Stadler and Will
Algorithms for Molecular Biology (2022) 17:10
https://doi.org/10.1186/s13015-022-00219-7

Algorithms for Molecular Biology

RESEARCH Open Access

Bi-alignments with affine gaps costs

Peter F. Stadler[1,2,3,4,5,6*] and Sebastian Will[7]

Abstract

Background: Commonly, sequence and structure elements are assumed to evolve congruently, such that homologous sequence positions correspond to homologous structural features. Assuming *congruent* evolution, alignments based on sequence *and* structure similarity can therefore optimize both similarities at the same time in a single alignment. To model *incongruent* evolution, where sequence and structural features diverge positionally, we recently introduced *bi-alignments*. This generalization of sequence and structure-based alignments is best understood as alignments of two distinct pairwise alignments of the same entities: one modeling sequence similarity, the other structural similarity.

Results: Optimal bi-alignments with affine gap costs (or affine shift cost) for two constituent alignments can be computed exactly in quartic space and time. Even bi-alignments with affine shift and gap cost, as well as bi-alignment with sub-additive gap cost are optimized efficiently. Affine gap-cost bi-alignment of large proteins (∼ 930 aa) can be computed.

Conclusion: Affine cost bi-alignments are of practical interest to study shifts of protein sequences and protein structures relative to each other.

Availability: The affine cost bi-alignment algorithm has been implemented in Python 3 and Cython. It is available as free software from https://github.com/s-will/BiAlign/releases/tag/v0.3 and as bioconda package `bialign`.

Keywords: Dynamic programming, Scoring functions, Multi-tape formal grammar, Recursion

Introduction

Incongruent evolution

While biological function is eventually encoded in a genomic sequence, it relies on the "decoding" of the sequence into a spatially structured RNA or protein, or into specific interactions, such as the binding of a DNA element by a transcription factor. Natural selection acts to conserve function over evolutionary times and therefore preserves functional RNA or protein structures, binding motifs, intron–exon boundaries, etc. Stabilizing selection on such a functional entity typically also causes the conservation of its encoding DNA sequence. Homologous functional units, i.e. those that share a common ancestry [1], are therefore represented by homologous sequences. As a consequence, functional elements often can be identified based on their similarity in sequence alignments. For RNA and proteins, this allows the detection of consensus structures [2, 3], enables the identification of transcription factor binding sites [4], and the detection of conserved (non-coding) transcripts through the conservation of splice junctions [5].

Homology of a feature or trait, however, does not require that all its constituent parts are homologous. Most obviously, insertions and deletions in a DNA sequence imply that not all nucleotides trace back to a common ancestor even if the sequence as a whole does. Similarly, homology of a structural feature does not imply that all its constituent contacts are preserved. There are indeed well-documented exceptions to the by far most common case of homologous features being produced

*Correspondence: studla@bioinf.uni-leipzig.de

[1] Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Universität Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany
Full list of author information is available at the end of the article

from homologous sequence positions. A well-studied, albeit apparently rare, example is *intron-sliding*, where the start and end of an intron "moves" in the same direction for the same number of nucleotides [6–9]. While the gene product is perfectly preserved, except possibly for some changes of the amino acids encoded by the few nucleotides involved in the sliding, both splice junctions are now encoded by non-homologous genomic positions. Promotors sometime exhibit a similar form of turnover, where a short binding site pattern at one site is replaced by the emergence of a matching sequence nearby [10]. In the context of biopolymer structures it is possible that contacts between nucleotides or amino acids are shifted relative to the underlying sequence in a way that preserves most features of the ancestral structure. Such transitions can be facilitated by the existence of kinetically accessible structural alternatives [11], of which different variants are stabilized by subsequent mutations in different lineages. In a preliminary survey, we recently observed that 72 of 1181 moderate-size `Rfam` families show evidence for this kind of incongruence between sequence and structure conservation [12]. This observation suggests that incongruent evolution of sequence and structure is relatively rare but still occurs with sufficient frequency to be non-negligible.

To our knowledge, incongruences between conserved protein sequence and conserved protein structures so far have not been studied systematically. However, the example of Fig. 1 demonstrates that (at least at the level of secondary structures) it is not at all difficult to obtain incongruence by performing a few mutations. Here, we (artificially) introduced substitutions into a peptide sequence such that predicted secondary structures shifted relative to the reference sequence. The comparative analysis of proteins occasionally reveals examples of natural incongruences between sequence and secondary structure; moreover, it shows that the phenomenon occurred at least occasionally in protein evolution. Figure 2(top) depicts the alignment of the extant human CYPB1 cytochrome P450 enzyme and its reconstructed ancestral mammalian counterpart, which was recently crystallized (PDB: 6OYU and 6OYV) and characterized functionally [15]. Despite the high level of similarity of the ancestral and extant folds, the bi-alignment (Fig. 2, bottom) reveals some differences in the extent of helices and suggests a shift of "helix D" by two amino acids, constituting an incongruence of the considered type. Another published example can be found in Fig. 5 of [16]: relative to the underlying sequence, one observes several small helix shifts in the evolution of the Pgp protein (MDR1) between human, mouse, and rat.

Incongruences between sequence homology and homology of structure or functional elements are rooted in the inherent redundancy of genotype-phenotype maps. For both RNA and proteins, very different sequences can encode the same fold or function [17–19], while at the same time identical sequences can appear in very different structural or functional contexts [20–22]. Together, these features sometimes lead to a sliding or migration of a functionally relevant structure in response to a fortuitously placed mutation. The occasional emergence of incongruences between sequence conservation and the conservation of structure thus is an expected consequence of the redundancies inherent in the sequence/structure relationship of biopolymers. It becomes a relevant empirical question, therefore, how frequent this process has been throughout evolutionary history.

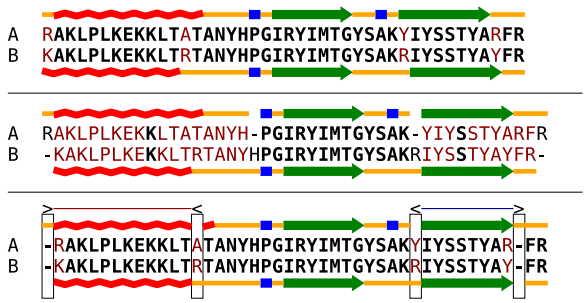

Fig. 1 Two pairwise alignments and a bi-alignment of peptide sequences and their predicted secondary structures (helix red, turn blue, β-sheet green, coil orange). Structure are predicted according to the Chou Fasman method [13] with `CFSSP` [14]. To facilitate quick visual assessment of sequence alignment quality, sequence mismatches are shown in bold black, sequence indels in non-bold black, and mismatches in dark red. The upper alignment optimizes sequence similarity, and shows the structure out of sync: the helix is moved to left, the last β-sheet is shifted to the right by 1 position. The second alignment maximizes structural similarity and thus shows little sequence similarity. The evolution of the two peptides is explained much better by a bi-alignment (third panel), which supports shift events (marked by rectangles) that can shift either sequence against its structure to the left (<) or to the right (>). The resulting *regions of shift* are indicated by in general k blue and red lines corresponding to shifts by k positions to the left or to the right. While the shift events shown in this example delete and insert structure of A with respect to both sequences and the structure of B, shift alignments also support as well analogous shifts of sequences and the second structure (which would be shown in the bottom row). In our representation, shift events are the only visible difference between the bi-alignment $\mathbb{A}$ in the third panel and the two alignments. Nevertheless, the representation can be mapped to our formalization of bi-alignments as alignments of two constituent alignments $\mathbb{U}$ and $\mathbb{V}$: $\mathbb{U}$ is obtained from the 2nd and 3rd bi-alignment row by removing the two all-gap columns (i.e. the first and the 3rd-to-last column). The secondary structure alignment $\mathbb{V}$ coincides with the 1st and 4th row since there is no column that contains only gaps in these to rows

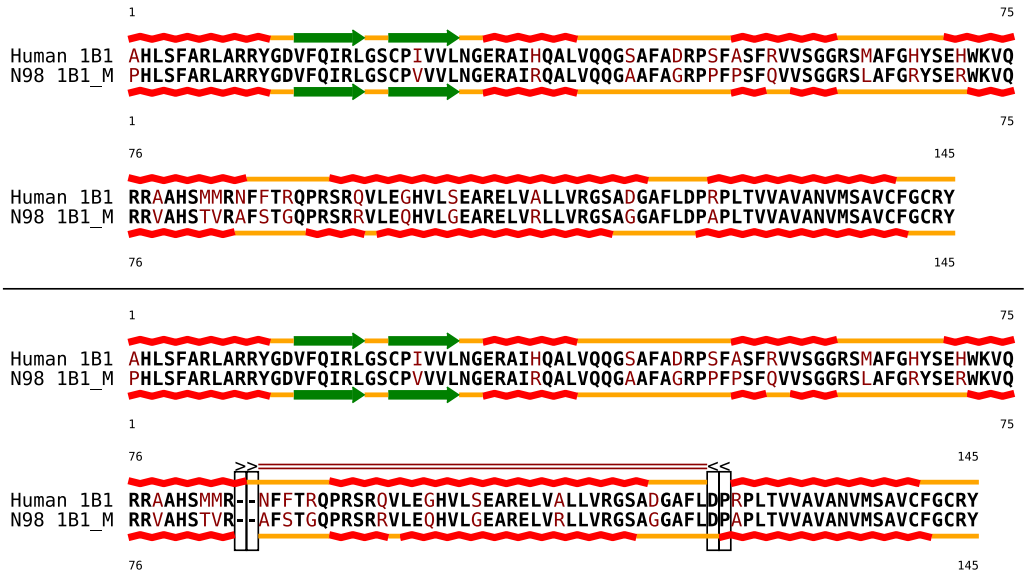

Fig. 2 Alignment (top) and Bi-alignment (bottom) of 145 N-terminal amino acids of two CYP1B1 cytochrome P450 enzymes: the extant human enzyme (Human 1B1) and the corresponding ancestral mammalian cytochrome (N98 1B1_M). See Fig. 1 for the representation of the alignments and secondary structure elements. Only the bi-alignment properly aligns the 'shifted' fifth helix and explains the structural incongruence by evolutionary shifts (two forward and two backward shifts

Not only is incongruent evolution of interest as an under-studied aspect of evolutionary dynamics, but it has practical implications for data analysis. Incongruences impact our ability to detect and reconstruct consensus structures, since corresponding structural features are formed by evolutionarily unrelated nucleotides or amino acids, while homologous sequence positions form disparate structural elements. This means that (in the presence of incongruent evolution) a single multiple sequence alignment cannot simultaneously represent the similarities of sequence and structure. In particular, conserved structure can no longer be represented as 'consensus structure', i.e. as an annotation of the columns of a sequence alignment.

Bi-alignments

We recently introduced bi-alignments [12, 23] as a mathematically consistent way of describing incongruent evolutionary relationships. Bi-alignments are motivated by treating shifts between sequence and structure explicitly as evolutionary events. It is important to realize that it is not necessarily possible to find an optimal reconciliation of sequence and structure alignments by identifying shifts events *a posteriori* from a pair of sequence and structure alignments that have been computed *separately*. Instead, bi-alignments allow *simultaneously* predicting sequence and structure homologies and their relation. For this purpose, we define a bi-alignment to consist of two alignments (one based one sequence similarity, the other one based on structure similarity) that are related by a third alignment, which captures the shift events. All three constituent alignments contribute to a common score.

While bi-alignments have similarities to *combined sequence and structure alignments* (which also optimize a joint score for sequence and structure similarity), bi-alignments extend such models by supporting shift events explicitly. Combined sequence/structure alignments therefore can be interpreted as the limit case of bi-alignments where arbitrarily high shift penalties completely prohibit shift events. As important consequence, bi-alignments overcome the requirement of a consensus structure, which is the key assumption underlying combined sequence/structure alignments.

As their main purpose, bi-alignments provide a *coherent framework* to detect shift-like incongruences, i.e. a local "movement" of conserved structures relative to the underlying sequence. It is worth noting that the formal concept of bi-alignments is not tied to applications in structural biology. Instead, it can be seen as a way to quantify the effect of differences in scoring schemes that focus on different aspects of the same sequence. The only requirement for bi-alignments is a position-wise one-to-one correspondence between the two different representations of each input object.

In this contribution, we extend bi-alignments with linear costs to a more realistic model with *affine gap costs*. We will illustrate our algorithmic developments using protein sequences and their secondary structures as an

example, because the position-wise annotation of a secondary structure elements fits well with the framework of sequence alignments. The (artificial) example in Fig. 1 shows that incongruence between sequence and secondary structure can indeed be caused a few well-place substitutions. It also shows that bi-alignments are capable, at least in principle, to reconcile incongruent sequence and structure homologies and to identify shift events.

A *bi-alignment* is formally defined as an alignment relating two, generally different, alignments of the same objects.

Definition 1 A *bi-alignment* $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ consists of two pairwise alignments $\mathbb{U}$ and $\mathbb{V}$ of the objects **a** and **b** and an alignment $\mathbb{W}$ of $\mathbb{U}$ and $\mathbb{V}$.

In Fig. 1, $\mathbb{U}$ is a sequence alignment (shown in the second row with the secondary structure annotation above and below the two sequences), while $\mathbb{V}$ is an alignment of the two respective secondary structures (shown in the second row with the two corresponding sequences between them). The columns of $\mathbb{U}$ and $\mathbb{V}$ are then aligned by $\mathbb{W}$. Since the pairwise alignment of two pairwise alignments is equivalent to a 4-way alignment, bi-alignments can be thought of as multiple alignments $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$. The input objects **a** and **b** appear twice in $\mathbb{A}$, once regarded as sequence (represented by the one-letter amino acid codes) and once regarded as secondary structure (shown a position-wise glyphs). Bi-alignments therefore differ from "structure-aware" sequence alignments by replacing the *annotation* of sequence positions with a secondary structure features by an *alignment* of both the sequence and the string of structural features. Importantly, $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ completely determines the alignments of the sequences of **a** and **b** with their secondary structures (shown in the third row of Fig. 1 as the first and last pair of rows, respectively.) These alignments in general contain gaps that indicate how the conserved "consensus" structure is shifted compared to the sequence positions.

Assuming a *linear scoring model*, i.e. scores for $\mathbb{U}$, $\mathbb{V}$, and $\mathbb{W}$ that are additively composed from single column contributions, it can be shown that the 4-way alignment $\mathbb{A}$ is scored additively as well [12, 23]. Linear bi-alignment problems therefore can be exactly solved by dynamic programming [24, 25] in quartic time. In this contribution we are interested in bi-alignments that are scored with affine gap costs.

Alignments as regular multi-tape grammars

To address this problem, it is helpful to describe the structure of alignments by multi-tape grammars, see e.g. [26] for a more detailed, formal discussion. In the simplest case, sequence alignments can be represented as regular grammars of the form $A \to Ac \mid \epsilon$. The only non-terminal symbol A denotes a (pairwise) alignment, the terminal ϵ is the empty alignment, and the terminal c denotes an alignment column, which may be a (mis) match $\binom{\bullet}{\bullet}$, a deletion $\binom{\bullet}{-}$, or an insertion $\binom{-}{\bullet}\}$. Since alignments compare extant sequences rather than an ancestor/descendant pair, the two "indels" (insertion/deletion) are biologically indistinguishable and hence receive the same score. The grammar simply expresses the fact that alignment can be constructed step-by-step by adding a column to an alignment of prefixes. For linear scoring functions, the production $A \to Ac$ allows adding the score of c to the previously accumulated score of the alignment A. Denote by $M(x)$ the *optimal* score of an alignment of the prefixes $\mathbf{a}[1..x_1]$ and $\mathbf{b}[1..x_2]$. As noted e.g. in [27, 28], the index vector of the penultimate column of the alignment is $x - c$, where $\bullet$ is interpreted as 1 and the gap character $-$ as 0. The Needleman–Wunsch recursions [29] thus can be written in compact form (see also [24]) as

$$M(x) = \max_c M(x - c) + s(x, c) \quad \text{with} \quad M(0) = 0\,. \tag{1}$$

Notably, in the non-affine case, the scoring function $s(x, c)$ is completely determined by a single column.

Affine gap cost. While linear gap costs are not very realistic in sequence alignment [30], arbitrary gap costs algorithmically require an additional factor $O(n)$ in running time [31, 32] and are difficult to parametrize in practice. The *affine gap cost* model serves as a useful and convenient compromise that is most often used in practice. Here, the opening and the extension of a gap are scored differently. It is therefore necessary to distinguish three different non-terminal $A\binom{\bullet}{\bullet}$, $A\binom{\bullet}{-}$, $A\binom{-}{\bullet}$ designating alignments that end in a (mis)match, deletion, and insertion column, respectively. Again one obtains a regular grammar with analogous productions of the form $A_c \to A_{c'}c \mid \epsilon$ for the three non-terminals. Denote by $M(x; c)$ the optimal score of an alignment of the prefixes $\mathbf{a}[1..x_1]$ and $\mathbf{b}[1..x_2]$ with end column of type c. We can then write Gotoh's well-known recursions [33] for pairwise affine gap cost alignment in the following compact form:

$$M(x; c) = \max_{c'} M(x - c; c') + s(x, c', c) \tag{2}$$

with initial conditions $M(0, \binom{\bullet}{\bullet}) = 0$, $M\left(0, \binom{-}{\bullet}\right) = M\left(0, \binom{\bullet}{-}\right) = -\infty$. In principle this

formulation accommodates any scoring function $s(x, c', c)$ for which the column score depends on the gap pattern of the previous column. For instance, we could also score the closing of a gap separately.

Both the Needleman–Wunsch algorithm and the Gotoh algorithm run in $O(n^2)$ space and time. Recursion Eq. (1) also describes the dynamic programming algorithm for k-ary alignments [24, 25, 34, 35], which requires $O(n^k)$ space and time. The situation is more complicated, however, for affine gap costs. Sum-of-pairs scoring functions simply sum over the scores of all pairwise alignments contained in a given multiple alignment. Surprisingly, computing the optimal alignment of alignments with affine gap costs under the sum-of-pairs-model is NP-complete unless the number of sequences in the constituent alignments is bounded [36]. On the other hand, scoring models of the form of Eq. (2) are of practical interest in particular for $k = 3$ [37–39].

In this contribution we show that the bi-alignment model with affine gap costs for the constituent alignments can be solved in polynomial time by dynamic programming. As we shall see, the recursions are of the form of Eq. (2) but require a subtle re-definition of $M(x; c)$.

Theory

Bi-alignments

Recall that we define a bi-alignment as an alignment of alignments (Def. 1). It is well known that an alignment of alignments can be represented again as an alignment. This compositional structure of alignments is discussed formally in [40]. In our case, $\mathbb{A}$ is a 4-way alignment from which $\mathbb{U}$ (and $\mathbb{V}$) are obtained as "projections", i.e. by extracting the corresponding pair of rows and removing all columns consisting of a pair of gap characters. The alignment $\mathbb{W}$, on the other hand, is obtained by considering each column in $\mathbb{U}$ and $\mathbb{V}$ as a single letter; and moreover interpreting the columns of the form $\begin{pmatrix}-\\-\end{pmatrix}$ (i.e. the ones that are removed in the projections to $\mathbb{U}$ and $\mathbb{V}$) as gap characters.

The Bi-Alignment Problem for two input sequences **a** and **b** consists in optimizing

$$\text{score}(\mathbb{A}) = u(\mathbb{U}) + v(\mathbb{V}) + w(\mathbb{W}) \tag{3}$$

with given scoring functions u, v, and w. The special case where u, v, and w are linear scoring functions has been discussed in [12, 23].

The alignment $\mathbb{W}$ of $\mathbb{U}$ and $\mathbb{V}$ describes the shifts distinguishing $\mathbb{U}$ and $\mathbb{V}$ in the following manner. First, consider a match column α of $\mathbb{W}$. It consists of a pair of columns with gap patterns $c(\alpha)$ and $d(\alpha)$, respectively. Using their numerical interpretation, we observe that

$$s(\alpha) := |c_1(\alpha) - d_1(\alpha)| + |c_2(\alpha) - d_2(\alpha)| \tag{4}$$

measures whether none, one, or both input sequences are shifted relative to each other (Fig. 3). Insertions and deletions in $\mathbb{W}$ correspond to inserting an all-gap column $\begin{pmatrix}-\\-\end{pmatrix}$ into $\mathbb{U}$ or $\mathbb{V}$, respectively, and always lead to incongruences. We note, furthermore, that there is a one-to-one correspondence between the columns of $\mathbb{W}$ and the columns of the 4-way alignment $\mathbb{A}$. Thus we can count the number of shifts $s(\mathbb{A}) = \sum_{\alpha\in\mathbb{A}} s(\alpha)$. The alignment $\mathbb{A}$ contains sub-alignments $\mathbb{A}^{(\mathbf{aa})}$ and $\mathbb{A}^{(\mathbf{bb})}$ of the first and second input sequence with itself. Let us denote the number of indels in these two projected alignments by $\delta_{\mathbf{a}}$ and $\delta_{\mathbf{b}}$, respectively.

Lemma 2 *If* $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ *is a bi-alignment of* **a** *and* **b**, *then* $s(\mathbb{A}) = \delta_{\mathbf{a}} + \delta_{\mathbf{b}}$.

Proof For column α of $\mathbb{A}$ we write $\delta_{\mathbf{a}}(\alpha) := |c_1(\alpha) - d_1(\alpha)|$ and $\delta_{\mathbf{b}}(\alpha) := |c_2(\alpha) - d_2(\alpha)|$. Thus $\delta_{\mathbf{a}}(\alpha) = 1$ if α is an indel column in the projected self-alignment of **a**, and $\delta_{\mathbf{a}}(\alpha) = 0$ if α is a (mis)match column. Note that all-gap columns are omitted in the projection and thus do not contribute to the indel count. Thus $\delta_{\mathbf{a}} = \sum_{\alpha\in\mathbb{A}} \delta_{\mathbf{a}}(\alpha)$ correctly counts the indels in $\mathbb{A}^{(\mathbf{aa})}$. An analogous equality holds for $\delta_{\mathbf{b}}$. A comparison with Eq. (4) completes the proof. □

A natural scoring function for $\mathbb{W}$ is thus to penalize the total number of shifts, setting $w(\mathbb{A}) = -\Delta s(\mathbb{A})$. This amounts to computing the shift contribution for each column $\begin{pmatrix}c\\d\end{pmatrix}$ of $\mathbb{A}$ as $\text{shift}(c, d) = -\Delta|c - d|$ $= -\Delta\,(|c_1 - d_1| + |c_2 - d_2|)$.

Bi-alignments with affine gaps costs

Lemma 2 provides an alternative interpretation in terms of a simple linear score for $\mathbb{A}^{(\mathbf{aa})}$ and $\mathbb{A}^{(\mathbf{bb})}$. We can

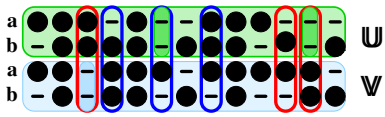

Fig. 3 Shifts in a bi-alignment. The bi-alignment consists of two alignments $\mathbb{U}$ or $\mathbb{V}$ (colored horizontal boxes) of the pair of objects **a** and **b** that are aligned with each other two different ways i.e. w.r.t. to two different objective functions. Since the actual letters in **a** and **b** are irrelevant for definition of shifts, we distinguish only letters (filled circles) and gaps (dashes). Note that **a** and **b** may be represented by different alphabets in **U** and **V**. Insertions and deletions in the alignment of alignments $\mathbb{W}$, i.e. the alignment of the columns of $\mathbb{U}$ with the columns of $\mathbb{W}$, are (highlighted by darker colors) correspond to all-gap columns in either $\mathbb{U}$ or $\mathbb{V}$. Aligned columns in $\mathbb{W}$ are shifts if the gap patterns in the upper pair and the lower pair differ. Colored outlines distinguish single (blue) and double shifts (red)

therefore think of Eq. (3) as a restricted sum-of-pairs model in which only four of the six pairwise alignments in $\mathbb{A}$ contribute. In this picture it is natural to assume that the constituent alignments $\mathbb{U}$ and $\mathbb{V}$ are scored with affine gap costs. In the light of the NP-hardness result of [36] it is not at all obvious, however, that the bi-alignment problem with affine gap costs can be solved in polynomial time.

In order to address this problem, we first recall the language of multi-way alignments. The following statement is "folklore", see e.g. [40]: every column of the 4-way alignment $\mathbb{A}$ is uniquely determined by

i. a four-dimensional index (x, y) identifying the prefixes $\mathbf{a}[1..x_1]$, $\mathbf{b}[1..x_2]$, $\mathbf{a}[1..y_1]$, and $\mathbf{b}[1..y_2]$ that are aligned up to the focal column.
ii. a gap pattern $(c, d) = ((c_1, c_2), (d_1, d_2))$ specifying whether the entry in a column is a letter or a gap character.

The language of 4-way alignments is generated by the regular language $A \to A\binom{c}{d} \mid \epsilon$, where the non-terminal A denotes a bi-alignment and the terminals $\binom{c}{d}$ correspond to one of the 15 possible gap patterns in a column of elements (excluding the all-gap column). Note that $c = \binom{-}{-}$ and $d = \binom{-}{-}$ respectively correspond to an insertion and deletion in $\mathbb{W}$, while $c, d \neq \binom{-}{-}$ corresponds to a match in $\mathbb{W}$. This regular language is sufficient for linear gap cost models [12, 23].

In order to handle affine gap costs for $\mathbb{U}$ and $\mathbb{V}$, we need to keep track of the gap patterns of the preceding alignment column in $\mathbb{U}$ and $\mathbb{V}$. This is *not* the same as considering the preceding column of $\mathbb{A}$ because gap patterns of the form $\binom{\binom{-}{-}}{d}$ and $\binom{c}{\binom{-}{-}}$ correspond to all-gap columns, which are removed in $\mathbb{U}$ or $\mathbb{V}$. Thus, we introduce a new notion of column type to address these 'preceding' gap patterns of the sub-alignments.

Definition 3 The *end column type* (p, q) of a bi-alignment $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ consists of the gap pattern p of the last column of $\mathbb{U}$ and the gap pattern q of the last column of $\mathbb{V}$. The end column type of the empty alignment is left arbitrary.

The definition is illustrated in Fig. 4. Note that by construction, neither p nor q consist only of gaps.

Now, we define a column-wise scoring function that captures the alignment score with affine gap cost. It

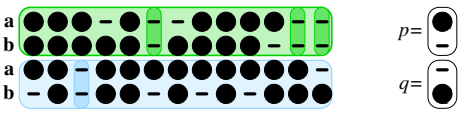

Fig. 4 The *end column type* of an bi-alignment is defined by the last column of each of the constituent pairwise alignments of **a** and **b** that is not an all-gap column

scores a single column of a bi-alignment $\mathbb{A}$, characterized by $\binom{x}{y}$ and $\binom{c}{d}$, depending on the end column type $\binom{c'}{d'}$ of the previous column. This function has the form

$$\begin{aligned}
&\text{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right) \\
&= \text{score}_{\mathbb{U}}(x, c', c) \\
&\quad + \text{score}_{\mathbb{V}}(y, d', d) \\
&\quad + \text{shift}(c, d)\text{withscore}(x, c', 0) \\
&= \text{score}(y, d', 0) = 0
\end{aligned} \tag{5}$$

Since $\text{score}(x, c', 0)$ and $\text{score}(y, d', 0)$, respectively, correspond to all-gap columns in $\mathbb{U}$ and $\mathbb{V}$, we observe that the sum of the $\text{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right)$ over all columns of $\mathbb{A}$ equals

$$\begin{aligned}
&\sum_{(x,c)\in\mathbb{U}} \text{score}_{\mathbb{U}}(x, c', c) + \sum_{(y,d)\in\mathbb{V}} \text{score}_{\mathbb{V}}(y, d', d) \\
&+ \sum_{(c,d)\in\mathbb{W}} \text{shift}(c, d) \\
&= u(\mathbb{U}) + v(\mathbb{V}) + \text{shift}(\mathbb{A})
\end{aligned} \tag{6}$$

Thus, Eq. (5) correctly scores the bi-alignment with general affine gap costs for both $\mathbb{U}$ and $\mathbb{V}$.

In order to derive a dynamic programming algorithm that solves the bi-alignment problem with this type of scoring function, we consider a decomposition of the search space in grammar form. The non-terminals $A_{(p,q)}$ correspond to bi-alignment with end column type (p, q). The terminals are the 15 possible column types of a 4-way alignment, which we write as $\binom{p}{q}$, with $p, q \neq \binom{-}{-}$ as well as $\binom{-}{q}\binom{q}{-}$ where the $-$ in the latter is a shorthand for $\binom{-}{-}$. In addition, we write ϵ for the empty 4-way column.

Lemma 4 *The language of bi-alignments with fixed end column type is generated by the productions*

$$A_{(p,q)} \to A_{(p',q')}\binom{p}{q} \mid A_{(p,q')}\binom{-}{q} \mid A_{(p',q)}\binom{p}{-} \mid \epsilon \tag{7}$$

Proof Consider an alignment $\mathbb{A}$ with last column (c, d) and end column type (p, q), and denote by $\mathbb{A}'$ the alignment without the last column. If $c, d \neq \binom{-}{-}$, i.e. the (mis)match case in $\mathbb{W}$, then $p = c$ and $q = d$ and $\mathbb{A}'$ may have any end-column type. If $c = \binom{-}{-}$, corresponding to the insertion case in $\mathbb{W}$, $\mathbb{A}$ inherits the first component c of its end column type from the previous alignment $\mathbb{A}'$. The other component is given by the second part of the last column, i.e. $d = q$. Thus the second component of the end column type of $\mathbb{A}'$ is arbitrary. The case $d = \binom{-}{-}$, deletion in $\mathbb{W}$ analogously yields $d = q$ and an end column type (p', q) for the $\mathbb{A}'$. □

Note that this grammar would allow terminating with any end column type. This is undesirable since we would like the first column to be scored as it was preceded by a match column in both $\mathbb{U}$ and $\mathbb{V}$. This is easily implemented by an appropriate initialization for $x = y = 0$, however.

Definition 5 Let $M_{p,q}(x, y)$ denote the optimal score of a 4-way alignment with end column type (p, q).

In order to enforce that empty alignment is treated as having end column type $\left(\binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}\right)$, we set $M_{\left(\binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}\right)}(0,0) = 0$ and $M_{(c,d)}(0,0) = -\infty$ for $(c, d) \neq \left(\binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}\right)$.

Theorem 6 *The matrices $M_{p,q}$ satisfy the recursion*

$$M_{(p,q)}(x,y) = \max \begin{cases} \max\limits_{\substack{p' \neq 0 \\ q' \neq 0}} M_{(p',q')}(x-p, y-q) + \text{score}\left(\binom{x}{y}, \binom{p'}{q'}, \binom{p}{q}\right) \\ \max\limits_{p' \neq 0} M_{(p',q)}(x-p, y) + \text{score}\left(\binom{x}{y}, \binom{p'}{q}, \binom{p}{0}\right) \\ \max\limits_{q' \neq 0} M_{(p,q')}(x, y-q) + \text{score}\left(\binom{x}{y}, \binom{p}{q'}, \binom{0}{q}\right) \end{cases} \tag{8}$$

Proof We first note that every column of $\mathbb{A}$ is either a (mis)match or an indel column w.r.t. $\mathbb{W}$. These correspond to the first three alternative productions in Eq. (7), and cover all alternatives. Since $\text{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right)$ depends only on the current column and the end column type, we obtain the optimal score of an alignment $\mathbb{A}$ with end column type (p, q) and last column (c, d) as the optimal score of an alignment $\mathbb{A}'$ with any of the matching column type plus the score $\text{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right)$ for the last column. The grammar in Eq. (7) specifies which end column types match. Furthermore, we note that, in the match case, the indices (x', y') of the last column of the alignment to the left are given by $x - p$ and $x - q$, where (p, q) is gap pattern on the last column of $\mathbb{A}$. Correspondingly we have $(x', y') = (x - p, y)$ for the insertion case and $(x', y') = (x, y' - q)$ in the insertion case. Taken together, this established the correctness of the recursion. □

As an immediate consequence we have

Corollary 7 *The bi-alignment problem with affine gap cost models for the two constituent alignments can be solved in $O(n^4)$ time and space.*

Affine shift costs

While bi-alignment with affine gap cost and linear shift costs may be of the most obvious practical relevance, we also discuss two variations with affine shift costs. First of all, we clarify how to attribute affine shift cost in our bi-alignment scoring model.

Let's take a step back to our original definition of the bi-alignment score (Eq. 3) and our previous suggestion to define the "shift" score component $w(\mathbb{A})$ as $-\Delta s(\mathbb{A})$, i.e. as a multiple of $s(\mathbb{A})$. Since the latter was defined as the number of gap columns in the alignments $\mathbb{A}^{(\mathbf{aa})}$ and $\mathbb{A}^{(\mathbf{bb})}$, this amounts to scoring shifts in a linear cost model, where every shift has a cost of Δ per column.

For affine shift costs, we take the view that every consecutive run of gap symbols in the pairwise alignments of the two copies of **a** and **b** represents one shift. This shift is scored in the same way as gaps are scored under affine gap cost, i.e. based on the shift opening cost Δ_o plus the shift extension cost Δ times the length of the shift (number of shift columns).

We first consider affine shift cost and non-affine (i.e. linear) gap cost. Since affine shifts are scored exactly in the same way as affine gaps, this situation is symmetric to the case of affine gap cost combined with linear shift cost. The corresponding bi-alignment problem can thus be solved efficiently by applying exactly the same idea as in our previous algorithm (Theorem 6), only now keeping track by p and q of the gap patterns in the respective alignments of rows 1&3 and 2&4. We immediately obtain

Corollary 8 *The bi-alignment problem with affine shift cost models (and linear gap cost) can be solved in $O(n^4)$ time and space.*

Combining affine gap and shift costs

More remarkably, we can even solve the general case of affine gap cost and affine shift cost in polynomial time by dynamic programming. Essentially, we combine the ideas of the above two algorithms. Our algorithm follows a grammar with general decomposition

$$A_p \to A_{p'}c \tag{9}$$

In order to evaluate affine gaps and affine shifts correctly at the same time, we need to know the last non-gap-only gap patterns of all four pairwise alignments of rows 1&2, 1&3, 2&4, and 3&4; thus, we utilize non-terminals A_p, for all p that encode the respective gap patterns $p = (p_{12}, p_{13}, p_{24}, p_{34})$. By the same argument as before, we can show this information to be sufficient to score shifts and gaps correctly in affine cost models for every possible last column c.

One keeps track of the correct gap patterns for all of the relevant pairwise alignments by setting the entries of p' as

$$p'_{ij} := \begin{cases} p_{ij} & c_i = - \text{ and } c_j = - \\ \binom{c_i}{c_j} & \text{otherwise} \end{cases} \tag{10}$$

for $ij \in \{12, 13, 24, 34\}$, depending on p and c in Eq. (9). For termination, we add the grammar rule:

$$A_{p^0} \to \epsilon \tag{11}$$

for $p^0 := \left(\binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}, \binom{\bullet}{\bullet}\right)$. This allows implicit accounting for gap and shift openings of respective gaps and shifts at the left end of alignment strings.

Remarks about generalizations and complexity Note that the existence of an efficient algorithm for general affine bi-alignment does not contradict the general hardness of multiple alignment with affine gap costs, even if it suggests the following generalization: Multiple (k-way) alignment with affine gap costs can be computed by dynamic programming following the above idea of keeping track of the right-most non-gap-only gap-patterns in all pairwise alignments. This requires considering $\binom{k}{2}$ many pairwise gap patterns, each out of three possibilities $\binom{\bullet}{\bullet}, \binom{\bullet}{-}, \binom{-}{\bullet}$. The resulting DP-algorithm for k-way alignment thus needs exponentially many matrices in k.

In bi-alignments of two sequences, we need to consider only four gap patterns, two for the two alignments and two for the shifts between the sequence copies. That is, there are (at most) $3^4 = 81$ combinations, which have to be represented by different matrices for the DP algorithm. This gets a little more practical, since many of these combinations cannot occur in valid bi-alignments. For example, having gap patterns $\binom{\bullet}{\bullet}$ for both alignments of **a** and **b**, rules out all patterns for the alignments of the copies that contradict having last columns $\binom{\bullet}{\bullet}\binom{\bullet}{\bullet}$, $\binom{\bullet}{-}\binom{\bullet}{-}$, or $\binom{-}{\bullet}\binom{-}{\bullet}$. Consequently, we find only 51 consistent gap pattern combinations, while we can proof 30 combinations inconsistent due to an analogous argument as sketched above.

Sub-additive gap costs

The affine gap cost model, despite its algorithmic convenience, has been criticized because empirical gap

length distributions usually are power laws thus suggesting a logarithmic gap costs [41]. However, gap costs of the form $w(\ell) = a + b\ell + c\ln\ell$ seem to yield better alignments in practice [42]. Pairwise alignments with subadditive gap costs can be computed by dynamic programming, considering insertions and deletions of arbitrary length:

$$M(x_1,x_2) = \max \begin{cases} M(x_1-1,x_2-1) + s(x_1,x_2) \\ \max_{\ell\geq 1} M(x_1-\ell,x_2) + w(\mathbf{a}[x_1-\ell+1..x_1]) \\ \max_{\ell\geq 1} M(x_1,x_2-\ell) + w(\mathbf{b}[x_2-\ell+1..x_2]) \end{cases} \tag{12}$$

This idea does not seem to generalize to bi-alignments. It is possible, however, to generalize the end column type. Instead of only distinguishing $\binom{1}{1}$, $\binom{1}{0}$, $\binom{0}{1}$, we can make each of them length dependent. This allows us to write the end column types $\langle p,\ell\rangle$, where $\ell \geq 1$ is the length of the run of columns of type p at the end of the alignment. With this notation we can write

$$\begin{aligned} M_{\langle p,\ell\rangle}(x) &= M_{\langle p,\ell-1\rangle}(x) + d(x,p,\ell) \text{ for } \ell \geq 1 \\ M_{\langle p,0\rangle}(x) &= \max_{p'\neq p} M_{\langle p',\ell\rangle}(x) \end{aligned} \tag{13}$$

with initial condition $M_{\langle p,0\rangle}(0) = 0$. Here $d(x,p,\ell)$ equals the match score $s(x)$ for $p = \binom{\bullet}{\bullet}$. For deletions, $p = \binom{\bullet}{-}$, we have $d(x,p,\ell) = w(\mathbf{a}[x_1-\ell+1..x_1]) - w(\mathbf{a}[x_1-\ell+1..x_1-1])$. The extensions of an insertion is scored by an analogous expression. The auxiliary entries $M_{\langle p,0\rangle}(x)$ are used to correctly score alignments in which the last column is different from the previous end gap pattern. This recursion runs in cubic time, but also requires cubic space (instead of quadratic space). For our purposes, however, it has the advantage that the score is again defined column-wise albeit at the expense of having to keep track of a linear instead of a constant number of end gap types. It generalizes to a recursion with four indices to compute the optimal bi-alignment.

Computational results

We implemented the bi-alignment algorithm with affine gap cost (Corollary 7) in Python 3. For improved performance, we adapted time-critical parts of the code to the Python C-extension Cython with some carefully chosen static typing. The new implementation was based on our previous implementation for RNA bi-alignment with linear gap cost [12, 23]. Like the earlier version, it allows the user to limit the number of positions either sequence can be shifted to the left or right against its own structure by a constant λ. The restricted recursions, following in essence the idea of [34, 35], have time complexity of $O(n^2\lambda^2)$ instead of the unrestricted, but often impractical complexity $O(n^4)$. In addition to efficient bi-alignment with affine gap cost, new features have been added to the software:

1. Protein sequences may be scored with an arbitrary, user-defined similarity matrix. The `BLOSUM62` matrix [43, 44] is supplied as default.
2. Protein secondary structures are scored using a simple bonus (here, 800) for matched secondary structure.
3. The dynamic programming matrices are stored as sparse matrices to to limit space consumption to $O(n^2\lambda^2)$ (compared to $O(n^4)$ space complexity of a hypothetical non-sparse implementation).
4. In case of ambiguity, simpler shifts are preferred (For example the bi-alignment of Fig 1 has co-optima with shift events in both sequences or shift events that shift longer sub-sequences).
5. Improved graphical output of bi-alignments. Figures 1 and 5 were produced using a Jupyter notebook that is included in the software distribution.
6. A flexible command line and Python interface is included.

As a proof of concept we generated optimal bi-alignments of DNA Polymerase I from *Escherichia* (length 928) and *Xanthomonas hortorum* (length 933), while allowing shifts of sequence against structure by up to two positions to the left or to the right in either protein ($\lambda = 2$). On an Intel(R) Core(TM) i7-10810U CPU, this (single-threaded) computation took 37 min. Note that a simple banding strategy on insertions and deletions could dramatically speed up such computations, typically without sacrificing alignment quality. The analogous computation allowing only one shift positions ($\lambda = 1$) was performed in 10.5 min. Due to filling 9 dynamic programming matrices and considering 15 recursion cases per entry, the same implementation still takes 26 s, if shifts are completely forbidden ($\lambda = 0$).

Figure 5 shows the resulting bi-alignment for $\lambda = 2$. For comparison, the results from $\lambda = 1$ and $\lambda = 0$ are given in Additional file 1. We chose a rather moderate shift cost $\Delta = -210$, compared to a bonus of 800 per structure match as well as gap extension and gap opening costs of

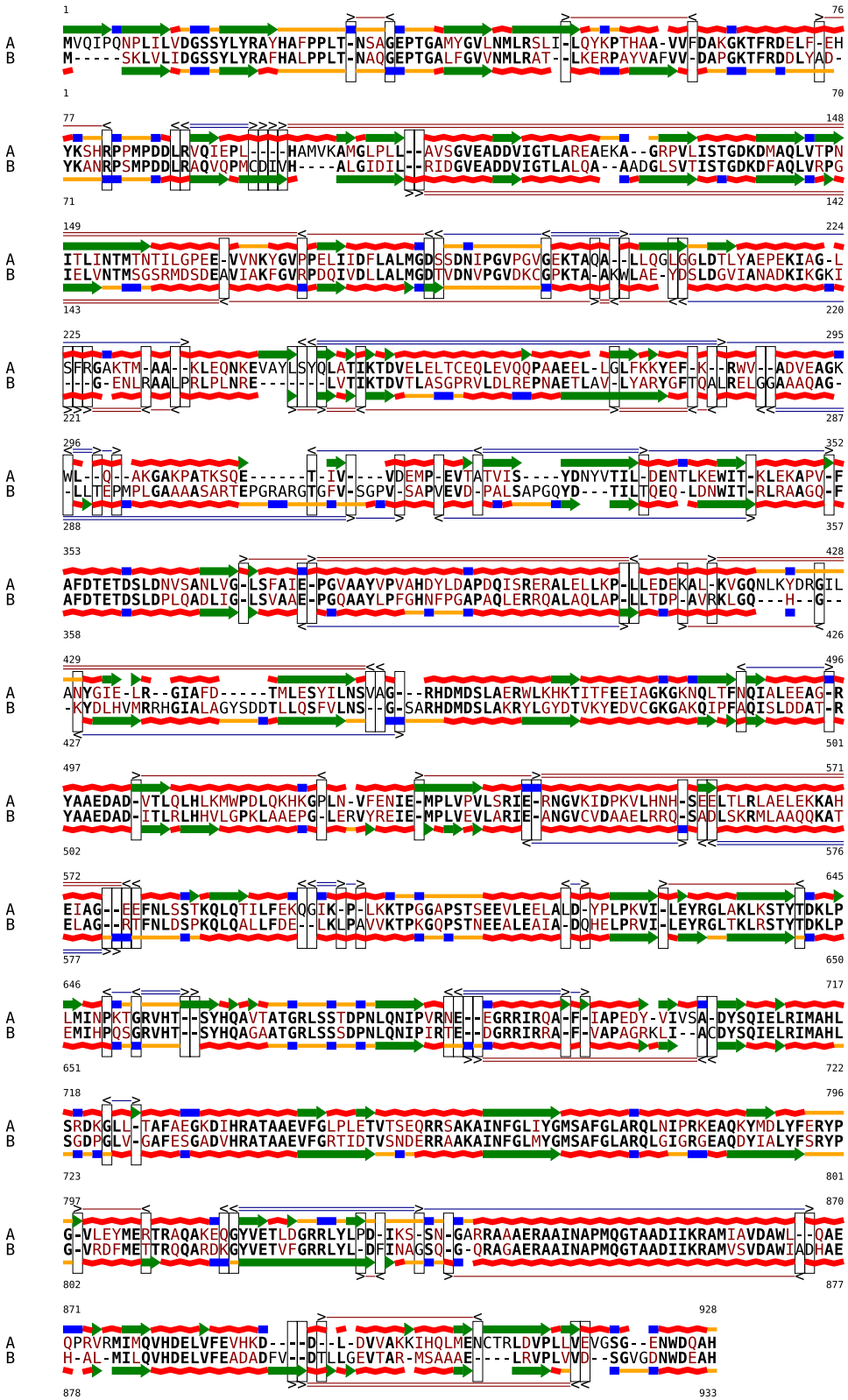

Fig. 5 Bi-alignment of the proteins DNA Polymerase I of *Escherichia* (`WP_016262675.1`) and *Xanthomonas hortorum* (`WP_095575020.1`). We use the same representation as in Fig. 1

−50 and −200, respectively. While we suspect that this parameter choice is too generous, it serves here to demonstrate that the algorithm readily predicts shifts that improve the compromise between primary and secondary structure alignment. The estimation of realistic shift costs is a non-trivial problem beyond the scope of this contribution.

Concluding remarks

We have shown here that bi-alignments with affine gap cost models for both constituent alignments and linear shift costs can be computed in quartic time by dynamic programming. Moreover, limiting the number of shifts to a constant reduces the cost to quadratic space and time. This makes the detection of locally-confined shifts computationally feasible for sequences of with length of realistic proteins or mRNAs. While we have illustrated our algorithmic innovations here using amino acid sequences and protein secondary structures as an example, the algorithm and its implementation is applicable to any linear representation of monomer-wise features along a biopolymer. In can be used, for instance, directly as an extension of the linear-gap-cost bi-alignments of RNAs described in [12].

We have focused here on the analysis of optimization problem and development of the algorithm. In addition to cost models for the constituent alignments $\mathbb{U}$ and $\mathbb{V}$, a bi-alignment problem also requires the specification of the shift costs, i.e. the scoring model for $\mathbb{W}$. Even though the scoring systems for $\mathbb{U}$ and $\mathbb{V}$ are borrowed from other studies, the choice of appropriate shift parameters remains an open problem for future work. This is a difficult problem for two reasons: (i) There is, at present, no collection of test cases with known shifts of sequences versus secondary structure for either proteins or RNAs that could be used to optimize the parameters. (ii) A biologically sound survey of proteins should presumably use a more elaborate scoring model for secondary structure elements that distinguishes amino acid positions depending on the distance from the element's ends. It stands to reason that the choice of the scoring model for the secondary structures would substantially influence estimates of the shift costs. Here, we are therefore content with a solution of algorithmic issues and a reference implementation. This provides the necessary tools for an in-depth empirical study of incongruent evolution of protein secondary structures in the future.

The formal framework of bi-alignments, Eq. (3), is much more general than the position-wise scoring models corresponding to regular multi-tape grammars. These were studied here because the corresponding optimization problems can be solved exactly by means of relatively simple dynamic programming algorithms. In a more general setting, one may want to consider $\mathbb{V}$ as an alignment of contact structures [45] or as an alignment of ordered sequences of 3D points, e.g. scored in terms of euclidean distances [46, 47]. This is of increasing practical interest as recent advances in protein folding [48, 49] provide access to high quality 3D structure predictions. The availability of accurately predicted protein structures of course also yields secondary structures, e.g. with the help of `DSSP` [50], which could be used for a systematic survey of incongruences in protein secondary structures. Alternatively, it seems promising to modify existing solutions to the protein structure alignment problems [51] to the corresponding bi-alignment problems. It is not obvious whether such a joint sequence and structure alignment problem implicitly contains a sequence-to-structure threading problem, which is known to be NP-complete [52]. In another forthcoming study, we are considering the corresponding problem for RNA secondary structures. In this case, the bi-alignment problem is amenable to a DP approach related to Sankoff's algorithm for the simultaneous folding and alignment of RNAs [53].

In [12] we further generalized bi-alignments to poly-alignments comprising $k > 2$ pairwise alignments $\mathbb{U}^{(i)}$, $1 \le i \le k \ge 2$ that are connected by a k-way alignment $\mathbb{W}$. Each of the alignments $\mathbb{U}^{(i)}$ then describes one particular aspect of the sequence. In addition to the individual amino acids and secondary structure elements, these may represent comparisons of profiles of physico-chemical parameters. It is not difficult to see that the grammar Eq. (7) generalizes to this case by defining end gap types $(p_1, p_2, \dots p_k)$ with $p_i \neq \binom{-}{-}$. The corresponding grammar then needs to consider all 2^k gap patterns for the last column of the k-way alignment $\mathbb{W}$. Optimal poly-alignments comprising k pairwise alignments with affine gap costs and additive cost contributions for the shifts between each pair of constituent alignments thus can be computed exactly in $O(n^{2k})$ space and time. Complementarily, one may consider alignments $\mathbb{U}$ and $\mathbb{V}$ of more than two sequences and their corresponding structures. The scoring of $\mathbb{W}$ then must accommodate more complex shift patterns, whose total number again increases exponentially in k. It is unlikely, therefore, that exact dynamic programming algorithms for these generalized problems will be practical. This begs the question whether poly-alignment problems can be approximated e.g. by progressive alignment schemes in a manner that is satisfactory from an applications point of view.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13015-022-00219-7.

Additional file 1. Bi-alignments with different choices of λ. Bi-alignments of the same data as in Fig. 5 using a more restrictive value (λ = 1) and a shift-free alignment (λ = 0). The latter corresponds to regular protein alignment with scores augmented by (mis)matches of the predicted secondary structure.

Acknowledgements
We thank Christian Höner zu Siederdissen for stimulating discussions and Verena Bender and Leonie Preker for preliminary work as part of the course *Advanced Methods in Bioinformatics* in 2020.

Author contributions
Both authors contributed to deriving the mathematical results, the interpretation of results and the writing of the manuscript. SW implemented the algorithms. Both authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL. This work was supported in part by the German Research Foundation (DFG) as part of the Collaborative Research Centre 1423 "Structural Dynamics of GPCR Activation and Signal Transduction" (SFB 1423/1 421152132).

Availability of data and materials
Implementations of the algorithms used in this contribution are available as free software from https://github.com/s-will/BiAlign/releases/tag/v0.3. For easy installation, we provide packages `bialign` on the Conda channel `bioconda` and the Python Package Index `PyPI`, respectively.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Universität Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany. [2]Competence Center for Scalable Data Services and Solutions Dresden/Leipzig, Interdisciplinary Center for Bioinformatics, German Centre for Integrative Biodiversity Research (iDiv), and Leipzig Research Center for Civilization Diseases, Universität Leipzig, Augustusplatz 12, 04107 Leipzig, Germany. [3]Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, 04109 Leipzig, Germany. [4]Department of Theoretical Chemistry, University of Vienna, Währinger Straße 17, 1090 Vienna, Austria. [5]Facultad de Ciencias, Universidad National de Colombia, Sede Bogotá, Ciudad Universitaria, 111321 Bogotá, D.C., Colombia. [6]Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA. [7]AMIBio, Laboratoire d'Informatique de l'École Polytechnique (LIX), Institute Polytechnique de Paris (IP Paris), Batiment Turing, 1 rue d'Estienne d'Orves, 91120 Palaiseau, France.

Received: 18 November 2021 Accepted: 1 May 2022
Published online: 16 May 2022

References

1. Wagner GP. Homology, genes, and evolutionary innovation. Princeton: Princeton Univ. Press; 2014.
2. Hofacker IL, Fekete M, Stadler PF. Secondary structure prediction for aligned RNA sequences. J Mol Biol. 2002;319:1059–66. https://doi.org/10.1016/S0022-2836(02)00308-X.
3. Marks DS, Hopf TA, Sander C. Protein structure prediction from sequence variation. Nat Biotechnol. 2012;30:1072–80. https://doi.org/10.1038/nbt.2419.
4. Chapman MA, Donaldson IJ, Gilbert J, Grafham D, Rogers J, Green AR, Göttgens B. Analysis of multiple genomic sequence alignments: a web resource, online tools, and lessons learned from analysis of mammalian SCL loci. Genome Res. 2004;14:313–8. https://doi.org/10.1101/gr.1759004.
5. Hiller M, Findeiß S, Lein S, Marz M, Nickel C, Rose D, Schulz C, Backofen R, Prohaska SJ, Reuter G, Stadler PF. Conserved introns reveal novel transcripts in *Drosophila melanogaster*. Genome Res. 2009;19:1289–300. https://doi.org/10.1101/gr.090050.108.
6. Stoltzfus A, Logsdon JM Jr, Palmer JD, Ford DW. Intron "sliding" and the diversity of intron positions. Proc Natl Acad Sci USA. 1997;94:10739–44. https://doi.org/10.1073/pnas.94.20.10739.
7. Lehmann J, Eisenhardt C, Stadler PF, Krauss V. Some novel intron positions in conserved *Drosophila* genes are caused by intron sliding or tandem duplications. BMC Evol Biol. 2010;10:156. https://doi.org/10.1186/1471-2148-10-156.
8. Bocco S, Csűrös M. Splice sites seldom slide: intron evolution in oomycetes. Genome Biol Evol. 2016;8:2340–50. https://doi.org/10.1093/gbe/evw157.
9. Fekete E, Flipphi M, Ág N, Kavalecz N, Cerqueira G, Scazzocchio C, Karaffa L. A mechanism for a single nucleotide intron shift. Nucleic Acids Res. 2017;45:9085–92. https://doi.org/10.1093/nar/gkx520.
10. Hare EE, Peterson BK, Iyer VN, Meier R, Eisen MB. Sepsid even-skipped enhancers are functionally conserved in *Drosophila* despite lack of sequence conservation. PLoS Genet. 2008;4:1000106. https://doi.org/10.1371/journal.pgen.1000106.
11. Flamm C, Fontana W, Hofacker I, Schuster P. RNA folding kinetics at elementary step resolution. RNA. 2000;6:325–38. https://doi.org/10.1017/s1355838200992161.
12. Waldl M, Will S, Wolfinger MT, Hofacker IL, Stadler PF. Bi-alignments as models of incongruent evolution of RNA sequence and secondary structure. In: Cazzaniga P, Besozzi D, Merelli I, Manzoni L, editors. Computational intelligence methods for bioinformatics and biostatistics (CIBB 2019)q, vol. 12313. Lecture notes in computer science. Cham: Springer; 2020. p. 159–70. https://doi.org/10.1007/978-3-030-63061-4_15.
13. Chou PY, Fasman GD. Prediction of protein conformation. Biochemistry. 1974;13:222–45. https://doi.org/10.1021/bi00699a002.
14. Ashok Kumar T. CFSSP: Chou and Fasman secondary structure prediction server. Wide Spectr Res J. 2013;1:15–9. https://doi.org/10.5281/zenodo.50733.
15. Bart AG, Harris KL, Gillam EMJ, Scott EE. Structure of an ancestral mammalian family 1B1 cytochrome P450 with increased thermostability. J Biol Chem. 2020;295:5640–53. https://doi.org/10.1074/jbc.RA119.010727.
16. Dong M, Ladavière L, Penin F, Deléage G, Baggetto LG. Secondary structure of P-glycoprotein investigated by circular dichroism and amino acid sequence analysis. Biochim Biophys Acta Biomembr. 1998;1371:317–34. https://doi.org/10.1016/S0005-2736(98)00032-7.
17. Schuster P, Fontana W, Stadler PF, Hofacker IL. From sequences to shapes and back: a case study in RNA secondary structures. Proc R Soc Lond B. 1994;255:279–84. https://doi.org/10.1098/rspb.1994.0040.
18. Babajide A, Hofacker IL, Sippl MJ, Stadler PF. Neutral networks in protein space: a computational study based on knowledge-based potentials of mean force. Fold Des. 1997;2:261–9. https://doi.org/10.1016/S1359-0278(97)00037-0.
19. Bornberg-Bauer E. How are model protein structures distributed in sequence space? Biophys J. 1997;73:2393–403. https://doi.org/10.1016/S0006-3495(97)78268-7.
20. Kabsch W, Sander C. On the use of sequence homologies to predict protein structure: identical pentapeptides can have completely different conformations. Proc Natl Acad Sci USA. 1984;81:1075–8. https://doi.org/10.1073/pnas.81.4.1075.
21. Schultes EA, Bartel DP. One sequence, two ribozymes: implications for the emergence of new ribozyme folds. Science. 2000;289(5478):448–52. https://doi.org/10.1126/science.289.5478.448.

22. Alexander PA, He Y, Chen Y, Orban J, Bryan PN. A minimal sequence code for switching protein structure and function. Proc Natl Acad Sci USA. 2009;106:21149–54. https://doi.org/10.1073/pnas.0906408106.
23. Waldl M, Will S, Wolfinger ML, Hofacker IL, Stadler PF. Bi-alignments as models of incongruent evolution and RNA sequence and structure. In: Cazzaniga P, Besozzi D, Merelli I, editors. CIBB'19 proceedings. 2019. p. 6. https://doi.org/10.1101/631606.
24. Sankoff D. Minimal mutation trees of sequences. SIAM J Appl Math. 1975;28:35–42. https://doi.org/10.1137/0128004.
25. Sankoff D. The early introduction of dynamic programming into computational biology. Bioinformatics. 2000;16:41–7. https://doi.org/10.1093/bioinformatics/16.1.41.
26. Höner zu Siederdissen C, Hofacker IL, Stadler PF. Product grammars for alignment and folding. IEEE/ACM Trans Comp Biol Bioinf. 2015;12:507–19. https://doi.org/10.1109/TCBB.2014.2326155.
27. Setubal JC, Meidanis J. Introduction to computational molecular biology. Boston: PWS Publishing Co.; 1997.
28. Retzlaff N, Stadler PF. Partially local multi-way alignments. Math Comp Sci. 2018;12:207–34. https://doi.org/10.1007/s11786-018-0338-4.
29. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48:443–53. https://doi.org/10.1016/0022-2836(70)90057-4.
30. Vingron M, Waterman MS. Sequence alignment and penalty choice: review of concepts, case studies and implications. J Mol Biol. 1994;235:1–12. https://doi.org/10.1016/S0022-2836(05)80006-3.
31. Waterman MS, Smith TF, Beyer WA. Some biological sequence metrics. Adv Math. 1976;20:367–87. https://doi.org/10.1016/0001-8708(76)90202-4.
32. Dewey TG. A sequence alignment algorithm with an arbitrary gap penalty function. J Comp Biol. 2001;8:177–90. https://doi.org/10.1089/106652701300312931.
33. Gotoh O. An improved algorithm for matching biological sequences. J Mol Biol. 1982;162:705–8. https://doi.org/10.1016/0022-2836(82)90398-9.
34. Carrillo H, Lipman D. The multiple sequence alignment problem in biology. SIAM J Appl Math. 1988;48:1073–82. https://doi.org/10.1137/0148063.
35. Lipman DJ, Altschul SF, Kececioglu JD. A tool for multiple sequence alignment. Proc Natl Acad Sci USA. 1989;86:4412–5. https://doi.org/10.1073/pnas.86.12.4412.
36. Kececioglu J, Starrett D. Aligning alignments exactly. In: Bourne PE, Gusfield D, editors. Proceedings of the 8th ACM conference on research in computational molecular biology (RECOMB). New York: ACM; 2004. p. 85–96. https://doi.org/10.1145/974614.974626.
37. Gotoh O. Alignment of three biological sequences with an efficient traceback procedure. J Theor Biol. 1986;121:327–37. https://doi.org/10.1016/S0022-5193(86)80112-6.
38. Konagurthu AS, Whisstock J, Stuckey PJ. Progressive multiple alignment using sequence triplet optimization and three-residue exchange costs. J Bioinf Comp Biol. 2004;2:719–45. https://doi.org/10.1142/S0219720004000831.
39. Kruspe M, Stadler PF. Progressive multiple sequence alignments from triplets. BMC Bioinform. 2007;8:254. https://doi.org/10.1186/1471-2105-8-254.
40. Berkemer SJ, Höner zu Siederdissen C, Stadler PF. Compositional properties of alignments. Math Comp Sci. 2021;15:609–30. https://doi.org/10.1007/s11786-020-00496-8.
41. Gonnet GH, Cohen MA, Benner SA. Exhaustive matching of the entire protein sequence database. Science. 1992;256:1443–5. https://doi.org/10.1126/science.1604319.
42. Cartwright RA. Logarithmic gap costs decrease alignment accuracy. BMC Bioinform. 2006;7:527. https://doi.org/10.1186/1471-2105-7-527.
43. Eddy SR. Where did the BLOSUM62 alignment score matrix come from? Nat Biotechnol. 2004;22:1035–6. https://doi.org/10.1038/nbt0804-1035.
44. Styczynski MP, Jensen KL, Rigoutsos I, Stephanopoulos G. BLOSUM62 miscalculations improve search performance. Nat Biotechnol. 2008;26:274–5. https://doi.org/10.1038/nbt0308-274.
45. Stadler PF. Alignments of biomolecular contact maps. Interface Focus. 2021;11:20200066. https://doi.org/10.1098/rsfs.2020.0066.
46. Poleksic A. Algorithms for optimal protein structure alignment. Bioinformatics. 2009;25:2751–6. https://doi.org/10.1093/bioinformatics/btp530.
47. Li SC. The difficulty of protein structure alignment under the RMSD. Algorithms Mol Biol. 2013;8:1. https://doi.org/10.1186/1748-7188-8-1.
48. ...Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Beck T, Petersen S, Reimann D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D. Highly accurate protein structure prediction with AlphaFold. Nature. 2021. https://doi.org/10.1038/s41586-021-03819-2.
49. ...Baek M, DiMaio F, Anishchenko I, Dauparas J, Ovchinnikov S, Lee GR, Wang J, Cong QC, Kinch LN, Schaeffer RD, Millán C, Park HP, Adams C, Glassman CR, DeGiovanni A, Pereira JH, Rodrigues AV, van Dijk AA, Ebrecht AC, Opperman DJ, Sagmeister T, Buhlheller C, Pavkov-Keller T, Rathinaswamy MK, Dalwadi U, Yip CKY, Burke JE, Garcia KC, Grishin NV, Adams PD, Read RJ, Baker D. Accurate prediction of protein structures and interactions using a three-track neural network. Science. 2021;373:871–6. https://doi.org/10.1126/science.abj8754.
50. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983;22:2577–637. https://doi.org/10.1002/bip.360221211.
51. Daniluk P, Lesyng B. Theoretical and computational aspects of protein structural alignment. 2014;1:557–98. https://doi.org/10.1007/978-3-642-28554-7_17.
52. Lathrop RH. The protein threading problem with sequence amino acid interaction preferences is NP-complete. Protein Eng Des Sel. 1994;7:1059–68. https://doi.org/10.1093/protein/7.9.1059.
53. Sankoff D. Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM J Appl Math. 1985;45:810–25. https://doi.org/10.1137/0145048.

Publisher's Note