Bi-Alignments with Affine Gaps Costs

Peter F. Stadler and Sebastian Will

Abstract. Bi-alignments were introduced recently to model incongruent evolution of different features of a biomolecules. They can be understood a alignment of two distinct pairwise alignments of the same entities, e.g., one modeling sequence similarity, the other structural similarity. Here we show that optimal bi-alignments with affine gap costs for two constituent alignments can be computed exactly in quartic space and time.

Mathematics Subject Classification (2000). 68R01, 03D25, 90C35.

Keywords. Dynamic Programming, Scoring Functions, Multi-tape Formal Grammar, Recursion.

1. Introduction

Many biopolymers, including RNA and proteins, require specific three-dimensional structures. These can often be detected as consensus structures from multiple sequence alignments [8, 14]. By definition, analogous structural features are thus formed by homologous sequence positions. This is not always the case, however. It is also possible that structural elements are shifted relative to the underlying sequence. In this situation of *incongruent evolution*, corresponding structural features are formed by evolutionarily unrelated nucleotides or aminoacid, while homologous sequence positions form disparate structural elements. As a consequence, an alignment annotated by a consensus structure is insufficient to model and represent the evolution of gene family. As a remedy, we recently introduced bi-alignments [20, 21]. Here the two alignments, one, $\mathbb{U}$, focusing on sequence conservation, the other one, $\mathbb{V}$, describing structural similarity, are represented separately. The shifts between them are modeled as an alignment $\mathbb{W}$ of the columns of $\mathbb{U}$ and the columns $\mathbb{V}$. A bi-alignment therefore can be represented as a 4-way alignment $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$, where $\mathbb{U}$ and $\mathbb{V}$ are (in general different) alignments of the same input sequences. Fig. 1 shows an example.

Assuming an linear scoring model, i.e., scores for $\mathbb{U}$, $\mathbb{V}$, and $\mathbb{W}$ that are completely determined by a single column, it can be shown that the 4-way alignment

```
StrA  CHHHHHHHHHHHHHCCCCTCEEEEEEECCTCEEEEEEEECCC
SeqA  RAKLPLKEKKLTATANYHPGIRYIMTGYSAKYIYSSTYARFR
       |||||||||||  |||||||||||||||||  ||||||| ||
SeqB  KAKLPLKEKKLTRTANYHPGIRYIMTGYSAKRIYSSTYAYFR
StrB  HHHHHHHHHHHHCCCCCCTCEEEEEEECCCCCEEEEEEEECC

SeqA  RAKLPLKEKKLTATANYH-PGIRYIMTGYSAK-YIYSSTYARFR
StrA  CHHHHHHHHHHHHHCCCC-TCEEEEEEECCTC-EEEEEEEECCC
       |||||||||||| ||||  |||||||||||  | ||||||||||
StrB  -HHHHHHHHHHHHCCCCCCTCEEEEEEECCCCCEEEEEEEECC-
SeqB  -KAKLPLKEKKLTRTANYHPGIRYIMTGYSAKRIYSSTYAYFR-
```

FIG. 1. Two peptide sequences and their predicted secondary structures (`H` helix, `T` turn, `E` β-sheet, `C` coil) predicted according to the Chou Fasman method with `CFSSP` []. The upper alignment traced sequence homology, 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 lower alignment maximizes structural similarity and thus shows little sequence similarity.

$\mathbb{A}$ is also scored additively [20, 21]. Linear bi-alignment problems therefore can be solved exactly by dynamic programming [2, 13].

It is helpful to describe the structure of alignments as multi-tape grammars, see e.g. [9]. In the simplest case we can write $A \to Ac \mid \epsilon$, where 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 $\binom{-}{\bullet}\}$. In the linear case, the production $A \to Ac$ adds the score of c. 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 [17, 16], the index vector of the penultimate column of the alignment is $x - c$ when $\bullet$ is interpreted as 1 and the gap character $-$ as 0. The Needleman-Wunsch recursions [15] thus can be written in compact form as

$$M(x) = \max_c M(x - c) + s(x, c) \qquad \text{with} \quad M(0) = 0 \tag{1.1}$$

with a scoring function $s(x, c)$ that is completely determined a single column.

Linear gap costs, however, are not very realistic in sequence alignment [18]. Arbitrary gap costs, on the other hand, algorithmically require an addition factor $O(n)$ in running time [22, 4] and are difficult to parametrize in practice. The affine gap cost model is 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 necessary, therefore, 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, resp. 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 write Gotoh's [6] well-known recursions for pairwise affine gap cost alignment in the following compact form:

$$M(x;p) = \max_{p'} M(x-c;c') + s(x,c',c) \tag{1.2}$$

with initial conditions $M(0,(\begin{smallmatrix}\bullet\\ \bullet\end{smallmatrix})) = 0$, $M(0,(\begin{smallmatrix}-\\ \bullet\end{smallmatrix})) = M(0,(\begin{smallmatrix}\bullet\\ -\end{smallmatrix})) = -\infty$. In principle this formulation accommodates any scoring function $s(x,c',c)$ for which a column score depends on the gap pattern of the previous column. For instance, we could also score the closing a gap separately.

Both the Needleman-Wunsch and the Gotoh algorithm run in $O(n^2)$ space and time. Recursion equ.(1.1) also describes the dynamic programming algorithm for k-ary alignments [2, 13], which requires $O(n^k)$ space and time. The situation is more complicated, however, for affine gap costs. Sum-of-pairs scoring functions simply 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 [10]. On the other hand, scoring models of the form equ.(1.2) are of practical interest in particular for $k = 3$ [7, 11, 12].

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 eq.(1.2) but require a subtle re-definition of $M(x;c)$.

2. Results

We start with a formal definition of bi-alignments and their scoring functions.

Definition 2.1. A bi-alignment $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ consists of an pairwise two alignments $\mathbb{U}$ and $\mathbb{V}$ of the sequences $\mathbf{a}$ and $\mathbf{b}$ and an alignment $\mathbb{W}$ of $\mathbb{U}$ and $\mathbb{V}$.

It is well known that an alignment of alignments can again be represented as an alignment, see [1] for a formal discussion of the compositional structure of alignments. 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.

A Bi-Alignment Problem for two input sequences $\mathbf{a}$ and $\mathbf{b}$ consists in optimizing

$$\text{score}(\mathbb{A}) = u(\mathbb{U}) + v(\mathbb{V}) + w(\mathbb{W}) \tag{2.1}$$

with given scoring functions u, v, and w. The special case where u, v, and w are linear scoring functions has been discussed in [20, 21]. In [19] we considered case where v scores the base pairs of a consensus RNA structure.

The alignment $\mathbb{W}$ 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

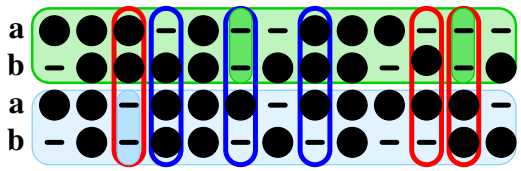

FIG. 2. Shifts in a bi-alignment. The bi-alignment consists of two alignments $\mathbb{U}$ or $\mathbb{V}$ (colored boxes) of $\mathbf{a}$ and $\mathbf{b}$ that are aligned with each other. Insertions and deletions in the alignment of alignments $\mathbb{W}$ are (highlighted by darker colors) correspond to all-gap columns. Aligned columns are shifts if they have different gap patterns. Colored outlines distinguish single (blue) and double shifts (red).

with gap patterns $c(\alpha)$ and $d(\alpha)$, respectively. Using their numerical interpretation observe that

$$s(\alpha) := |c_1(\alpha) - d_1(\alpha)| + |c_2(\alpha) - d_2(\alpha)| \tag{2.2}$$

measures whether none, one, or both input sequences are shifted relative to each other, Fig. 2 Insertions and deletions in $\mathbb{W}$ correspond to inserting an all-gap column $\binom{-}{-}$ into $\mathbb{U}$ or $\mathbb{V}$, respectively, and always lead to incongruences. We note, furthermore, that there is a 1-1 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 2.2. *Let* $\mathbb{A} \cong (\mathbb{U}, \mathbb{V}, \mathbb{W})$ *be a bi-alignment of* $\mathbf{a}$ *and* $\mathbf{b}$. $s(\mathbb{A}) = \delta_{\mathbf{a}} + \delta_{\mathbf{b}}$.

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 $\mathbf{a}$, 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 equ.(2.2) 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 of of $\mathbb{A}$ as $\text{shift}(c, d) = -\Delta\,|c - d|$. Lemma 2.2 provides an alternative interpretation in terms of a simple linear score for $\mathbb{A}^{(\mathbf{aa})}$ and $\mathbb{A}^{(\mathbf{aa})}$. We can therefore think of equ.(2.1) as a restricted sum-of-pair model in which only four of the six pairwise alignments in $\mathbb{A}$ contribute. In the light of the NP-hardness result of [10] it is not at all obvious that the bi-alignment problem with affine gap costs can be solved in polynomial time.

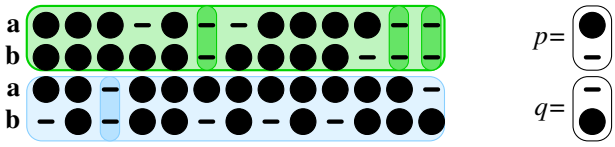

FIG. 3. The *end column type* of an a bi-alignment is defined by the last column of each of the constituent pairwise alignments of **a** and **a** that is not an all-gap column.

The following statement is "folklore", see e.g. [1]: 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 column of elements (excluding the all-gap column). Note that $c = \binom{-}{-}$ and $d = \binom{-}{-}$ correspond to an insertion and deletion, resp., 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 [20, 21].

In order to handle affine gap costs for $\mathbb{U}$ and $\mathbb{V}$, we need to keep track of the gap patterns of previous alignment column in $\mathbb{U}$ and $\mathbb{V}$. This is *not* the same as considering the previous column of $\mathbb{A}$ because gap patterns of the form $\left(\binom{-}{-}\right)$ and $\left(\binom{c}{-}\right)$ correspond to all-gap columns in $\mathbb{U}$ or $\mathbb{V}$. We therefore introduce

Definition 2.3. 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.

By construction, neither p nor q consists of gaps only. The definition is illustrated in Fig. 3. Now we consider a scoring function of the form

$$\begin{gathered}\mathrm{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right) = \mathrm{score}_{\mathbb{U}}(x, c', c) + \mathrm{score}_{\mathbb{V}}(y, d', d) + \mathrm{shift}(c, d) \\ \text{with}\, \mathrm{score}(x, c', 0) = \mathrm{score}(y, d', 0) = 0\end{gathered} \tag{2.3}$$

with $\mathrm{score}(x, c', 0) = \mathrm{score}(y, d', 0) = 0$, where (x, y) and (c, d) together determine column of $\mathbb{A}$ and (c', d') is the end column type of the previous column. Since $\mathrm{score}(x, c', 0) = \mathrm{score}(y, d', 0) = 0$ corresponds to all-gap columns in $\mathbb{U}$ and $\mathbb{V}$, we observe that the sum of the $\mathrm{score}\left(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d}\right)$ over all columns of $\mathbb{A}$ equals

$$\sum_{(x,c)\in\mathbb{U}} \mathrm{score}_{\mathbb{U}}(x, c', c) + \sum_{(y,d)\in\mathbb{V}} \mathrm{score}_{\mathbb{V}}(y, d', d) + \sum_{(c,d)\in\mathbb{W}} \mathrm{shift}(c, d) = u(\mathbb{U}) + v(\mathbb{A}) + \mathrm{shift}(\mathbb{A}) \tag{2.4}$$

Thus equ.(2.3) 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 functions 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 2.4. *The language of bi-alignments with fixed end column type is generated by the productions*

$$A_{(p,q)} \to A_{(p',q')}\binom{p}{q} \mid A_{(p,q')}\binom{-}{q} \mid A_{(p',q)}\binom{p}{-} \mid \epsilon \tag{2.5}$$

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 insert 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 2nd part of the last column, i.e., $d = q$. Thus the 2nd 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 allows a termination 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 $\mathbb{U}$. This is easily implemented by an appropriate initialization for $x = y = 0$, however.

Definition 2.5. 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 $((\bullet),(\bullet))$, we set $M_{((\bullet),(\bullet))}(0, 0) = 0$ and $M_{(c,d)}(0, 0) = -\infty$ for $(c, d) \neq ((\bullet),(\bullet))$.

Theorem 2.6. *The matrices $M_{p,q}$ satisfy the recursion*

$$M_{(p,q)}(x, y) = \max \begin{cases} \max\limits_{\substack{p'\neq 0\\ q'\neq 0}} M_{(p',q')}(x-p, y-q) + \text{score}(\binom{x}{y}, \binom{p'}{q'}, \binom{p}{q}) \\ \max\limits_{p'\neq 0} M_{(p',q)}(x-p, y) + \text{score}(\binom{x}{y}, \binom{p'}{q}, \binom{p}{0}) \\ \max\limits_{q'\neq 0} M_{p,q'}(x, y-q) + \text{score}(\binom{x}{y}, \binom{p}{q'}, \binom{0}{q}) \end{cases} \tag{2.6}$$

Proof. We first note that every column of $\mathbb{A}$ is either a (mis)match, insertion, or deletion column w.r.t. $\mathbb{W}$. These correspond to the first three alternative productions in equ.(2.5), and cover all alternatives. Since $\text{score}(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d})$ 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 score$(\binom{x}{y}, \binom{c'}{d'}, \binom{c}{d})$ for the last column. The grammar in equ.(2.5) 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 2.7. *The bi-alignment problem with affine gap cost models for the two constituent alignments can be solved in $O(n^4)$ time and space.*

2.1. 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 step back to our original definition of the bi-alignment score (Equ. 2.1) 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 2.6), 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 get

Corollary 2.8. *The bi-alignment problem with affine shift cost models (and linear gap cost) can be solved in $O(n^4)$ time and space.*

2.2. Combining Affine Gap and Shift Costs

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

decomposition

$$A_p \to A_{p'} c \tag{2.7}$$

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

$$p'_{ij} := \begin{cases} p_{ij} & c_i = - \text{ and } c_j = - \\ \binom{c_i}{c_j} & \text{otherwise} \end{cases} \tag{2.8}$$

for $ij \in \{12, 13, 24, 34\}$, depending on p and c in Equ. (2.7).

For termination, we add the grammar rule:

$$A_{p^0} \to \epsilon \tag{2.9}$$

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

Remark about generalizations and complexity. Note that the 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 $\begin{pmatrix}\bullet\\\bullet\\\bullet\\\bullet\end{pmatrix}$,$\begin{pmatrix}\bullet\\\bullet\\-\\-\end{pmatrix}$, or $\begin{pmatrix}-\\-\\\bullet\\\bullet\end{pmatrix}$. Consequently, we find only 51 consistent gap pattern combinations, while we can proof 30 combinations inconsistent due to an analogous argument as sketched above.

One can elaborate: Depending on the patterns of 1,2 & 3,4 there are only 3 different possible last columns (either satisfying both patterns or being $-,-$ for exactly one of them; the same holds for each pattern combination of 1,3 & 2,4. If one cannot generate some common last column in the two possible ways, the combination is inconsistent.

3. Sub-Additive Gap Costs

The affine gap cost model, despite its algorithmic convenience, has been critisized because empirical gap length distributions usually are power laws thus suggesting a logarithmic gap costs [5]. However, gap costs of the form $w(\ell) = a + b\ell + c\ln\ell$ seem to yield better alignments in practice [3]. Pairwise alignments with subadditive gap costs can be computed by dynamic programming, considering insertions and deletions of arbitrary length:

$$M(x_1,x_2) = \max\begin{cases} M(x_1-1,x_2-1) + s(x_1,x_2) \\ \max_{\ell\geq 1} M(x_1-\ell,x_2) + w(\mathbf{a}[x_1-\ell+1..x_1]) \\ \max_{\ell\geq 1} M(x_1,x_2-\ell) + w(\mathbf{b}[x_2-\ell+1..x_2]) \end{cases} \tag{3.1}$$

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

$$\begin{aligned} M_{\langle p,\ell\rangle}(x) &= M_{\langle p,\ell-1\rangle}(x) + d(x,p,\ell) \text{ for } \ell \geq 1 \\ M_{\langle p,0\rangle}(x) &= \max_{p'\neq p} M_{\langle p',\ell\rangle}(x) \end{aligned} \tag{3.2}$$

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 $d(x,p,\ell) = w(\mathbf{a}[x_1-\ell+1..x_1]) - w(\mathbf{a}[x_1-\ell+1..x_1-1])$ for $p = \binom{\bullet}{-}$. 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. The recursion runs in cubic time, but requires also cubic instead of quadratic memory. For our purposes, however, it has the advantage that the score is again defined column-wise albeith 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.

4. 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. Limiting the number of shifts to a constant thus reduces the cost to quadratic space and time.

In [21] we further generalized bi-alignments to poly-alignments comprising pairwise alignments $\mathbb{U}^{(i)}$, $1 \leq i \leq k \geq 2$ connected by a k-way alignment $\mathbb{W}$. It is not difficult to see that the grammar equ.(2.5) generalizes to this case by defining end gap types $(p_1, p_2, \ldots p_k)$ 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 constributions for the shifts between each pair of constituent alignments thus can be computed exactly in $O(n^{2k})$ space and time.

References

[1] S. Berkemer, C. Höner zu Siederdissen, and P. F. Stadler. Alignments as compositional structures. 2018. submitted; arXiv:1810.07800.

[2] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48:1073–1082, 1988.

[3] R. A. Cartwright. Logarithmic gap costs decrease alignment accuracy. *BMC Bioinformatics*, 7:527, 2006.

[4] T. G. Dewey. A sequence alignment algorithm with an arbitrary gap penalty function. *J. Comp. Biol.*, 8:177–190, 2001.

[5] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[6] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[7] O. Gotoh. Alignment of three biological sequences with an efficient traceback procedure. *J. theor. Biol.*, 121:327–337, 1986.

[8] I. L. Hofacker, M. Fekete, and P. F. Stadler. Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.*, 319:1059–1066, 2002.

[9] C. Höner zu Siederdissen, I. L. Hofacker, and P. F. Stadler. Product grammars for alignment and folding. *IEEE/ACM Trans. Comp. Biol. Bioinf.*, 12:507–519, 2015.

[10] J. Kececioglu and D. Starrett. Aligning alignments exactly. In P. E. Bourne and D. Gusfield, editors, *Proceedings of the 8th ACM Conference on Research in Computational Molecular Biology (RECOMB)*, pages 85–96, New York, NY, 2004. ACM.

[11] A. S. Konagurthu, J. Whisstock, and P. J. Stuckey. Progressive multiple alignment using sequence triplet optimization and three-residue exchange costs. *J. Bioinf. and Comp. Biol.*, 2:719–745, 2004.

[12] M. Kruspe and P. F. Stadler. Progressive multiple sequence alignments from triplets. *BMC Bioinformatics*, 8:254, 2007.

[13] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, 86:4412–4415, 1989.

[14] D. S. Marks, T. A. Hopf, and C. Sander. Protein structure prediction from sequence variation. *Nature Biotech*, 30:1072–1080, 2012.

[15] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48:443–453, 1970.

[16] N. Retzlaff and P. F. Stadler. Partially local multi-way alignments. *Math. Comp. Sci.*, 12:207–234, 2018.

[17] J. C. Setubal and J. Meidanis. *Introduction to computational molecular biology*. PWS Pub., 1997.

[18] M. Vingron and M. S. Waterman. Sequence alignment and penalty choice: Review of concepts, case studies and implications. *J. Mol. Biol.*, 235:1–12, 1994.

[19] M. Waldl, C. Flamm, T. Gatter, C. Höner zu Siederdissen, M. T. Wolfinger, S. Will, I. L. Hofacker, and P. F. Stadler. Incongruences between sequence and secondary structure alignments of nucleic acids. 2020. in preparation.

[20] M. Waldl, S. Will, M. Wolfinger, H. I. L., and P. F. Stadler. Bi-alignments as models of incongruent evolution and RNA sequence and structure. In *CIBB'19 Proceedings*, 2019. BioArxiv.

[21] M. Waldl, S. Will, M. Wolfinger, H. I. L., and P. F. Stadler. Bi-alignments as models of incongruent evolution of RNA sequence and secondary structure. In *Proceedings of the CIBB 2019*, Heidelberg, 2020. Springer. in preparation.

[22] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. *Adv. Math.*, 20:367–387, 1976.

Peter F. Stadler
Bioinformatics Group, Department of Computer Science; Interdisciplinary Center for Bioinformatics; Competence Center for Scalable Data Services and Solutions Dresden/Leipzig; German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig; Centre for Biotechnology and Biomedicine, and Leipzig Research Center for Civilization Diseases (LIFE); Leipzig University, Härtelstraße 16-18, D-04107 Leipzig, Germany

MPI Mathematics in the Sciences
Inselstraße 22, D-04103 Leipzig, Germany

Institute for Theoretical Chemistry, University of Vienna
Währingerstraße 17, A-1090 Wien, Austria

Facultad de Ciencias, Universidad National de Colombia, Bogotá, Colombia

Santa Fe Institute
1399 Hyde Park Rd., Santa Fe, NM 87501 USA
e-mail: `studla@bioinf.uni-leipzig.de`

Sebastian Will
Institute for Theoretical Chemistry, University of Vienna
Währingerstraße 17, A-1090 Wien, Austria

Institut d'Administration Adiabatique
e-mail: `will@tbi.univie.ac.at`