but the formulation of the lemma is to general to make the assumption necessary. The following example however suggests when the assumption is necessary. Consider a deletion of three nucleotides that transforms the six amino acids ABEFCD to ABGCD, i.e. $X =$ EF and $X' =$ G. Consider the two alignments shown in Fig. 4.

$$\begin{bmatrix} \mathrm{A\ B\ E\ F\ C\ D} \\ \mathrm{A\ B\ G\ -\ C\ D} \end{bmatrix} \qquad \begin{bmatrix} \mathrm{A\ B\ E\ F\ -\ C\ D} \\ \mathrm{A\ B\ -\ -\ G\ C\ D} \end{bmatrix}$$

Fig. 4. Two alignments of the amino acids ABEFCD and ABGCD.

If we assume that $c_p(\mathrm{E}, \mathrm{G}) \leq c_p(\mathrm{F}, \mathrm{G})$ then the cost of the alignment in Fig. 4 (left) is $c_p(\mathrm{E}, \mathrm{G}) + g_p(1)$ while the cost of the alignment in Fig. 4 (right) is $g_p(2)+g_p(1)$. If the assumption in lemma 1 does not hold then $g_p(2)+g_p(1)$ might be less than $c_p(\mathrm{E}, \mathrm{G}) + g_p(1)$ because $c_p(\mathrm{E}, \mathrm{G})$ can be arbitrary large. Hence, the protein level cost of the deletion would not be the cost of an alignment describing the minimum number of insertion-deletions.

3 The cost of an alignment

Before we can describe how to compute the cost of an optimal alignment of two sequence in the DNA/protein model, we need to know how to compute the cost of a given alignment in the model.

An alignment of two sequences describes a set of events necessary to transform one of the sequences into the other sequence but it does not describe the order of the events. As observed in Sect. 2.2 the DNA level cost of an alignment is independent of the order of the events, while the protein level cost of an alignment depends on the order of the events. This implies that the DNA level cost of an alignment is just the sum of the DNA level cost of the events described by the alignment, while the protein level cost of the same alignment is somewhat harder to determine. An obvious way to determine the protein level cost of an alignment is to minimize over all possible sequences of the events described by the alignment. This method is however not feasible in practice due to the factorial number of possible sequences one has to consider.

If Lemma 1 is fulfilled then we know that the protein level cost of a nucleotide event only depends on the affected codons. We can use this property to decompose the computation of the protein level cost of an alignment into smaller subproblems. The idea is to decompose the alignment into *codon alignments*. A codon alignment is a minimal part of the alignment that corresponds to a path connecting two nodes $(3i', 3j')$ and $(3i, 3j)$ in the alignment graph. We can decompose an alignment uniquely into codon alignments as illustrated in Fig. 5.

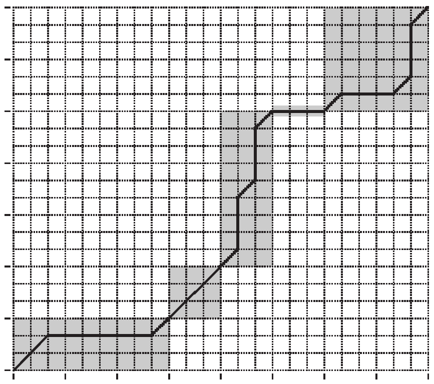


Fig. 5. An alignment of two sequences decomposed into codon alignments.

The assumption that the length of an insertion or deletion is a multiple of three implies that a codon alignment either describes no substitutions (see Type 2 and 3 in Fig. 7) or exactly three substitutions. If a codon alignment describes exactly three substitutions then it can also describe one or more insertion-deletions. More precisely, between any two consecutive substitutions in the codon alignment there can be an alternating sequence of insertions and deletions each with length a multiple of three. Such a sequence of insertion-deletions corresponds to a “staircase” in the alignment graph. The set of codon alignments that describe three substitutions and at most one insertion and deletion between two consecutive substitutions, i.e. the “staircase” is limited to at most one “step”, is illustrated in Fig. 6. A particular codon alignment in this set corresponds to a choice of which sides of the two rectangles to traverse.

Now consider the decomposition of an alignment into codon alignments (this could be as illustrated in Fig. 5). We observe that nucleotide events described by two different codon alignments in the decomposition do not affect the same codons. Hence, the protein level cost of a codon alignment can be computed independently of the other codon alignments in the decomposition. We can thus compute the protein level cost of an alignment as the sum of the protein level cost of each of the codon alignments in the decomposition.

Since a codon alignment can describe an alternating sequence of insertion-deletions between two consecutive substitutions it is possible that a decomposition of an alignment of two sequences of length n and m contains codon alignments describing $\Theta(n+m)$ events. This implies that the problem of computing the cost of the codon alignments in a decompsition of an alignment is, in the worst case, not any easier than computing the cost of the alignment itself. One way to circumvent this problem is to only consider alignments that can be decomposed into (or built of) codon alignments with at most some maximum number of insertion-deletions between two consecutives substitutions. This upper bounds the number of events described by any codon alignment by some constant. Hence, we can determine the cost of a codon alignment in constant time simply by minimizing over all possible sequences of the events described by the codon alignment. The protein level cost of an alignment can thus be determined in time proportional to the number of codon alignments in the decomposition of the alignment.

In Hein [4] at most one insertion *or* one deletion is allowed between two consecutive substitutions in a codon alignment. Besides the two codon alignments that describe no substitutions, this corresponds to the set of codon alignments we obtain from Fig. 6 when we require that the width and/or the height of each of the two rectangles must be zero. This implies that there are eleven different types of codon alignments. The eleven types are shown in Fig. 7.

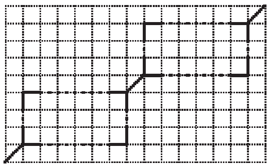


Fig. 6. A summary of all possible codon alignments with at most one insertion and one deletion between two consecutive substitutions.

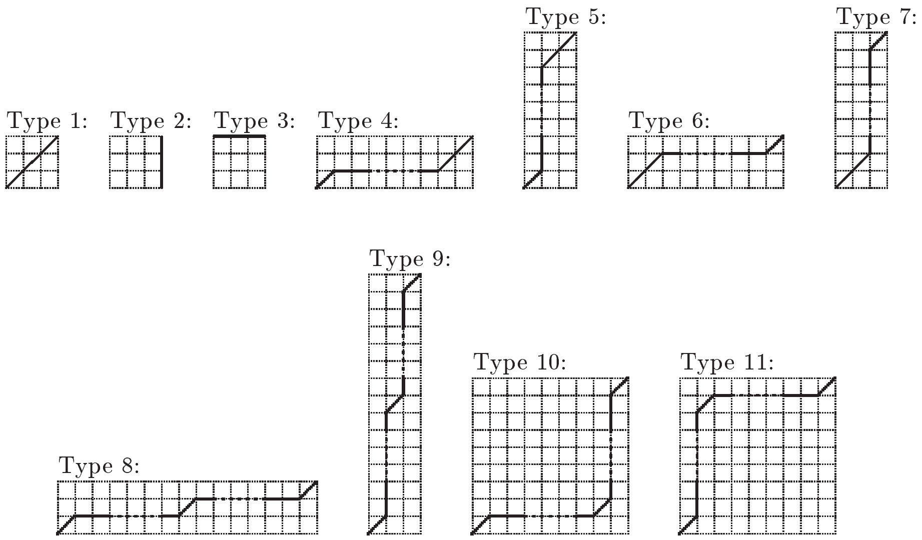


Fig. 7. The eleven types of codon alignments with at most one inseriton or one deletion between two consecutive substitutions.

4 A simple alignment algorithm

Let $a_1a_2\cdots a_{3n}$ and $b_1b_2\cdots b_{3m}$ be two coding sequences of DNA. Hein [4] describes how the decomposition into codon alignments makes it possible to compute the cost of an optimal alignment of a and b in the DNA/protein model in time $\mathrm{O}(n^2m^2)$. The algorithm assumes that Lemma 1 is fulfilled and that we only allow codon alignments with some maximum number of insertion-deletions between two consecutive substitutions, e.g. the eleven types of codon alignments in Fig. 7. The algorithm can be summarized as follows.

Let $D(i,j)$ denote the cost of an optimal alignment of $a_1a_2\cdots a_{3i}$ and $b_1b_2\cdots b_{3j}$. We define $D(0,0)$ to be zero and $D(i,j)$ to be infinity for $i<0$ or $j<0$. An optimal alignment of $a_1a_2\cdots a_{3i}$ and $b_1b_2\cdots b_{3j}$ can be decomposed into codon alignments $ca_1, ca_2, \ldots, ca_k$. We say that ca_k is the last codon alignment and that $ca_1, ca_2, \ldots, ca_{k-1}$ is the remaining alignment.

If the last codon alignment ca_k is an alignment of $a_{3i'+1}a_{3i'+2}\cdots a_{3i}$ and $b_{3j'+1}b_{3j'+2}\cdots b_{3j}$ for some $(i',j') < (i,j)$[2], then $D(i,j)$ is equal to $D(i',j') + \mathit{cost}(ca_k)$. This is the cost of the last codon alignment plus the cost of the remaining alignment. We can compute $D(i,j)$ by minimizing the expression $D(i',j') + \mathit{cost}(ca)$ over all $(i',j') < (i,j)$ and all possible codon alignments ca of $a_{3i'+1}a_{3i'+2}\cdots a_{3i}$ and $b_{3j'+1}b_{3j'+2}\cdots b_{3j}$.

The upper bound on the number of insertion-deletions in a codon alignment implies that the number of possible codon alignments of $a_{3i'+1}a_{3i'+2}\cdots a_{3i}$ and $b_{3j'+1}b_{3j'+2}\cdots b_{3j}$, for all $(i',j') < (i,j)$, is upper bounded by some constant. For example, if we only consider the eleven types of codon alignments in Fig. 7 then there are at most three possible codon alignments of $a_{3i'+1}a_{3i'+2}\cdots a_{3i}$ and $b_{3j'+1}b_{3j'+2}\cdots b_{3j}$. Hence, if we assume that $D(i',j')$ is known for all $(i',j') < (i,j)$ then we can compute $D(i,j)$ in time $\mathrm{O}(ij)$. By dynamic programming this implies that we can compute $D(n,m)$ in time $\mathrm{O}(n^2m^2)$ and space $\mathrm{O}(nm)$. By back-tracking we can also get the optimal alignment (and not only the cost) within the same time and space bound.

5 An improved alignment algorithm

Let a and b be coding sequences of DNA as introduced in the previous section. We will describe how to compute the cost of an optimal alignment of a and b in the DNA/protein model in time $\mathrm{O}(nm)$ and space $\mathrm{O}(n)$. Besides the assumptions of the simple algorithm described in the previous section we also assume that the function $g(k) = g_d(3k) + g_p(k)$ is affine $\alpha + \beta k$ for some $\alpha, \beta \geq 0$. We say that g is the *combined gap cost function*.

The idea behind the improved algorithm is similar to the idea behind the simple algorithm in the sense that we compute the cost of an optimal alignment by minimizing the cost over all possible last codon alignments. We define $D^t(i,j)$ to be the cost of an optimal alignment of $a_1a_2\cdots a_{3i}$ and $b_1b_2\cdots b_{3j}$ under the

[2] We say that $(i',j') < (i,j)$ iff $i' \leq i \wedge j' \leq j \wedge (i' \neq i \vee j' \neq j)$.

assumption that the last codon alignment is of type t. Remember that we only allow codon alignments with some maximum number of insertion-deletions between two consecutive substitutions. This implies that the number of possible codon alignment, i.e. types of codon alignments, is upper bounded by some constant. We define $D^t(0,0)$ to be zero and $D^t(i,j)$ to be infinity for $i < 0$ or $j < 0$. We can compute $D(i,j)$ as

$$D(i,j) = \min_t D^t(i,j). \tag{5}$$

Lemma 1 ensures that $D^t(i,j)$ is the cost of some last codon alignment (of type t) plus the cost of the corresponding remaining alignment. This allows us to compute $D^t(i,j)$ by minimizing the cost over all possible last codon alignments of type t. The assumption that g is affine makes it possible to compute $D^t(i,j)$ in constant time if $D^t(k,l)$, for all t, is known for some $(k,l) < (i,j)$. Since we only consider a constant number of possible codon alignments (e.g. the types in Fig. 7) this implies that we can compute $D(n,m)$ in time $\mathrm{O}(nm)$ and space $\mathrm{O}(n)$. By adapting the technique in Hirschberg [7] or the variant described in Durbin et al. [2, Page 35–36] we can also get the optimal alignment (and not only the cost) within the same time and space bound.

The method we use to compute $D^t(i,j)$ can be used for any type of last codon alignment but the bookkeeping and thereby the constant overhead increases with the number of gaps described by the last codon alignment. By restricting ourselves to the eleven types of codon alignments shown in Fig. 7 we can compute the cost of an optimal alignment with a reasonable constant overhead. In the rest of this paper we therefore focus on these eleven types of codon alignments. We divide the explanation of how to compute $D^t(i,j)$ for $t = 1, 2, \ldots, 11$ according to the number of gaps within a codon (denoted *internal gaps*) described by a codon alignment of type t. Codon alignments of type 1–3 describe no internal gaps, codon alignments of type 4–7 describe one internal gap and codon alignments of type 8–11 describe two internal gaps.

For use in the explanation we introduce $c_p^* : \{\mathrm{A},\mathrm{C},\mathrm{G},\mathrm{T}\}^3 \times \{\mathrm{A},\mathrm{C},\mathrm{G},\mathrm{T}\}^3 \to \mathbb{R}$. We define $c_p^*(\sigma_1\sigma_2\sigma_3, \tau_1\tau_2\tau_3)$ as the distance between $\sigma_1\sigma_2\sigma_3$ and $\tau_1\tau_2\tau_3$ in the DNA/protein model. This is the minimum over the cost[3] of the six possible sequences of the three substitutions $\sigma_1 \to \tau_1$, $\sigma_2 \to \tau_2$ and $\sigma_3 \to \tau_3$.

5.1 Codon alignments with no internal gaps

The cost $D^1(i,j)$ is the cost of the last codon alignment of type 1 plus the cost of the remaining alignment. The last codon alignment is an alignment of $a_1^i a_2^i a_3^i$ and $b_1^j b_2^j b_3^j$. By definition of c_p^* the cost is $c_p^*(a_1^i a_2^i a_3^i, b_1^j b_2^j b_3^j)$. The cost of the remaining alignment is $D(i-1, j-1)$.

$$D^1(i,j) = D(i-1, j-1) + c_p^*(a_1^i a_2^i a_3^i, b_1^j b_2^j b_3^j) \tag{6}$$

[3] We use the term *cost* to denote the DNA level cost plus the protein level cost.

A codon alignment of type 2 or type 3 describes a gap between codons. Since the combined gap cost function is affine we can use the technique introduced in [3] saying that a gap ending in (i,j) is either a continuation of an existing gap ending in $(i-1,j)$ or $(i,j-1)$, or the start of a new gap.

$$D^2(i,j) = \min\{D(i,j-1)+\alpha+\beta, D^2(i,j-1)+\beta\} \tag{7}$$

$$D^3(i,j) = \min\{D(i-1,j)+\alpha+\beta, D^3(i-1,j)+\beta\} \tag{8}$$

5.2 Codon alignments with one internal gap

We describe how to compute $D^6(i,j)$. The other cases where the last codon alignment describes one internal gap are handled similarly. The cost $D^6(i,j)$ is the cost of the last codon alignment of type 6 plus the cost of the remaining alignment. The last codon alignment of type 6 describes three substitutions and one deletion. If the deletion has length k (a deletion of $3k$ nucleotides) then the cost of the remaining alignment is $D(i-k-1, j-1)$ and the last codon alignment is an alignment of $a_1^{i'} a_2^{i'} a_3^{i'} \cdots a_1^i a_2^i a_3^i$ and $b_1^j b_2^j b_3^j$ where $i' = i-k$. This is illustrated in Fig. 8.

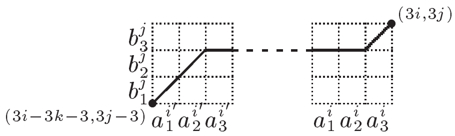


Fig. 8. The last codon alignment of type 6.

The cost of the last codon alignment is the minimum cost of a sequence of the three substitutions and the deletion. Any sequence of these four events can be divided into three steps: The substitutions occurring before the deletion, the deletion and the substitutions occurring after the deletion. Figure 9 illustrates the three steps of the evolution of $a_1^{i'} a_2^{i'} a_3^{i'} \cdots a_1^i a_2^i a_3^i$ to $b_1^j b_2^j b_3^j$. The nucleotides x_1, x_2 and x_3 are the result of the up to three substitutions before the deletion. For example, if the substitution $a_1^{i'} \to b_1^j$ occurs before the deletion, then x_1 is b_1^j, otherwise it is $a_1^{i'}$. We say that $x_1 \in \{a_1^{i'}, b_1^j\}$, $x_2 \in \{a_2^{i'}, b_2^j\}$ and $x_3 \in \{a_3^i, b_3^j\}$ are the status of the three substitutions before the deletion. We observe that $x_1 x_2 x_3$ is the remaining codon of the deletion.

The substitutions occurring before the deletion change codon $a_1^{i'} a_2^{i'} a_3^{i'}$ to $x_1 x_2 a_3^{i'}$ and codon $a_1^i a_2^i a_3^i$ to $a_1^i a_2^i x_3$. The substitutions occurring after the deletion change codon $x_1 x_2 x_3$ to $b_1^j b_2^j b_3^j$. We recall that the cost of changing codon $\sigma_1\sigma_2\sigma_3$ to codon $\tau_1\tau_2\tau_3$ by a sequence of the substitutions $\sigma_1 \to \tau_1$, $\sigma_2 \to \tau_2$ and $\sigma_3 \to \tau_3$ is $c_p^*(\sigma_1\sigma_2\sigma_3, \tau_1\tau_2\tau_3)$. Since an identical substitution has cost zero then the cost of the three substitutions in the last codon alignment is equal to the

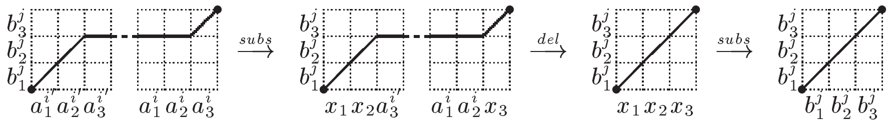


Fig. 9. The evolution of $a_1^{i'} a_2^{i'} a_3^{i'} \cdots a_1^i a_2^i a_3^i$ to $b_1^j b_2^j b_3^j$ described by the last codon alignment.

cost of the induced codon changes. The cost is

$$cost(subs) = c_p^*(a_1^{i'} a_2^{i'} a_3^{i'}, x_1 x_2 a_3^{i'}) + \\ c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i x_3) + c_p^*(x_1 x_2 x_3, b_1^j b_2^j b_3^j). \quad (9)$$

The cost of the deletion of $3k$ nucleotides in the last codon alignment is the sum of the DNA level cost $g_d(3k)$ and the protein level cost as given by (4). By using the combined gap cost function $g(k) = g_d(3k) + g_p(k) = \alpha + \beta k$ and our knowledge of the remaining codon $x_1 x_2 x_3$ of the deletion, we can formulate this sum as

$$cost(del) = \min \begin{cases} \alpha + \beta k + c_p(a_1^i a_2^i x_3, x_1 x_2 x_3)^4 \\ \alpha_p + \alpha + \beta k + \min_{0<l<k} c_p(a_1^{i-l} a_2^{i-l} a_3^{i-l}, x_1 x_2 x_3) \\ \alpha + \beta k + c_p(x_1 x_2 a_3^{i'}, x_1 x_2 x_3) \end{cases} . \quad (10)$$

The cost of the deletion depends on the deletion length, the remaining codon $x_1 x_2 x_3$ and a witness. The witness can be the start-codon $x_1 x_2 a_3^{i'}$, the end-codon $a_1^i a_2^i x_3$ or one of the internal codons $a_1^{i-l} a_2^{i-l} a_3^{i-l}$ for some $0 < l < k$. The witness encodes the amino acid aligned with the remaining amino acid.

We can now compute $D^6(i,j)$ under the assumption of a certain deletion length k and remaining codon $x_1 x_2 x_3$ of the deletion in the last codon alignment as the sum $cost(subs) + cost(del) + D(i-k-1, j-1)$. We can thus compute $D^6(i,j)$ by minimizing this sum over all possible combinations of deletion length k and remaining codon $x_1 x_2 x_3$. A combination of deletion length k and remaining codon $x_1 x_2 x_3$ is possible if $x_1 \in \{a_1^{i'}, b_1^j\}$, $x_2 \in \{a_2^{i'}, b_2^j\}$ and $x_3 \in \{a_3^i, b_3^j\}$ where $i' = i - k$. The terms $c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i x_3)$ and $c_p^*(x_1 x_2 x_3, b_1^j b_2^j b_3^j)$ of $cost(subs)$ do not depend on the deletion length, so we can split the minimization as

$$D^6(i,j) = \min_{x_1 x_2 x_3} \{c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i x_3) + \\ c_p^*(x_1 x_2 x_3, b_1^j b_2^j b_3^j) + D^6_{x_1 x_2 x_3}(i,j)\} \quad (11)$$

where

$$D^6_{x_1 x_2 x_3}(i,j) = \min_{0<k<i} \{D(i-k-1, j-1) + \\ c_p^*(a_1^{i-k} a_2^{i-k} a_3^{i-k}, x_1 x_2 a_3^{i-k}) + cost(del)\} \quad (12)$$

[4] We use $c_p(\sigma_1\sigma_2\sigma_3, \tau_1\tau_2\tau_3)$ as a convenient notation for $c_p(\sigma, \tau)$ where σ and τ are the amino acids coded by the codons $\sigma_1\sigma_2\sigma_3$ and $\tau_1\tau_2\tau_3$ respectively.

is the minimum cost of the terms that depend on both the deletion length and the remaining codon under the assumption that the remaining codon is $x_1x_2x_3$. If we expand the term $cost(del)$ we get

$$D^6_{x_1x_2x_3}(i,j) = \min_{0<k<i} \{\text{len}^6_{x_1x_2}(i,j,k) + \\ \min \begin{cases} c_p(a_1^i a_2^i x_3, x_1x_2x_3) \\ \alpha_p + \min_{0<l<k} c_p(a_1^{i-l} a_2^{i-l} a_3^{i-l}, x_1x_2x_3) \\ c_p(x_1x_2a_3^{i-k}, x_1x_2x_3) \end{cases} \} \quad (13)$$

where

$$\text{len}^6_{x_1x_2}(i,j,k) = D(i-k-1, j-1) + \\ c_p^*(a_1^{i-k} a_2^{i-k} a_3^{i-k}, x_1x_2a_3^{i-k}) + \alpha + \beta k \quad (14)$$

is the cost of the remaining alignment plus the part of the cost of the last codon alignment that does not depend on the codon $a_1^i a_2^i a_3^i$ and the witness. The cost $\text{len}^6_{x_1x_2}(i,j,k)$ is defined if $x_1 \in \{a_1^{i-k}, b_1^j\}$ and $x_2 \in \{a_2^{i-k}, b_2^j\}$. The cost $D^6_{x_1x_2x_3}(i,j)$ is defined if there exists a deletion length k such that k and $x_1x_2x_3$ is a possible combination of deletion length and remaining codon.

We observe that there are at most 32 possible remaining codons $x_1x_2x_3$. The observation follows because we known that x_3 must be one of the two known nucleotides a_3^i or b_3^j. If we can compute $D^6_{x_1x_2x_3}(i,j)$ in constant time for each of the possible remaining codons then we can also compute $D^6(i,j)$ in constant time. To compute $D^6_{x_1x_2x_3}(i,j)$ we must determine a combination of witness and deletion length that minimizes the cost. We say that we must determine the witness and deletion length of $D^6_{x_1x_2x_3}(i,j)$.

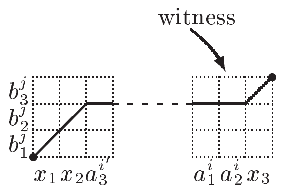
It is easy to see that witness and deletion lenghth of $D^6_{x_1x_2x_3}(i,j)$ must be one of the four combinations illustrated in Fig. 10. We can thus compute $D^6_{x_1x_2x_3}(i,j)$ as the minimum over the cost of the four cases illustrated in Fig. 10. The cost of case 1–3 is computed by simplifying (13) for a particular witness and deletion length. The cost of case 4 cannot be computed this way because both the witness and the deletion length are unknown.

Case 1. The end-codon is the witness and the deletion length is at least one. The cost is $\min_{0<k<i} \text{len}^6_{x_1x_2}(i,j,k) + c_p(a_1^i a_2^i x_3, x_1x_2x_3)$.

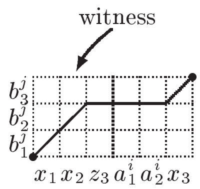
Case 2. The last internal codon is the witness and the deletion length is at least two. The cost is $\min_{1<k<i} \text{len}^6_{x_1x_2}(i,j,k) + \alpha_p + c_p(a_1^{i-1} a_2^{i-1} a_3^{i-1}, x_1x_2x_3)$.

Case 3. The start-codon is the witness and the deletion length is one. The cost is $\text{len}^6_{x_1x_2}(i,j,1) + c_p(x_1x_2a_3^{i-1}, x_1x_2x_3)$.

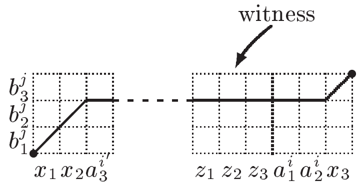
Case 4. The witness is neither the end-codon nor the last internal codon and the deletion length is at least two. We observe that if the witness of $D^6_{x_1x_2x_3}(i-1,j)$ is not the end-codon $a_1^{i-1}a_2^{i-1}x_3$ then by optimality of $D^6_{x_1x_2x_3}(i-1,j)$ this witness must also be the witness of case 4. If this is the case then the cost of case 4 is $D^6_{x_1x_2x_3}(i-1,j) + \beta$.



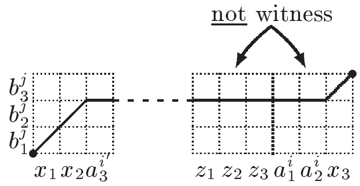
(a) Case 1



(c) Case 3



(b) Case 2



(d) Case 4

Fig. 10. The four cases in the computation of $D^6_{x_1x_2x_3}(i, j)$. We use $z_1z_2z_3$ as notation for $a_1^{i-1}a_2^{i-1}a_3^{i-1}$.

The observation in case 4 suggests that we can use dynamic programming to keep track of $D^6_{x_1x_2x_3}(i, j)$ under the assumption that the end-codon is not the witness, i.e. use dynamic programming to keep track of the minimum cost of case 2–4. We introduce tables $F^6_{x_1x_2x_3}$ corresponding to the 64 combinations of $x_1x_2x_3$. We maintain that if $x_1x_2x_3$ is a possible remaining codon and the end-codon $a_1^ia_2^ix_3$ is not the witness of $D^6_{x_1x_2x_3}(i, j)$, then $F^6_{x_1x_2x_3}(i, j)$ is equal to $D^6_{x_1x_2x_3}(i, j)$. If we define $F^6_{x_1x_2x_3}(0, j)$ to infinity, then

$$F^6_{x_1x_2x_3}(i, j) = \min \begin{cases} \text{cost of } \textit{Case 2} \\ \text{cost of } \textit{Case 3} \\ F^6_{x_1x_2x_3}(i-1, j) + \beta \end{cases} \tag{15}$$

In order to compute the cost of case 1 and 2 in constant time we maintain the minimum of $\mathrm{len}^6_{x_1x_2}(i, j, k)$ over k by dynamic programming. We introduce tables $L^6_{x_1x_2}$ corresponding to the 16 combinations of x_1x_2 such that $L^6_{x_1x_2}(i, j)$ is equal to $\min_{0<k<i} \mathrm{len}^6_{x_1x_2}(i, j, k)$. If we define $L^6_{x_1x_2}(0, j)$ to infinity, then

$$L^6_{x_1x_2}(i, j) = \min \begin{cases} \mathrm{len}^6_{x_1x_2}(i, j, 1) \\ L^6_{x_1x_2}(i-1, j) + \beta \end{cases} \tag{16}$$

We can now compute $D^6_{x_1x_2x_3}(i, j)$ in constant time as the minimum cost of case 1–4. The cost of case 1 is $L^6_{x_1x_2}(i, j) + c_p(a_1^ia_2^ix_3, x_1x_2x_3)$ and the minimum cost of case 2–4 is $F^6_{x_1x_2x_3}(i,j)$, so

$$D^6_{x_1x_2x_3}(i,j) = \min \begin{cases} L^6_{x_1x_2}(i,j) + c_p(a_1^i a_2^i x_3, x_1x_2x_3) \\ F^6_{x_1x_2x_3}(i,j) \end{cases} \tag{17}$$

The computation of $D^6(i,j)$ by (11) requires us to compute $D^6_{x_1x_2x_3}(i,j)$ for each of the 32 possible remaining codons. To do this we must compute entry (i,j) in the 16 tables $L^6_{x_1x_2}$ and entry (i,j) in the 64 tables $F^6_{x_1x_2x_3}$. As explained in this section all this can be done in constant time.

The other three cases where the last codon alignment describes one internal gap (type 4, 5 and 7) are handled similarly. However, if the last codon alignment is of type 4 or 5, then only the first nucleotide x_1 in the remaining codon depends on the deletion (or insertion) length. This limits the number of possible remaining codons to 16 and implies that only four tables are needed to keep track of $\min_{0<k<i} \mathrm{len}^t_{x_1}(i,j,k)$ for $t = 4,5$. Hence, to compute $D^t(i,j)$ for $t = 4,5,6,7$, we compute $2 \cdot 4 + 2 \cdot 16 + 4 \cdot 64 = 296$ table entries in total.

5.3 Codon alignments with two internal gaps

We describe how to compute $D^8(i,j)$. The other cases where the last codon alignment describes two internal gaps are handled similarly. The cost $D^8(i,j)$ is the cost of the last codon alignment of type 8 plus the cost of the remaining alignment. The last codon alignment of type 8 describes three substitutions and two deletions. If the first deletion has length k' and the second deletion has length k then the cost of the remaining alignment is $D(i-k-k'-1, j-1)$ and the last codon alignment is an alignment of $a_1^{i''} a_2^{i''} a_3^{i''} \cdots a_1^{i'} a_2^{i'} a_3^{i'} \cdots a_1^i a_2^i a_3^i$ and $b_1^j b_2^j b_3^j$ where $i' = i-k$ and $i'' = i'-k'$. This is illustrated in Fig. 11.

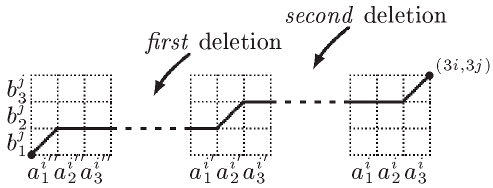


Fig. 11. The last codon alignment of type 8

We will compute $D^8(i,j)$ as we computed $D^6(i,j)$ by minimizing the cost over all possible combinations of deletion length k and remaining codon $x_1x_2x_3$ of the second deletion. This reduces the problem of computing $D^8(i,j)$ to computing $D^8_{x_1x_2x_3}(i,j)$, the cost under the assumption of a certain remaining codon of the second deletion, for each of the 32 possible remaining codons of the second deletion. We can compute $D^8_{x_1x_2x_3}(i,j)$ similar to the way we computed $D^6_{x_1x_2x_3}(i,j)$. An inspection of (15), (16) and (17) reveals that all we essentially have to do is to replace $\text{len}^6_{x_1x_2}(i,j,1)$ with the corresponding part of $D^8(i,j)$. We denote this part of the cost $\text{len}^8_{x_1x_2x_3}(i,j,1)$.

More precisely. If we assume that the second deletion has length k and remaining codon $x_1x_2x_3$ then $\text{len}^8_{x_1x_2x_3}(i,j,k)$ is the part of $D^8(i,j)$ that does not depend on the codon $a_1^i a_2^i a_3^i$ and the witness of the second deletion. This cost depends on the order of the two deletions in the last codon alignment. Hence, we introduce $\text{len}^{8'}_{x_1x_2x_3}(i,j,k)$ and $\text{len}^{8''}_{x_1x_2x_3}(i,j,k)$ to denote the cost when the first deletion occurs before the second deletion and vice versa. We define $\text{len}^8_{x_1x_2x_3}(i,j,k)$ as $\min\{\text{len}^{8'}_{x_1x_2x_3}(i,j,k), \text{len}^{8''}_{x_1x_2x_3}(i,j,k)\}$. Since we only have to compute $\text{len}^8_{x_1x_2x_3}(i,j,1)$ we can restrict ourselves to the case where the second deletion has length one and the first deletion has length k'. In the rest of this section we use the notation that $i' = i-1$ and $i'' = i'-k'$. We split into two cases depending on the order of the first and second deletion.

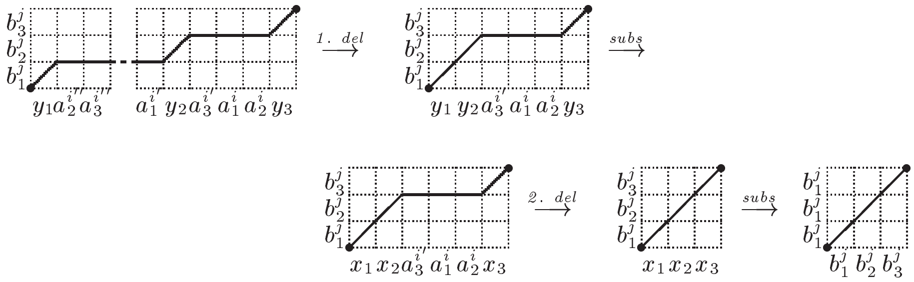


Fig. 12. The first deletion occurs before the second deletion and the second deletion has length one.

Figure 12 illustrates the evolution of the last codon alignment (of type 8) when the second deletion has length one and occurs after the first deletion. The nucleotides y_1, y_2 and y_3 are the status of the substitutions before the first deletion and the nucleotides x_1, x_2 and x_3 are the status of the substitutions before the second deletion. Similar to (9) we compute the cost of the three substitutions in the last codon alignment as the cost of the induced codon changes. An inspection of Fig. 12 reveals that the cost of the three substitutions is

$$\begin{aligned} cost(subs) = {} & c_p^*(a_1^{i''} a_2^{i''} a_3^{i''}, y_1 a_2^{i''} a_3^{i''}) + c_p^*(a_1^{i'} a_2^{i'} a_3^{i'}, a_1^{i'} y_2 a_3^{i'}) + \\ & c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i y_3) + c_p^*(y_1 y_2 a_3^{i'}, x_1 x_2 a_3^{i'}) + \\ & c_p^*(a_1^i a_2^i y_3, a_1^i a_2^i x_3) + c_p^*(x_1 x_2 x_3, b_1^j b_2^j b_3^j). \end{aligned} \quad (18)$$

We can compute the cost of the two deletions similar to (10). Remember that the first deletion has length k' and the second deletion has length one and that we use the notation $i' = i-1$ and $i'' = i' - k'$.

$$cost(del_1) = \min \begin{cases} \alpha + \beta k' + c_p(a_1^{i'} y_2 a_3^{i'}, y_1 y_2 a_3^{i'}) \\ \alpha_p + \alpha + \beta k' + \min_{0<l<k'} c_p(a_1^{i'-l} a_2^{i'-l} a_3^{i'-l}, y_1 y_2 a_3^{i'}) \\ \alpha + \beta k' + c_p(y_1 a_2^{i''} a_3^{i''}, y_1 y_2 a_3^{i'}) \end{cases} \quad (19)$$

$$cost(del_2) = \alpha + \beta + \min \begin{cases} c_p(x_1 x_2 a_3^{i'}, x_1 x_2 x_3) \\ c_p(a_1^i a_2^i x_3, x_1 x_2 x_3) \end{cases} \quad (20)$$

If we assume that the first deletion occurs before the second deletion and that the second deletion has length one and remaining codon $x_1x_2x_3$ then $D^8(i,j)$ is given by the sum $cost(subs) + cost(del_1) + cost(del_2) + D(i'-k'-1, j-1)$ minimized over all possible combinations of $y_1y_2y_3$ and k'. Remember that $\text{len}^{8'}_{x_1x_2x_3}(i,j,1)$ is the part of this minimum that does not depend on $a_1^i a_2^i a_3^i$ or the witness of the second deletion. An inspection of the above expressions reveals that $\text{len}^{8'}_{x_1x_2x_3}(i,j,1)$ includes everything but $c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i y_3)$ and $c_p^*(a_1^i a_2^i y_3, a_1^i a_2^i x_3)$ of $cost(subs)$ and $\min\{c_p(x_1 x_2 a_3^{i'}, x_1 x_2 x_3), c_p(a_1^i a_2^i x_3, x_1 x_2 x_3)\}$ of $cost(del_2)$. It is easy to verify that $D^4_{y_1 y_2 a_3^{i'}}(i', j)$ is equal to the sum $D(i'-k'-1, j-1) + c_p^*(a_1^{i''} a_2^{i''} a_3^{i''}, y_1 a_2^{i''} a_3^{i''}) + cost(del_1)$ minimized over the deletion length k' of the first deletion. This observation makes it possible to compute $\text{len}^{8'}_{x_1x_2x_3}(i,j,1)$ as

$$\begin{aligned} \text{len}^{8'}_{x_1x_2x_3}(i,j,1) &= \alpha + \beta + c_p^*(x_1 x_2 x_3, b_1^j b_2^j b_3^j) + \\ &\min_{y_1 y_2} \{ c_p^*(a_1^{i'} a_2^{i'} a_3^{i'}, a_1^{i'} y_2 a_3^{i'}) + D^4_{y_1 y_2 a_3^{i'}}(i', j) + c_p^*(y_1 y_2 a_3^{i'}, x_1 x_2 a_3^{i'}) \} \end{aligned} \quad (21)$$

where we minimize over $y_1 \in \{a_1^{i''}, x_1\}$ and $y_2 \in \{a_2^{i'}, x_2\}$. The cost $\text{len}^{8'}_{x_1x_2x_3}(i,j,1)$ is defined if $x_1x_2x_3$ allows the second deletion to have length one, i.e. if $x_1 \in \{a_1^{i''}, b_1^j\}$, $x_2 \in \{a_2^{i'}, b_2^j\}$ and $x_3 \in \{a_3^i, b_3^j\}$. The nucleotide $a_1^{i''}$ depends on the unknown length of the first deletion, so we must assume that it can be any of the four nucleotides.

Figure 13 illustrates the evolution of the last codon alignment when the second deletion has length one and occurs before the first deletion. The nucleotides z_1, x_2 and x_3 are the status of the substitutions before the second deletion and the nucleotides y_1, y_2 and y_3 are the status of the substitutions before the first deletion. Observe that x_1 is just $a_1^{i'}$. The cost of this case can be described as above. This would reveal that $\text{len}^{8''}_{x_1x_2x_3}(i,j,1)$ includes everything but $c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i x_3) + c_p^*(a_1^i a_2^i x_3, a_1^i a_2^i y_3) + c_p(x_1 x_2 x_3, w_1 w_2 w_3)$ where $w_1 w_2 w_3$ is the witness of the second deletion. Furthermore it would reveal that $D^4_{y_1 y_2 y_3}(i', j)$ is equal to the sum of the cost of the remaining alignment, the cost of the first deletion and the cost of changing codon $a_1^{i''} a_2^{i''} a_3^{i''}$ to $z_1 a_2^{i''} a_3^{i''}$ to $y_1 a_2^{i''} a_3^{i''}$ minimized over the deletion length k' of the first deletion. This makes it possible to compute $\text{len}^{8''}_{x_1x_2x_3}(i,j,1)$ as

$$\begin{aligned} \text{len}^{8''}_{x_1x_2x_3}(i,j,1) &= \alpha + \beta + c_p^*(a_1^{i'} a_2^{i'} a_3^{i'}, x_1 x_2 a_3^{i'}) + \\ &\min_{y_1 y_2 y_3} \{ c_p^*(a_1^{i'} x_2 x_3, a_1^{i'} y_2 y_3) + D^4_{y_1 y_2 y_3}(i', j) + c_p^*(y_1 y_2 y_3, b_1^j b_2^j b_3^j) \} \end{aligned} \quad (22)$$

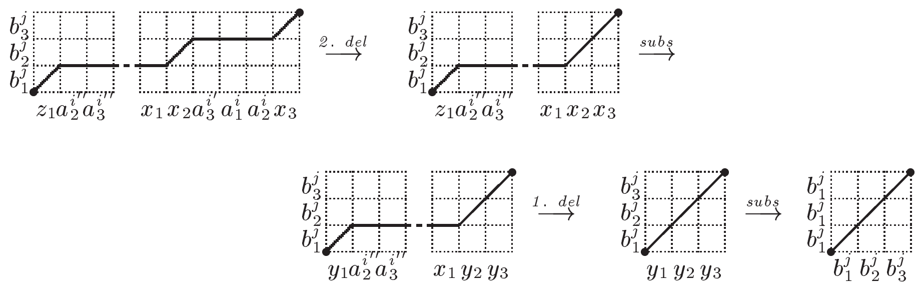


Fig. 13. The second deletion occurs before the first deletion and the second deletion has length one.

where we minimize over $y_1 \in \{z_1, b_1^j\}$, $y_2 \in \{x_2, b_2^j\}$ and $y_3 \in \{x_3, b_3^j\}$. The nucleotide z_1 depends on the unknown length of the first deletion, so we must assume that z_1 can be any of the four nucleotides. The cost $\text{len}^{8''}_{x_1x_2x_3}(i,j,1)$ is defined if $x_1x_2x_3$ allows the second deletion to have length one, i.e. if $x_1 = a_1^{i'}$, $x_2 \in \{a_2^{i'}, b_2^j\}$ and $x_3 \in \{a_3^i, b_3^j\}$.

We are finally in a position where we can describe how to use the method from the previous section to compute $D^8(i,j)$. The cost $\text{len}^8_{x_1x_2x_3}(i,j,k)$ depends on x_1, x_2 and x_3, so instead of 16 tables we need 64 tables $L^8_{x_1x_2x_3}$ to keep track of $\min_{0<k<i} \text{len}^8_{x_1x_2x_3}(i,j,k)$. We still need 64 tables $F^8_{x_1x_2x_3}$ to keep track of the cost under the assumption that the end-codon $a_1^i a_2^i x_3$ is not the witness (of the second deletion). We compute table entry (i,j) in these tables as

$$L^8_{x_1x_2x_3}(i,j) = \min \begin{cases} \text{len}^8_{x_1x_2x_3}(i,j,1) \\ L^8_{x_1x_2x_3}(i-1,j) + \beta \end{cases} \tag{23}$$

$$F^8_{x_1x_2x_3}(i,j) = \min \begin{cases} L^8_{x_1x_2x_3}(i-1,j) + \beta + \alpha_p + c_p(a_1^{i-1}a_2^{i-1}a_3^{i-1}, x_1x_2x_3) \\ \text{len}^8_{x_1x_2x_3}(i,j,1) + c_p(x_1x_2a_3^{i-1}, x_1x_2x_3) \\ F^8_{x_1x_2x_3}(i-1,j) + \beta \end{cases} \tag{24}$$

We compute $D^8_{x_1x_2x_3}(i,j)$ using the above tables and we compute $D^8(i,j)$ by minimizing over the 32 possible remaining codons of the second deletion.

$$D^8_{x_1x_2x_3}(i,j) = \min \begin{cases} L^8_{x_1x_2x_3}(i,j) + c_p(a_1^i a_2^i x_3, x_1x_2x_3) \\ F^8_{x_1x_2x_3}(i,j) \end{cases} \tag{25}$$

$$D^8(i,j) = \min_{x_1x_2x_3} \{c_p^*(a_1^i a_2^i a_3^i, a_1^i a_2^i x_3) + D^8_{x_1x_2x_3}(i,j)\} \tag{26}$$

This constant time computation of $D^8(i,j)$ requires us to compute entry (i,j) in 128 tables. The other three cases where the last codon alignment describes two internal gaps are handled similarly. Hence, to compute $D^t(i,j)$ for $t = 8, 9, 10, 11$ we compute $4 \cdot 128 = 512$ table entries in total.

5.4 Combining the computation

We observe that the only real difference between the computation of $D^6(i,j)$ and $D^8(i,j)$ is between $\mathrm{len}^6_{x_1x_2}(i,j,1)$ and $\mathrm{len}^8_{x_1x_2x_3}(i,j,1)$. The similarity stems from the fact that a codon alignment of type 6 and type 8 ends in the same way. By "end in the same way" we mean that the events described on the codon $a^i_1a^i_2a^i_3$ are the same. Figure 7 reveals that a codon alignment of type 11 also ends in the same way as codon alignments of type 6 and 8.

The similarity between the computation of $D^t(i,j)$ for $t = 6, 8, 11$ makes it possible to combine the computation of the three costs and thereby reduce the number of tables. We can replace the three tables $L^6_{x_1x_2}$, $L^8_{x_1x_2x_3}$ and $L^{11}_{x_1x_2x_3}$ with one table $L^{6,8,11}_{x_1x_2x_3}$ where $L^{6,8,11}_{x_1x_2x_3}(i,j)$ is the minimum of entry (i,j) in the three tables it replaces. Similarly we can replace $F^6_{x_1x_2x_3}$, $F^8_{x_1x_2x_3}$ and $F^{11}_{x_1x_2x_3}$ with $F^{6,8,11}_{x_1x_2x_3}$. We can compute $L^{6,8,11}_{x_1x_2x_3}(i,j)$ and $F^{6,8,11}_{x_1x_2x_3}(i,j)$ by expressions similar to (23) and (24). All we essentially have to do is to replace $\mathrm{len}^8_{x_1x_2x_3}(i,j,1)$ by

$$\mathrm{len}^{6,8,11}_{x_1x_2x_3}(i,j,1) = \min_{t=6,8,11} \mathrm{len}^t_{x_1x_2x_3}(i,j,1) \tag{27}$$

where we in order to ensure that $\mathrm{len}^t_{x_1x_2x_3}(i,j,1)$ for $t = 6, 8, 11$ describes the same part of the total cost must redefine $\mathrm{len}^6_{x_1x_2x_3}(i,j,1)$ as $\mathrm{len}^6_{x_1x_2}(i,j,1) + c^*_p(x_1x_2x_3, b^j_1b^j_2b^j_3)$.

We introduce $D^{6,8,11}(i,j)$ as the minimum of $D^t(i,j)$ over $t = 6, 8, 11$. We can compute $D^{6,8,11}(i,j)$ by using $L^{6,8,11}_{x_1x_2x_3}$ and $F^{6,8,11}_{x_1x_2x_3}$ in expressions similar to (25) and (26). The computation of $D^{6,8,11}(i,j)$ requires us to compute only $64 + 64 = 128$ table entries while the individual computation of $D^t(i,j)$ for $t = 6, 8, 10$ requires us to compute $80 + 128 + 128 = 336$ table entries. Figure 7 also reveals that codon alignments of type 7, 9 and 10 end in the same way. Hence, we can also combine the computation of $D^t(i,j)$ for $t = 7, 9, 10$.

Finally, to compute $D(i,j)$ by (5) we must minimize over $D^1(i,j)$, $D^2(i,j)$, $D^3(i,j)$, $D^4(i,j)$, $D^5(i,j)$, $D^{6,8,11}(i,j)$ and $D^{7,9,10}(i,j)$. In total this computation requires us to compute $1 + 7 + 68 + 68 + 128 + 128 = 400$ table entries.

6 Future work

We are working on implementing the alignment algorithm described in the previous section in order to compare it to the heuristic alignment algorithm described in [5]. The heuristic algorithm allows frame shifts, so an obvious extension of our exact algorithm would be to allow frame shifts, e.g. to allow insertion-deletions of arbitrary length. This however makes it difficult to split the evaluation of the alignment cost into small independent subproblems (codon alignments) of known size.

Another interesting extension would be to annotate the DNA sequence with more information. For example, if the DNA sequence codes in more than one reading frame (overlapping reading frames) then the DNA sequence should be annotated with all the amino acid sequences encoded and the combined cost of a nucleotide event should summarize the cost of changes induced on all the amino acid sequences encoded by the DNA sequence. This extension also makes it difficult to split the evaluation of the alignment cost into small independent subproblems. To implement these extensions efficiently it might be fruitful to investigate reasonable restrictions of the cost functions.

References

1. L. Arvestad. Aligning coding DNA in the presence of frame-shift errors. In *Proceedings of the 8th Annual Symposium of Combinatorial Pattern Matching (CPM 97)*, volume 1264 of *Lecture Notes in Computer Science*, pages 180–190, 1997.
2. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probalistic Models of Proteins and Nucleic Acids.* Cambrigde University Press, 1998.
3. O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1981.
4. J. Hein. An algorithm combining DNA and protein alignment. *Journal of Theoretical Biology*, 167:169–174, 1994.
5. J. Hein and J. Støvlbæk. Genomic alignment. *Journal of Molecular Evolution*, 38:310–316, 1994.
6. J. Hein and J. Støvlbæk. Combined DNA and protein alignment. *Methods in Enzymology*, 266:402–418, 1996.
7. D. S. Hirschberg. A linear space algortihm for computing maximal common subseqeunce. *Communication of the ACM*, 18(6):341–343, 1975.
8. Y. Hua, T. Jiang, and B. Wu. Aligning DNA sequences to minimize the change in protein. Accepted for CPM 98.
9. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid seqeunce of two proteins. *Journal of Molecular Biology*, 48:433–443, 1970.
10. H. Peltola, H. Söderlund, and E. Ukkonen. Algorithms for the search of amino acid patterns in nucleic acid sequences. *Nuclear Acids Research*, 14(1):99–107, 1986.
11. D. Sankoff. Matching sequences under deletion/insertion constraints. In *Proceedings of the National Acadamy of Science USA*, volume 69, pages 4–6, 1972.
12. P. H. Sellers. On the theory and computation of evolutionary distance. *SIAM Journal of Applied Mathematics*, 26:787–793, 1974.
13. R. A. Wagner and M. J. Fisher. The string to string correction problem. *Journal of the ACM*, 21:168–173, 1974.
14. Z. Zhang, W. R. Pearson, and W. Miller. Aligning a DNA sequence with a protein sequence. *Journal of Computational Biology*, 4(3):339–349, Fall 1997.