Algorithms for Normalized Multiple Sequence Alignments

Eloi Araujo[1]
Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil

Luiz C. Rozante
Centro de Matemática Computação e Cognição, Universidade Federal do ABC, Santo André, MS, Brasil

Diego P. Rubert
Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil

Fábio V. Martinez
Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil

Abstract

Sequence alignment supports numerous tasks in bioinformatics, natural language processing, pattern recognition, social sciences, and other fields. While the alignment of two sequences may be performed swiftly in many applications, the simultaneous alignment of multiple sequences proved to be naturally more intricate. Although most multiple sequence alignment (MSA) formulations are NP-hard, several approaches have been developed, as they can outperform pairwise alignment methods or are necessary for some applications. Taking into account not only similarities but also the lengths of the compared sequences (i.e. normalization) can provide better alignment results than both unnormalized or post-normalized approaches. While some normalized methods have been developed for pairwise sequence alignment, none have been proposed for MSA. This work is a first effort towards the development of normalized methods for MSA. We discuss multiple aspects of normalized multiple sequence alignment (NMSA). We define three new criteria for computing normalized scores when aligning multiple sequences, showing the NP-hardness and exact algorithms for solving the NMSA using those criteria. In addition, we provide approximation algorithms for MSA and NMSA for some classes of scoring matrices.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Approximation algorithms analysis; Applied computing → Molecular sequence analysis

Keywords and phrases Multiple sequence alignment, Normalized multiple sequence alignment, Algorithms and complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.40

Related Version *Full Version*: `https://arxiv.org/abs/2107.01607` [4]

Acknowledgements We thank the three anonymous reviewers for their valuable comments and suggestions on our manuscript.

1 Introduction

Sequence alignment lies at the foundation of bioinformatics. Several procedures rely on alignment methods for a range of distinct purposes, such as detection of sequence homology, secondary structure prediction, phylogenetic analysis, identification of conserved motifs or genome assembly. On the other hand, alignment techniques have also been reshaped and found applications in other fields, such as natural language processing, pattern recognition, or social sciences [1, 3, 8, 18].

[1] Corresponding author

Given its range of applications in bioinformatics, extensive efforts have been made to improve existing or developing novel methods for sequence alignment. The simpler ones compare a pair of sequences in polynomial time on their lengths, usually trying to find editing operations (insertions, deletions, and substitutions of symbols) that transform one sequence into another while maximizing or minimizing some objective function called edit distance [16]. This concept can naturally be generalized to align multiple sequences [28], adding another new layer of algorithmic complexity, though. In this case, most multiple sequence alignment (MSA) formulations lead to NP-hard problems [13]. Nevertheless, a variety of methods suitable for aligning multiple sequences have been developed, as they can outperform pairwise alignment methods on tasks such as phylogenetic inference [21], secondary structure prediction [12] or identification of conserved regions [24].

In order to overcome the cost of exact solutions, a number of MSA heuristics have been developed in recent years, most of them using the so-called progressive or iterative methods [17, 23, 25, 27]. Experimental data suggest that the robustness and accuracy of heuristics can still be improved, however [28].

Most approaches for pairwise sequence alignment define edit distances as absolute values, lacking some normalization that would result in edit distances relative to the lengths of the sequences. However, some applications may require sequence lengths to be taken into account. For instance, a difference of one symbol between sequences of length 5 is more significant than between sequences of length 1000. In addition, experiments suggest that normalized edit distances can provide better results than both unnormalized or post-normalized edit distances [18]. While normalized edit distances have been developed for pairwise sequence alignment [6, 18], none have been proposed for MSA to the best of our knowledge.

In this work, we propose exact and approximation algorithms for normalized MSA (NMSA). This is a first step towards the development of methods that take into account the lengths of sequences for computing edit distances when multiple sequences are compared.

The remainder of this paper is organized as follows. Section 2 introduces concepts related to sequence alignment and presents normalized scores for NMSA, followed by the complexity analysis of NMSA using those scores in Section 3. Next, Sections 4 and 5 describe exact and approximation algorithms, respectively. Section 6 closes the paper with the conclusion and prospects for future work.

2 Preliminaries

An *alphabet* Σ is a finite non-empty set of *symbols*. A finite sequence s with n symbols in Σ is seen as $s(1)\cdots s(n)$. We say that the *length* of s, denoted by $|s|$, is n. The (sub)sequence $s(p)\cdots s(q)$ of s, with $1 \le p \le q \le n$, is denoted by $s(p\,{:}\,q)$. If $p > q$, $s(p\,{:}\,q)$ is the *empty* sequence, whose length is zero, and it is denoted by ε. We denote the sequence resulting from the concatenation of sequences s and t by st. A sequence of n symbols $\mathtt{a}$ is denoted by $\mathtt{a}^n$. A k-tuple S over Σ^* is called a *k-sequence* and we write $s_1, \dots, s_k$ to refer to S, where s_i is the i-th sequence in S. Let $\Sigma_{\mathtt{-}} := \Sigma \cup \{\mathtt{-}\}$, where $\mathtt{-} \notin \Sigma$ and the symbol $\mathtt{-}$ is called a *space*. Let $S = s_1, \dots, s_k$ be a k-sequence. An *alignment* of S is a k-tuple $A = [s'_1, \dots, s'_k]$ over $\Sigma_{\mathtt{-}}^*$, where (a) each sequence s'_h is obtained by inserting spaces in s_h; (b) $|s'_h| = |s'_i|$ for each pair h, i, with $1 \le h, i \le k$; and (c) there is no j in $\{1, \dots, k\}$ such that $s'_1(j) = \ldots = s'_k(j) = \mathtt{-}$. Notice that k-tuples over $\Sigma_{\mathtt{-}}^*$ are written enclosed by square brackets "[ ]". The sequence $[s'_1(j), \dots, s'_k(j)]$ is the *column* j of the aligment $[s'_1, \dots, s'_k]$. We denote the column j of the alignment A by $A(j)$ and by $A[j_1\,{:}\,j_2]$ the columns $j_1, j_1 + 1, \dots, j_2$ of A. We say that the pair $[s'_h(j), s'_i(j)]$ *aligns* in A or, simply, that $s'_h(j)$ and $s'_i(j)$ are *aligned* in A, and $|A| = |s'_i|$ is the *length* of the alignment A. It is easy to check that $\max_i\{|s_i|\} \le |A| \le \sum_i |s_i|$.

An alignment can be used to represent *editing operations* of *insertions*, *deletions* and *substitutions* of symbols in sequences, where the symbol `-` represents insertions or deletions. An alignment can also be represented in the matrix format. Thus, alignments [`aaa-`, `ab--`, `-cac`] and [`-aaa-`, `ab---`, `-ca-c`] of `aaa`, `ab`, `cac` can be represented respectively as

$$\begin{bmatrix} \texttt{a} & \texttt{a} & \texttt{a} & \texttt{-} \\ \texttt{a} & \texttt{b} & \texttt{-} & \texttt{-} \\ \texttt{-} & \texttt{c} & \texttt{a} & \texttt{c} \end{bmatrix} \text{ and } \begin{bmatrix} \texttt{-} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{-} \\ \texttt{a} & \texttt{b} & \texttt{-} & \texttt{-} & \texttt{-} \\ \texttt{-} & \texttt{c} & \texttt{a} & \texttt{-} & \texttt{c} \end{bmatrix}.$$

Let $I = \{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}$ be a set of indices such that $i_1 < \cdots < i_m$ and let $A = [s'_1, \dots, s'_k]$ be an alignment of $S = s_1, \dots, s_k$. We write S_I to denote the m-tuple $s_{i_1}, \dots, s_{i_m}$. The alignment of S_I *induced* by A is the alignment A_I obtained from the alignment A, considering only the corresponding sequences in S_I and, from the resulting structure, removing columns where all symbols are `-`. In the following example, $A =$ [`aaa-`, `ab--`, `-cac`] is an alignment of `aaa`, `ab`, `cac` and

$$\begin{bmatrix} \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{b} & \texttt{-} \end{bmatrix}$$

is an alignment of `aaa`, `ab` induced by A. We denote by $\mathcal{A}_S$ the set of all alignments of S.

For a problem P, we call $\mathbb{I}_\mathrm{P}$ the set of instances of P. If P is a decision problem, then $\mathrm{P}(I) \in \{\texttt{Yes}, \texttt{No}\}$ is the image of an instance I. If P is an optimization (minimization) problem, there is a set $\mathrm{Sol}(I)$ for each instance I, a function v defining a non-negative rational number for each $X \in \mathrm{Sol}(I)$, and a function $\mathrm{opt}_v(I) = \min_{X \in \mathrm{Sol}(I)}\{v(X)\}$. We use opt instead of opt_v if v is obvious. Let $\mathbf{A}(I)$ be a solution computed by an algorithm $\mathbf{A}$ with input I, and $\mathbf{A}(I) \geq \mathrm{opt}(I)$. We say that $\mathbf{A}$ is an α*-approximation* for P if $\mathbf{A}(I) \leq \alpha\,\mathrm{opt}(I)$, for each $I \in \mathbb{I}_\mathrm{P}$, with $\alpha \geq 1$. We say that α is an *approximation factor* for P.

The *alignment problem* is a collection of decision and optimization problems whose instances are finite subsets of Σ^* and $\mathrm{Sol}(S) = \mathcal{A}_S$, for each instance S. Function v, used for scoring alignments, is called *criterion* for P and we call $v[A]$ the *cost of the aligment* A. The v*-optimal alignment* A of S is such that $v[A] = \mathrm{opt}(S)$. Thus, we state the following general optimization problems using the criterion v:

▶ Problem 1 (Alignment with criterion v). Given a k-sequence S, with $k \in \mathbb{N}$, find a v-optimal alignment of S.

We also need the decision version of the alignment problem with criterion v, where we are given a k-sequence S and a number $d \in \mathbb{Q}_\geq$, and we want to decide whether there exists an alignment A of S such that $v[A] \leq d$.

Usually the cost of an alignment v is defined from a scoring matrix. A *scoring matrix* γ is a rational matrix such that the elements in $\Sigma_\texttt{-}$ are indices of its rows and columns. For $\texttt{a}, \texttt{b} \in \Sigma_\texttt{-}$ and a scoring matrix γ, we denote by $\gamma_{\texttt{a}\to\texttt{b}}$ the entry of γ in line `a` and column `b`. The value $\gamma_{\texttt{a}\to\texttt{b}}$ defines the score for a substitution if $\texttt{a}, \texttt{b} \in \Sigma$, for an insertion if $\texttt{a} = \texttt{-}$, and for a deletion if $\texttt{b} = \texttt{-}$. The entry $\gamma_{\texttt{-}\to\texttt{-}}$ is not defined.

Given a scoring function v_γ for alignments that depends on a scoring matrix γ, we say that two scoring matrices γ and ρ are *equivalent* considering v when $v_\gamma[A] \leq v_\gamma[B]$ if and only if $v_\rho[A] \leq v_\rho[B]$ for any pair of alignments A, B of sequences s, t. If ρ is a matrix obtained from γ by multiplying each entry of γ by a constant $c > 0$, then $v\mathrm{A}_\rho[A] = c\,v\mathrm{A}_\gamma[A]$ and $v\mathrm{N}_\rho[A] = c\,v\mathrm{N}_\gamma[A]$, which implies that γ and ρ are equivalent. As a consequence, when the scoring function is $v\mathrm{A}_\gamma$ or $v\mathrm{N}_\gamma$ and it is convenient, we can suppose that all entries of γ are integers instead of rationals, according to the definition.

A k*-vector* $\vec{\jmath} = [j_1, \dots, j_k]$ is a k-tuple, where $j_i \in \mathbb{N} = \{0, 1, 2, \dots\}$. We say that j_i is the i-th element of $\vec{\jmath}$. The k-vector $\vec{0}$ is such that all its elements are zero. If $\vec{\jmath}$ and $\vec{h}$ are k-vectors, we write $\vec{\jmath} \leq \vec{h}$ if $j_i \leq h_i$ for each i; and $\vec{\jmath} < \vec{h}$ if $\vec{\jmath} \leq \vec{h}$ and $\vec{\jmath} \neq \vec{h}$. A sequence of k-vectors $\vec{\jmath}_1, \vec{\jmath}_2, \dots$ is in *lexicographical order* if $\vec{\jmath}_i \leq \vec{\jmath}_{i+1}$ for each i.

Consider $S = s_1, \dots, s_k$ a k-sequence with $n_i = |s_i|$ for each i and $\vec{n} = [n_1, \dots, n_k]$. Let $V_{\vec{n}} = \{\vec{\jmath} : \vec{\jmath} \le \vec{n}\}$ be the set of all k-vectors $\vec{\jmath}$ such that $\vec{\jmath} \le \vec{n}$. For example, if $k = 3$ and $\vec{n} = [1, 2, 1]$, then $V_{\vec{n}} = \{[x, y, z] : x, y, z \in \mathbb{N}, x \le 1, y \le 2, z \le 1\}$. Notice that if $n_i = n$ for all i, then $|V_S| = (n + 1)^k$. Define $S(\vec{\jmath}) = s_1(j_1), \dots, s_k(j_k)$ a column $\vec{\jmath}$ in S and we say that $S(1{:}\vec{\jmath}) = s_1(1{:}j_1), \dots, s_k(1{:}j_k)$ is the *prefix of S ending in $\vec{\jmath}$*. Thus, $S = S(1{:}\vec{n})$. Besides that, if A is an alignment and $\vec{v} = [j, j, \dots, j]$, then $A[\vec{v}] = A(j)$.

Denote by $\mathcal{B}^k$ the set of k-bit vectors $[b_1, \dots, b_k]$, where $b_i \in \{0, 1\}$ for each i. Now, for $\vec{b} \le \vec{\jmath}$, define $\vec{b} \cdot S(\vec{\jmath}) = [x_1, \dots, x_k] \in \Sigma_{\text{-}}^k$ such that $x_i = s_i(j_i)$ if $b_i = 1$ and $x_i =$ - otherwise. Therefore, given an alignment A of $S(1{:}\vec{\jmath})$, there exists $\vec{b} \in \mathcal{B}^k$, with $\vec{b} \le \vec{\jmath}$, such that $A(|A|) = \vec{b} \cdot S(\vec{n})$. In other words, if $\vec{n} = [n_1, \dots, n_k]$ and $\vec{b} = [b_1, \dots, b_k]$, we have $b_i = 1$ if and only if $s_i(n_i)$ is in the i-th row of the last column of A. We also define the operation $\vec{\jmath} - \vec{b} = [j_1 - b_1, \dots, j_k - b_k]$. Notice that $|\mathcal{B}_k| = 2^k$.

Consider a scoring matrix γ. Let $s, t \in \Sigma^*$, with $n = |s|, m = |t|$. A simple criterion for scoring alignments using the function $v\mathrm{A}_\gamma$ follows. For an alignment $[s', t']$ of s, t we define

$$v\mathrm{A}_\gamma[s', t'] = \textstyle\sum_{j=1}^{|[s',t']|} \gamma_{s'(j) \to t'(j)} \,.$$

We say that $v\mathrm{A}_\gamma[s', t']$ is a *$v\mathrm{A}_\gamma$-score* of s, t. The optimal function for this criterion is denoted by optA_γ and an alignment A of s, t is called an A-*optimal alignment* of s, t if $v\mathrm{A}_\gamma[A] = \mathrm{optA}_\gamma(s, t)$.

Now, suppose that $n \ge m$. Needleman and Wunch [20] proposed an $O(n^2)$-time algorithm for computing $\mathrm{optA}_\gamma(s, t)$. If optA_γ is a Levenstein distance, Masek and Paterson [19] presented an $O(n^2/\log n)$-time algorithm using the "*Four Russian's Method*". Crochemore, Landau and Ziv-Ukelson [11] extended this result for real arrays, describing an $O(n^2/\log n)$-time algorithm. Indeed, there is no algorithm to determine $\mathrm{optA}_\gamma(s, t)$ in $O(n^{2-\delta})$-time for any $\delta > 0$, unless SETH is false [7]. Andoni, Krauthgamer and Onak [2] described a nearly linear time algorithm approximating the edit distance within an approximation factor poly$(\log n)$. Later, Chakraborty et al. [10] presented an $O(n^{2-2/7})$-time α-approximation for this problem, where α is constant.

Marzal and Vidal [18] defined another criterion for scoring alignments of two sequences called *$v\mathrm{N}_\gamma$-score*, which is a normalization of $v\mathrm{A}_\gamma$-score, as follows:

$$v\mathrm{N}_\gamma[A] = \begin{cases} 0\,, & \text{if } |A| = 0\,, \\ v\mathrm{A}_\gamma[A]/\,|A|\,, & \text{otherwise}\,. \end{cases}$$

The optimal function for this criterion is $\mathrm{optN}_\gamma(s, t) = \min_{A \in \mathcal{A}_{s,t}} \left\{v\mathrm{N}_\gamma[A]\right\}$, and an N-*optimal* alignment A of s, t is such that $v\mathrm{N}_\gamma[A] = \mathrm{optN}_\gamma(s, t)$.

A naive dynamic programming algorithm was proposed by Marzal and Vidal [18] to obtain an N-optimal alignment of two sequences in $O(n^3)$-time. Using fractional programming, Vidal, Marzal and Aibar [26] presented an algorithm with running time $O(n^3)$, requiring $O(n^2)$-time in practice, similarly to the classical (unnormalized) edit distance algorithm. Further, Arslan and Egecioglu [6] described an $O(n^2 \log n)$-time algorithm to solve this problem.

Let A be an A-optimal alignment of maximum length of 2-sequence $S = s, t$, with $|s| = n$ and $|t| = m$. Considering $\vec{n} = [n, m]$ and $\vec{b}$ a bit vector such that $A(|A|) = \vec{b} \cdot S(\vec{n})$, the length of a maximum length A-optimal alignment of $S(1{:}\vec{n} - \vec{b})$ must be $|A| - 1$. Thus, the maximum length $L(n, m)$ can be found by a dynamic programming formula as following:

$$L(0, 0) = 0\,, \quad L(0, j) = j\,, \quad L(i, 0) = i\,,$$

$$L(i,j) = \max \left\{ \begin{array}{ll} L(i-1,j)\,, & \text{if } d(i,j) = d(i-1,j) + \gamma_{s(i)\to\texttt{-}} \\ L(i,j-1)\,, & \text{if } d(i,j) = d(i,j-1) + \gamma_{\texttt{-}\to t(j)} \\ L(i-1,j-1)\,, & \text{if } d(i,j) = d(i-1,j-1) + \gamma_{s(i)\to s(j)} \end{array} \right\} + 1\,,\ i,j > 0\,,$$

where $d(i,j) = \mathrm{optA}_\gamma(s(1{:}i), t(1{:}j))$. Therefore, the maximum length A-optimal alignment of s, t can be obtained in $O(nm)$-time. The following theorem shows that we can propose a simple approximation algorithm to find an A-optimal alignment of maximum length.

▶ **Theorem 2.1.** *Let s, t be sequences of lengths n, m, respectively, and let $L(n,m)$ be the maximum length of an A-optimal alignment of s, t. Then,*

$$\mathrm{optA}_\gamma(s,t)/L(n,m) \le 2\,\mathrm{optN}_\gamma(s,t)\,,$$

and it can be computed in $O(n^2)$-time if $n = m$. Moreover, this ratio is tight, i.e., for any positive rational ε, there exists a scoring matrix γ, sequences s, t and an A-optimal alignment of s, t with maximum length A such that $\mathrm{optA}_\gamma(s,t)/|A| = v\mathrm{A}_\gamma[A]/|A| = (2-\varepsilon)\,\mathrm{optN}_\gamma(s,t)$.

Proof. Let A be an A-optimal alignment with maximum length computed by the heuristic above in $O(nm)$-time and space. Let B be an N-optimal alignment. Thus, $v\mathrm{A}_\gamma[A] \le v\mathrm{A}_\gamma[B]$. Moreover, $|B| \le n + m \le 2\max\{n,m\} \le 2\,|A|$, that is, $|A| \ge |B|/2$. Therefore, $v\mathrm{N}_\gamma[A] = \frac{v\mathrm{A}_\gamma[A]}{|A|} \le \frac{v\mathrm{A}_\gamma[B]}{|A|} \le \frac{v\mathrm{A}_\gamma[B]}{|B|/2} = 2\,\mathrm{optN}_\gamma(s,t)\,.$

We present now two sequences and a scoring matrix γ such that the solution given by the heuristic is at least $2-\varepsilon$ times the $v\mathrm{N}_\gamma$-score of an N-optimal alignment, for any ε in $\mathbb{Q}_>$. Let $\Sigma = \{\texttt{a},\texttt{b}\}$, γ be a scoring matrix such that $\gamma_{\texttt{a}\to\texttt{-}} = \gamma_{\texttt{b}\to\texttt{-}} = 1/\varepsilon$ and $\gamma_{\texttt{a}\to\texttt{b}} = 2/\varepsilon - 1$ and $\texttt{a}^n, \texttt{b}^n \in \Sigma^*$, with n in $\mathbb{N}^*$. Observe that the $v\mathrm{A}_\gamma$-score of any alignment of $(\texttt{a}^n, \texttt{b}^n)$, where $[\texttt{a},\texttt{b}]$ is aligned in k columns, is $2n/\varepsilon - k$. Thus, $\mathrm{optA}_\gamma(\texttt{a}^n,\texttt{b}^n) = \min_{0\le k\le n}\{2n/\varepsilon - k\} = 2n/\varepsilon - n = (2/\varepsilon - 1)\,n$ which implies $[\texttt{a}^n, \texttt{b}^n]$ is the A-optimal alignment with maximum length. Since $\mathrm{optN}_\gamma(\texttt{a}^n,\texttt{b}^n) \le v\mathrm{N}_\gamma([\texttt{a}^n\texttt{-}^n, \texttt{-}^n\texttt{b}^n]) = 1/\varepsilon$, it follows

$$\frac{\mathrm{optA}_\gamma(s,t)}{|[\texttt{a}^n,\texttt{b}^n]|} = \frac{v\mathrm{A}_\gamma(s,t)}{|[\texttt{a}^n,\texttt{b}^n]|} = \frac{(2/\varepsilon - 1)\,n}{n} = (2-\varepsilon)/\varepsilon \ge (2-\varepsilon)\,\mathrm{optN}_\gamma(\texttt{a}^n,\texttt{b}^n)\,. \qquad ◀$$

We define now classes of scoring matrices. The usual class of scoring matrices $\mathbb{M}^{\mathrm{C}}$ has the following properties: for all symbols $\texttt{a},\texttt{b},\texttt{c}, \in \Sigma_\texttt{-}$, we have (a) $\gamma_{\texttt{a}\to\texttt{b}} > 0$ if $\texttt{a} \ne \texttt{b}$, and $\gamma_{\texttt{a}\to\texttt{b}} = 0$ if $\texttt{a} = \texttt{b}$; (b) $\gamma_{\texttt{a}\to\texttt{b}} = \gamma_{\texttt{b}\to\texttt{a}}$; and (c) $\gamma_{\texttt{a}\to\texttt{c}} \le \gamma_{\texttt{a}\to\texttt{b}} + \gamma_{\texttt{b}\to\texttt{c}}$. The class $\mathbb{M}^{\mathrm{A}}$ of scoring matrices is such that, for all symbols $\texttt{a},\texttt{b},\texttt{c} \in \Sigma$, we have (a) $\gamma_{\texttt{a}\to\texttt{-}} = \gamma_{\texttt{-}\to\texttt{a}} > 0$; (b) $\gamma_{\texttt{a}\to\texttt{b}} > 0$ if $\texttt{a} \ne \texttt{b}$, and $\gamma_{\texttt{a}\to\texttt{b}} = 0$ if $\texttt{a} = \texttt{b}$; (c) if $\gamma_{\texttt{a}\to\texttt{b}} < \gamma_{\texttt{a}\to\texttt{-}} + \gamma_{\texttt{-}\to\texttt{b}}$, then $\gamma_{\texttt{a}\to\texttt{b}} = \gamma_{\texttt{b}\to\texttt{a}}$; (d) $\gamma_{\texttt{a}\to\texttt{-}} \le \gamma_{\texttt{a}\to\texttt{b}} + \gamma_{\texttt{b}\to\texttt{-}}$; and (e) $\min\{\gamma_{\texttt{a}\to\texttt{c}}, \gamma_{\texttt{a}\to\texttt{-}} + \gamma_{\texttt{-}\to\texttt{c}}\} \le \gamma_{\texttt{a}\to\texttt{b}} + \gamma_{\texttt{b}\to\texttt{c}}$. Moreover, the class $\mathbb{M}^{\mathrm{N}}$ is such that (a) $\mathbb{M}^{\mathrm{N}} \subseteq \mathbb{M}^{\mathrm{A}}$ and (b) $\gamma_{\texttt{a}\to\texttt{-}} \le 2\,\gamma_{\texttt{b}\to\texttt{-}}$ for each $\texttt{a},\texttt{b} \in \Sigma$.

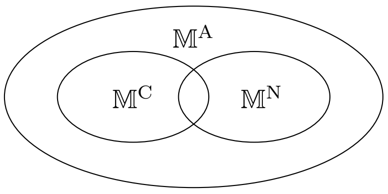
For a set S, we say that the (distance) function $f : S \times S \to \mathbb{R}$ is a *metric* on S if, for all $s,t,u \in S$, the distance f satisfies: (1) $f(s,s) = 0$ (*reflexive*); (2) $f(s,t) > 0$ if $s \ne t$ (*positive*); (3) $f(s,t) = f(t,s)$ (*symmetry*); and (4) $f(s,u) \le f(s,t) + f(t,u)$ (*triangle inequality*).

If a given criterion v depends on a scoring matrix γ and it is a metric on Σ^*, we say that the scoring matrix γ *induces* a v-distance on Σ^*. Sellers [22] showed that matrices in $\mathbb{M}^{\mathrm{C}}$ induce an optA_γ-distance on Σ^* and Araujo and Soares [5] showed that $\gamma \in \mathbb{M}^{\mathrm{A}}$ if and only if γ induces an optA_γ-distance on Σ^*. Furthermore, $\gamma \in \mathbb{M}^{\mathrm{N}}$ if and only if γ induces an optN_γ-distance on Σ^*. Figure 1 shows the relationship between these classes.

2.1 $v\mathrm{SP}_\gamma$-score for k sequences

Consider a scoring matrix γ. Let $S = s_1, \ldots, s_k$ be a k-sequence and $A = [s'_1, \ldots, s'_k]$ be an alignment of S. The criterion $v\mathrm{SP}_\gamma$, also called *SP-score*, for scoring the alignment A is

$$v\mathrm{SP}_\gamma[A] = \textstyle\sum_{h=1}^{k-1} \sum_{i=h+1}^{k} v\mathrm{A}_\gamma[A_{\{h,i\}}]\,. \qquad (1)$$



$\mathbb{M}^{\mathrm{A}}$
$\mathbb{M}^{\mathrm{C}}$
$\mathbb{M}^{\mathrm{N}}$

Figure 1 Relationship between scoring matrices. Araujo and Soares [5] showed that $\mathbb{M}^{\mathrm{C}} \subseteq \mathbb{M}^{\mathrm{A}}$, $\mathbb{M}^{\mathrm{N}} \subseteq \mathbb{M}^{\mathrm{A}}$, $\mathbb{M}^{\mathrm{C}} \not\subseteq \mathbb{M}^{\mathrm{N}}$ and $\mathbb{M}^{\mathrm{N}} \not\subseteq \mathbb{M}^{\mathrm{C}}$. Moreover, the scoring matrix γ such that $\gamma_{\mathtt{a}\to\mathtt{a}} = 0$ for each $\mathtt{a}$ and $\gamma_{\mathtt{a}\to\mathtt{b}} = 1$ for each $\mathtt{a} \neq \mathtt{b}$ is in $\mathbb{M}^{\mathrm{C}} \cap \mathbb{M}^{\mathrm{N}}$, which implies that $\mathbb{M}^{\mathrm{C}} \cap \mathbb{M}^{\mathrm{N}} \neq \emptyset$.

We define optSP_γ as the optimal function for the criterion $v\mathrm{SP}_\gamma$. An alignment A of S such that $v\mathrm{SP}_\gamma[A] = \mathrm{optSP}_\gamma(S)$ is called $v\mathrm{SP}_\gamma$*-optimal alignment*. Regardless its decision or optimization version, we call this the *multiple sequence alignment problem* (**MSA**). Formally,

▶ Problem 2 (Multiple sequence alignment). Let γ be a fixed scoring matrix. Given a k-sequence S, find a $v\mathrm{SP}_\gamma$-optimal alignment of S.

In order to compute optSP_γ, we extend the definition of $v\mathrm{SP}_\gamma$ considering a column of an alignment $A = [s'_1, \ldots, s'_k]$ as its parameter. Thus, $v\mathrm{SP}_\gamma(A(j)) = \sum_{i<h} \gamma_{s'_i(j)\to s'_h(j)}$ assuming that $\gamma_{\mathtt{-}\to\mathtt{-}} = 0$ and

$$\mathrm{optSP}_\gamma(S) = \mathrm{optSP}_\gamma(S(1\,{:}\,\vec{n})) = \min_{\vec{b}\in\mathcal{B}^k, \vec{b}\leq\vec{\jmath}}\{\mathrm{optSP}_\gamma(S(1\,{:}\,\vec{n}-\vec{b})) + v\mathrm{SP}_\gamma[\vec{b}\cdot S(\vec{n})]\}. \quad (2)$$

Recurrence (2) can be computed using a dynamic programming algorithm, obtaining $D(\vec{\jmath}) = \mathrm{optSP}_\gamma(S(1\,{:}\,\vec{\jmath}))$ for all $\vec{\jmath} \leq \vec{n}$. This task can be performed by generating all indexes of D in lexicographical order, starting with $D(\vec{0}) = 0$, as presented in Algorithm 1.

Algorithm 1 $v\mathrm{SP}_\gamma$-optimal alignment of S.

```
Input: $S = s_1, \ldots, s_k \in (\Sigma^*)^k$
Output: $\mathrm{optSP}_\gamma(S)$
1: $D(\vec{0}) \leftarrow 0$
2: for each $\vec{\jmath} \leq \vec{n}$ in lexicographical order do
3:     $D(\vec{\jmath}) \leftarrow \min_{\vec{b}\in\mathcal{B}^k,\, \vec{b}\leq\vec{\jmath}}\{D(\vec{\jmath}-\vec{b}) + v\mathrm{SP}_\gamma[\vec{b}\cdot S(\vec{\jmath})]\}$
4: return $D(\vec{n})$
```

Suppose that $|s_i| = n$ for each i. Notice that the space to store the matrix D is $\Theta((n+1)^k)$ and thus Algorithm 1 uses $\Theta((n+1)^k)$-space. Besides that, Algorithm 1 checks, in the worst case, $\Theta(2^k)$ entries for computing all entries in the matrix D and each computation spends $\Theta(k^2)$-time. Therefore, its running time is $O(2^k k^2 (n+1)^k)$. Observe that when the distances between sequences are small, not all entries in D need to be computed, such as in the Carrillo and Lipman's algorithm [9].

2.2 V^i_γ-score for k sequences

In this section we define a new criteria to normalize the $v\mathrm{SP}_\gamma$-score of a multiple alignment. The symbol - aligned to the same symbol - does not contribute to the definition of scoring, and thus this entry is not defined. However, as all the criteria are additive, it is convenient to consider $\gamma_{\mathtt{-}\to\mathtt{-}} = 0$. The new criteria for aligning sequences takes into account the length of the alignments according to the following:

$$\mathrm{V}^1_\gamma[A] = \begin{cases} 0\,, & \text{if } |A| = 0\,,\\ v\mathrm{SP}_\gamma[A]/\,|A|\,, & \text{otherwise}\,, \end{cases} \quad (3)$$

$$\mathrm{V}^2_\gamma[A] = \textstyle\sum_{h=1}^{k-1}\sum_{i=h+1}^{k} v\mathrm{N}_\gamma[A_{\{h,i\}}]\,, \tag{4}$$

$$\mathrm{V}^3_\gamma[A] = \begin{cases} 0\,, & \text{if } |A| = 0\,, \\ v\mathrm{SP}_\gamma[A]/\big(\sum_{h=1}^{k-1}\sum_{i=h+1}^{k}|A_{\{h,i\}}|\big)\,, & \text{otherwise}\,. \end{cases} \tag{5}$$

We define optNSP^z_γ as the optimal function for the criterion V^z_γ. An alignment A of S such that $\mathrm{V}^z_\gamma[A] = \mathrm{optNSP}^z_\gamma(S)$ is called V^z_γ*-optimal alignment*. Moreover, regardless its decision or optimization version, we establish the *criterion* V^z_γ *for the normalized multiple sequence alignment problem* (**NMSA**-z), for $z = 1, 2, 3$. Formally,

▶ Problem 3 (Normalized multiple sequence alignment with score V^z_γ). Let γ be a fixed scoring matrix and $z \in \{1, 2, 3\}$. Given a k-sequence S, find a V^z_γ-optimal alignment of S.

3 Complexity

We study now the complexity of the multiple sequence alignment problem for each new criterion defined in Section 2. We consider the decision version of the computational problems and we prove **NMSA**-z is NP-complete for each z when the following additional restrictions for the scoring matrix γ hold: $\gamma_{\mathtt{a}\to\mathtt{b}} = \gamma_{\mathtt{b}\to\mathtt{a}}$ and $\gamma_{\mathtt{a}\to\mathtt{b}} = 0$ if and only if $\mathtt{a} = \mathtt{b}$ for each pair $\mathtt{a}, \mathtt{b} \in \Sigma_-$. Elias [13] shows that, even considering such restrictions, **MSA** is NP-complete. We show a polynomial time reduction from **MSA** to **NMSA**-z.

Consider a fixed alphabet Σ and a scoring matrix γ with the restrictions above. Let $\sigma \notin \Sigma_-$ be a new symbol and $\Sigma^\sigma = \Sigma \cup \{\sigma\}$. Let G be a fixed (constant) positive integer such that each entry in γ is at most G. We define a scoring matrix γ^σ such that $\gamma^\sigma_{\mathtt{a}\to\mathtt{b}} = \gamma_{\mathtt{a}\to\mathtt{b}}, \gamma^\sigma_{\mathtt{a}\to\sigma} = \gamma^\sigma_{\sigma\to\mathtt{a}} = G$ and $\gamma^\sigma_{\sigma\to\sigma} = 0$, for each pair $\mathtt{a}, \mathtt{b} \in \Sigma_-$.

For an instance $(S = s_1, \ldots, s_k, C)$ of **MSA**, let $S^L = s_1\sigma^L, \ldots, s_k\sigma^L$, where $L = Nk^2MG$, $M = \max_i\{|s_i|\}$ and $N = \binom{k}{2}M$. Define l as the *tail length* of an alignment A if $A[i, j] = \sigma$ for each $i = 1, \ldots, k$ and $j = l + 1, l + 2, \ldots, |A|$, i.e., every symbol in the last l columns of A is σ. We say that an alignment of S^L is *canonical* if its tail length is L. If $A = [s''_1, \ldots, s''_k]$ is an alignment of S, we denote by A^L the canonical alignment $[s''_1\sigma^L, \ldots, s''_k\sigma^L]$ of S^L. The two following results are useful to prove Theorem 3.3, which is the main result of this section. The proofs of these two lemmas can be found in [4].

▶ **Lemma 3.1.** *There exists a canonical alignment of S^L which is $\mathrm{V}^z_{\gamma^\sigma}$-optimal for each z.*

▶ **Lemma 3.2.** *If $C \geq k^2MG$, then* **MSA**(S, C) = **NMSA**-$z(S^L, C^z)$ = Yes, *for each z.*

▶ **Theorem 3.3.** **NMSA**-z *is NP-complete for each z.*

Proof. Given a k-sequence S over Σ^*, an alignment A of S and a integer C, it is easy to check in polynomial time on the length of A that $\mathrm{V}^z_\gamma[A] \leq C$, for $z \in \{1, 2, 3\}$, and then **NMSA**-z is in NP.

Consider now $C^1 := C^2 := C/L, C^3 := C/\big(\binom{k}{2}L\big)$ and $L := Nk^2MG$ and then we prove that **MSA**(S, C) = Yes if and only if **NMSA**-$z(S^L, C^z)$ = Yes for each $z \in \{1, 2, 3\}$. If $C \geq k^2MG$, the Lemma 3.2 holds trivially. Thus, we assume $C < k^2MG$. Suppose that **MSA**(S, C) = Yes and, hence, there exists an alignment A such that $v\mathrm{SP}_\gamma[A] \leq C$:

$$\mathrm{V}^1_{\gamma^\sigma}[A^L] = \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{|A^L|} \leq \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{L} = \frac{v\mathrm{SP}_\gamma[A]}{L} \leq \frac{C}{L} = C^1\,,$$

$$\mathrm{V}^2_{\gamma^\sigma}[A^L] = \sum_{h=1}^{k-1}\sum_{i=k+1}^{k}\frac{v\mathrm{A}_{\gamma^\sigma}[A^L_{\{h,i\}}]}{|A^L_{\{h,i\}}|} \leq \sum_{h=1}^{k-1}\sum_{i=h+1}^{k}\frac{v\mathrm{A}_{\gamma^\sigma}[A^L_{\{h,i\}}]}{L} = \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{L} = \frac{v\mathrm{SP}_\gamma[A]}{L} \leq \frac{C}{L} = C^2,$$

$$\mathrm{V}^3_{\gamma^\sigma}[A^L] = \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{\sum_{h=1}^{k-1}\sum_{i=k+1}^{k}|A^L_{\{h,i\}}|} \le \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{\sum_{h=1}^{k-1}\sum_{i=k+1}^{k} L} = \frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{\binom{k}{2}L} = \frac{v\mathrm{SP}_{\gamma}[A]}{\binom{k}{2}L} \le \frac{C}{\binom{k}{2}L} = C^3\,,$$

where the first inequality in each equation follows since either A^L or each alignment induced by A^L has length at least L and the second inequality follows since $v\mathrm{SP}_\gamma[A] \le C$. Thus, if $\mathbf{MSA}(S, C) = \mathsf{Yes}$ then $\mathbf{NMSA}\text{-}z(S^L, C^z) = \mathsf{Yes}$.

Suppose that $\mathbf{NMSA}\text{-}z(S^L, C^z) = \mathsf{Yes}$. It follows from Lemma 3.1 that, for each z, there exists a canonical alignment A^L such that $\mathrm{V}^z_{\gamma^\sigma}[A^L] \le C^z$. Thus, considering $\mathrm{V}^1_{\gamma^\sigma}[A^L] \le C^1$, we have

$$\begin{aligned} v\mathrm{SP}_\gamma[A] &= v\mathrm{SP}_{\gamma^\sigma}[A^L] = (N+L)\,\frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{N+L} \le (N+L)\,\frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{|A^L|} = (N+L)\,\mathrm{V}^1_{\gamma^\sigma}[A^L] \\ &\le (N+L)\,C^1 = (N+L)\,\frac{C}{L} = \frac{NC}{L} + C < \frac{Nk^2MG}{L} + C = 1 + C\,, \end{aligned}$$

where the first equality holds since A^L is canonical, the first inequality holds since $|A^L| \le N+L$ and the second and the third inequalities hold by hypothesis. Considering $\mathrm{V}^2_{\gamma^\sigma}[A^L] \le C^2$, we have

$$\begin{aligned} v\mathrm{SP}_\gamma[A] &= v\mathrm{SP}_{\gamma^\sigma}[A^L] = (N+L)\,\frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{N+L} = (N+L)\sum_{h=1}^{k-1}\sum_{i=h+1}^{k}\frac{v\mathrm{A}_{\gamma^\sigma}[A^L_{\{h,i\}}]}{N+L} \\ &\le (N+L)\sum_{h=1}^{k-1}\sum_{i=h+1}^{k}\frac{v\mathrm{A}_{\gamma^\sigma}[A^L_{\{h,i\}}]}{|A^L_{\{h,i\}}|} = (N+L)\,\mathrm{V}^2_{\gamma^\sigma}[A^L] \\ &\le (N+L)\,C^2 = (N+L)\,\frac{C}{L} = \frac{NC}{L} + C < \frac{Nk^2MG}{L} + C = 1 + C\,, \end{aligned}$$

where the first equality holds since A^L is canonical, the first inequality holds since, for each h, i, $|A^L_{\{h,i\}}| \le N + L$, and the second and the third inequalities hold by hypothesis. And finally, considering $\mathrm{V}^3_{\gamma^\sigma}[A^L] \le C^3$, we have

$$\begin{aligned} v\mathrm{SP}_\gamma[A] &= v\mathrm{SP}_{\gamma^\sigma}[A^L] = \big(N + \tbinom{k}{2}L\big)\,\frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{N+\binom{k}{2}L} \\ &\le \big(N + \tbinom{k}{2}L\big)\,\frac{v\mathrm{SP}_{\gamma^\sigma}[A^L]}{\sum_{h=1}^{k-1}\sum_{i=h+1}^{k}|A^L_{\{h,i\}}|} = \big(N + \tbinom{k}{2}L\big)\,\mathrm{V}^3_{\gamma^\sigma}[A^L] \le \big(N + \tbinom{k}{2}L\big)\,C^3 \\ &= \big(N + \tbinom{k}{2}L\big)\,\frac{C}{\binom{k}{2}L} = \frac{NC}{\binom{k}{2}L} + C < \frac{Nk^2MG}{\binom{k}{2}L} + C = \frac{1}{\binom{k}{2}} + C \le 1 + C\,, \end{aligned}$$

where the first equality holds since A^L is canonical, the first inequality holds since the sum of lengths of two sequences induced by a canonical alignment is at most $N + \binom{k}{2}L$ and the second and the third inequalities hold by hypothesis.

Therefore, if $\mathbf{NMSA}\text{-}z(S^L, C^z) = \mathsf{Yes}$ then $v\mathrm{SP}_\gamma[A] < 1 + C$, for any $z \in \{1, 2, 3\}$. Since the entries in the scoring matrix are integers, we have that $v\mathrm{SP}_\gamma[A]$ is an integer. And since C is an integer, it follows that $v\mathrm{SP}_\gamma[A] \le C$. ◀

4 Exact algorithms

4.1 NMSA-1

Let $S = s_1, \ldots, s_k$ be a k-sequence and $A = [s'_1, \ldots, s'_k]$ be an alignment of S. As defined in Equation (3), $\mathrm{V}^1_\gamma[A]$ takes into account the length of A, and the optimal function is given by $\mathrm{optV}^1_\gamma(S) = \min_{A \in \mathcal{A}_S} \{\mathrm{V}^1_\gamma[A]\}$. The V^1_γ-optimal alignment of S is an alignment A such that $\mathrm{V}^1_\gamma[A] = \mathrm{optV}^1_\gamma(S)$. Thus, in **NMSA**-1 we are given a k-sequence S and we want to compute $\mathrm{optV}^1_\gamma(S)$ for a fixed matrix γ. We can solve **NMSA**-1 by calculating the minimum SP-score considering every possible length of an alignment. That is, we compute the entries of a table D indexed by $V_S \times \{0, 1, \ldots, N\}$, where $N = \sum_{i=1}^{k} |s_i|$. The entry $D(\vec{v}, L)$ stores the score of an alignment of $S(\vec{v})$ of length L with lowest SP-score. Notice that $D(\vec{0}, 0) = 0$, $D(\vec{v} \neq \vec{0}, 0) = D(\vec{0}, L \neq 0) = \infty$. Therefore, the table entries can be calculated as:

$$D(\vec{v}, L) = \begin{cases} 0\,, & \text{if } \vec{v} = \vec{0}, L = 0\,, \\ \infty\,, & \text{if } \vec{v} = \vec{0}, L \neq 0 \text{ or } \vec{v} \neq \vec{0}, L = 0\,, \\ \min_{\vec{b} \in \mathcal{B}_k, \vec{b} \leq \vec{v}} \left\{ D(\vec{v} - \vec{b}, L - 1) + v\mathrm{SP}_\gamma[\vec{b} \cdot S(\vec{v})] \right\}, & \text{otherwise}\,. \end{cases}$$

Table D is computed for all possible values of $L = 0, \ldots, N$. Consequently, $\mathrm{optV}^1_\gamma(S) = \min_L \{D(\vec{n}, L)/L\}$ is returned. Algorithm 2 describes this procedure more precisely.

Algorithm 2 V^1_γ-optimal alignment of S.

```
Input: k-sequence $S = s_1, \ldots, s_k$ such that $n_i = |s_i|$
Output: $\mathrm{optV}^1_\gamma(S)$
1: $D(\vec{0}, 0) \leftarrow 0$
2: for each $L \neq 0$ do $D(\vec{0}, L) \leftarrow \infty$
3: for each $\vec{v} \neq \vec{0}$ do $D(\vec{v}, 0) \leftarrow \infty$
4: for each $\vec{0} < \vec{v} \leq \vec{n}$ in lexicographical order do
5:     for each $L \leftarrow 1, 2, \ldots, N$ do
6:         $D(\vec{v}, L) \leftarrow \min_{\vec{b} \in \mathcal{B}^k, \vec{b} \leq \vec{v}} \{D(\vec{v} - \vec{b}, L - 1) + v\mathrm{SP}_\gamma[\vec{b} \cdot S(\vec{v})]\}$
7: return $\min_L \{D(\vec{n}, L)/L\}$
```

Suppose that $n_i = |s_i| = n$ for each i. Notice that the space to store the matrix D is $\Theta(N(n+1)^k)$. The time consumption of Algorithm 2 corresponds to the time needed to fill the table D up, plus the running time of line 7. Each entry of D can be computed in $O(2^k k^2)$-time. Therefore, the algorithm spends $O(2^k k^2 \cdot N(n+1)^k)$-time to compute the entire table D, since D has $\Theta(N(n+1)^k)$ entries. Line 7 is computed in $\Theta(N)$-time. Therefore, the running time of Algorithm 2 is $O(2^k k^2 \cdot N(n+1)^k) + \Theta(N) = O(2^k k^2 \cdot N(n+1)^k)$. If $N = kn$, it follows that the total running time is $O(2^k k^3 (n+1)^{k+1})$.

4.2 NMSA-2

Let $S = s_1, \ldots, s_k$ be a k-sequence and $A = [s'_1, \ldots, s'_k]$ be an alignment of S. As defined in Equation (4), $\mathrm{V}^2_\gamma[A]$ takes into account the length of the induced alignment A, and the optimal function is given by $\mathrm{optV}^2_\gamma(S) = \min_{A \in \mathcal{A}_S} \{\mathrm{V}^2_\gamma[A]\}$. The V^2_γ-optimal alignment of S is an alignment A such that $\mathrm{V}^2_\gamma[A] = \mathrm{optV}^2_\gamma(S)$. Then, in **NMSA**-2 we are given a k-sequence S and we want to compute $\mathrm{optV}^2_\gamma(S)$ for a fixed matrix γ.

Let $\vec{L} = [L_{12}, L_{13}, \ldots, L_{1k}, L_{23}, \ldots L_{2k}, \ldots, L_{(k-1)k}]$ be a $\binom{k}{2}$-vector indexed by sets of two integers $\{h, i\}$ such that $1 \leq h < i \leq k$ and L_{hi} denotes the element of $\vec{L}$ of index $\{h, i\}$. The lengths of the induced alignments by an alignment can be represented by a vector

$\vec{L}$. Thus, if A is an alignment and $|A_{\{h,i\}}| = L_{hi}$ for each pair h, i, we say that $\vec{L}$ is the *induced length* of A. For a k-sequence $S = s_1, \dots, s_k$, where $n_i = |s_i|$ for each i, we define $\mathbb{L} = \{\vec{L} = [L_{12}, L_{13}, \dots, L_{1k}, L_{23}, \dots L_{2k}, \dots, L_{(k-1)k}] : 0 \le L_{hi} \le n_h + n_i\}$. Note that if n is the length of each sequence in S, then $|\mathbb{L}| = (2n+1)^{\binom{k}{2}}$. Let $\vec{b} = [b_1, \dots, b_k]$ be a k-bit vector. Overloading the minus operator "$-$", we define $\vec{L} - \vec{b}$ to be a $\binom{k}{2}$-vector $\vec{L'}$ which is obtained from $\vec{L}$ and from $\vec{b}$ such that, for each pair h, i, we have $L'_{hi} = L_{hi}$ if $b_h = b_i = 0$, and $L'_{hi} = L_{hi} - 1$, otherwise. Observe that if $\vec{L}$ is the induced length of an alignment A of $S(\vec{v})$ and $\vec{b}$ is a k-bit vector such that $\vec{b} \cdot S(\vec{v})$ is the last column of A, then $\vec{L'} = \vec{L} - \vec{b}$ is the induced length of the alignment $A(1 : |A| - 1)$.

Let $\vec{\gamma}$ be a vector of $\binom{k}{2}$ scoring matrices indexed by two integers $\{h, i\}$, with $1 \le h < i \le k$. We denote by $\gamma^{(hi)}$ the element of $\vec{\gamma}$ with index $\{h, i\}$. Then, we have $\vec{\gamma} = [\gamma^{(12)}, \gamma^{(13)}, \dots, \gamma^{(1k)}, \gamma^{(23)}, \dots, \gamma^{(2k)}, \dots, \gamma^{((k-1)k)}]$, and define the $\vec{\gamma}$-SP-score of A as $v\mathrm{SP}_{\vec{\gamma}}[A] = \sum_{h=1}^{k-1} \sum_{i=h+1}^{k} v\mathrm{A}_{\gamma^{(hi)}}[A_{\{h,i\}}]$. If we define the $\vec{\gamma}$-SP-score of a vector $\vec{\sigma} = [\sigma_1, \dots, \sigma_k]$ in $\Sigma_{\text{-}}$ as $v\mathrm{SP}_{\vec{\gamma}}[\vec{\sigma}] = \sum_{h=1}^{k-1} \sum_{i=h+1}^{k} \gamma^{(hi)}_{\sigma_h \to \sigma_i}$, then we can alternatively calculate the $\vec{\gamma}$-SP-score of the alignment A as $v\mathrm{SP}_{\vec{\gamma}}[A] = \sum_j v\mathrm{SP}_{\vec{\gamma}}[A(j)]$.

4.2.1 Computing optV^2_γ

In this section we describe an algorithm in two steps for computing optV^2_γ for a given k-sequence S: in Step 1 we consider the particular case where we have three sequences, and in Step 2 we treat the general case.

Step 1: $k = 3$

Let $S = s_1, s_2, s_3$ be a 3-sequence. Suppose that we have induced lengths $\vec{\mathcal{L}} = [\mathcal{L}_{12}, \mathcal{L}_{13}, \mathcal{L}_{23}]$ of a V^2_γ-optimal alignment A of S. In consequence, we have that $\mathcal{L}_{hi} = |A_{\{h,i\}}|$ for each pair h, i. Notice that knowing the lengths $\mathcal{L}_{12}, \mathcal{L}_{13}$ and $\mathcal{L}_{23}$ does not imply knowing the V^2_γ-optimal alignment A. In general, we cannot even infer what $|A|$ is. For example, the alignments

$$\begin{bmatrix} s_1(1) & s_1(2) & \text{-} \\ s_2(1) & \text{-} & s_2(2) \\ \text{-} & s_3(1) & s_3(2) \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} s_1(1) & s_1(2) & \text{-} & \text{-} \\ s_2(1) & \text{-} & s_2(2) & \text{-} \\ s_3(1) & \text{-} & \text{-} & s_3(2) \end{bmatrix}$$

have different lengths but same induced lengths for s_1, s_2, s_3, where $|s_1| = |s_2| = |s_3| = 2$ and $\mathcal{L}_{12} = \mathcal{L}_{13} = \mathcal{L}_{23} = 3$. However, if we know $\vec{\mathcal{L}} = [\mathcal{L}_{12}, \mathcal{L}_{13}, \mathcal{L}_{23}]$, we have

$$\begin{aligned} \mathrm{optV}^2_\gamma(S) &= \min_{A \in \mathcal{A}_S : \mathcal{L}_{hi} = |A_{\{h,i\}}|, \forall h,i} \left\{ \textstyle\sum_{h=1}^{k-1} \sum_{i=h+1}^{k} v\mathrm{A}_\gamma[A_{\{h,i\}}] / \mathcal{L}_{hi} \right\} \\ &= \min_{A \in \mathcal{A}_S} \left\{ \textstyle\sum_{h=1}^{k-1} \sum_{i=h+1}^{k} v\mathrm{A}_{\gamma^{(hi)}}[A_{\{h,i\}}] \right\}, \end{aligned}$$

where $\gamma^{(hi)}$ is a scoring matrix obtained by multiplying the elements of γ by $1/\mathcal{L}_{hi}$. Since we guarantee it is the induced length of an alignment V^2_γ-optimal, we fix $\vec{\mathcal{L}}$ and compute $\vec{\gamma}$ in order to calculate the entries of a table $D_{\vec{\mathcal{L}}}$, such that

$$D_{\vec{\mathcal{L}}}(\vec{v} = [v_1, v_2, v_3], \vec{L} = [L_{12}, L_{13}, L_{23}]) = \min_{A \in \mathcal{A}_{S(\vec{1}:\vec{v})}, L_{hi} = |A_{\{h,i\}}|, \forall h,i} \{ \textstyle\sum_{h<i} v\mathrm{A}_{\gamma^{(hi)}}[A_{\{h,i\}}] \}$$

corresponds to the score of an alignment with the lowest $\vec{\gamma}$-SP-score when the induced length is $\vec{L}$. The table $D_{\vec{\mathcal{L}}}$ can then be computed using the following recurrence

$$D_{\vec{\mathcal{L}}}(\vec{v}, \vec{L}) = \begin{cases} 0, & \text{if } \vec{v} = \vec{0}, \vec{L} = \vec{0}, \\ \infty, & \text{if } \vec{v} = \vec{0}, \vec{L} \ne \vec{0} \text{ or } \vec{v} \ne \vec{0}, \vec{L} = \vec{0}, \\ \min_{\vec{b} \in \mathcal{B}_k, \vec{b} \le \vec{v}, \vec{b} \le \vec{L}} \{ D_{\vec{\mathcal{L}}}(\vec{v} - \vec{b}, \vec{L} - \vec{b}) + v\mathrm{SP}_{\vec{\gamma}}[\vec{b} \cdot S(\vec{v})] \}, & \text{otherwise}, \end{cases}$$

where $\vec{b} \le \vec{L}$ is also an overloading, meaning that $\vec{L} - \vec{b} \ge \vec{0}$.

In this case, if $\vec{\mathcal{L}}$ is the induced length of a V^2_γ-optimal alignment of S, then $\mathrm{optV}^2_\gamma(S) = D_{\vec{\mathcal{L}}}(\vec{n}, \vec{\mathcal{L}})$. If each sequence has length n, then the total space to store the table $D_{\vec{\mathcal{L}}}$ is $(2n+1)^{\binom{3}{2}} \cdot (n+1)^3 = \Theta(n^6)$. When $\vec{\mathcal{L}}$ is unknown, the computation must be repeated for each $\vec{\mathcal{L}} \in \mathbb{L}$, but the space can be reused and no additional space required. If $\vec{\mathcal{L}}$ is known, the algorithm runs to compute all entries of $D_{\vec{\mathcal{L}}}$. As $D_{\vec{\mathcal{L}}}$ has $\Theta(n^6)$ entries and each entry takes $O(1)$-time to be computed, the total time spent is $O(n^6)$. If $\vec{\mathcal{L}}$ is unknown, the time needed to compute $\vec{\mathcal{L}}$ must be multiplied by the total of elements in $\mathbb{L}$ which is $(2n+1)^3$. Therefore, in the latter case, the total time is $O(n^6 \cdot (2n+1)^3) = O(n^9)$.

Step 2: $k > 3$

Algorithm 3 is a natural extension of the algorithm described in Step 1. Given a scoring matrix γ and an induced length $\vec{\mathcal{L}}$, let $\gamma \times \vec{\mathcal{L}} = [\gamma^{(12)}, \ldots, \gamma^{(k(k-1))}]$ be the vector of $\binom{k}{2}$ scoring matrices, where $\gamma^{(hi)}$ is obtained dividing each entry of γ by $\mathcal{L}_{hi}$ for each $h < i$.

Algorithm 3 V^2_γ-optimal alignment of S.

```
Input: A $k$-sequence $S = s_1, \ldots, s_k$ such that $n_i = |s_i|$
Output: $\mathrm{optV}^2_\gamma(S)$
1: for each $\vec{\mathcal{L}} \in \mathbb{L}$ do
2:    $D_{\vec{\mathcal{L}}}(\vec{0}, \vec{0}) \leftarrow 0$
3:    for each $\vec{L} \neq \vec{0}$ do $D_{\vec{\mathcal{L}}}(\vec{0}, \vec{L}) \leftarrow \infty$
4:    for each $\vec{v} \neq \vec{0}$ do $D_{\vec{\mathcal{L}}}(\vec{v}, \vec{0}) \leftarrow \infty$
5:    $\vec{\gamma} \leftarrow \gamma \times \mathcal{L}$
6:    for each $\vec{0} < \vec{v} \leq \vec{n}$ in lexicographical order do
7:       for each $\vec{L} \neq \vec{0}$ in lexicographical order do
8:          $D_{\vec{\mathcal{L}}}(\vec{v}, \vec{L}) = \min_{\vec{b} \in \mathcal{B}^k, \vec{b} \leq \vec{v}, \vec{b} \leq \vec{L}} \{D_{\vec{\mathcal{L}}}(\vec{v} - \vec{b}, \vec{L} - \vec{b}) + v\mathrm{SP}_{\vec{\gamma}}[\vec{b} \cdot S(\vec{v})]\}$
9: return $\min_{\vec{\mathcal{L}} \in \mathbb{L}}\{D_{\vec{\mathcal{L}}}(\vec{n}, \vec{\mathcal{L}})\}$
```

For k sequences of length n, Algorithm 3 needs $(2n+1)^{\binom{k}{2}} \cdot (n+1)^k$ space to store the table $D_{\vec{\mathcal{L}}}$. For each of the $(2n+1)^{\binom{k}{2}}$ values $\vec{\mathcal{L}} \in \mathbb{L}$, table $D_{\vec{\mathcal{L}}}$ is recalculated. Since the computation of each entry takes $O(2^k k^2)$-time, the total time is

$$O\big(2^k k^2 \cdot (2n+1)^{\binom{k}{2}} \cdot (2n+1)^{\binom{k}{2}} (n+1)^k\big) = O\big(\big(1 + 1/(2n+1)\big)^k (2n+1)^{k^2} k^2\big)\,.$$

If $k \leq 2n+1$, the total time can be written as $O\big((2n+1)^{k^2} k^2)\big)$, since $(1+1/k)^k \leq e = 2.718281828\ldots$ Notice that $(1+1/(2n+1))^k \leq (1+1/k)^k \leq e$ is also constant.

4.3 NMSA-3

Let $S = s_1, \ldots, s_k$ be a k-sequence and $A = [s'_1, \ldots, s'_k]$ be an alignment of S. As defined in Equation (5), $\mathrm{V}^3_\gamma[A]$ takes into account the length of A, and the optimal function is given by $\mathrm{optV}^3_\gamma(S) = \min_{A \in \mathcal{A}_S} \{\mathrm{V}^3_\gamma[A]\}$. The V^3_γ-*optimal* alignment of S is an alignment A such that $\mathrm{V}^3_\gamma[A] = \mathrm{optV}^3_\gamma(S)$. Then **NMSA-3** is defined as follows: for a fixed matrix γ, given a k-tuple S, determine $\mathrm{optV}^3_\gamma(S)$.

4.3.1 Computing optV^3_γ

Here, each entry $D(\vec{v}, L)$ of D stores the SP-score of an alignment A of the prefix $S(\vec{v})$ with the lowest SP-score, such that $\sum_{i<h} |A_{\{i,h\}}| = L$. If $\vec{b}$ is a k-vector, define $\|\vec{b}\| = \sum_{h<i} b_h b_i$. Notice that if $\vec{v} - \vec{b}$ is the last column of an alignment A and $L = \sum_{h=1}^{k-1} \sum_{i=h+1}^{k} |A_{\{h,i\}}|$

is the sum of lengths of the alignments induced by A. Thus, the sum of lengths of the alignments induced by $A(1{:}|A|-1)$ is $L - \|\vec{b}\|$. Therefore,

$$D(\vec{v},L)=\begin{cases}0\,, & \text{if } \vec{v}=\vec{0}, L=0,\\ \infty\,, & \text{if } \vec{v}=\vec{0}, L\neq 0 \text{ or } \vec{v}\neq\vec{0}, L=0,\\ \{D(\vec{v}-\vec{b}, L-\|\vec{b}\|)+v\mathrm{SP}_\gamma[\vec{b}\cdot S(\vec{v})]\}, & \text{otherwise}.\end{cases}$$

Algorithm 4 provides more details about the procedure for computing optV^3_γ.

Algorithm 4 V^3_γ-optimal alignment of S.

```
Input: a $k$-sequence $S = s_1, \dots, s_k$ such that $n_i = |s_i|$
Output: $\mathrm{optV}^3_\gamma(S)$
1: $D(\vec{0}, 0) \leftarrow 0$
2: for each $L \neq 0$ do $D(\vec{0}, L) \leftarrow \infty$
3: for each $\vec{v} \neq \vec{0}$ do $D(\vec{v}, 0) \leftarrow \infty$
4: for each $\vec{0} < \vec{v} \leq \vec{n}$ in lexicographical order do
5:     for $L \leftarrow 1, 2, \dots, N(k-1)$ do
6:         $D(\vec{v}, L) \leftarrow \min_{\vec{b}\in\mathcal{B}^k, \vec{b}\leq\vec{v}, \|\vec{b}\|\leq L} \{D(\vec{v}-\vec{b}, L-\|\vec{b}\|) + v\mathrm{SP}_\gamma[\vec{b}\cdot S(\vec{v})]\}$
7: return $\min_L \{D(\vec{n}, L)/L\}$
```

Assume that all sequences in S have length n. The table D is computed for all possible values of $L = 1, \dots, \binom{k}{2}(2n)(= nk^2 - nk)$ and, after this, we determine $\mathrm{optSP}_\gamma(S) = \min_L \{D(\vec{n}, L)/L\}$. Thus, table D needs space equivalent to $(nk^2 - nk + 1)\cdot(n+1)^k = \Theta(k^2(n+1)^{k+1})$. Since the time required to determine each entry of D is $O(2^k k^2)$, the running time of Algorithm 4 is $O(2^k k^4 (n+1)^{k+1})$.

5 Approximation algorithms for MSA and NMSA-2

Gusfield [15] described a 2-approximation algorithm for **MSA**. It assumes that $\gamma \in \mathbb{M}^{\mathrm{C}}$. In this section, we adapt Gusfield's algorithm, proposing a 6-approximation algorithm for **MSA** when $\gamma \in \mathbb{M}^{\mathrm{A}}$ and a 12-approximation algorithm for **NMSA**-2 problem when $\gamma \in \mathbb{M}^{\mathrm{N}}$.

We consider here a generic function v to score an alignment of a 2-sequence such that $\mathrm{opt}(s,s) = 0$ (identity) and $\mathrm{opt}(s,t) = \mathrm{opt}(t,s)$ (symmetry), where $\mathrm{opt}(s,t)$ is the score of a *v-optimal alignment* of a 2-sequence s, t. Notice that $v\mathrm{A}_\gamma$ and $v\mathrm{N}_\gamma$ have these properties when $\gamma \in \mathbb{M}^{\mathrm{A}}$ and $\gamma \in \mathbb{M}^{\mathrm{N}}$, respectively. Let S be a k-sequence and A in $\mathcal{A}_S$ be an alignment. We define V and OPT as functions such that $V[A] = \sum_{h=1}^{k-1}\sum_{i=h+1}^{k} v[A_{\{h,i\}}]$ and $\mathrm{OPT}(S) = \min_{A\in\mathcal{A}_S} V(A)$. Thus, a *$V$-optimal alignment* is an alignment A such that $V[A] = \mathrm{OPT}(S)$.

Let c be an integer with $1 \leq c \leq k$. A *star X with center c* of $S = s_1, \dots, s_k$, also called a *c-star*, is a collection of $k-1$ alignments: alignment $X_h = [s'_h, s^h_c]$ of s_h, s_c, for each $h < c$, where $v[s'_h, s^h_c] = v[s^h_c, s'_h]$, and alignment $X_h = [s^h_c, s'_h]$ of s_c, s_h, for each $h > c$, where $v[s^h_c, s'_h] = v[s'_h, s^h_c]$. The set of all c-stars is denoted by $\mathcal{X}_c$. The score of the c-star X is $\mathrm{cStar}(X) = \sum_{h\neq c} v[X_h]$ and a *v-optimal star* is one whose score is $\mathrm{optStar}(S) = \min_{X\in\mathcal{X}_c, c\in\mathbb{N}}\{\mathrm{cStar}(X)\}$. Notice that $\mathrm{optStar}(S) = \min_c \{\sum_{h\neq c} \mathrm{opt}(s_h, s_c)\}$, and if $v = v\mathrm{A}_\gamma$ and $\gamma \in \mathbb{M}^{\mathrm{A}}$, $\mathrm{optStar}(S)$ can be computed in $O(k^2n^2)$-time, and if $v = v\mathrm{N}_\gamma$ and $\gamma \in \mathbb{M}^{\mathrm{N}}$, $\mathrm{optStar}(S)$ can be computed in $O(k^2n^3)$-time when $|s_i| \leq n$, for each s_i in S.

We say that an alignment A of a k-sequence S and a c-star X of S are *compatible* (A is compatible with X or X is compatible with A) in S when either $A_{\{h,c\}}$ or $A_{\{c,h\}}$ is equal to X_h, for each h. It is easy to obtain, from an alignment A and c in $\mathbb{N}$, the unique c-star X

that is compatible with A. On the other hand, it is known from Feng and Doolitte [14] that one can find an alignment A compatible with a given c-star X in $O(kn)$-time, where $|s_i| \leq n$ for each sequence s_i in the k-sequence S. In this case, there exists one or more compatible alignments with X.

The next result is a straightforward consequence of a result from Gusfield [15], where v is the scoring function. See the proof in [4].

▶ **Lemma 5.1.** *Given a k-sequence S,* $\mathrm{optStar}(S) \leq (2/k)\,\mathrm{OPT}(S)$.

From now on, we consider $v = v\mathrm{A}_\gamma$, or $v = v\mathrm{N}_\gamma$ and $\gamma = \mathbb{M}^{\mathrm{A}}$ or $\gamma = \mathbb{M}^{\mathrm{N}}$, respectively. Let $s, t \in \Sigma^*$ be sequences. Suppose that $A = [s', t']$ is an alignment of s, t. We say that a column j is *splittable in* A if $s'(j) \neq \texttt{-}$, $t'(j) \neq \texttt{-}$ and $\min\{\gamma_{t'(j)\to\texttt{-}}, \gamma_{s'(j)\to\texttt{-}}\} \leq \gamma_{s'(j)\to t'(j)}$. Let $J := \{j_i \in \mathbb{N} : 1 \leq j_1 < \cdots < j_m \leq |A| \text{ and } j_i \text{ is splittable in } A\}$. An A-*splitting* is the alignment

$$\begin{bmatrix} s'(1\colon j_1-1) & s'(j_1) & \texttt{-} & s'(j_1+1\colon j_2-1) & s'(j_2) & \texttt{-} & \dots & s'(j_m+1\colon |A|) \\ t'(1\colon j_1-1) & \texttt{-} & t'(j_1) & t'(j_1+1\colon j_2-1) & \texttt{-} & t'(j_2) & \dots & t'(j_m+1\colon |A|) \end{bmatrix}.$$

We say that J is *required to split* A. The following proposition is used to check properties of an A-splitting. See its proof in [4].

▶ **Proposition 5.2.** *Consider $\gamma \in \mathbb{M}^{\mathrm{A}}$ and $\mathtt{a}, \mathtt{b} \in \Sigma$. If $\gamma_{\mathtt{a}\to\texttt{-}} > \gamma_{\mathtt{a}\to\mathtt{b}}$ or $\gamma_{\mathtt{a}\to\texttt{-}} > \gamma_{\mathtt{b}\to\mathtt{a}}$, then $\gamma_{\mathtt{a}\to\mathtt{b}} = \gamma_{\mathtt{b}\to\mathtt{a}}$.*

Let $X = \{X_1, \dots, X_{c-1}, X_{c+1}, X_k\}$ be a c-star. The X-*starsplitting* is the c-star $Y = \{Y_1, \dots, Y_{c-1}, Y_{c+1}, Y_k\}$, where Y_j is the X_j-splitting for each j. The next result shows that the v-score of the star Y is bounded by the v-score of the star X when $\gamma \in \mathbb{M}^{\mathrm{A}}$ and $v = v\mathrm{A}_\gamma$, or $\gamma \in \mathbb{M}^{\mathrm{N}}$ and $v = v\mathrm{N}_\gamma$. Thus, as a consequence of Preposition 5.2, we have the following lemma. The formal proof can be seen in [4].

▶ **Lemma 5.3.** *Let $S = s_1, \dots, s_k$ be a k-sequence, X be a c-star of S, Y be the X-starsplitting and v be a function to score alignments. Consider $\gamma \in \mathbb{M}^{\mathrm{A}}$ and $v = v\mathrm{A}_\gamma$, or $\gamma \in \mathbb{M}^{\mathrm{A}}$ and $v = v\mathrm{N}_\gamma$. Then, Y is also a c-star and cStar$(Y) \leq 3\,$cStar(X).*

Proof sketch. As a consequence of $\gamma \in \mathbb{M}^{\mathrm{A}}$ and Proposition 5.2, we have that Y is also a c-star. Let $h \in \{1, \dots, k\}$ and J be a set required to split X. We prove that $v\mathrm{A}_\gamma[Y_h] \leq 3\,v\mathrm{A}_\gamma[X_h]$ and $v\mathrm{N}_\gamma[Y_h] \leq 3\,v\mathrm{N}_\gamma[X_h]$ hold since $\gamma \in \mathbb{M}^{\mathrm{A}}$ when $v = v\mathrm{A}_\gamma$ and $\gamma \in \mathbb{M}^{\mathrm{N}}$ when $v = v\mathrm{N}_\gamma$, and $J \subseteq \{1, 2, \dots, |A|\}$. Therefore, in these cases, we have

$$\mathrm{cStar}(Y) = \sum_{h\neq c} v[Y_h] = \sum_{h<c} v[Y_h] + \sum_{h>c} v[Y_h] \leq 3 \sum_{h\neq c} v[X_h] = 3\,\mathrm{cStar}(X)\,. \qquad ◀$$

Notice that the time consumption for computing an X-splitting from X is $O(kn)$ when $|s_i| \leq n$, for each $s_i \in S$. Considering a star X of $S = s_1, \dots, s_k$, there can exist many compatible alignments with a v-star Y which is a X-splitting. Let CompatibleAlign be a subroutine that receives the c-star Y and returns an alignment A compatible with Y. It is quite simple: if symbols $s_h(j_1)$ and $s_c(j_2)$ are aligned in X_h, they are also aligned in A; otherwise, $s_h(j)$ aligns only with $\texttt{-}$ in A. This property is enough to guarantee the approximation factor of **MSA** and **NMSA**-2.

Let $Q_{\max} := \max_{\mathtt{a}\in\Sigma}\{\gamma_{\mathtt{a}\to\texttt{-}}, \gamma_{\texttt{-}\to\mathtt{a}}\}$ and consider the following result, whose proof can be found in [4].

▶ **Proposition 5.4.** *Let S be a k-sequence, X be a c-star of S and Y be a X-starsplitting. Assume that $\gamma \in \mathbb{M}^{\mathrm{A}}$ and that* CompatibleAlign(Y) *returns $A = [s'_1, \ldots, s'_k]$. If $h \neq c$ and $i \neq c$, we have that*

(i) $\gamma_{s'_h(j) \to s'_i(j)} \leq \gamma_{s'_h(j) \to s'_c(j)} + \gamma_{s'_c(j) \to s'_i(j)}$ *for each $j = 1, \ldots, |A|$, and*

(ii) $v\mathrm{N}_\gamma[A_{\{h,i\}}] \leq 2\,Q_{\max}$ *when $\gamma \in \mathbb{M}^{\mathrm{N}}$.*

Proposition 5.4 is an auxiliary result to show the following lemma.

▶ **Lemma 5.5.** *Let S be a k-sequence, X be a c-star of S, Y be a X-starsplitting and* CompatibleAlign$(Y) = A$. *Then, for each $h < i$, $h \neq c$ and $i \neq c$,*

(i) $v\mathrm{A}_\gamma[A_{\{h,i\}}] \leq v\mathrm{A}_\gamma[A_{\{h,c\}}] + v\mathrm{A}_\gamma[A_{\{c,i\}}]$ *when $\gamma \in \mathbb{M}^{\mathrm{A}}$, and*

(ii) $v\mathrm{N}_\gamma[A_{\{h,i\}}] \leq 2\left(v\mathrm{N}_\gamma[A_{\{h,c\}}] + v\mathrm{N}_\gamma[A_{\{c,i\}}]\right)$ *when $\gamma \in \mathbb{M}^{\mathrm{N}}$.*

Proof. Let $A = [s'_1, \ldots, s'_k]$ and $Z = \{j : s'_c(j) \neq \texttt{-} \text{ and } s'_h(j) = s'_i(j) = \texttt{-}\}$. We have $v\mathrm{A}_\gamma[A_{\{h,i\}}] \leq v\mathrm{A}_\gamma[A_{\{h,c\}}] + v\mathrm{A}_\gamma[A_{\{c,i\}}] - \sum_{j \in Z}(\gamma_{\texttt{-} \to s'_c(j)} + \gamma_{s'_c(j) \to \texttt{-}})$ from a consequence of Proposition 5.4. Besides, since $\gamma \in \mathbb{M}^{\mathrm{A}}$, we have that $\gamma_{\texttt{-} \to s'_c(j)}, \gamma_{s'_c(j) \to \texttt{-}} > 0$. It implies that (i) is proven.

For proving (ii), observe first that, by definition of $\mathbb{M}^{\mathrm{N}}$, we have that $Q_{\max} \leq \gamma_{\texttt{-} \to s'_c(j)} + \gamma_{s'_c(j) \to \texttt{-}}$ for every j. Furthermore, following these statements, we have that

$$\begin{aligned}
v\mathrm{N}_\gamma[A_{\{h,i\}}] &= \frac{v\mathrm{A}_\gamma[A_{\{h,i\}}]}{|A_{\{h,i\}}|} \\
&\leq \frac{v\mathrm{A}_\gamma[A_{\{h,i\}}] + 2 \cdot Q_{\max}\,|Z|}{|A_{\{h,i\}}| + |Z|} \\
&\leq 2 \cdot \frac{v\mathrm{A}_\gamma[A_{\{h,i\}}] + Q_{\max}\,|Z|}{|A_{\{h,i\}}| + |Z|} && (6)\\
&\leq 2 \cdot \frac{v\mathrm{A}_\gamma[A_{\{h,c\}}] + v\mathrm{A}_\gamma[A_{\{c,i\}}] - \sum_{j \in Z}(\gamma_{\texttt{-} \to s'_c(j)} + \gamma_{s'_c(j) \to \texttt{-}}) + Q_{\max}\,|Z|}{|A_{\{h,i,c\}}| - |Z| + |Z|} && (7)\\
&\leq 2 \cdot \frac{v\mathrm{A}_\gamma[A_{\{h,c\}}] + v\mathrm{A}_\gamma[A_{\{c,i\}}] - Q_{\max}\,|Z| + Q_{\max}\,|Z|}{|A_{\{h,i,c\}}| - |Z| + |Z|} \\
&= 2 \cdot \left(\frac{v\mathrm{A}_\gamma[A_{\{h,c\}}]}{|[A_{\{h,i,c\}}]|} + \frac{v\mathrm{A}_\gamma[A_{\{c,i\}}]}{|[A_{\{h,i,c\}}]|}\right) && (8)\\
&\leq 2 \cdot \Big(v\mathrm{N}_\gamma[A_{\{h,c\}}] + v\mathrm{N}_\gamma[A_{\{c,i\}}]\Big)\,, && (9)
\end{aligned}$$

where the first inequality of (6) is a consequence of Proposition 5.4 and the second inequality follows since every entry of γ is nonnegative, (7) follows from the result in the first paragraph and from $|A_{\{h,i\}}| = |A_{\{h,i,c\}}| - |Z|$, (8–9) follow as a consequence of the definition of $Q_{\max}$, and as a consequence of $|A_{\{h,c\}}| \leq |A_{\{h,i,c\}}|$ and $|A_{\{c,i\}}| \leq |A_{\{h,i,c\}}|$. ◀

We can now describe the approximation algorithm (Algorithm 5).

Algorithm 5 Approximation algorithm for **MSA** and **NMSA**-2.

Input: k-sequence $S = s_1, \ldots, s_k$

Output: $v[A]$, where A is an alignment of S, and $v\mathrm{SP}_\gamma[A] \leq 6\,\mathrm{optSP}_\gamma(S)$ if $v = v\mathrm{A}_\gamma$ and $\gamma \in \mathbb{M}^{\mathrm{A}}$, and $\mathrm{V}^2_\gamma[A] \leq 12\,\mathrm{optNSP}^2_\gamma(S)$ if $v = v\mathrm{N}_\gamma$ and $\gamma \in \mathbb{M}^{\mathrm{N}}$.

```
1: Let $X$ be a $v$-optimal star of $S$ with center $c$
2: Compute the $X$-splitting $Y$
3: $A \leftarrow$ CompatibleAlign($Y$)
4: return $v[A]$
```

Clearly, Algorithm 5 is correct. Furthermore, Lemmas 5.1, 5.3 and 5.5 are auxiliary to prove its approximation factor. Since the running time of CompatibleAlign is $O(k^2n)$, a straightforward running time analysis allows us to state the following theorem. The detailed proof is presented in [4].

▶ **Theorem 5.6.** *Let $S = s_1, \ldots, s_k$ be a k-sequence and γ be a scoring matrix. Then, Algorithm 5 computes $v[A]$ correctly:*

(i) *in $O(k^2n^2)$-time such that $v\mathrm{SP}_\gamma[A] \le 6\,\mathrm{optSP}_\gamma(S)$, if $v = v\mathrm{A}_\gamma$ and $\gamma = \mathbb{M}^\mathrm{A}$, or*

(ii) *in $O(k^2n^3)$-time such that $\mathrm{V}^2_\gamma[A] \le 12\,\mathrm{optNSP}^2_\gamma(S)$, if $v = v\mathrm{N}_\gamma$ and $\gamma \in \mathbb{M}^\mathrm{N}$, where A is the alignment of S computed by the algorithm.*

6 Conclusion and future work

We presented and discussed several aspects of normalized multiple sequence alignment (NMSA). We defined three new criteria for computing normalized scores when aligning multiple sequences, showing the NP-hardness and exact algorithms for solving the **NMSA**-z given criterion V^z_γ for each z. In addition, we adapted an existing 2-approximation algorithm for **MSA** when the scoring matrix γ is in the common class $\mathbb{M}^\mathrm{C}$, leading to a 6-approximation algorithm for **MSA** when γ is in the broader class $\mathbb{M}^\mathrm{A} \supseteq \mathbb{M}^\mathrm{C}$, and to a 12-approximation for **NMSA**-2 when γ is in $\mathbb{M}^\mathrm{N} \subseteq \mathbb{M}^\mathrm{A}$, a slightly more restricted class compared to $\mathbb{M}^\mathrm{A}$ such that the cost of a deletion for any symbol is at most twice the cost for any other.

This work is an effort to expand the boundaries of multiple sequence alignment algorithms towards normalization, an unexplored domain that can produce results with higher accuracy in some applications. In future work, we will implement our algorithms in order to verify how large are the sequences that our algorithms are able to handle. Also, we plan to perform practical experiments, measuring how well alignments provided by our algorithms and other MSA algorithms agree with multiple alignment benchmarks. In addition, we intend to measure the accuracy of phylogenetic tree reconstruction based on our alignments for simulated and real genomes. Finally, we will work on heuristics and parallel versions of our algorithms in order to faster process large datasets.

References

1 A. Abbott and A. Tsay. Sequence analysis and optimal matching methods in sociology: Review and prospect. *Sociol Method Res*, 29(1):3–33, 2000. doi:10.1177/0049124100029001001.

2 A. Andoni, R. Krauthgamer, and K. Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proc. of FOCS*, pages 377–386. IEEE, 2010. doi:10.1109/FOCS.2010.43.

3 A. Apostolico and Z. Galil. *Pattern Matching Algorithms*. Oxford University Press, 1997.

4 E. Araujo, L. C. Rozante, D. P. Rubert, and F. V. Martinez. Algorithms for normalized multiple sequence alignments, 2021. arXiv:2107.01607.

5 E. Araujo and J. Soares. Scoring matrices that induce metrics on sequences. In *Proc. of LATIN*, pages 68–79, 2006. doi:10.1007/11682462_11.

6 A. N. Arslan and Ö. Egecioglu. An efficient uniform-cost normalized edit distance algorithm. In *Proc. of SPIRE*, pages 9–15. IEEE, 1999. doi:10.1109/SPIRE.1999.796572.

7 A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J Comput*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

8 R. Barzilay and L. Lee. Bootstrapping lexical choice via multiple-sequence alignment. In *Proc. of EMNLP*, pages 164–171. ACL, 2002. doi:10.3115/1118693.1118715.

9 H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J Appl Math*, 48(5):1073–1082, 1988. doi:10.1137/0148063.

10 D. Chakraborty, D. Das, E. Goldenberg, M. Koucký, and M. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. *J ACM*, 67(6):1–22, 2020. doi:10.1145/3422823.

11 M. Crochemore, G. M. Landau, and M. Ziv-Ukelson. A sub-quadratic sequence alignment algorithm for unrestricted cost matrices. In *Proc. of SODA*, pages 679–688. SIAM, 2002. doi:10.5555/545381.545472.

12 J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, 34(4):508–519, 1999. doi:10.1002/(SICI)1097-0134(19990301)34:4<508::AID-PROT10>3.0.CO;2-4.

13 I. Elias. Settling the intractability of multiple alignment. *J Comput Biol*, 13(7):1323–1339, 2006. doi:10.1089/cmb.2006.13.1323.

14 D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *J Mol Evol*, 25(4):351–360, 1987. doi:10.1007/BF02603120.

15 D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull Math Biol*, 55(1):141–154, 1993. doi:10.1007/BF02460299.

16 W. Haque, A. Aravind, and B. Reddy. Pairwise sequence alignment algorithms: A survey. In *Proc. of ISTA*, pages 96–103. ACM, 2009. doi:10.1145/1551950.1551980.

17 M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. *Bioinformatics*, 11(1):13–18, 1995. doi:10.1093/bioinformatics/11.1.13.

18 A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE T Pattern Anal*, 15(9):926–932, 1993. doi:10.1109/34.232078.

19 W. J. Masek and M. S. Paterson. A faster algorithm computing string edit distances. *J Comput Syst Sci*, 20(1):18–31, 1980. doi:10.1016/0022-0000(80)90002-1.

20 S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, 1970. doi:10.1016/0022-2836(70)90057-4.

21 T. H. Ogden and M. S. Rosenberg. Multiple sequence alignment accuracy and phylogenetic inference. *Syst Biol*, 55(2):314–328, 2006. doi:10.1080/10635150500541730.

22 P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM J Appl Math*, 26(4):787–793, 1974. doi:10.1137/0126070.

23 F. Sievers and D. G. Higgins. Clustal Omega. *Curr Protoc Bioinfo*, 48(1):3.13.1–3.13.16, 2014. doi:10.1002/0471250953.bi0313s48.

24 F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol Syst Biol*, 7(1):539, 2011. doi:10.1038/msb.2011.75.

25 J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res*, 27(13):2682–2690, 1999. doi:10.1093/nar/27.13.2682.

26 E. Vidal, A. Marzal, and P. Aibar. Fast computation of normalized edit distances. *IEEE T Pattern Anal*, 17(9):899–902, 1995. doi:10.1109/34.406656.

27 I. M. Wallace, O. O'Sullivan, D. G. Higgins, and C. Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res*, 34(6):1692–1699, 2006. doi:10.1093/nar/gkl091.

28 X.-D. Wang, J.-X. Liu, Y. Xu, and J. Zhang. A survey of multiple sequence alignment techniques. In *Proc. of ICIC*, pages 529–538. Springer, 2015. doi:10.1007/978-3-319-22180-9_52.