Math.Comput.Sci. (2018) 12:207–234
https://doi.org/10.1007/s11786-018-0338-4

Mathematics in Computer Science

Partially Local Multi-way Alignments

Nancy Retzlaff · Peter F. Stadler

Received: 9 September 2017 / Revised: 25 January 2018 / Accepted: 23 February 2018 / Published online: 19 March 2018

Abstract Multiple sequence alignments are an essential tool in bioinformatics and computational biology, where they are used to represent the mutual evolutionary relationships and similarities between a set of DNA, RNA, or protein sequences. More recently they have also received considerable interest in other application domains, in particular in comparative linguistics. Multiple sequence alignments can be seen as a generalization of the string-to-string edit problem to more than two strings. With the increase in the power of computational equipment, exact, dynamic programming solutions have become feasible in practice also for 3- and 4-way alignments. For the pairwise (2-way) case, there is a clear distinction between local and global alignments. As more sequences are considered, this distinction, which can in fact be made independently for both ends of each sequence, gives rise to a rich set of partially local alignment problems. So far these have remained largely unexplored. Here we introduce a general formal framework that gives raise to a classification of partially local alignment problems. This leads to a generic

NR gratefully acknowledges the hospitality of the Santa Fe Institute, where much of this work was performed.

N. Retzlaff · P. F. Stadler
MPI Mathematics in the Sciences, Inselstraße 22, 04103 Leipzig, Germany
e-mail: nancy@bioinf.uni-leipzig.de

N. Retzlaff · P. F. Stadler (✉)
Bioinformatics Group Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Leipzig University, Härtelstraße 16-18, 04107 Leipzig, Germany
e-mail: studla@bioinf.uni-leipzig.de

P. F. Stadler
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, Leipzig, Germany

P. F. Stadler
FHI Cell Therapy and Immunology, Perlickstraße 1, 04109 Leipzig, Germany

P. F. Stadler
Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, 1090 Wien, Austria

P. F. Stadler
Center for non-coding RNA in Technology and Health, Copenhagen University, Grønnegårdsvej 3, Frederiksberg C, Denmark

P. F. Stadler
Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA

scheme that guides the principled design of exact dynamic programming solutions for particular partially local alignment problems.

Keywords Multiple sequence alignment · Dynamic programming · Sum-of-pairs score · Formal grammar · Recursion

Mathematics Subject Classification 68R01 · 03D25 · 90C35

1 Introduction

Intuitively, a multiple string alignment (MSA) is a rectangular arrangement of the characters that preserves the order of characters in each string, see Fig. 1 for an example. The order of the strings (rows) is arbitrary, while the columns preserve the order of the characters in the input strings. Each sequence is stretched to a common length by inserting gap characters in such a way that similarities between sequences are maximized. The concept of (multiple) alignment generalizes from biological sequences and string in general to sequences of arbitrary objects.

Global multiple alignments frequently appear as a key intermediate result in computational biology. Figure 1 shows an alignment of short peptide sequences from eleven different animal species. Algorithmic approaches to computing MSAs typically are phrased as optimization problems: given a scoring scheme for the similarities observed across an alignment column, usually evaluated as the sum of pairwise similarities of all pairs of characters (including gaps) in a column, the objective is to maximize total similarity or minimize total dissimilarity. Several variants of this problem have been shown to be NP-hard [15,27,40,42,51,77]. Multiple sequence alignments therefore are usually computed with the help of heuristic approximation algorithms, see [9] for a recent review.

Many of these methods rely on the exact solution of the 2-way alignment problems for some or all of the $N(N-1)/2$ pairs of input sequences, which can be computed by means of a simple dynamic programming recursions known as the Needleman–Wunsch algorithm [58] or its variants. Several conceptually different approaches are used in practise to combine the pairwise alignments to MSAs: Progressive approaches reuse pair wise alignment algorithms to combine pairwise alignments usually in a tree-like fashion [37]. Many of the software packages that are widely used in bioinformatics belong to this class, such as `clustal` [71], `mafft` [41], `t-coffee` [59]. Iterative methods employ additional optimization steps to improve partial alignments usually interleaved with a progressive scheme to add sequences or partial alignments [35,79]. An unusual member of this class is `dialign` [55], which uses near-perfect local seed alignments [16]. Consensus methods such as `M-COFFEE` [3] or `MergeAlign` [21] combine many—usually partially contradictory—alignments using voting-like procedures.

The maximum weighted trace problem [42] provides a combinatorial way to describe MSAs: given weights for pairs of positions i and j from distinct input sequences, the problem is to extract a subset with maximal total

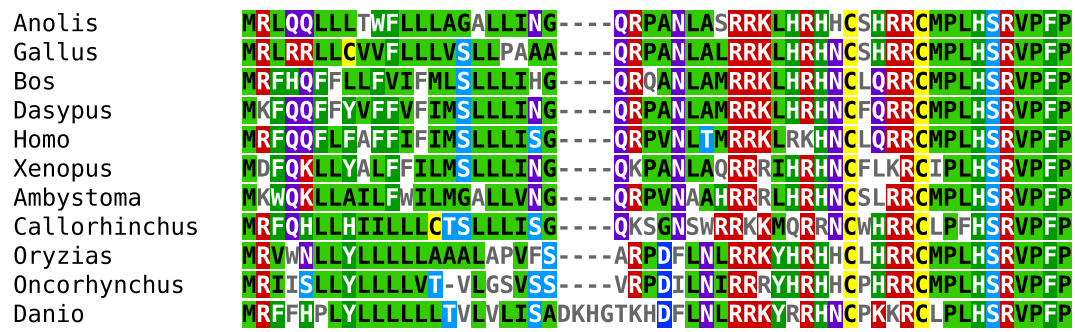


Fig. 1 Example of a multiple alignment of ELABELA protein sequences from $N = 11$ different vertebrate species. Each amino acid is represented by a single character, the column-wise consensus is highlighted in color. Gaps are indicated by - signs as usual in the bioinformatics literature (color figure online)

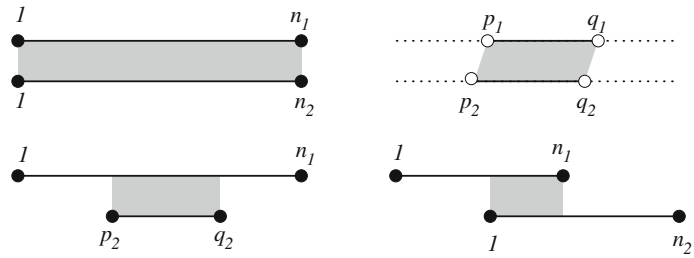


Fig. 2 Basic types of pairwise alignments of two sequences of lengths n_1 and n_2, respectively. Top: global and local alignments. Below: semi-global/overhang alignments. The variables p_1, p_2 and q_1, q_2 denote the local start and end positions, respectively. They $1 \leq p_i \leq q_i \leq n_i$ for $i = 1, 2$

weight that forms a so-called trace. In essence, a trace [68] represents the matches and mismatches in the MSA. The maximum weighted trace problem serves as a convenient framework for consensus methods. The trace framework is limited to relatively simple scoring models and gaps usually are not scored at all. Practical solutions are often based on integer linear programming [44], which also makes it possible to include arbitrary gap scores.

Many authors have used general optimization methods such as genetic algorithms or simulated annealing [20], which allow essentially arbitrary scoring models. Exact solutions for N-way alignments with a wide range of scoring models can be obtained by a straightforward generalization of the Needleman–Wunsch algorithm, albeit with space and time requirements scaling exponentially with the number sequences N [19]. In this contribution we focus on this class, mostly for ease of presentation.

Different variants of pairwise alignments are used in computational biology. The distinction between global, local, and semi-local *pairwise* alignments is standard material for introductory courses in algorithmic bioinformatics, Fig. 2. It is well known that all these problems are solved by slight variations of the same basis dynamic programming algorithms: the Needleman–Wunsch recursion [58] for the global problem and Smith–Waterman algorithm [72] for the local version. The key recursion step compares the scores of extensions of shorter alignments by a (mis)match, insertion, or deletion, $\mathfrak{N}(i, j) := \max\{S_{i-1,j-1} + m(i, j); S_{i-1,j} + \gamma; S_{i,j-1} + \gamma\}$, where $m(i, j)$ returns the match score for the positions i and j and γ is the gap score. In the local case, a character in one input sequence may not only be (mis)matched with a character in the other sequence, but it may also belong to a prefix or suffix that remains entirely unaligned. One say that is in the *aligned* or *unaligned* state, respectively. Since only prefixes or suffixes may remain unaligned, the transition from *unaligned* to *aligned* can occur only once. In the dynamic programming recursion this translates to an extra choice with score 0 signifying that the prefixes up to positions i and j remain *unaligned* [72]; thus one computes $\max\{\mathfrak{N}(i, j); 0\}$. In addition, the two algorithms differ in the initialization and the entry of the S-matrix that harbours the final result, i.e., the score of the optimal global of local alignment: To account for the possibility of returning to the *unaligned* state, one seeks the index pair (i^*, j^*) that maximizes and the S-matrix and regards the suffixes beyond i^* and j^* as *unaligned.*

Local alignments are of key practical importance in computational biology in a variety of different application scenarios. Most commonly they are not computed exactly but one relies on fast heuristics, in particular `blast` [6] or `blat` [38]. In high throughput sequencing [28], where huge numbers of short fragments are compared to a long reference with few differences expected in the best matches, index data structures such as suffix arrays are employed, as in our `segemehl` suite [36]. Local alignments also form the basis for whole genome alignments, for which a diverse set of integrated workflows are available, see e.g., [12,61].

The identification of conserved sequence elements, such as clusters of transcription factor binding sites or splice enhancers, has become a key task in comparative genomics, often referred to as phylogenetic footprinting. It can be formalized as a local multiple alignment problem [52,84]. In [60,63] we considered the problem of combining pre-computed local pairwise alignments to a single local MSA using a consistency approach similar to [22,56]. Alternatively, phylogenetic footprinting can be viewed as a motif discovery task, as in `meme` [10], which required motives to match without gaps. The substring parsimony problem [13,14,70] formalizes a more general model that also allows gaps.

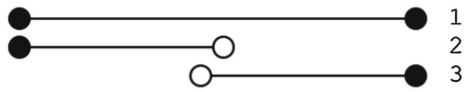


Fig. 3 Schematic representation of a breakpoint alignment for a reference sequence (1), a prefix (2), and a suffix (3). Sequences (2) and (3) may or may not overlap

Several variations of alignment problems share the same basic structure of the recursion but do not score some or all of the end-gaps. The simplest form of semi-global alignment is used primarily for homology search. It asks for a complete match of a small sequence in a potentially large database. The associated dynamic programming algorithm only differs in the initialization, setting scores $S_{0,j} = 0$ to allow cost-neutral deletions of the prefixes of the long, second sequence. Correspondingly, the optimal match is found as $\max_k S_{n_1,k}$. Dedicated implementations exists for this task, e.g., `gotohscan` [34]. More general overlap alignments [39] allow free end gaps on all sequences and play a role e.g., in sequence assembly [65].

With the advances in available computing power, the exact dynamic programming algorithms have become feasible beyond pairwise alignments. The basic recursion for the simultaneous alignment of N sequences is straightforward generalization of recursion $\mathfrak{N}$ and simply enumerates all $2^N - 1$ possible patterns of gaps in the last column of a alignment ending at position i_p in the pth sequences [19]. Despite the extra effort, 3-way alignments have at least occasionally found practical applications in computational biology [24,32,45,48]. In computational linguistics, 4-way alignments [73] have been used to align words from related natural languages, an approach that is feasible owing to the short sequence length.

Despite the importance of alignment problems in computational biology, and the need to distinguish global and local versions of the problem, there does not seem to be an accessible theory of partially local alignments beyond the pairwise case. The notion of "local multiple sequence alignment" typically found in the literature refers to the identification of short, conserved, usually gap free patterns [14,50]. This problem is also known as phylogenetic footprinting in applications to DNA sequences. In [74] the local multiple sequence alignment problem is phrased as finding substrings (with given minimal length in a minimal number of input sequences) to that the global alignment score of these subsequences is maximized. There are, furthermore, alignment problems of practical interest that involve complex mixtures of local and global alignments. Mitochondrial genome rearrangements, for example, can be accompanied by the duplication or loss of sequence in the vicinity of a breakpoint. A natural model for this problem considers the alignment of a reference (the ancestral state, 1 in Fig. 3) and the two sequences close to the breakpoint in the derived state (2 and 3 in Fig. 3) [4]. Beyond the breakpoint, the derived sequences are unrelated to the reference. Thus they are naturally treated as local on one side and global (anchored) at the other. Regions in which both derived fragments overlap are rewarded to increase the sensitivity for the detection of a potentially duplicated stretch of sequence. As it turns out, the corresponding alignment algorithm has been quite efficient in detecting previously undescribed tandem duplication random loss events in mitochondrial evolution [4].

In a variety of applications it is useful to consider a probabilistic version of alignments. In the pairwise case, the relation between score minimization and a corresponding "partition function version" is well understood [17,54,57,83] and in a more general context explained within the framework of Algebraic Dynamic Programming [30]. More recently probabilistic models were also studied systematically for local pairwise alignments [26]. Exact probabilistic algorithms beyond pairwise alignments seem to have received very little attention so far.

Most 3-way and 4-way alignment algorithms were designed with very specific application in mind and made no attempt to map the world of mixed global and local alignment problems. The alignment problem appearing in [4], covers just a very specific special case and pertains to a specialized question: how does mitochondrial DNA evolve close to genomic breakpoints? Mixtures of local and global alignments also arise naturally in the context of historical linguistics: Words from different but related languages can be compared by means of alignments [23,46,47,49,73]. However, some languages have specific morphological patterns such as prefixes or suffixes marking grammatical features, or use an inherited root only in a composition. In such cases, cross-language comparisons could naturally resort to alignment techniques that determine for each input sequence, whether it is to be used globally or weather

```
Latin                 s t u d i u [m]
German                s t u d i u [m]
Dutch/Afrikaans       s t u d i e
German                s t u d i e [ren]
English               s t u d y -
Spanish            [e] s t u d i a [r]
Spanish            [e] s t u d i o
Norwegian             s t u d i o
Croatian              s t u d i j [a]
French             ['e] - t u d - e
```

Fig. 4 Example of a partially local alignment of words with the Indo-European root for "studeō". This form of alignment, we might be able to directly infer a cross-linguistic kind of stemming serving as the first step to infer a proto form. We show unaligned prefixes and suffixes in brackets. Depending on prior knowledge of the inflection structure or other peculiarities of a language, it is desirable to decide for each word individually whether it is to be aligned locally or globally. As common in graphical displays of alignments, - signifies a gap

only a substring is expected to match with the cognates from other languages. A discussion of this issue in terms of pairwise alignments can be found in [47]. Figure 4 gives a conceptual example.

The purpose of this contribution is to develop a concise formal framework in which partially local N-way multiple sequence alignments can be studied systematically. To this end we generalize the systematic differences and similarities of local and global alignment problems and show that they can be represented in a manner that already implies exact DP solutions for them. In our presentation we proceed stepwisely. First we introduce a compact notation for the global alignment problem and argue that (the generalizations of) semi-global and "overhang" alignments are better viewed as global alignments with modified scoring functions at the ends of the input sequences. On this basis we then give a complete classification of partially local problems and derive a DP algorithm to solve them. Subsequently we consider the problem from the point of grammars, deriving an unambiguous version that, albeit computationally even more expensive, is suitable for a full probabilistic treatment. We close with some comments on possible future developments.

2 Global N-Way Alignments

2.1 Notation and Basic Properties

Let us start with a formal definition. For completeness of the presentation we also consider some of the basic properties of multiple alignments that are folklore. We include proofs where we could not find a convenient reference.

Definition 2.1 A *global multiple sequence alignment* (gMSA) of N sequences of objects X_p of finite length ℓ_p for $1 \le p \le N$ is a set of N strictly monotonic maps $\varphi_p : [1, \ell_p] \to [1, L]$ for some $L \in \mathbb{N}$, where L is the length of the alignment. We write $\varphi = \prod_{p=1}^{N} \varphi_p$. The pre-image $\varphi^{-1}(k) = \left(\varphi_1^{-1}(k), \varphi_2^{-1}(k), \ldots, \varphi_N^{-1}(k)\right)$ with $1 \le k \le L$ is called an *alignment column*. A pair (p, k) is a *gap* if $\varphi_p^{-1}(k) = \emptyset$. An alignment is *well formed* if no alignment column consists entirely of gaps, i.e., if $\bigcup_{p=1}^{N} \varphi_p^{-1}(k) \neq \emptyset$ for all $k \in [1, L]$.

Some simple properties follow immediately from the definition:

Lemma 2.2 *Let φ be gMSA of N sequences with length $\ell_1, \ell_2, \ldots, \ell_N$. Then* $\max_{1 \le p \le N} \ell_p \le L$. *If the alignment is well-formed we also have* $L \le \sum_{p=1}^{N} \ell_p$.

Proof The lower bound follows directly from strict monotonicity. The longest possible well-formed gMSAs correspond to the concatenation of the N sequences. □

Definition 2.3 The *pattern* $\tilde{\pi} : [1, L] \to \{0, 1\}^N$ of a gMSA is $\tilde{\pi}_p(k) = 0$ if (p, k) is a gap, i.e., $\varphi_p^{-1}(k) = \emptyset$, and $\pi_p(k) = 1$ otherwise.

The *index* $\Upsilon : [0, L] \to \prod_{p=1}^{N} [0, \ell_p]$ of a gMSA is $\Upsilon_p(k) = \max\{j | \varphi_p(j) \leq k\}$.

Note that $\Upsilon_p(k) = 0$ for $0 \leq k < \varphi_p(1)$ and thus in particular $\Upsilon_p(0) = 0$. We write π here as a *function* of the row index in the alignment; hence it can be viewed as a vector of length L.

Lemma 2.4 *Let $\tilde{\pi}$ and Υ be the column pattern and index of a gMSA φ. Then $\Upsilon_p(k) = \sum_{k'=1}^{k} \tilde{\pi}_p(k)$ and $\Upsilon_p(k) = \varphi_p^{-1}(k)$ whenever $\tilde{\pi}_p(k) = 1$.*

Proof For $k = 1$ we have $\tilde{\pi}_p(1) = 1$ if and only if $\varphi_p(1) = 1$ and thus if and only if $\varphi_p^{-1}(1) = \Upsilon_p(1) = 1$. Now consider an arbitrary column k. By definition we have $\Upsilon_p(k) = I_p(k)$ if $\pi_p(k) = 0$ and $I_p(k) = \Upsilon_p(k-1) + 1$ for $\Upsilon_p(k) = 1$ because of strict monotonicity of φ_p. Thus $\Upsilon_p(k)$ counts the columns $k' \leq k$ with $\tilde{\pi}_p(k') = 1$. Strictly monotonic also implies $\Upsilon_p(k) = \max_{k' \leq k} \varphi_p^{-1}(k')$ and thus $\Upsilon_p(k) = \varphi_p^{-1}(k)$ if and only if $\tilde{\pi}_p(k) = 1$. □

The key implication of Lemma 2.4 is that both $\tilde{\pi}$ and Υ completely determine the gMSA.

In the following it will be convenient to use the symbol X to denote an index set that identifies the sequences X_p, i.e., $p \in \{1, \ldots, |X|\}$. Furthermore, we denote by $\mathfrak{A}(X)$ the set of gMSAs that can be formed by the sequence X_p. Note that $\mathfrak{A}(X)$ only depends on the lengths of the sequences.

2.2 Scoring Functions and Optimization

Given sequences (X_p) of input objects, not all MSAs are of equal interest. To distinguish different MSAs of the same sequences, a scoring function is used. We consider here only the case that the total score is the sum of contributions $s(k)$ of the alignment column. Still there are many different ways how the column score $s(k)$ depends on φ and X. Consider the score of the partial gMSA φ up to column k. This amounts to scoring an alignment of the prefixes $X_1[1, \Upsilon_1(k)], X_2[1, \Upsilon_2(k)], \ldots, X_N[1, \Upsilon_N(k)]$, or, in other words, the restriction of φ_p to the intervals $[1, \Upsilon_p(k)]$ for $1 \leq p \leq N$. We write $S(k, \Upsilon)$ for this value, suppressing the explicit reference to the dependence of the score on the input sequences. Since columns are scored independently, it satisfies the recursion $S(k, \Upsilon) = S(k-1, \Upsilon) + \text{score}(k, \Upsilon, X)$.

Similar to the exposition in [69], we fix a set of indices $I \in \prod_{p=1}^{N} [0, \ell_p]$ and a pattern π and define S_I^{π} as the best possible score of a partial alignment with its last column k constrained to the prescribed values $\Upsilon(k) = I$ and $\tilde{\pi}(k) = \pi$, i.e.,

$$S_I := \max_{\Upsilon, k : \Upsilon(k) = I} S(k, \Upsilon) \tag{2.1}$$

Let us assume that the column score depends only on the pattern π in the current column and the indices $i_1, i_2, \ldots, i_p$ of the current column. Usually, the scoring function will only evaluate the characters and gaps observed in the current column. Subject to this constraint, the function $\text{score}(\pi, I, X)$ thus can be tailored freely to match the application at hand. For instance, there is no reason that prevents us from using a scoring function that also considers the adjacent positions in the input sequence, e.g., to score two columns with the identical character and gap patterns differently. Such scoring models are indeed often used in the context of protein alignments to incorporate e.g., knowledge on structural features [7,53,62]. There is also no need to require that the column score satisfies any other constraints, such as being composed as sum or average of pairwise comparisons, although this is often used in practical applications. There is also no need for the scoring function to satisfy any symmetries. Scores that are sensitive to permutations of the input order are e.g., used in comparative linguistics, where sound shifts between languages are directional [73], or in the analysis of ancient DNA to account for degradation by cytosine deamination in the ancient sequence only [64].

Since the last column has index I by assumption, the previous column has index $I - \pi$, where π is the pattern of the last column. Thus

$$\begin{aligned} S_I &:= \max_{\substack{\Upsilon,k:\\ \Upsilon(k)=I}} S(k,\Upsilon) = \max_{\pi\neq o} \max_{\substack{\Upsilon,k:\\ \Upsilon(k)=I-\pi}} \left\{S(k,\Upsilon) + s(k,\Upsilon,X)\right\} \\ &= \max_{\pi} \left\{S_{I-\pi} + \text{score}(\pi, I, X)\right\} \end{aligned} \tag{2.2}$$

Here we write o for the pattern of an all-gap column, which does not occur in well-formed gMSAs.

Before we proceed, it is instructive to consider the more complicated case of alignments with the naïve affine gap cost model. This scoring model cannot deal exactly with pairwisely affine gap costs [43]. It is, however, an approach that is frequently employed for N-way alignments. In this model, the score of a column depends not only on its own pattern, but also on the pattern in the previous column, since the continuation of runs of gaps is scored differently from initiating a new run of gaps. It does not account for the fact that pairs of gaps between two input sequences are not present in the corresponding pairwise alignment at all. We will briefly return to this issue below. Formally, the score of the optimal alignment is

$$S_I^\pi := \max_{\substack{\Upsilon,k:\\ \Upsilon(k)=I, \tilde\pi(k)=\pi}} S(k,\Upsilon) \tag{2.3}$$

Using the same reasoning as above, yields the recursion

$$S_I^\pi = \max_{\tau} \left\{S_{I-\pi}^\tau + \text{score}(\tau, \pi, I, X)\right\}, \tag{2.4}$$

The explicit reference to the pattern $\tau = \tilde\pi(k-1)$ in the preceding column makes it possible to distinguish gap opening ($\pi_p = 0, \tau_p = 1$) from gap extension ($\pi_p = \tau_p = 0$). Equation (2.4) is the obvious generalization of Gotoh's algorithm [31] to more than two sequences. In general, the score contributions take the form $\text{score}(\tau, \pi, I, X)$ depending on the gap pattern π of the current column, the gap pattern τ of the previous column, the multi-index I, and the actual values of the input X. We formally make the score explicitly dependent on I (rather than only the corresponding characters X_I that are aligned) to allow for context-dependent scoring, which has been shown to have large benefits in particular for protein alignments [7,53,62]. We shall see below, furthermore, that the explicit dependence on I is also important when dealing with semi-global alignments. In the following we will, however, suppress the explicit reference to X to somewhat simplify the notation.

2.3 Sets of Global Alignments and Formal Grammars

Instead of directly focusing on the scores it is useful to consider the sets of gMSAs. First we observe that each index set I, by virtue of determining prefixes of sequences, also specifies the set $\mathfrak{A}(X|I)$ of gMSAs. The column pattern π, furthermore determines a unique alignment column. Therefore

$$\mathfrak{A}(X|I) = \bigcup_{\pi} \mathfrak{A}(X|I-\pi) +\!\!+ \mathfrak{c}(X, I, \pi) \tag{2.5}$$

where $+\!\!+$ denotes the component-wise concatenation operation and $\mathfrak{c}(X, I, \pi)$ denotes the alignment column with entries X_{p,I_p} if $\pi_p = 1$ and `gap` otherwise. The base case $\mathfrak{A}(I|X, \vec o)$, for $I = \vec o$ the zero vector, is the empty alignment. More precisely, given a gMSA $\varphi : \prod_{p=1}^N [1, J_p] \to [1, L]$ and $\pi_p \in \{0, 1\}$, we explain $\tilde\varphi = \varphi +\!\!+ \pi$ as the function $\tilde\varphi : \prod_{p=1}^N [1, J_p + \pi_p] \to [1, L+1]$ such that $\varphi_p(i) = \tilde\varphi_p(i)$ for all $i \le I_p$ and $\varphi_p(I_p + 1) = L + 1$ for all p with $\pi_p = 1$. Thus $\mathfrak{A}(X|I) = \{\varphi +\!\!+ \pi \,|\, \varphi \in \mathfrak{A}(X|I-\pi)\}$.

In the case of alignment with naïve affine gap costs, the scoring distinguishes between opening and extension of gaps. This makes it necessary to consider for the extending column, not only its own pattern π but also the pattern τ at the end of the gMSA that is extended. This amounts to partitioning each set $\mathfrak{A}(X|I)$ of gMSAs in terms of the end pattern and yields the recursion

$$\mathfrak{A}(X|I,\pi) = \bigcup_{\tau} \mathfrak{A}(X|I-\pi,\tau) \dotplus \mathfrak{c}(X, I, \tau \to \pi). \tag{2.6}$$

with base case $\mathfrak{A}(X|\vec{o}, \vec{1})$, which again denotes the empty alignment. The scoring of the column $\mathfrak{c}$ may now explicitly depend on the patterns τ of penultimate column and π of the last column. The empty alignment is assigned the end pattern $\vec{1}$ to enforce that leading gaps are score as gap-open. This partitioning is sufficient for affine gap costs in two species but does not allow exact affine gaps costs in general. The well-known problem is that after a double gap in pair of sequences the information about their previous alignment state is lost. We will return to exact affine gap cost model at the end of this section. In the same approximation, this partitioning supports also scoring functions that depend on overlapping bigrams, as used e.g., in `blastR` for alignments of RNA sequences [18] and in word alignments in phylolinguistics [11,66].

Note that in alignment model above it is not logically necessary to include the reference to τ also in the term $\mathfrak{c}(X, I, \tau \to \pi)$. It is convenient, however, to use this expanded notation because it emphasizes the differences, e.g., between gap extension ($\tau_i = \pi_i = 0$) and gap opening ($\tau_i = 1, \pi_i = 0$) already in the symbol $\mathfrak{c}(X, I, \tau \to \pi)$. Using the set-based notation here we obtain the recursions for the optimal alignment scores by optimizing of the scores of the alignment set $\mathfrak{A}(X|I,\pi)$ and identifying the score of $\mathfrak{c}(X, I, \tau \to \pi)$ with the scoring function $\mathrm{score}(\tau, \pi, I, X)$. A complete, formal proof of the correctness of Eq. (2.6) will be given in the following section in a more general setting.

This reasoning matches the formalism of algebraic dynamic programming (ADP) [30], which separates the recursion of the search space (here explicitly spelled out in terms of sets of gMSAs), the scoring algebra (here a simple addition of score contributions of prefix alignments and the score of the last column), and the choice function (optimization of the score in our case). The theory of ADP emphasizes that this separation of concerns is possible for a large class of dynamic programming problems. An important advantage for practical applications is that the ADP framework makes it possible to use the same implementation for forward recursion and backtracking steps. A recent extension even provides a generic way to construct outside algorithms and thus posterior probabilities [86]. For our purposes, the recursive construction of the state space is the most important issue.

A key idea of ADP is to describe the recursive construction of the search space in terms of a formal grammar $\mathcal{G} = (N, \Sigma, P, S)$ comprising a finite set N of non-terminals, a finite alphabet Σ disjoint from N usually called the terminals, a start symbol $\mathbb{S} \in N$, and set P of production rules. Here we are only concerned with context-free grammars, where the production rules $p \in P$ are of the form $p : Q \to \alpha$ with $Q \in N$ and $\alpha \in (N \cup \Sigma)^*$. In our case, the set of terminals Σ are the possible patterns of alignment columns $\langle \pi \rangle$ (or $\langle \tau \to \pi \rangle$) and the empty alignment $\varnothing$. For emphasis, we use angular brackets to denote terminals of the grammar that correspond to alignment columns. Note that each pattern corresponds to a different terminal. The non-terminals are formal variables that designate different types of alignments.

In the simplest case of additive scoring functions, there is only a single alignment type $\mathbb{A}$ and the formal language of gMSAs is produced by the simple grammar $\mathbb{A} \to \mathbb{A}\langle \pi \rangle$ for all end patterns π as well as the termination step $\mathbb{A} \to \varnothing$. Since there is only a single non-terminal, the initialization step is trivial, i.e., $\mathbb{A}$ also serves as the start symbol.

In the case of naïve affine gap costs we need a more elaborate grammar that distinguishes the end patterns of the alignments. Hence we need to distinguish non-terminals $\mathbb{A}_\pi$ referring to alignments with different end patterns. Again, a terminal corresponding to the empty alignment $\varnothing$ is needed to represent termination. The production are of the form $\mathbb{A}_\pi \to \mathbb{A}_\tau \langle \tau \to \pi \rangle$ and $\mathbb{A}_\pi \to \varnothing \langle o \to \pi \rangle$. Here we also need a start rule that accounts for the fact that the gMSA may have every π as an end pattern. This is encoded by $\mathbb{S} \to \mathbb{A}_\pi$ for all patterns π.

More details on the ADP formalism for alignment problems can be found in [85]. We note that alignments correspond to regular expressions or linear grammars in the Chomsky hierarchy since terminals are emitted only on one side of a single non-terminal, here to the right. The computational complexity of the multiple alignment problem arises from the exponential increase in the number of non-terminals and thus the exponential number of production rules (in terms of the number of input sequences) that are necessary to specify the problem. The compact notation above, which uses the end patterns to index terminals and non-terminals will be very convenient to investigate the grammars and dynamic programming recursions associated with partially local alignment problems that are the main topic of this contribution. Before we proceed, however, we briefly consider the most commonly used scoring function for multiple alignment problems in some more detail.

Equation (2.6) is a good starting point to investigate just how general scoring functions may be. Let $\mathbf{A} \in \mathfrak{A}(X|I, \lambda)$, where λ now is an index that identifies a "subtype" of alignment. The gap pattern π is one possible choice that is relevant for naïve affine gap cost alignments. A more complex construction that supports arbitrary gap scores, is briefly discussed in the appendix. Denote the last column of $\mathbf{A}$ by $\mathbf{c}$. Structurally, then $\mathbf{A} = \mathbf{B} +\!\!+ \mathbf{c}$. The previous alignment $\mathbf{B} \in \mathfrak{A}(X|I - \pi, \lambda')$ is of type λ' and has sequence end points $I - \pi$ determined by (I, λ) and the pattern π of the column $\mathbf{c}$. The class of scoring functions we are interested in here assumes additivity

$$\text{score}(\mathbf{A}) = \text{score}(\mathbf{B}) + \text{score}(\mathbf{c}, I, \lambda', \lambda'') \tag{2.7}$$

There are, however, very few restrictions on the column score $\text{score}(\mathbf{c}, I, \lambda', \lambda'')$, which in particular can model score increments for extending a particular gap pattern or a pattern of shapes. In particular, thus, the formalism includes affine gap scores in both the naïve version and the exact form used e.g., in the alignment of alignments problem [43]. In order for dynamic programming to be applicable, we only need to ensure global subadditivity: Concatenating $\mathbf{A} = \mathbf{B} +\!\!+ \mathbf{C}$ from two shorter alignments, we require

$$\text{score}(\mathbf{A}) \geq \text{score}(\mathbf{B}) + \text{score}(\mathbf{C}) \tag{2.8}$$

which in particular specializes to the usual condition that gap penalties must be subadditive [78].

The condition that scores are of the form Eq. (2.7) and alignments can be constructed by stepwisely appending individual columns, i.e., that the recursion conforms to a regular grammar, excludes models such as Sankoff's algorithm [67] for the simultaneous alignment and structure prediction of RNAs. The reason is that RNA structure prediction is inherently non-local, giving rise to recursive steps of the form $\mathbf{A} = \overline{\mathbf{p}}\mathbf{A}'\overline{\mathbf{p}}$, where $\overline{\mathbf{p}}$ and $\overline{\mathbf{p}}$ correspond to alignment columns that are base-paired with each other. We refer to [25] for an introduction to this kind of models.

2.4 Sum-of-Pairs Scores

By far the most widely used scoring model for multiple sequence alignments is the sum-of-pairs score [5,33] defined as

$$\text{score}(\tau, \pi, I) = \sum_{p>q} \text{score}\begin{pmatrix} \tau_p & \pi_p \\ \tau_q & \pi_q \end{pmatrix}\begin{matrix} i_p \\ i_q \end{matrix} \tag{2.9}$$

Pairwise score components expressed in terms of π and τ can be interpreted in the following way: A term of the form $\text{score}\begin{pmatrix} \tau_p & 1 \\ \tau_q & 1 \end{pmatrix}\begin{matrix} i_p \\ i_q \end{matrix} = s(X_{p,i_p}, X_{p,i_q})$ corresponds to a (mis)match of the letters X_{p,i_p} at position i_p of sequence X_p and X_{q,i_q} at position i_q of sequence X_q. The terms $\text{score}\begin{pmatrix} 0 & 0 \\ \tau_q & 1 \end{pmatrix}\begin{matrix} i_p \\ i_q \end{matrix}$ and $\text{score}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{matrix} i_p \\ i_q \end{matrix}$ evaluate as gap extension

and gap open scores, respectively. Since projects of multiple sequence alignments to pairwise alignments may contain gap columns, terms of the form score $\left(\begin{smallmatrix}\tau_p & 0 \\ \tau_q & 0\end{smallmatrix}\middle|\begin{smallmatrix} i_p \\ i_q\end{smallmatrix}\right) = 0$ also appear.

The end patterns π and τ can be more complex that just the gap patterns in the last and penultimate column. The strict definition of the sum-of-pairs score requires that columns consisting of two gap are removed before scoring a pairwise subalignment. The following two examples show that this cannot be done based only on the gap pattern of the current and the previous column.

$$(a)\quad \begin{array}{cccc} a & b & c & d \\ a' & - & - & - \\ - & - & - & d' \end{array} \qquad (b)\quad \begin{array}{cccc} a & b & c & d \\ a' & - & - & d' \\ a'' & - & - & - \end{array}$$

The naïve generalization of affine gap cost score given above treats $\begin{smallmatrix}-\\ d'\end{smallmatrix}$ as gap extension, while it should be scored as opening a new gap. These issues are discussed at length in the context of "aligning alignments" and shown to be resolved by characterizing alignment columns by more elaborate "shapes" that describe, for each sequence, the relative position of the immediately preceding character [43]. Another variant that requires more elaborate end patterns is the use of distinct gap types as in the case of piece-wise linear gap cost functions [82]. These problems are interesting because they emphasize that the NP-completeness of alignment problems is not just related to the exponential number of gap/non-gap patterns required in the last column. In the case of pairwise alignments of multiple alignments, there are only three types of extensions. Nevertheless the problems is NP-complete [43], because exact solutions need to keep track of an exponential number of "shapes". We briefly sketch a related encoding in "Appendix B".

The term $I - \pi$ in Eq. (2.6) already accounts for situations with different types of gaps. The recursion can be generalized further to more elaborate descriptions of the subtypes of alignments, as in the example given in "Appendix B". Suppose the subtypes of alignments are characterized by some variable λ' and the extension of such an alignment by a column with gap pattern π gives rise to an alignment of type λ uniquely determined by λ' and π. Writing $\lambda = \lambda' \oplus \pi$ to express this fact, we recast the recursion in the form $\mathfrak{A}(I, \lambda) = \bigcup_\pi \bigcup_{\lambda' : \lambda = \lambda' \oplus \pi} \mathfrak{A}(I - \pi, \lambda') +\!\!+ \mathfrak{c}(\pi)$ and score it with an arbitrary column score depending on $\pi, \lambda, \lambda', I$ and the sequence data X.

2.5 End Gaps and Overhangs

In pairwise semi-global and overhang alignments it is customary to use the global alignment recursions unaltered. Overhangs on the left hand side are taken care of with the initialization of the DP matrix: one simply sets $S_{i,0} = 0$ if the deletion of a prefix of length i from the first sequences is supposed to be score-neutral. The right hand end of the alignment is handled differently. Here the "free" deletion is handled by searching for the maximal entry $\max_i S_{i,n_2}$ to determine the best position beyond which to delete the suffix of the first sequence. While this is convenient for score maximization, this trick does not carry over to probabilistic frameworks. The problem is that S_{n_1,n_2} does not correctly score the best alignment but includes gap scores for the deletions from the first sequence also beyond the last character of the second sequence. The easy remedy is to explicitly treat the cost-neutral end gap in the scoring function. Conceptually, this is what is done in the initialization step as well.

For N-way alignment problems this opens a can of worms: in principle one might want to be able to specify for any pair of sequences whether an overhang should be cost-neutral or not. Furthermore, this choice of scoring can be made independently on the left hand end and the right hand end of the alignment. In the case of a sum-of-pairs scoring model, these choices are conveniently represented by graphs, see Fig. 5. The pairwise scores now depend explicitly on the sequences to which they apply: For instance, we have $s_{13}(X_{1,i_1}, \emptyset) = 0$ if $i_3 = 0$ or $i_3 = n_3$, while a regular gap score is used for all other values of i_3.

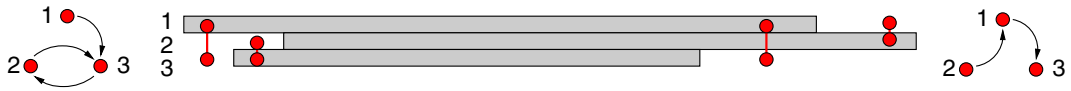


Fig. 5 Scoring of free end gaps in a global alignment. The graphs at both ends denote the behavior of unscored deletions, which can differ from sequence pair to sequence pair. For example deletions from 1 to 3 on the left side are allowed while on the same side deletion from 3 to 1 would be penalized. Hence one would call 1 left-global; 3 on the other hand is not. In this example we have $S(i, 0, 0) = S(0, j, 0) = S(0, 0, k) = 0$. The entries $S(i, j, 0)$ coincide with the pairwise score of 1 and 2 and do not incur a gap score for deletion of the third sequences

3 Partially Local Alignments

3.1 General Framework

In a gMSA, each input sequence is contained in its entirety. The idea of local alignments is to relax this condition, and to consider also—in the most general case—multiple alignments of arbitrary infixes of the input sequences. In order to model this case, we extend the notion of an alignment column, $\varphi^{-1}(k)$ in the following manner:

Definition 3.1 A MSA of length L of $\{X_p | p = 1 \dots N\}$ is defined by a sequence of alignment columns $\mu = \prod_{i=p}^{N} \mu_p$ with $\mu_p : [0, L] \to [1, \ell_p] \cup \{\bullet, \varnothing, \circ\}$ such that the following properties are satisfied:

1. If $\mu_p(i), \mu_p(j) \in [1, \ell_p]$ then $\mu_p(i) = \mu_p(j)$ implies $i = j$ and $\mu_p(i) < \mu_p(j)$ implies $i < j$.
2. If $\mu_p(i), \mu_p(j) \in [1, \ell_p]$ then there is $k \in [1, L]$ such that $\mu_p(k) = h$ for all $h \in [\mu_p(i), \mu_p(j)]$.
3. If $\mu_p(i) = \bullet$, $\mu_p(j) \in [1, \ell_p] \cup \{\varnothing\}$, and $\mu_p(k) = \circ$ then $i < j < k$.
4. $\mu_p(0) \in \{\bullet, \varnothing\}$. If $\mu_p(0) = \varnothing$ then $1 \in \mu_p([1, L])$;
5. If $\mu_p(k) \notin [1, \ell_p] \cup \{\varnothing\}$ for all p, then $\mu_p(0) = \bullet$ and $\mu_p(j) = \circ$ for all $j \in [1, L]$

Furthermore, we say that the MSA is well-formed if, for each $k \in [1, L]$ there is at least one $p \in [1, N]$ such that $\mu_p(k) \notin \{\bullet, \varnothing, \circ\}$.

Condition 1 ensures that every sequence position is mapped at most one alignment column and that the order of elements in input sequence is preserved. Condition 2 ensures that if the end points of an interval of X_p, i.e., an infix of X_p, appear in an alignment column, then all elements of the infix are represented in an alignment column. The restriction μ to $[1, L]$, upon ignoring the distinction between the three gap symbols $\{\bullet, \varnothing, \circ\}$, therefore describes the columns φ^{-1} of a gMSA of length L of infixes of $\{X_1, \dots, X_N\}$. In this gMSA we have, in particular, $\varphi_p^{-1}(k) = \emptyset$ if and only if $\mu_p(k) \in \{\bullet, \varnothing, \circ\}$. The MSA is well-formed if and only if this gMSA of infixes is well-formed.

The three distinct gap types $\bullet$, $\circ$, and $\varnothing$ are introduced to distinguish, for each alignment column k, whether (an infix of) an input sequence X_p "participates" in the alignment, the `aligned` state $\mu(k) \in [1, \ell_p] \cup \{\varnothing\}$, or not. The `unaligned` state is represented by $\mu(k) \in \{\bullet, \circ\}$. The distinction between `aligned` and an `unaligned` state was the key idea behind the dynamic programming algorithms for pairwise local alignments [72]. The distinction between initial and terminal unaligned states, $\bullet$ and $\circ$, resp., is a technical convenience that could be omitted. Condition 3 ensures that the `aligned` state forms a contiguous interval of alignment columns for each input sequence X_p. As a special case we also allow that sequences have no `aligned` states at all. In order to enforce a unique representation we stipulate that $\mu(0) = \bullet$ to $\mu(i) = \circ$ for $i \in [1, L]$. In practise, this case refers to an input sequence that is so different from all others that it does not significantly match anywhere. The purpose of the "empty" alignment column 0 is the encode a definite state at the beginning of the alignment. Thus, if X_p is in the aligned state from the beginning, its first element must appear in the MSA. This is stipulated by Condition 4. The final Condition 5 establishes a convention for representing in a unique manner the case that sequence X_p is never `aligned`.

The notion of a pattern $\tilde{\pi}$ naturally generalized from gMSAs to arbitrary MSAs: The pattern of a MSA of length L is function $[1, L] \to \{0, 1\}^N$ such that $\tilde{\pi}_p(k) = 1$ iff $\mu_p(k) \in [1, \ell_p]$ and $\tilde{\pi}_p(k) = 0$ otherwise. Again, the pattern just distinguishes cells in the alignment that contain elements of the input sequences from cells that contain a gap.

Definition 3.2 The state function $\tilde{\Psi} : [0, L+1]^N \to \{-\infty, 0, +\infty\}^N$ of an MSA of length L is defined $\Psi_p(k) = 0$ if $\mu_p(k) \in [1, \ell_p] \cup \{\varnothing\}$, $\Psi_p(k) = -\infty$ if $\mu_p(k) = \bullet$ and $\Psi_p(k) = +\infty$ if $\mu_p(k) = \circ$.

Finally, we extend the index function Υ to MSAs.

Definition 3.3 The index function $\Upsilon : [0, L+1]^N \to \prod_{p=1}^N [1, \ell_p]$ of an MSA of length L is given by

$$\Upsilon_p(0) := \min_{j \in 1 \ldots L} \mu_p(j) - 1 \quad \Upsilon_p(k) := \Upsilon_p(0) + \sum_{k'=1}^{k} \tilde{\pi}_p(k) \quad \text{for } k > 0 \tag{3.1}$$

The value $\Upsilon_p(0)$ is the length of the prefix of X_p that remains unaligned. Definition 3.3 does not unambiguously settle the special case that a sequence is completely deleted. In principle, this could be represented by the deletion of any prefix followed by the deletion of the remaining suffix. To avoid ambiguities, we elect—without loss of generality—to interpret empty alignment rows as the deletion of the full-length suffix. Hence, $\Upsilon_p(0) = 0$ is this case and thus, consistent with Definition 3.3, $\Upsilon_p(i) = 0$ for all $i \in [0, L+1]$.

One easily verifies

$$\Upsilon_p(k) = \max\{\Upsilon_p(0), \max_{k' \le k} \mu_p(k)\} \tag{3.2}$$

i.e., the index for sequence X_p at alignment column k is the sequence position up to which X_p is represented in the alignment; this includes also `unaligned` prefixes that are omitted. Since $\Upsilon_p(0) = 0$ whenever sequences X_p starts out in the `aligned` state, the definition is consistent with definition of the index of gMSAs. The index function completely determines the pattern function of a MSA by virtue of $\tilde{\pi}_p(k) = \Upsilon_p(k) - \Upsilon_p(k-1)$.

Lemma 3.4 *For fixed L, there is a bijection between MSAs μ and pairs (Υ, Ψ) of state and index functions that represent MSAs.*

Proof The definitions above determine how (Υ, Ψ) is computed from μ. It is sufficient, therefore, to show that μ is uniquely determined by (Υ, Ψ). To this end we observe that $\mu_p(i) = \Upsilon_p(i)$ exactly for alignment columns i where row p contains an element of the input sequence, i.e., if and only if $\tilde{\pi} = 1$ or, equivalently, $\Upsilon_p(i) = \Upsilon_p(i-1) + 1$. Since all other columns have values $\mu_p(i) \in \{\bullet, \varnothing, \circ\}$ it is clear that different index functions also give rise to different functions μ. The state function now uniquely determines for all other values of i whether $\mu_p(i)$ is an `unaligned` position $\bullet$, and `unaligned` position end gap, or a regular gap character. Since the state function is by construction $\Psi_p(i) = 0$ whenever $\tilde{\pi}_p(i) = 1$, we see that different state functions imply different functions μ. Thus there is a one-to-one correspondence between MSAs μ and pairs (Υ, Ψ) that are derived from μ. □

We note for later reference that a MSA μ is gMSA if and only if the state function Ψ is zero everywhere. All resulted proved below for sets of MSAs thus also hold for gMSAs when restricted to everywhere zero state functions.

Let us now turn to the recursive construction of set of all local alignments. For this purpose we also need to formally define the extension of a MSA μ by an additional column.

Definition 3.5 Let $\mu : [0, L] \to \prod_{p=1}^N ([1, \ell_p] \cup \{\bullet, \varnothing, \circ\})^N$ be a MSA of length L and let (π, Ψ) be a pair consisting of a pattern π and a state vector Ψ. Then the concatenation $\tilde{\mu} = \mu +\!\!+ (\pi, \Psi)$ is the MSA of length $L+1$ with $\tilde{\mu}(i) = \mu(i)$ for $0 \le i \le L$ and $\tilde{\mu}_p(L+1) = 1 + \max_{1 \le j \le L} \mu_p(j)$ if $\pi_p = 1$, and $\tilde{\mu}_p(L+1) = \bullet, \varnothing, \circ$ depending on whether $\Psi_p = -\infty, 0$, or $+\infty$ whenever $\pi_p = 0$.

For a set $\mathfrak{A}$ of MSAs we write $\mathfrak{A} +\!\!+ \pi := \{\mu +\!\!+ \pi \,|\, \mu \in \mathfrak{A}\}$.

Let us write $\mathfrak{A}(X|I, \pi, \Psi)$ for the set of MSAs of prefixes of X_p with length I_p for $p \in [1, N]$ whose last column has pattern π and state function Ψ irrespective of their length. Instead of a single empty global alignment we now

have a large set of empty local alignment that differ in length of the prefixed that have been omitted. These MSAs of length $L = 0$ are of the form $\mathfrak{A}(X|I, o, \Psi^o)$, where the special pattern o features no aligned element in the zero-th column. The values of I_p encode the lengths of the prefixes of the input sequences X_p that are omitted. The state vector therefore must satisfies $\Psi^o_p = -\infty$ whenever $I_p > 0$, i.e., for all rows in which a prefix if X_p is omitted. By definition we have $\Psi^o_p \in \{-\infty, 0\}$ whenever $I_p = 0$.

In the following we use the notation $\mathfrak{c}(\tau \to \pi; \Phi \to \Psi)$ for an alignment column with pattern π and state vector Ψ that succeeds a column with pattern τ and state vector Φ. While it is not strictly necessary to include the information on the preceding column, it is a convenient notational device to restrict transitions between patterns and to make scoring functions context dependent. Formally, we think of $\mathfrak{c}(\tau \to \pi; \Phi \to \Psi)$ as a set that either contains the unique column (π, Ψ) if the transition is allowed, or as the empty set. Note that the set-wise concatenation with the empty set yields the empty set. Depending on the scoring function not all transitions $\tau \to \pi; \Phi \to \Psi$ might be allowed. For instance, the state function cannot change from $+\infty$ ("omitted suffix") to 0 ("aligned") or $-\infty$ ("omitted prefix"). The user may want to further restrict transitions. In Sect. 4, for instance, we briefly consider a model that allows changes in the state function only simultaneously for all input sequences.

Theorem 3.6 *Let $\mathfrak{A}(X|I, \pi, \Psi)$ the set of all MSAs of prefixes of X_p with length I_p for $p \in [1, N]$ whose last column has pattern π and state function Ψ. Then*

1. *$\mathfrak{A}(X|I, \pi, \Psi) \cap \mathfrak{A}(X|I', \pi', \Psi') \neq \emptyset$ implies $I = I', \pi = \pi'$ and $\Psi = \Psi'$.*
2. *$\bigcup_I \bigcup_\pi \bigcup_\Psi \mathfrak{A}(X|I, \pi, \Psi)$ is the set of all MSAs of X*
3. *The sets can be constructed recursively using*

$$\mathfrak{A}(X|I, \pi, \Psi) = \bigcup_\tau \bigcup_\Phi \mathfrak{A}(X|I - \pi, \tau, \Phi) \dotplus \mathfrak{c}(\tau \to \pi; \Phi \to \Psi)\,, \tag{3.3}$$

The alignments of the form $\mathfrak{A}(X|I, o, \Psi^o)$, in which all non-empty prefixes are completely `unaligned`, i.e., $\Psi^o_p = -\infty$ whenever $I_p > 0$, and $\Psi^o_p \in \{-\infty, 0\}$ for $I_p = 0$, serve as base case.

Proof (1) Lemma 3.4 implies that different index functions as well as different state functions yield MSAs that are distinct from each other. Alignments are therefore in particular different from each other if the index function or the state function differ in the last alignment column. Now suppose the last alignment column has the same values of index and state function, then the pattern π can be different only if the penultimate columns have different values of the index function. Hence no alignment can simultaneously belong to sets $\mathfrak{A}(X|I, \pi, \Psi)$ with different values of I, Ψ, or π.

(2) Every alignment of prefixed ends with an index $0 \leq I_p \leq \ell_p$, and corresponding pattern π_p (defined by the difference of the index function between the last two columns) and a value of the state function; hence every MSA of X is contained in one of the sets $\mathfrak{A}(X|I, \pi, \Psi)$.

(3) Let us partition the sets $\mathfrak{A}(X|I, \pi, \Psi)$ w.r.t. to the alignment length L. We first show that the length-stratified version of Eq. (3.4)

$$\mathfrak{A}(X|I, \pi, \Psi; L) = \bigcup_\tau \bigcup_\Phi \mathfrak{A}(X|I - \pi, \tau, \Phi; L - 1) \dotplus \mathfrak{c}(\tau \to \pi; \Phi \to \Psi)\,, \tag{3.4}$$

is true. To this end, observe that every alignment (consisting) A with $L \geq 2$ columns is the concatenation of an alignment A' that is one column shorter and the final column. A ends in (I, π, Ψ) while A' has a last column characterized by (J, τ, Φ). From $J_p + \pi_p = I_p$ we conclude that $J = I - \pi$. Any potential restrictions on transitions between patterns and/or state function are encoded in the context dependent columns $\mathfrak{c}$. Hence the recursion is correct for $L \geq 2$. The first column follows an empty alignment, which features the special pattern o (nothing is aligned yet) and an index J specifying the length of prefixes that are omitted in `unaligned` states, thus the state of the zero-th column must be $\Psi_p = -\infty$ whenever $J_p > 0$. If $J_p = 0$, our definition leaves us the choice between

$\Psi_p = -\infty$, a "deletion" of an empty prefix, or $\Psi_p = 0$, enforcing that the first column is treated as a regular part of an alignment. Thus Eq. (3.4) is also valid for $L = 1$. Furthermore, any alignment of the form $\mathfrak{A}(X|I, o, \Psi^o)$ by definition does not contain an alignment column and thus has length $L = 0$, while all alignments with a pattern $\pi \neq o$ contain at least one column, i.e., $\mathfrak{A}(X|I, o, \Psi^o) = \mathfrak{A}(X|I, o, \Psi^o; 0)$. Thus every alignment with a finite length L can be constructed by L concatenations of columns. Taking the union over all alignment lengths L now immediately yields Eq. (3.3). □

The correctness of Eq. (2.6) for gMSAs now follows as an immediate corollary: gMSA are characterized as the MSAs whose state function is 0 everywhere, thus the base case reduces to alignment with pattern o and state function $\Psi^o_p = 0$ for all p, whence $I_p = 0$. In fact, the set $\mathfrak{A}(0, o, 0)$ contains a simple empty alignment.

From a modelling point of view, it seems desirable in most contexts to exclude alignments in which left-local sequences begin with an `aligned` gap and right-local sequences end with an `aligned` gap. For this purpose we have to limit transitions $(\tau, \Phi \to \pi, \Psi)$ in such a way that

(i) $\Phi_p = -\infty$ and $\Psi_p = 0$ implies $\pi_p = 1$; and
(ii) $\Phi_p = 0$ and $\Psi_p = +\infty$ implies $\tau_p = 1$.

This is achieved by setting $\mathfrak{c}(\tau \to \pi; \Phi \to \Psi) = \emptyset$ whenever these conditions are violated. Different restrictions may also be useful, as in the case of block alignments, see Sect. 4 below.

As in the case of gMSAs, this set-wise recursion can be re-cast in the form of a linear grammar of the form

$$\mathbb{A}_{\pi,\Psi} \to \mathbb{A}_{\tau,\Phi}\,[\tau \to \pi; \Phi \to \Psi] \tag{3.5}$$

The formally simple recursion now also contains rules that stepwisely remove prefixed and suffixes in productions with $\Psi = \Phi = \pm\infty$. We note in passing that the base cases $\mathfrak{A}(X|I, o, \Psi^o)$ can, in principle, also be generated recursively by step-wisely deleting longer prefixes. This is irrelevant for our present discussion, but relevant for convenient ADP implementations in practise [85].

Instead of using the state vector $\Psi \in \{-\infty, 0, +\infty\}$ will be more convenient in the following to encode the state as two subsets $A, D \subseteq \{1, \ldots, |X|\}$ such that $p \in A$ for $\Psi_p = 0$ and $p \in D$ whenever $\Psi_p = +\infty$.

3.2 Classification of Partially Local Alignment Problems

Partially local alignment problems are naturally classified in terms of constraints on the individual sequences X_p. For each sequence we can therefore specify whether the set of alignments must contain X_p in its entirety, whether a prefix or a suffix is required, or whether an arbitrary, possibly empty infix also constitutes an acceptable solution. In the language of the previous section, this amounts to specifying the state Ψ for the zero-th and the last alignment column.

It is convenient to associate these constraints directly with each input sequence X_p. We say that X_p is *right-global* or *left-global* if its first or last position must be included in every valid MSA. If X_p is not right-global, it is *right-local*, implying that valid MSAs may exclude suffixes of X_p. Similarly, prefixes my be omitted for *left-local* sequences. A sequence is global if it is both right- and left-global. Global sequences must be completely contained in the MSA. For local sequences, i.e., those that are both right- and left-local, any infix suffices. In the following we write R and L for the set of right-global and left-global sequences, respectively. See Fig. 6 for a more complicated example.

The sets of valid alignments for given set L and R are obtained from the general case by constraining the state of the initial and the final alignment. Valid initial conditions, i.e., alignments of length 0, are of the form $\mathfrak{A}(I, o, (A, D))$ and consist of all MSAs that satisfy $A = L$ and $I_p = 0$ for all $p \in L$. On the other hand, solutions of the partially local alignment problem are of the form $\mathfrak{A}(I, \pi, (A, D))$ such that $A = R$ and $I_p = \ell_p$ for all $p \in R$.

Conditions 3 and 4 imply that state changes from one alignment column to the next are only possible from the unaligned • state, i.e., $p \in X \backslash A \cup D$, to the aligned state A (or in the special case of empty rows directly

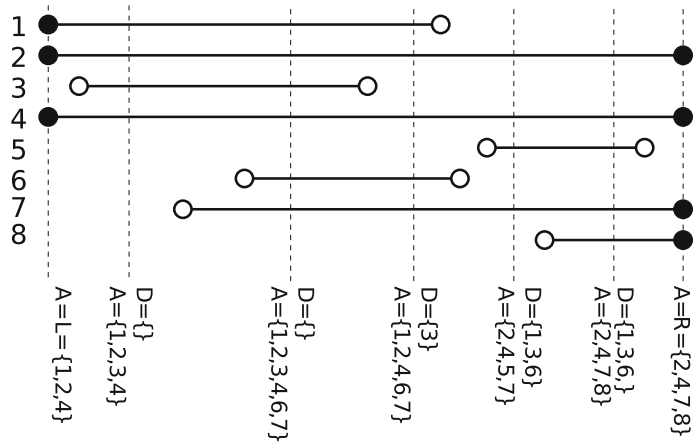


Fig. 6 Example of an 8-way alignment. Black circles indicate global, i.e., non-truncatable sequence ends, open circles denote local, i.e., truncatable ends. Hence, $L = \{1, 2, 4\}$ and $R = \{2, 4, 7, 8\}$. The state (A, D) is indicated for a few alignment columns denoted by dashed vertical lines

to the unaligned $\circ$ state), and from A to D. Denoting by (A', D') and (A, D) the states of the previous and the current alignment columns, respectively, we can rephrase these constraints in the form $D' \subseteq D$ and $X \backslash (A \cup D) \subseteq X \backslash (A' \cup D')$. The latter inclusion is equivalent to $A' \cup D' \subseteq A \cup D$. This simple observation defines a predecessor relation on the set of states:

$$(A', D') \preceq (A, D) \quad \Longleftrightarrow \quad A' \cup D' \subseteq A \cup D \text{ and } D' \subseteq D \tag{3.6}$$

Lemma 3.7 *Consider an arbitrary partially local alignment problem. Then $\preceq$ is a partial order and the set of states forms form a lattice w.r.t. $\preceq$.*

Proof We first show that $\preceq$ is a partial order. Since set inclusion $\subseteq$ is reflexive and symmetric, Eq. (3.6) implies that $\preceq$ is also reflexive and symmetric. Now assume $(A'', D'') \preceq (A', D')$ and $(A', D') \preceq (A, D)$, i.e., $A'' \cup D'' \subseteq A' \cup D' \subseteq A \cup D$ and $D'' \subseteq D' \subseteq D$; thus $(A'', D'') \preceq (A, D)$ and $D'' \subseteq D$, i.e., $(A'', D'') \preceq (A, D)$. Hence $\preceq$ is transitive.

In order to construct the infimum (A', D') of an arbitrary collection of states (A_i, D_i), $i \in J$, we observe that by construction $A' \cup D' \subseteq A_i \cup D_i$ and $D' \subseteq D_i$ for all $i \in I$. This implies immediately $A' \cup D' \subseteq \bigcap_i (A_i \cup D_i) = \bigcap A_i \cup \bigcap_i D_i$ and $D' \subseteq \bigcap D_i$. Obviously, the $\prec$-largest pair of sets that satisfies this condition is (A', D') with $A' = \bigcap_i A_i$ and $D' = \bigcap_i D_i$. Since it represents a valid state, (A', D') is the unique infimum.

In order to construct the supremum, consider $A_i \cup D_i \subseteq A' \cup D'$ and $D_i \subseteq D'$ for all i, i.e., $\bigcup A_i \cup \bigcup D_i \subseteq A' \cup D'$ and $\bigcup D_i \subseteq D$. The smallest sets satisfying both conditions are $D' = \bigcup D_i$ and $A' \cup D' = \bigcup_i A_i \cup D'$, i.e., $A' = \bigcup A_i$. Since (A', D') is a valid state, it is the unique supremum. □

Given $R, L \subseteq X$, there is a unique sublattice with minimal element $(L, \emptyset)$ and maximal element $(R, X \backslash R)$ that defines all allowed states $(X \backslash L, \emptyset) \preceq (A, D) \preceq (R, X \backslash R)$ that may appear in a solution of a partially local alignment problem with left-global set L and right-global set L. The pair R, L therefore characterizes the partially local alignment problem and completely specifies the set of relevant MSAs. For an example see Fig. 7.

3.3 Explicit Recursions for Optimal Alignments

Let us briefly consider the problem of computing the optimal alignment score for a partially local alignment problem of type R, L. Denote by

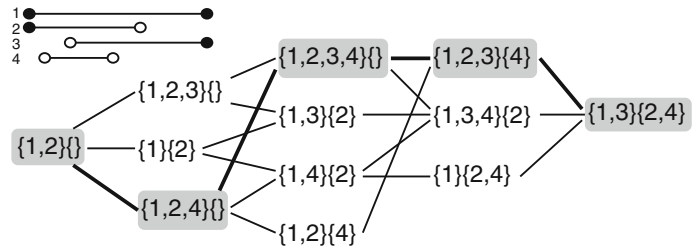


Fig. 7 Hasse diagram of the partial order of states $\prec$ for a 4-way alignment problem with $L = \{1, 2\}$ and $R = \{1, 3\}$. The highlighted path from L to R corresponds to the alignment in the inset

$$S_I^{\pi,(A,D)} := \max_{\mu \in \mathfrak{A}(X|I,\pi,(A,D))} \text{score}(\mu) \tag{3.7}$$

the score of the optimal alignment with index I, pattern π, and state (A, D) in the last column. An immediate consequence of Theorem 3.6 is

Corollary 3.8 *The values of* $S_I^{\pi,(A,D)}$ *satisfy the recursion*

$$S_I^{\pi,(A,D)} = \max_{\tau} \max_{(A',D') \preceq (A,D)} \left[S_{I-\pi}^{\tau,(A',D')} + \text{score}(X, I|\tau \to \pi, (A', D') \to (A, D)) \right] \tag{3.8}$$

In the scoring models that are usually employed for sequence alignments, one usually assumes that scores associated with state changes (if they are not omitted altogether) are at least independent of details of the emitted alignment column, i.e., the score(.) consists of additive terms $\text{score}(X, I|\tau \to \pi)$ and $\text{score}(A', D' \to A, D)$. In this scenario, it possible to separate the extension of the alignment without state change from state changes without extension. This amounts to transforming the recursion (3.8) into the form

$$S_I^{\pi,(A,D)} = \max \begin{cases} \max_{\tau} \left[S_{I-\pi}^{\tau,(A,D)} + \text{score}(X, I|\tau \to \pi) \right] \\ \max_{(A',D') \prec (A,D)} \left[S_I^{\pi,(A,D)} + \text{score}(A', D' \to A, D) \right] \end{cases} \tag{3.9}$$

The first alternative amounts to the usual recursion and accounts for all $2^N - 1$ possible gap patterns π of the last and the preceding alignment columns. The second term separately considers all possible state changes. Instead of considering all states that are $\prec$-smaller than (A, D) one could also make use of the Hasse diagram of this partial order, Fig. 7, and recurse only over the immediate predecessors of (A, D).

Lemma 3.9 (A', D') *is an immediate predecessor of* (A, D)*, in symbols* $(A', D') \prec\!\!\prec (A, D)$*, if and only if either (i)* $D' = D$*,* $A' \subseteq A$*, and* $|A| = |A'| + 1$*, or (ii)* $D \backslash D' = A' \backslash A = A' \cap D$ *and* $|A' \backslash A| = |D \backslash D'| = 1$*.*

Proof It is easy to check that $(A', D') \prec (A, D)$ is true in both cases and there cannot be another state between (A', D') and (A, D), i.e., both alternatives imply $(A', D') \prec\!\!\prec (A, D)$. Now suppose that $(A'', D'') \prec (A, D)$ but $(A'', D'') \not\prec\!\!\prec (A, D)$. If $A'' \cap D$ contains two or more elements, say X_p and X_q then $(A \backslash \{X_p\}, D \cup \{X_p\})$ is $\prec$-between (A, D) and (A'', D''). Similarly, if $D = D'$ and $A \backslash A'$ contains at least two elements (again called X_p and X_q), then $(A' \cup \{X_p\}, D)$ lies $\prec$-between (A', D') and (A, D). Finally if there $X_p \in D \backslash (D' \cup A')$, then $(A' \cup \{p\}, D')$ lies $\prec$-between (A', D') and (A, D). Thus, conditions (i) or (ii) together are indeed sufficient. □

It follows immediately that (A, D) has at most N immediate predecessors. Scoring models for local alignments are usually parameterized in such a way that state changes do not incur a score contribution. In this case Eq. (3.9)

can be simplified further

$$S_I^{\pi,(A,D)} = \max \begin{cases} \max\limits_{\tau} \left[S_{I-\pi}^{\tau,(A,D)} + \text{score}(X, I|\tau \to \pi) \right] \\ \max\limits_{(A',D') \lessdot (A,D)} S_I^{\pi,(A,D)} \end{cases} \tag{3.10}$$

Here we have omitted the possibility to attach scores to the transitions between states since this seems to be the most natural choice in the light of commonly used local alignment algorithms. One may add, however, a scoring term $\text{score}(A', D' \to A, D)$ to the second case in Eq. (3.10) as in Eq. (3.9). Such a score may of course depend on I and X, and may be used e.g., to favor the aligned portions to start with a particular nucleotide or to end a given sequence motif. Since only transitions from immediate predecessors are admitted, the scoring model must assume additive contributions from each sequence that is "activated" or "retired". In contrast, Eq. (3.8) allows arbitrary $\preceq$-preserving transitions and thus can also handle non-additive transitions scores.

The production rules of the grammar corresponding to Eq. (3.9) are of the form

$$\begin{aligned} &(A, D, \pi) \to (A, D, \tau)\mathbf{c}^{\pi \leftarrow \tau} \quad \text{for all } \tau \\ &(A, D, \pi) \to (A', D', \tilde{\pi}) \quad \text{for all } (A', D') \lessdot (A, D) \end{aligned} \tag{3.11}$$

In addition we need a starting rule of the form $S \to (R, X\backslash R, \tau)$ for all τ that generates all alignments with the correct state $(R, X\backslash R)$ at the right end and all corresponding end patterns.

3.4 Explicit Recursions for Partition Functions of Alignments

The partition function over the set of alignments $\mathfrak{A}(X|I, \pi, (A, D))$ is defined as sum of the Boltzmann factors of their scores, i.e.,

$$Z_I^{\pi,(A,D)} := \sum_{\mu \in \mathfrak{A}(X|I,\pi,(A,D))} \exp(\beta \, \text{score}(\mu)) \tag{3.12}$$

where β is the "inverse temperature" governing the relative importance of alignments with different scores. As an immediate consequence of Theorem 3.6 we obtain

$$Z_I^{\pi,(A,D)} = \sum_{\pi} \sum_{(A',D') \prec (A,D)} Z_{I-\pi}^{\tau,(A',D')} \exp(\beta \, \text{score}(X, I|\tau \to \pi, (A', D') \to (A, D))) \tag{3.13}$$

The rules of ADP would suggest that a partition function could be obtained from a "partition function version" of the recursion (3.9), which takes the form

$$Z_I^{A,D,\pi} = \sum_{\tau} Z_{I-\pi}^{A',D',\tau} \exp(\text{score}(\mathbf{c}^{\pi \leftarrow \tau})) + \sum_{(A',D') \lessdot (A,D)} Z_I^{A',D',\tau} \tag{3.14}$$

However, semantically, this is not what one wants to compute in a probabilistic setting because the grammar of Eq. (3.11) is ambiguous, i.e., it allows multiple distinct parses for the same alignment [29]. First, alignments in which a sequence X_p that is both left and right local is completely deleted are represented multiple times, because the direct transition from not yet active to done may occur in every alignment step and all these possibilities are accounted for as separate alignments. This contradicts our intuition for the conditions under which two alignments should be regarded as the same. Of course this "overcounting" has no effect when the goal is to maximize the score. It does affect the result, however, when the task is compute probabilities over ensembles of alignments.

A second, even worse ambiguity arises from the fact that we allowed ourselves to perform a sequence state changes without emitting any alignment column in between. This amounts to multiple paths in the lattice of Fig. 7 that lead to the same overall state change. To remedy this problem, we need to restructure the grammar. The basic idea is to ensure that every production emits a column, and makes at most one state change at the same time. Given the order structure of the alignment columns and the states, this ensures unambiguity. This modification comes at a cost, however. Now we have to allow any combination of state changes in a single step—so as to count it only once. Thus, in general there are exponentially many (in N) state transitions that need to be considered. The resulting grammar is of the form

$$(A, D, \pi) \to (A', D', \tau)\mathbf{c}^{\pi \leftarrow \tau} \quad \text{for all } (A', D') \preceq (A, D) \text{ and all } \tau \tag{3.15}$$

This grammar gives rise to the correct recursion (3.13).

To ensure that this grammar is unambiguous we need to enforce that a sequence X_p can be activated or retired only if one of its characters is emitted in the column $\mathbf{c}^{\pi \leftarrow \tau}$ that is emitted in the same step. This restriction now requires a separate treatment of sequences that remain completely unaligned, i.e., the first type of ambiguity mentioned above. Note that this is implicit in the definition of MSAs μ, which allows a transition from $\bullet$ to $\circ$ only after the zero-th column of the alignment.

4 Block-Local N-Way Alignments

The general treatment of partially local alignments suggests that further variations on the theme may also be of interest. In phylogenetic footprinting the goal is to find intervals of common length k such that their gap-less alignment maximizes a score [14]. In the present setting it would be natural to relax the no-gaps condition. Instead, one may ask for intervals $[b_p, e_p]$ for all $p \in X$ such that the global alignment of the infixes/substrings $X_p[b_p, e_p]$ maximizes the score. We call this the *block alignment problem*. This is can be seen as variant of the partial local N-way alignment where (1) all sequences are local at both ends, and (2) the state transitions are restricted to the rather trivial Hasse diagram $(\emptyset, \emptyset) \to (X, \emptyset) \to (\emptyset, X)$, i.e., all sequences are activated and then retired at common points in time. Naturally, end-gaps should be costly in this setting, i.e., in contrast to the unambiguous grammars discussed in the previous section we would also allow the first and last rows to contain gaps.

5 Concluding Remarks

In this contribution we have outlined a general formal framework towards treating partial, complex locality constraints in sequence alignment. Our approach was guided by exact DP algorithms for this class of problems. Of course the resulting algorithms are exponential in the number of sequences; after all they use the same recursive scheme as the well-known DP solution for global N-way MSA. This is not a fundamental shortcoming, however, since (a) the (decision version of the) global N-way sequence alignment problem is well known to be NP-hard [40,77], and (b) the problem is tractable in practice and of relevance for practical applications for small numbers of input sequences. Recent results from complexity theory, furthermore, strongly indicate that N-way MSA problems cannot be solved in $O(n^{N-\epsilon})$ time for input sequences of length n unless common believed conjectures in the theory of computational complexity are false [1,2]. That is, the dynamic programming solutions are optimal (up to logarithmic factors e.g., by means of the well-known "Four Russians" methods [8]).

Much of the notational complications in this contribution are caused by the need to show that very general classes of scoring models are compatible with the notion of partially local alignments. This separation of concerns should in particular become practically useful when the grammar formalism is used to construct DP algorithms. The alignment types λ then simply become classes in a classified DP framework.

At present, we do not provide an implementation. A special case, however, is in practical use for breakpoint determination [4]. In [85] it has been demonstrated that N-way global alignments as well a variety of complex alignment algorithms can be constructed quite easily with the help of grammar products. In this context a general implementation in Haskell has been provided. A complete implementation of the framework outline in this contribution, however, requires some further developments, in particular an extension of the underlying theory to corresponding products of the scoring algebras and the development of a principled manner to construct the lattice of alignment states. While it is of course possible to implement models for a small number of sequences explicitly, this quickly becomes tedious and prone to errors. For alignment models are more complex than the example outlined in "Appendix A" a machinery that generates the code automatically along the line of [85] will be necessary in practice.

In practical applications we expect that partially local alignments will in general appear as components in workflows. This is indeed the case in application described in [4]: there, the putative location of a breakpoint is located approximately from a comparison of the gene orders in a pair of mitogenomes and the sequencing containing this region is extracted. The partially local alignment is then applied to these preprocessed input sequences. It is unlikely that the exact DP solutions to partially local alignments will be of much practical use for more than 3 input sequences—at least in computational biology application where sequences are typically long. This may well turn out differently in alignment applications to language data, such as the one briefly outlined in the introduction. At the very least, they may serve for comparisons in benchmarking heuristics for partially local alignment tasks. The grammar versions can in principle also be used to generate benchmark data for such problems. Currently test sets for benchmarking alignments such as `BaliBase` [75] are geared towards global alignments even though in some cases of the performance on particular blocks of conserved sequence enters the performance metrics [76].

The theoretical framework developed here is not limited to pure sequence alignments. It also pertains, for instance, to the concurrent alignments of RNA secondary structures. The Sankoff algorithm [67] also comes with local and semi-global variants, see e.g., [80,81] that could in principle be generalized to partially local N-way problems. Of course, the resulting algorithms will be even more expensive, scaling with $O(n^{3N})$ rather than $O(n^N)$ with sequence length due to the cubic scaling of the RNA folding part. Nevertheless it is at least of theoretical interest that the global alignment part, above denoted by $\mathfrak{A}_I(.)$, can be replaced without affecting the handling of partial locality constraints by any scheme that produces global alignments.

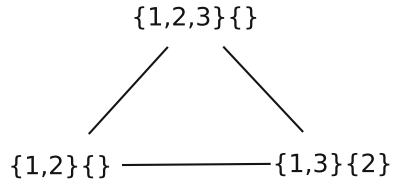
Acknowledgements Open access funding provided by Max Planck Society.

Appendix

A. Breakpoint Alignments with Naïve Affine Gap Costs

The breakpoint detection algorithm of [4] is a specialized three-way alignment comprising one global, one left-local, and one right-local sequence. The global one forms the reference in the ancestral gene order, the two partially local ones are the sequences adjacent to the breakpoint in the derived gene order. In [4] a simple additive gap cost model was used. In order to illustrate use of naïve affine gap costs in our setting we show below the complete recursion.

The Hasse diagram of this problem reads



In the recursion we use the functions score() to evaluate the pairwise contributions to the sum-of-pairs score, as well as lookup() to evaluate the contribution of pairwise sequence (mis)matches. Their definitions for the specific application will be defined at the end of the section.

The indices k, l, and m referring to sequence positions vary in the ranges $k \in [2, \ell_1]$, $l \in [2, \ell_2]$, and $m \in [2, \ell_3]$. We start with $S^{\{1,2\},\emptyset}$. The base cases are

$$
\begin{aligned}
S^{\{1,2\},\emptyset}_{0,0} &= 0 \\
S^{\{1,2\},\emptyset,\bullet\circ}_{1,0} &= S^{\{1,2\},\emptyset}_{0,0} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\bullet\circ}_{k,0} &= S^{\{1,2\},\emptyset}_{k-1,0} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\circ\bullet}_{0,1} &= S^{\{1,2\},\emptyset}_{0,0} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\circ\bullet}_{0,l} &= S^{\{1,2\},\emptyset}_{0,l-1} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\bullet\circ}_{1,1} &= S^{\{1,2\},\emptyset}_{0,0} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\circ\bullet}_{1,1} &= S^{\{1,2\},\emptyset}_{0,0} + \text{gapopen} \\
S^{\{1,2\},\emptyset,\bullet\bullet}_{1,1} &= S^{\{1,2\},\emptyset}_{0,0} + \text{lookup}(x_1(1), x_2(1))
\end{aligned}
$$

The recursions for the general case of the pairwise alignments of the first two sequences conform to the usual form of Gotoh's algorithm:

$$
\begin{aligned}
S^{\{1,2\},\emptyset,\bullet\bullet}_{k,l} &= \max\Big\{ S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l-1} + \text{score}((1,1),(1,1),(k,l),\{1,2\}),\ S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,l-1} + \text{score}((1,0),(1,1),(k,l),\{1,2\}), \\
&\qquad S^{\{1,2\},\emptyset,\circ\bullet}_{k-1,l-1} + \text{score}((0,1),(1,1),(k,l),\{1,2\}) \Big\} \\
S^{\{1,2\},\emptyset,\bullet\circ}_{k,l} &= \max\Big\{ S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l} + \text{score}((1,1),(1,0),(k,l),\{1,2\}),\ S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,l} + \text{score}((1,0),(1,0),(k,l),\{1,2\}), \\
&\qquad S^{\{1,2\},\emptyset,\circ\bullet}_{k-1,l} + \text{score}((0,1),(1,0),(k,l),\{1,2\}) \Big\} \\
S^{\{1,2\},\emptyset,\circ\bullet}_{k,l} &= \max\Big\{ S^{\{1,2\},\emptyset,\bullet\bullet}_{k,l-1} + \text{score}((1,1),(0,1),(k,l),\{1,2\}),\ S^{\{1,2\},\emptyset,\bullet\circ}_{k,l-1} + \text{score}((1,0),(0,1),(k,l),\{1,2\}), \\
&\qquad S^{\{1,2\},\emptyset,\circ\bullet}_{k,l-1} + \text{score}((0,1),(0,1),(k,l),\{1,2\}) \Big\}
\end{aligned}
$$

Since we can reach $S^{\{1,3\},\{2\}}$ also from $S^{\{1,2,3\},\emptyset}$, we will first define the latter, 3-dimensional matrix. The bases cases are the following:

$$
\begin{aligned}
S^{\{1,2,3\},\emptyset}_{0,0,0} &= 0 \\
S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{1,0,0} &= \max\Big\{ S^{\{1,2,3\},\emptyset}_{0,0,0},\ S^{\{1,2\},\emptyset}_{0,0} \Big\} + \text{gapopen} \\
S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,0,0} &= \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,0,0},\ S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,0} \Big\} + \text{score}((1,0,0),(1,0,0),(k,0,0),\{1,2,3\}) \\
S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,1,0} &= \max\Big\{ S^{\{1,2,3\},\emptyset}_{0,0,0},\ S^{\{1,2\},\emptyset}_{0,0} \Big\} + \text{gapopen}
\end{aligned}
$$

$$S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l,0} = \max\left\{S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l-1,0}, S^{\{1,2\},\emptyset,\circ\bullet}_{0,l-1}\right\} + \text{score}((0,1,0),(0,1,0),(0,l,0),\{1,2,3\})$$

$$S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,0,1} = S^{\{1,2,3\},\emptyset}_{0,0,0} + \text{gapopen}$$

$$S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,0,m} = S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,0,m-1} + \text{score}((0,0,1),(0,0,1),(0,0,m),\{1,2,3\})$$

$$S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{1,1,0} = \max\left\{S^{\{1,2,3\},\emptyset}_{0,0,0}, S^{\{1,2\},\emptyset}_{0,0}\right\} + \text{lookup}(x_1(1), x_2(1))$$

The entries for the first two sequences are:

$$\begin{aligned}
S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l,0} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l-1,0} + \text{score}((1,1,0),(1,1,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l-1,0} + \text{score}((1,0,0),(1,1,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l-1,0} + \text{score}((0,1,0),(1,1,0),(k,l,0),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l,0} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,0} + \text{score}((1,1,0),(1,0,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l,0} + \text{score}((1,0,0),(1,0,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l,0} + \text{score}((0,1,0),(1,0,0),(k,l,0),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l,0} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l-1,0} + \text{score}((1,1,0),(0,1,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l-1,0} + \text{score}((1,0,0),(0,1,0),(k,l,0),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l-1,0} + \text{score}((0,1,0),(0,1,0),(k,l,0),\{1,2,3\})\Big\}
\end{aligned}$$

For the first and third sequence we get the following recursions, which in contrast to the above also include the activation of the third sequence:

$$\begin{aligned}
S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,0,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,0,m-1} + \text{score}((1,0,1),(1,0,1),(k,0,m),\{1,2,3\}), \max\left\{S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,0,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,0}\right\}\\
& + \text{score}((1,0,0),(1,0,1),(k,0,m),\{1,2,3\}), S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,0,m-1} + \text{score}((0,0,1),(1,0,1),(k,0,m),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,0,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,0,m} + \text{score}((1,0,1),(1,0,0),(k,0,m),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,0,m} + \text{score}((1,0,0),(1,0,0),(k,0,m),\{1,2,3\}),\\
& S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,0,m} + \text{score}((0,0,1),(1,0,0),(k,0,m),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,0,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,0,m-1} + \text{score}((1,0,1),(0,0,1),(k,0,m),\{1,2,3\}), \max\left\{S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,0,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k,0}\right\}\\
& + \text{score}((1,0,0),(0,0,1),(k,0,m),\{1,2,3\}), S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,0,m-1} + \text{score}((0,0,1),(0,0,1),(k,0,m),\{1,2,3\})\Big\}
\end{aligned}$$

Similarly, the recursions for the alignments of the second and third sequence involve terms referring to the activation of the third sequence:

$$\begin{aligned}
S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{0,l,m} = \max\Big\{ & \max\left\{S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{0,l-1,m-1}, S^{\{1,2\},\emptyset,\circ\bullet}_{0,l-1}\right\} + \text{score}((0,1,1),(0,1,1),(0,l,m),\{1,2,3\}), S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l-1,m-1}\\
& + \text{score}((0,1,0),(0,1,1),(0,l,m),\{1,2,3\}), S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,l-1,m-1} + \text{score}((0,0,1),(0,1,1),(0,l,m),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{0,l-1,m} + \text{score}((0,1,1),(0,1,0),(0,l,m),\{1,2,3\}), S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l-1,m}
\end{aligned}$$

$$+\,\text{score}((0,1,0),(0,1,0),(0,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,l-1,m} + \text{score}((0,0,1),(0,1,0),(0,l,m),\{1,2,3\})\Big\}$$

$$S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{0,l,m-1} + \text{score}((0,1,1),(0,0,1),(0,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{0,l,m-1}$$
$$+\,\text{score}((0,1,0),(0,0,1),(0,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{0,l,m-1} + \text{score}((0,0,1),(0,0,1),(0,l,m),\{1,2,3\})\Big\}$$

Since we "look back" two steps, it is necessary to initialize the first the diagonal step explicitly:

$$S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{1,1,1} = \max\Big\{ S^{\{1,2,3\},\emptyset}_{0,0,0} + \text{lookup}(x_1(1),x_2(1)) + \text{lookup}(x_1(1),x_3(1)) + \text{lookup}(x_2(1),x_3(1)),$$

We are now in the position to consider the general case. As in some of the pairwise alignments above, we have to track the activation of the third sequence. This may only occur when sequence 3 contributes a letter to the current alignment column, not when a gap is inserted for this sequence. Note that this handling of transitions to and from the active state is a modeling decision: it may be handled differently, although it seems most natural to assume that the activation of a sequence has to occur together with the insertion of a character to the alignment.

$$S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l-1,m-1} + \text{score}((1,1,1),(1,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l-1,m-1}, S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l-1}\Big\}$$
$$+\,\text{score}((1,1,0),(1,1,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,l-1,m-1} + \text{score}((1,0,1),(1,1,1),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l-1,m-1} + \text{score}((0,1,1),(1,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l-1,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,l-1}\Big\}$$
$$+\,\text{score}((1,0,0),(1,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l-1,m-1}, S^{\{1,2\},\emptyset,\circ\bullet}_{k-1,l-1}\Big\}$$
$$+\,\text{score}((0,1,0),(1,1,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,l-1,m-1} + \text{score}((0,0,1),(1,1,1),(k,l,m),\{1,2,3\})\Big\}$$

$$S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l-1,m} + \text{score}((1,1,1),(1,1,0),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l-1,m} + \text{score}((1,1,0),(1,1,0),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,l-1,m} + \text{score}((1,0,1),(1,1,0),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l-1,m} + \text{score}((0,1,1),(1,1,0),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l-1,m} + \text{score}((1,0,0),(1,1,0),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l-1,m} + \text{score}((0,1,0),(1,1,0),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,l-1,m} + \text{score}((0,0,1),(1,1,0),(k,l,m),\{1,2,3\})\Big\}$$

$$S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l,m-1} + \text{score}((1,1,1),(1,0,1),(k,l,m),\{1,2,3\}), \max\{S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,m-1}, S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l}\}$$
$$+\,\text{score}((1,1,0),(1,0,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,l,m-1} + \text{score}((1,0,1),(1,0,1),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l,m-1} + \text{score}((0,1,1),(1,0,1),(k,l,m),\{1,2,3\}), \max\{S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,l}\}$$
$$+\,\text{score}((1,0,0),(1,0,1),(k,l,m),\{1,2,3\}), \max\{S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k-1,l}\}$$
$$+\,\text{score}((0,1,0),(1,0,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,l,m-1} + \text{score}((0,0,1),(1,0,1),(k,l,m),\{1,2,3\})\Big\}$$

$$S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k,l-1,m-1} + \text{score}((1,1,1),(0,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l-1,m-1}, S^{\{1,2\},\emptyset,\bullet\bullet}_{k,l-1}\Big\}$$
$$+\,\text{score}((1,1,0),(0,1,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,l-1,m-1} + \text{score}((1,0,1),(0,1,1),(k,l,m),\{1,2,3\}),$$
$$S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k,l-1,m-1} + \text{score}((0,1,1),(0,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l-1,m-1}, S^{\{1,2\},\emptyset,\bullet\circ}_{k,l-1}\Big\}$$
$$+\,\text{score}((1,0,0),(0,1,1),(k,l,m),\{1,2,3\}), \max\Big\{ S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l-1,m-1}, S^{\{1,2\},\emptyset,\circ\bullet}_{k,l-1}\Big\}$$
$$+\,\text{score}((0,1,0),(0,1,1),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,l-1,m-1} + \text{score}((0,0,1),(0,1,1),(k,l,m),\{1,2,3\})\Big\}$$

$$S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l,m} = \max\Big\{ S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l,m} + \text{score}((1,1,1),(1,0,0),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,m}$$
$$+\,\text{score}((1,1,0),(1,0,0),(k,l,m),\{1,2,3\}),\, S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k-1,l,m} + \text{score}((1,0,1),(1,0,0),(k,l,m),\{1,2,3\}),$$

$$
\begin{aligned}
&S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l,m} + \text{score}((0,1,1),(1,0,0),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k-1,l,m} + \text{score}((1,0,0),(1,0,0),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k-1,l,m} + \text{score}((0,1,0),(1,0,0),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k-1,l,m} + \text{score}((0,0,1),(1,0,0),(k,l,m),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k,l-1,m}\\
&+ \text{score}((1,1,1),(0,1,0),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l-1,m} + \text{score}((1,1,0),(0,1,0),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,l-1,m} + \text{score}((1,0,1),(0,1,0),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k,l-1,m} + \text{score}((0,1,1),(0,1,0),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l-1,m} + \text{score}((1,0,0),(0,1,0),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l-1,m} + \text{score}((0,1,0),(0,1,0),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,l-1,m} + \text{score}((0,0,1),(0,1,0),(k,l,m),\{1,2,3\})\Big\}\\
S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,l,m} = \max\Big\{ & S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k,l,m-1} + \text{score}((1,1,1),(0,0,1),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l,m-1} + \text{score}((1,1,0),(0,0,1),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\bullet\circ\bullet}_{k,l,m-1} + \text{score}((1,0,1),(0,0,1),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k,l,m-1} + \text{score}((0,1,1),(0,0,1),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\bullet\circ\circ}_{k,l,m-1} + \text{score}((1,0,0),(0,0,1),(k,l,m),\{1,2,3\}),\\
&S^{\{1,2,3\},\emptyset,\circ\bullet\circ}_{k,l,m-1} + \text{score}((0,1,0),(0,0,1),(k,l,m),\{1,2,3\}),\ S^{\{1,2,3\},\emptyset,\circ\circ\bullet}_{k,l,m-1} + \text{score}((0,0,1),(0,0,1),(k,l,m),\{1,2,3\})\}
\end{aligned}
$$

It remains to fill the matrix $S^{\{1,3\},\{2\}}$. Again we allow the termination of the second sequence only to together with the insertion of a non-gap character. The special case that sequence 2 remains completely unaligned is taken care of in the initialization of this matrix:

$$
\begin{aligned}
S^{\{1,3\},\{2\}}_{0,0} &= S^{\{1,2\},\emptyset}_{0,0}\\
S^{\{1,3\},\{2\},\bullet\circ}_{1,0} &= \max\left\{S^{\{1,3\},\{2\}}_{0,0}, S^{\{1,2,3\},\emptyset}_{0,0}\right\} + \text{gapopen}\\
S^{\{1,3\},\{2\},\bullet\circ}_{k,0} &= \max\left\{S^{\{1,3\},\{2\},\bullet\circ}_{k-1,0}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,0}\right\} + \text{score}((1,0),(1,0),(k,0),\{1,3\})\\
S^{\{1,3\},\{2\},\circ\bullet}_{0,1} &= \max\left\{S^{\{1,3\},\{2\}}_{0,0}, S^{\{1,2,3\},\emptyset}_{0,0}, S^{\{1,2\},\emptyset}_{0,0}\right\} + \text{gapopen}\\
S^{\{1,3\},\{2\},\circ\bullet}_{0,m} &= \max\left\{S^{\{1,3\},\{2\},\circ\bullet}_{0,m-1}, \max_l S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{0,l,m-1}\right\} + \text{score}((0,1),(0,1),(0,l),\{1,3\})\\
S^{\{1,3\},\{2\},\bullet\bullet}_{1,1} &= S^{\{1,3\},\{2\}}_{0,0} + \text{lookup}(x_1(1), x_3(1))\\
S^{\{1,3\},\{2\},\bullet\circ}_{1,1} &= \max\left\{S^{\{1,3\},\{2\},\circ\bullet}_{0,1}, S^{\{1,2,3\},\emptyset,\bullet\circ}_{1,0}\right\} + \text{score}((0,1),(1,0),(1,1),\{1,3\})\\
S^{\{1,3\},\{2\},\circ\bullet}_{1,1} &= \max\left\{S^{\{1,3\},\{2\},\bullet\circ}_{1,0}, S^{\{1,2,3\},\emptyset,\bullet\circ}_{1,0}\right\} + \text{score}((1,0),(0,1),(1,1),\{1,3\})\\
S^{\{1,3\},\{2\},\bullet\bullet}_{k,m} &= \max\Big\{\max\left\{S^{\{1,3,\},\{2\},\bullet\bullet}_{k-1,m-1}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l,m-1}\right\} + \text{score}((1,1)(1,1),(k,m),\{1,3\}),\\
&\quad \max\left\{S^{\{1,3\},\{2\},\bullet\circ}_{k-1,m-1}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,m-1}\right\} + \text{score}((1,0),(1,1),(k,m),\{1,3\}),\\
&\quad \max\left\{S^{\{1,3\},\{3\},\circ\bullet}_{k-1,m-1}, \max_l S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l,m-1}\right\} + \text{score}((0,1),(1,1),(k,m),\{1,3\}),\\
&\quad \max_l\left\{S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l}, S^{\{1,2\},\emptyset,\circ\bullet}_{k-1,l}\right\} + \text{score}((0,0),(1,1),(k,m),\{1,3\})\Big\}\\
S^{\{1,3\},\{2\},\bullet\circ}_{k,m} &= \max\Big\{\max\left\{S^{\{1,3\},\{2\},\bullet\bullet}_{k-1,m}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k-1,l,m}\right\} + \text{score}((1,1),(1,0),(k,m),\{1,3\}),\\
&\quad \max\left\{S^{\{1,3\},\{2\},\bullet\circ}_{k-1,m}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k-1,l,m}\right\} + \text{score}((1,0),(1,0),(k,m),\{1,3\}),
\end{aligned}
$$

$$
\begin{aligned}
S^{\{1,3\},\{2\},\circ\bullet}_{k,m} = \max \Bigg\{ & \max \left\{ S^{\{1,3\},\{2\},\circ\bullet}_{k-1,m}, \max_l S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k-1,l,m} \right\} + \text{score}((0,1),(1,0),(k,m),\{1,3\})\} \\
& \max \left\{ S^{\{1,3\},\{2\},\bullet\bullet}_{k,m-1}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\bullet}_{k,l,m-1} \right\} + \text{score}((1,1),(0,1),(k,m),\{1,3\}), \\
& \max \left\{ S^{\{1,3\},\{2\},\bullet\circ}_{k,m-1}, \max_l S^{\{1,2,3\},\emptyset,\bullet\bullet\circ}_{k,l,m-1} \right\} + \text{score}((1,0),(0,1),(k,m),\{1,3\}), \\
& \max \left\{ S^{\{1,3\},\{2\},\circ\bullet}_{k,m-1}, \max_l S^{\{1,2,3\},\emptyset,\circ\bullet\bullet}_{k,l,m-1} \right\} + \text{score}((0,1),(0,1),(k,m),\{1,3\}), \\
& \max_l \left\{ S^{\{1,2\},\emptyset,\bullet\bullet}_{k-1,l}, S^{\{1,2\},\emptyset,\circ\bullet}_{k-1,l} \right\} + \text{score}((0,0),(0,1),(k,m),\{1,3\}) \Bigg\}
\end{aligned}
$$

The *scoring function* score() appears in the recursions to evaluate the pairwise contributions to the sum of pairs score. It depends on two position indices (p, q), the restriction of the gap patterns τ and π to these two sequences, and two sequences i and j. In extension of the model in [4] we set

$$
\text{score}(\tau,\pi,(p,q),\{i,j\}) = \begin{cases}
0 & \text{if } \pi_1 = 0, i = 2, p = \ell_2 & \text{end seq 2} \\
0 & \text{if } \pi_2 = 0, j = 2, q = \ell_2 & \text{end seq 2} \\
0 & \text{if } \tau_2 = \pi_2 = q = 0, j = 3 & \text{start seq 3} \\
\mathit{gapextend} & \text{if } \tau_1 = 0, \pi_1 = 0 & \text{gap extend} \\
\mathit{gapextend} & \text{if } \tau_2 = 0, \pi_2 = 0 & \text{gap extend} \\
\mathit{gapopen} & \text{if } \tau_1 = 1, \pi_1 = 0 & \text{gap open} \\
\mathit{gapopen} & \text{if } \tau_2 = 1, \pi_2 = 0 & \text{gap open} \\
\mathit{lookup}(x_{i,p-1}, x_{j,q-1}) & \text{else} & \text{look up (mis)match score}
\end{cases}
$$

The sum-of-pair scoring model, finally, implies $\text{score}(\tau,\pi,(k,l,m),\{1,2,3\}) = \text{score}(\tau,\pi,(k,l),\{1,2\}) + \text{score}(\tau,\pi,(k,m),\{1,3\}) + \text{score}(\tau,\pi,(l,m),\{2,3\})$.

B. A Shape Encoding for Exact Handling of Arbitrary Gap Costs

A problematic issue with cap cost function in the multiple sequence alignments is that the projected pairwise alignments can contain all-gap columns. It is not sufficient, therefore, to characterize alignments by the gap pattern π in their last column. The situation is depicted in Fig. 8. For each pair (p, q) of sequences we define λ_{pp} as the number of terminal gaps in the alignment. The off-diagonal elements λ_{pq} counts the number of positions deleted from p relative to q and we set $\lambda_{qp} = -\lambda_{pq}$ to account for positions are inserted in p relative to q. In particular, thus $\lambda_{pq} = 0$ identifies a (mis)match at the end of the alignment. This description is inspired by the notion of shapes introduced in [43] in the related problem of aligning alignments with exact affine sum of pair costs.

The values of λ_{pq} can be easily tracked at the end of the alignment. We write λ' for the pattern up to the penultimate column, λ for the pattern up to the last column, and, as usual, π for gap pattern of the last column. By construction, we can keep track of pairs of rows separately. The diagonal terms are simply $\lambda_{pp} = \lambda'_{pp} + 1$ if $\pi = 0$, i.e., the gap in p is extended, and $\lambda_{pp} = 1$. For the offdiagonal terms we have

$$
\lambda_{pq} = \begin{cases}
0 & \text{if } \pi_p = \pi_q = 1 & \text{(mis)match} \\
\lambda'_{pq} & \text{if } \pi_p = \pi_q = 0 & \text{double gap} \\
\lambda'_{pq} + 1 & \text{if } \pi_p = 1,\ \pi_q = 0,\ \lambda'_{pq} \geq 0 & \text{extension of deletion in } q \\
+1 & \text{if } \pi_p = 1,\ \pi_q = 0,\ \lambda'_{pq} \leq 0 & \text{opening of deletion in } q \\
-1 & \text{if } \pi_p = 0,\ \pi_1 = 0,\ \lambda'_{pq} \geq 0 & \text{opening of deletion in } p \\
\lambda'_{pq} - 1 & \text{if } \pi_p = 0,\ \pi_1 = 0,\ \lambda'_{pq} \leq 0 & \text{extension of deletion in } p
\end{cases} \tag{5.1}
$$

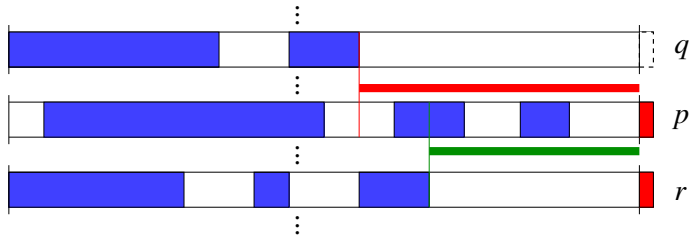


Fig. 8 Encoding of gap patterns for very general gap cost models. Occupied positions are shown in blue, gaps in white. Only three rows p, q, and r of a large MSA are show. The terminal gap between p and q is shown by the red bar; it corresponds to a deletion of the two rightmost blue boxes from p relative to q. Similarly, the blue part above the green bar are deleted from p relative to r. Comparing q and r, the last blue box is deleted from r, that terminal gap matches the red bar. The last column shows a possible extension of the alignment. In this case, the deletion from p to q is extended, while the gap in r relative to p (and q) is closed by a (mis)match shown in red (color figure online)

Note the cases $\lambda'_{pq} = 0$ are consistently defined in the overlapping cases. Denoting this pairwise rule as $\lambda = \lambda' \oplus \pi$ allows us to write

$$\mathfrak{A}(X, I, \lambda) = \bigcup_{\pi} \bigcup_{\lambda' : \lambda' \oplus \pi = \lambda} \mathfrak{A}(X, I - \pi, \lambda') +\!\!+ \mathfrak{c}(X, I, \pi, \lambda', \lambda) \tag{5.2}$$

The representation in particular supports sum-of-pair scores with arbitrary (subadditive) pairwise gap scores, since the type λ provides information on the true length of the pairwise gaps. We have made no effort here to devise a an encoding that is most efficient in practise. For instance, in an affine gap cost setting, the diagonal elements of λ are not used.

References

1. Abboud, A., Backurs, A., Williams, V.V.: Tight hardness results for LCS and other sequence similarity measures. In: 56th Annual Symposium on Foundations of Computer Science (FOCS), pp. 59–78. IEEE (2015)
2. Abboud, A., Hansen, T.D., Williams, V.V., Williams, R.: Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC '16), New York, NY, pp. 375–388 (2016)
3. Al Ait, L., Yamak, Z., Morgenstern, B., Morgenstern, B.: DIALIGN at GOBICS—multiple sequence alignment using various sources of external information. Nucleic Acids Res. **41**, W3–W7 (2013)
4. Al Arab, M., Bernt, M., zu Siederdissen, C.H., Tout, K.: Partially local three-way alignments and the sequence signatures of mitochondrial genome rearrangements. Alg. Mol. Biol. **12**, 22 (2017)
5. Altschul, S.F.: Gap costs for multiple sequence alignment. J. Theor. Biol. **138**, 297–309 (1989)
6. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. **215**, 403–410 (1990)
7. Angermüller, C., Biegert, A., Söding, J.: Discriminative modelling of context-specific amino acid substitution probabilities. Bioinformatics **28**, 3240–3247 (2012)
8. Arlazarov, V., Dinic, E., Kronrod, M., Faradzev, I.: On economical construction of the transitive closure of a directed graph. Dokl. Akad. Nauk. **11**, 1209–1210 (1970)
9. Baichoo, S., Ouzounis, C.A.: Computational complexity of algorithms for sequence comparison, short-read assembly and genome alignment. Biosystems **156**(157), 72–85 (2017)
10. Bailey, T.L., Williams, N., Misleh, C., Wilfred, W.L.: MEME: discovering and analyzing DNA and protein sequence motifs. Nucleic Acids Res. **34**, W369–W373 (2006)
11. Bhattacharya, T., Retzlaff, N., Blasi, D., Croft, W., Cysouw, M., Hruschka, D., Maddieson, I., Müller, L., Smith, E., Stadler, P.F., Starostin, G., Youn, H.: Studying language evolution in the age of big data. J. Lang. Evol. (2018)
12. Blanchette, M.: Computation and analysis of genomic multi-sequence alignments. Annu. Rev. Genomics Hum. Genet. **8**, 193–213 (2007)
13. Blanchette, M., Tompa, M.: `FootPrinter`: a program designed for phylogenetic footprinting. Nucleic Acids Res. **31**, 3840–3842 (2003)
14. Blanchette, M., Schwikowski, B., Tompa, M.: Algorithms for phylogenetic footprinting. J. Comput. Biol. **9**, 211–223 (2002)

15. Bonizzoni, P., Della Vedova, G.: The complexity of multiple sequence alignment with SP-score that is a metric. Theor. Comput. Sci. **259**, 63–79 (2001)
16. Brudno, M., Chapman, M., Göttgens, B., Batzoglou, S., Morgenstern, B.: Fast and sensitive multiple alignment of large genomic sequences. BMC Bioinform. **4**, 66 (2003)
17. Bucher, P., Hoffmann, K.: A sequence similarity search algorithm based on a probabilistic interpretation of an alignment scoring system. In: States, D.J., Agarwal, P., Gaasterland, T., Hunter, L., Smith, R.F. (eds.) Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology (ISMB '96), pp. 44–50. AAAI Press, Menlo Park, CA (1996)
18. Bussotti, G., Raineri, E., Erb, I., Zytnicki, M., Wilm, A., Beaudoing, E., Bucher, P., Notredame, C.: BlastR-fast and accurate database searches for non-coding RNAs. Nucleic Acids Res. **39**, 6886–6895 (2011)
19. Carrillo, H., Lipman, D.: The multiple sequence alignment problem in biology. SIAM J. Appl. Math. **48**, 1073–1082 (1988)
20. Chowdhurya, B., Garaib, G.: A review on multiple sequence alignment from the perspective of genetic algorithm. Genomics **109**, 419–431 (2017)
21. Collingridge, P.W., Kelly, S.: Mergealign: improving multiple sequence alignment performance by dynamic reconstruction of consensus multiple sequence alignments. BMC Bioinform. **13**, 117 (2012)
22. Corel, E., Pitschi, F., Morgenstern, B.: A min-cut algorithm for the consistency problem in multiple sequence alignment. Bioinformatics **26**, 1015–1021 (2010)
23. Cysouw, M., Jung, H.: Cognate identification and alignment using practical orthographies. In: Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology, pp. 109–116. Association for Computational Linguistics (2007)
24. Dewey, T.G.: A sequence alignment algorithm with an arbitrary gap penalty function. J. Comput. Biol. **8**, 177–190 (2001)
25. Dowell, R.D., Eddy, S.R.: Evaluation of several lightweight stochastic context free grammars for RNA secondary structure prediction. BMC Bioinform. **5**, 71 (2004)
26. Eddy, S.R.: A probabilistic model of local sequence alignment that simplifies statistical significance estimation. PLoS Comput. Biol. **4**, e1000069 (2008)
27. Elias, I.: Settling the intractability of multiple alignment. J. Comput. Biol. **13**, 1323–1339 (2006)
28. Fonseca, N.A., Rung, J., Brazma, A., Marioni, J.C.: Tools for mapping high-throughput sequencing data. Bioinformatics **28**, 3169–3177 (2012)
29. Giegerich, R.: Explaining and controlling ambiguity in dynamic programming. In: Giancarlo, R., Sankoff, D. (eds.) Combinatorial Pattern Matching CPM 2000, vol. 1848, pp. 46–59. Springer, Berlin (2002)
30. Giegerich, R., Meyer, C., Steffen, P.: A discipline of dynamic programming over sequence data. Sci. Comput. Prog. **51**, 215–263 (2004)
31. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**, 705708 (1982)
32. Gotoh, O.: Alignment of three biological sequences with an efficient traceback procedure. J. Theor. Biol. **121**, 327–337 (1986)
33. Gupta, S.K., Kececioglu, J.D., Schäffer, A.A.: Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. J. Comput. Biol. **2**, 459–472 (1995)
34. Hertel, J., de Jong, D., Marz, M., Rose, D., Tafer, H., Tanzer, A., Schierwater, B., Stadler, P.F.: Non-coding RNA annotation of the genome of Trichoplax adhaerens. Nucleic Acids Res. **37**, 1602–1615 (2009)
35. Hirosawa, M., Totoki, Y., Hoshida, M., Ishikawa, M.: Comprehensive study on iterative algorithms of multiple sequence alignment. Comput. Appl. Biosci. **11**, 13–18 (1995)
36. Hoffmann, S., Otto, C., Doose, G., Tanzer, A., Langenberger, D., Christ, S., Kunz, M., Holdt, L.M., Teupser, D., Hackermüller, J., Stadler, P.F.: A multi-split mapping algorithm for circular RNA, splicing, trans-splicing, and fusion detection. Genome Biol. **15**, R34 (2014)
37. Hogeweg, P., Hesper, B.: The alignment of sets of sequences and the construction of phyletic trees: an integrated method. J. Mol. Evol. **20**, 175–186 (1984)
38. James Kent, W.: BLAT—the BLAST-like alignment tool. Genome Res. **12**, 656–664 (2002)
39. Jones, N.C., Pevzner, P.A.: An Introduction to Bioinformatics. MIT Press, Cambride (2004). Problem 6.22
40. Just, W.: Computational complexity of multiple sequence alignment with SP-score. J. Comput. Biol. **8**, 615–623 (2001)
41. Katoh, K., Standley, D.M.: MAFFT: iterative refinement and additional methods. Methods Mol. Biol. **1079**, 131–146 (2014)
42. Kececioglu, J.D.: The maximum weight trace problem in multiple sequence alignment. In: Proceedings of the 4th Symposium on Combinatorial Pattern Matching, volume 684 of Lecture Notes Computer Science, pp. 106–119. Springer, Berlin (1993)
43. Kececioglu, J., Starrett, D.: Aligning alignments exactly. In: Bourne, P.E., Gusfield, D. (eds.) Proceedings of the 8th ACM Conference on Research in Computational Molecular Biology (RECOMB), pp. 85–96. ACM, New York, NY (2004)
44. Kececioglu, J.D., Lenhof, H.-P., Mehlhorn, K., Mutzel, P., Reinert, K., Vingron, M.: A polyhedral approach to sequence alignment problems. Discrete Appl. Math. **104**, 143–186 (2000)
45. Konagurthu, A.S., Whisstock, J., Stuckey, P.J.: Progressive multiple alignment using sequence triplet optimization and three-residue exchange costs. J. Bioinform. Comput. Biol. **2**, 719–745 (2004)
46. Kondrak, G.: A new algorithm for the alignment of phonetic sequences. In: Proceedings of NAACL 2000 1st Meeting of the North American Chapter of the Association for Computational Linguistics, pp. 288–295. Morgan Kaufmann Publishers, San Francisco, CA (2000)
47. Kondrak, G.: Phonetic alignment and similarity. Comput. Humanit. **37**, 273–291 (2003)

48. Kruspe, M., Stadler, P.F.: Progressive multiple sequence alignments from triplets. BMC Bioinform. **8**, 254 (2007). https://doi.org/10.1186/1471-2105-8-254
49. List, J.-M., Greenhill, S.J., Gray, R.D.: The potential of automatic word comparison for historical linguistics. PLoS ONE **12**, e0170046 (2017)
50. Lukashin, A.V., Rosa, J.J.: Local multiple sequence alignment using dead-end elimination. Bioinformatics **15**, 947–953 (1999)
51. Manthey, B.: Non-approximability of weighted multiple sequence alignment. Theor. Comput. Sci. **296**, 179–192 (2003)
52. Margulies, E.H., Blanchette, M., Haussler, D., Green, E.D.: Identification and characterization of multi-species conserved sequences. Genome Res. **13**, 2507–2518 (2003)
53. Meier, A., Söding, J.: Context similarity scoring improves protein sequence alignments in the midnight zone. Bioinformatics **31**, 674–681 (2015)
54. Miyazawa, S.: A reliable sequence alignment method based on probabilities of residue correspondences. Protein Eng. **8**, 999–1009 (1994)
55. Morgenstern, B., Frech, K., Dress, A., Werner, T.: DIALIGN: finding local similarities by multiple sequence alignment. Bioinformatics **14**(3), 290–294 (1998)
56. Morgenstern, B., Stoye, J., Dress, A.W.M.: Consistent equivalence relations: a set-theoretical framework for multiple sequence alignments. Technical report, University of Bielefeld, FSPM (1999)
57. Mückstein, U., Hofacker, I.L., Stadler, P.F.: Stochastic pairwise alignments. Bioinformatics **60**, S153-S118 (2002)
58. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**, 443–453 (1970)
59. Notredame, C., Higgins, D., Heringa, J.: `T-Coffee`: a novel method for multiple sequence alignments. J. Mol. Biol. **302**, 205–217 (2000)
60. Otto, W., Stadler, P.F., Prohaska, S.J.: Phylogenetic footprinting and consistent sets of local alignments. In: Giancarlo, R., Manzini, G. (eds.) CPM 2011, volume 6661 of Lecture Notes in Computer Science, pp. 118–131. Springer, Heidelberg (2011)
61. Ovcharenko, I., Loots, G.G., Giardine, B.M., Hou, M., Ma, J., Hardison, R.C., Stubbs, L., Miller, W.: Mulan: multiple-sequence local alignment and visualization for studying function and evolution. Genome Res. **15**, 184–194 (2005)
62. Overington, J., Donnelly, D., Johnson, M.S., Šali, A., Blundell, T.L.: Environment-specific amino acid substitution tables: tertiary templates and prediction of protein folds. Protein Sci. **1**, 216–226 (1992)
63. Prohaska, S., Fried, C., Flamm, C., Wagner, G., Stadler, P.F.: Surveying phylogenetic footprints in large gene clusters: applications to Hox cluster duplications. Mol. Phylogen. Evol. **31**, 581–604 (2004)
64. Prüfer, K., Stenzel, U., Hofreiter, M., Pääbo, S., Kelso, J., Green, R.E.: Computational challenges in the analysis of ancient DNA. Genome Biol. **11**, R47 (2010)
65. Rausch, T., Koren, S., Denisov, G., Weese, D., Emde, A.-K., Döring, A., Reinert, K.: A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads. Bioinformatics **25**, 1118–1124 (2009)
66. Retzlaff, N.: A two-step scoring model for computational phylolinguistics. In: de Haan, Ronald (ed.), Proceedings of the ESSLLI 2014 Student Session, pp. 196–206. TU Wien, Vienna, A (2014). www.kr.tuwien.ac.at/drm/dehaan/stus2014/proceedings.pdf. Accessed 21 Feb 2018
67. Sankoff, D.: Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM J. Appl. Math. **45**, 810–825 (1985)
68. Sankoff, D., Kruskal, J.B.: Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, Reading, MA (1983)
69. Setubal, J.C., Meidanis, J.: Introduction to Computational Molecular Biology. PWS Pub, Boston, MA (1997)
70. Shigemizu, D., Maruyama, O.: Searching for regulatory elements of alternative splicing events using phylogenetic footprinting. In: Jonassen, I., Kim, J. (eds.) 4th International Workshop on Algorithms in Bioinformatics, vol. 3240, pp. 147–158. Springer, Heidelberg (2004)
71. Sievers, F., Higgins, D.G.: Clustal Omega for making accurate alignments of many protein sequences. Protein Sci. **27**, 135–145 (2018)
72. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**, 195–197 (1981)
73. Steiner, L., Stadler, P.F., Cysouw, M.: A pipeline for computational historical linguistics. Lang. Dyn. Change **1**, 89–127 (2011)
74. Tabei, Y., Asai, K.: A local multiple alignment method for detection of non-coding RNA sequences. Bioinformatics **25**, 1498–1505 (2009)
75. Thompson, J.D., Koehl, P., Ripp, R., Poch, O.: BAliBASE 3.0: latestdevelopments of the multiple sequence alignment benchmark. Proteins **61**, 127–136 (2005)
76. Thompson, J.D., Linard, B., Lecompte, O., Poch, O.: A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. PLoS ONE **6**, e18093 (2011)
77. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. J. Comput. Biol. **1**, 337–348 (1994)
78. Waterman, M.S., Smith, T.F., Beyer, W.A.: Some biological sequence metrics. Adv. Math. **20**, 367–387 (1976)
79. Wheeler, T.J., Kececioglu, J.D.: Multiple alignment by aligning alignments. Bioinformatics **23**, i559–i568 (2007)
80. Will, S., Missal, K., Hofacker, I.L., Stadler, P.F., Backofen, R.: Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. PLoS Comput. Biol. **3**, e65 (2007)
81. Will, S., Siebauer, M.F., Heyne, S., Engelhardt, J., Stadler, P.F., Reiche, K., Backofen, R.: `LocARNAscan`: Incorporating thermodynamic stability in sequence and structure-based RNA homology search. Alg. Mol. Biol. **8**, 14 (2013)

82. Yamada, S., Gotoh, O., Yamana, H.: Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. BMC Bioinform. **7**, 524 (2006)
83. Yi-Kuo, Y., Hwa, T.: Statistical significance of probabilistic sequence alignment and related local hidden markov models. J. Comput. Biol. **8**, 249–282 (2001)
84. Zhang, Z., Gerstein, M.: Of mice and men: phylogenetic footprinting aids the discovery of regulatory elements. J. Biol. **2**, 11 (2003)
85. zu Siederdissen, C.H., Hofacker, I.L., Stadler, P.F.: Product grammars for alignment and folding. IEEE/ACM Trans. Comput. Biol. Bioinform. **12**, 507–519 (2015)
86. zu Siederdissen, C.H., Prohaska, S.J., Stadler, P.F.: Algebraic dynamic programming over general data structures. BMC Bioinform. **16**, 19:S2 (2015)