On the Construction of Elastic Degenerate Strings

Nicola Rizzo
Department of Computer Science, University of Helsinki, Finland

Veli Mäkinen
Department of Computer Science, University of Helsinki, Finland

Nadia Pisanti
Universitá di Pisa, Italy

Abstract

An *elastic degenerate string* (EDS) is a sequence of sets of strings. In the context of bioinformatics, EDSes can be used to represent the variations observed in a population from its consensus genome. Pattern matching and comparison problems on EDSes have been widely studied in the literature, but their construction has been largely omitted. We fill this gap by showing how algorithms originally developed for related problems of founder reconstruction can be adapted to minimize the total cardinality of the EDS sets and total length of the EDS strings in linear time, given suitable multiple alignments representing the input data.

2012 ACM Subject Classification Theory of computation → Pattern matching; Theory of computation → Sorting and searching; Theory of computation → Dynamic programming; Applied computing → Genomics

Keywords and phrases multiple sequence alignment, pattern matching, data structures, segmentation algorithms, founder reconstruction, dynamic programming, semi-dynamic range minimum queries, positional Burrows–Wheeler transform

Digital Object Identifier 10.4230/OASIcs.Grossi.2

Category Research

Funding This work was partially supported by the PANGAIA, ALPACA, and TeamPerMed projects that received funding from the European Union's Horizon 2020 research and innovation programme with two first under the Marie Skłodowska-Curie grant agreements No. 872539 and 956229, respectively, and the third under grant agreement No. 101060011. NP was also partially supported by MUR PRIN 2022 YRB97K PINC.

1 Introduction

Computing an optimal partition of an interval $[1..c]$, that is, a *segmentation*, is a problem that arises naturally when dealing with the transformation of sequences that are aligned into c positions, after trivial and contradicting constraints are imposed on the desired features of the solution. Figure 1 gives an example of such segmentation on a *multiple sequence alignment* (MSA) of 6 sequences. In an MSA, the r input sequences are made to be of equal length c by adding gaps "$-$" forming a matrix of r rows and c columns which we will denote $\mathsf{MSA}[1..r, 1..c]$. The quality of such MSA can be measured in many ways, but already the problem of placing gaps such that the number of columns containing a single symbol is maximized equals the problem of finding the length of the longest common subsequence (LCS) of the input sequences, which is NP-hard [19]. Nevertheless, MSAs play a vital role in bioinformatics, and practical heuristics have been developed to obtain high quality MSAs [23]. In this paper, we assume an MSA to be given as input, and we look for rigorous ways to convert the MSA into a sequence of sets of strings, that is, into an *elastic degenerate string*

(EDS) [18] (also known as elastic-degenerate text). In Figure 1, we can observe that a segmentation of the MSA directly induces an EDS by interpreting the strings of each segment to be part of a set.

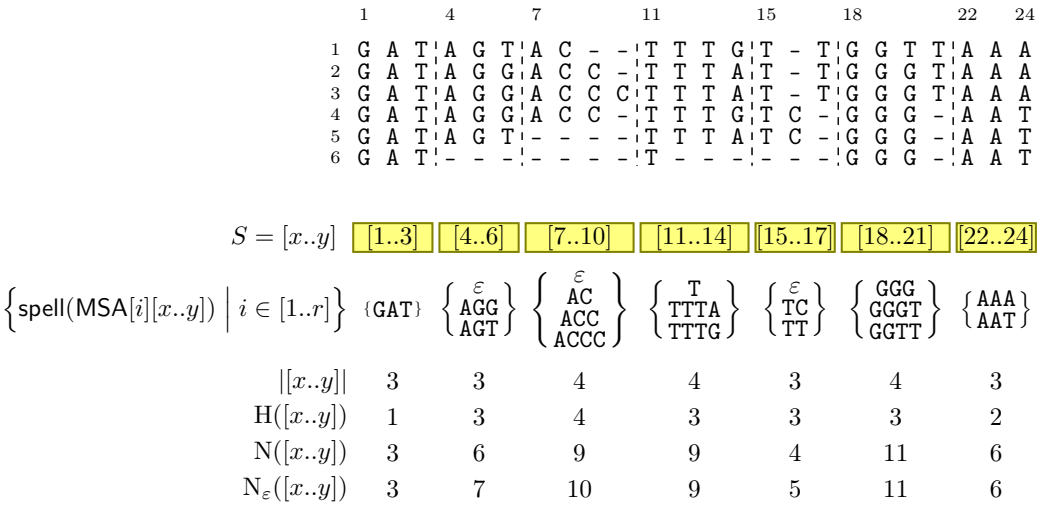

Figure 1 Example of an $\mathsf{MSA}[1..6, 1..24]$ and a segmentation S containing $k = 7$ segments. The maximum length $\mathrm{L}(S)$ is 4, the height $\mathrm{H}(S)$ is 4, the cardinality $\mathrm{m}(S)$ is 19, the size $\mathrm{N}(S)$ is 43, and the gap-aware size $\mathrm{N}_\varepsilon(S)$ is 51.

The transformation from an MSA to an EDS through segmentation can be seen as lossy compression: we merge together equal strings inside each segment and we ignore the exact connections between segments. Such lossy compression can be a desired feature, for example, for hiding the sensitive full sequence input by publishing only the EDS. A more rudimentary reason for such conversion is that EDSes and related pangenome graphs do compress well and enable fast processing and analysis of variation in populations [9]. A notable feature of this scheme is that if a query string occurs in a row of the MSA, then it will also occur in the EDS, but not the other way around. Various ways to optimize the transformation to minimize false positives while not sacrificing privacy could be considered. Minimizing a specific parameter is useful when the downstream analyses of the resulting EDS employ algorithms whose complexity depends from that parameter.

This paper is initiating the study of EDS optimization by looking at some selected optimization criteria (Section 3) for which we can modify algorithms from related literature. We give linear-time algorithms (Section 4) to minimize the total cardinality of the EDS sets (*cardinality*) and total length of the strings in the EDS sets (*size*) under specific assumptions: the algorithms directly work for gapless MSAs, and they serve as heuristics when gaps are interpreted as part of the alphabet. In the latter case, the optimization criteria should be adjusted accordingly: see Section 5 for the multiple problems that we leave open for general MSAs. We start by reviewing the related work in Section 2.

2 Related work

Founder reconstruction. Ukkonen [25] studied the problem of explaining a given set of r aligned sequences representing c haplotype sites (thus no insertions or deletions) with the recombination of a few *founder sequences*, under the simple *crossover* model of recombination (sequence switching at aligned positions); minimizing the founder set is

trivial if the recombination can happen at any position, but if the recombinations are allowed only at selected positions, then the resulting segmentation problems – minimizing the maximum cardinality of a segment (the number of founders in [25]) given a lower bound on the segment length, or given a target average segment length size to surpass – can be solved in polynomial time.

Norri et al. [22] solved one of the problems posed by Ukkonen – the segmentation of aligned sequences minimizing the maximum height given a segment-length lower bound L – in linear $O(rc)$ time and $O(r + L)$ space. The solution proceeds in a dynamic programming fashion, from left to right, and for each aligned position y it uses the *positional Burrows–Wheeler Transform* [10], enhanced by a few other arrays, to efficiently compute all the possible heights of a segment ending at y and their optimal recursive values.

Cazaux et al. [8] studied the related problem of maximizing the minimum segment length or the average segment length of any segmentation that has height of at most a given H, motivated by the minimization of the size of the resulting pangenomic index based on founder sequences. To do so, they recognize that the approach by Norri et al. [22] fits into a more general left-to-right *column-stream* model of computation, and use *range maximum queries* to efficiently compute the recursive values. For this last task, they modify existing techniques for the semi-dynamic setting where an array is indexed for amortized constant-time range queries in an online fashion.

Algorithms on EDSes. There is a wide literature on pattern search and comparison on (elastic) degenerate strings [18, 17, 5, 6, 1, 21, 4, 2, 3] where algorithmic results exhibit complexity that depend on both the cardinality and the size of input EDSes. Moreover, the latest algorithms for EDS-based pangenome comparison (intersection, matching statistics, similarity and distance measures) also depend on the cardinality and size metrics [15, 14, 16]. These results give further motivation for our focus on optimizing these measures.

Indexable Founder Graphs. *Elastic Founder Graphs* (EFGs), introduced by Equi et al. [13], take the simplified recombination model of founder sequences (recombination at selected positions only) to transform an MSA of r sequences and c aligned positions – with insertions and deletions and thus gaps – into an acyclic Elastic Block Graph: the segmentation results into consecutive blocks of nodes. In the general case, the resulting graph is still hard to index just like more general graphs [12, 11], so Equi et al. [13] prove that if each chosen segment spells strings that exclusively occur from the segments' starting column, then this property is conserved in the resulting graph, which is easy to index. This results in a framework of linearithmic (i.e. $O(rc \log(rc))$) time constructions of indexable EFGs, which support fast pattern matching.

Rizzo et al. [24] improved some construction solutions to linear time (i.e. $O(rc)$) while studying the problem of minimizing the maximum segment length or maximum height of a segmentation resulting in an indexable EFG.

MSA covers. While segmentation induces a limited class of acyclic pangenome graph representations, there is an analogous approach by Cartes et al. [7] to cover MSAs and generate arbitrary acyclic graphs; the related optimization problems are much harder, although small instances can be solved using integer linear programming.

3 Preliminaries

We denote integer interval $\{1,\ldots,n\}$ as $[1..n]$. Let $\mathsf{MSA}[1..r,1..c] \in (\Sigma \cup \{-\})^{r\times c}$ be a *multiple sequence alignment* of r rows and c columns representing the input sequences (or *strings*) and the aligned positions, respectively. It is built from a finite integer alphabet $\Sigma = [1..\sigma]$ ($\Sigma = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$ in our examples) augmented with the *gap symbol* $- \notin \Sigma$ to represent insertions and deletions. It is immediate to see that the alignment takes at most $O(rc)$ words of space, or $O(rc\log|\Sigma|)$ bits of space. We denote the i-th row of the MSA as $\mathsf{MSA}[i,1..c]$, and the concatenation of its characters from position x to y, with $x \leqslant y$, as $\mathsf{MSA}[i,x..y]$. We say that $\mathsf{MSA}[1..r,1..c]$ is *gapless* if no gap symbol is used. The operator $\mathrm{spell}(T)$ removes all the gap symbols from a given string $T \in (\Sigma\cup\{-\})^c$: for example, $\mathrm{spell}(\mathsf{MSA}[i,1..c])$ is the i-th (unaligned) sequence. If T contains only gaps then $\mathrm{spell}(T) = \varepsilon$, the empty string. Consider the following definition alongside Figure 1.

▶ **Definition 1** (Segmentation). *Given* $\mathsf{MSA}[1..r,1..c] \in (\Sigma\cup\{-\})^{r\times c}$, *a* segmentation *of the MSA is a partition* $S = S_1, S_2, \ldots, S_k$ *of* $[1..c]$, *i.e. a sequence of contiguous and non-overlapping intervals covering* $[1..c]$. *In symbols*

$$S_1 = [x_1..y_1],\ \ldots,\ S_k = [x_k..y_k] \qquad \text{with} \qquad \begin{array}{c} x_1 = 1, \\ y_k = c, \text{ and} \\ y_j + 1 = x_{j+1}\ \forall j \in [1..k-1]. \end{array}$$

We denote:

- *the* number of segments *of* S *as* $|S| = k$*;*
- *the* maximum segment length $\max_{j\in[1..k]}|S_j|$ *of* S *as* $\mathrm{L}(S)$*, where* $|[x..y]| = y - x + 1$*;*
- *the* height $\max_{j\in[1..k]}\mathrm{H}(S_j)$ *of* S *as* $\mathrm{H}(S)$*, where the height* $\mathrm{H}(S_j)$ *of a segment* $S_j = [x..y]$ *is defined as*

$$\mathrm{H}([x..y]) = \left|\left\{\, \mathrm{spell}(\mathsf{MSA}[i,x..y]) \;\middle|\; i \in [1..r] \right\}\right|$$

  (note that the empty string is counted by H*);*
- *the* cardinality $\sum_{j\in[1..k]}\mathrm{H}(S_j)$ *as* $\mathrm{m}(S)$*;*
- *the* size $\sum_{j\in[1..k]}\mathrm{N}(S_j)$ *as* $\mathrm{N}(S)$*, where the size* $\mathrm{N}(S_j)$ *of a segment* $S_j = [x..y]$ *is defined as*

$$\mathrm{N}([x..y]) = \sum_{T\in\,\mathrm{spell}(\mathsf{MSA}[1..r,x..y])} |T| \qquad \text{with}$$

$$\mathrm{spell}(\mathsf{MSA}[1..r,x..y]) = \left\{\, \mathrm{spell}(\mathsf{MSA}[i,x..y]) \;\middle|\; i\in[1..r] \right\}$$

  (note that the empty string ε *contributes* 0 *units to* $\mathrm{N}([x..y])$*); and*
- *the* gap-aware size $\mathrm{N}_\varepsilon(S)$ *as above, substituting* $\mathrm{N}([x..y])$ *with*

$$\mathrm{N}_\varepsilon([x..y]) = \begin{cases} \mathrm{N}([x..y]) + 1 & \text{if } \varepsilon \in \mathrm{spell}(\mathsf{MSA}[1..r,x..y]), \\ \mathrm{N}([x..y]) & \text{otherwise.} \end{cases}$$

Consider the problem of finding a segmentation S of minimum cardinality $\mathrm{m}(S)$. If the MSA contains few but very long sequences, that is, $r \ll c$, the trivial segmentation $S^{\equiv} = [1..c]$ might be optimal. We can encourage a certain level of recombination in the corresponding EDS by setting an upper bound U on the length of the accepted segments.

▶ **Problem 2** (min-U-cardinality). Given $\mathsf{MSA}[1..r,1..c]$ and an upper bound U on the segment length, find a segmentation S containing segments of length at most U and minimizing the cardinality $\mathrm{m}(S)$.

On the other hand, consider finding a segmentation S of minimum gap-aware size $\mathrm{N}_\varepsilon(S)$. In Section 4.1 we show that the trivial segmentation $S^{|||} = [1], \ldots, [c]$ has always minimum size in the gapless setting, resulting in a highly recombinant EDS. We can symmetrically discourage recombination in regions of low sequence similarity by fixing a lower bound L on the length of the accepted segments.

▶ **Problem 3** (min-L-size)**.** Given $\mathsf{MSA}[1..r, 1..c]$ and a lower bound L on the segment length, find a segmentation S containing segments of length at least L and minimizing the gap-aware size $\mathrm{N}_\varepsilon(S)$.

One crucial data structure that we use in Section 4 is based on sorting iteratively the MSA rows while reading the alignment from left to right, column by column. Let Σ have an implicit total order $\leqslant$ on its characters. Given strings $S, T \in \Sigma^c$, we say that S is *lexicographically smaller* than T, in symbols $S \prec_{\text{lex}} T$, if $S[1..x] = T[1..x]$ and $S[x+1] < T[x+1]$ for some $x \in [0..c-1]$ (where $S[1..0]$ is equal to the empty string ε). Symmetrically, S is *co-lexicographically smaller* than T, in symbols $S \prec_{\text{colex}} T$, if $S[x..c] = T[x..c]$ and $S[x-1] < T[x-1]$ for some $x \in [1..c+1]$ (where $S[c+1..c] = \varepsilon$). Moreover, we call $S[1..x]$ a *prefix* of S and $S[1..x]$ a *suffix* of S. They are non-empty if $x \geqslant 1$ and $x \leqslant c$, respectively.

▶ **Definition 4** (Positional Burrows–Wheeler transform [10])**.** *The* positional Burrows–Wheeler transform *(pBWT) of a* gapless $\mathsf{MSA}[1..r, 1..c]$ *for position* $x \in [1..c]$ *consists of array* $\mathtt{a}_x[1..r]$*, representing the prefixes* $\mathsf{MSA}[i, 1..x]$ *for* $i \in [1..r]$ *sorted co-lexicographically, and array* $\mathtt{d}_x[1..r]$*, describing the length of the longest common suffixes of adjacent prefixes in the sorted order. More specifically:*

- $\mathtt{a}_x[1..r]$ *is the permutation of* $[1..r]$ *such that*

$$\mathsf{MSA}[\mathtt{a}_x[1], 1..x] \preceq_{\text{colex}} \mathsf{MSA}[\mathtt{a}_x[2], 1..x] \preceq_{\text{colex}} \cdots \preceq_{\text{colex}} \mathsf{MSA}[\mathtt{a}_x[r], 1..x];$$

- $\mathtt{d}_x[i]$ *is an integer in* $[1..x+1]$ *such that the longest common suffix between* $\mathsf{MSA}[\mathtt{a}_x[i], 1..x]$ *and* $\mathsf{MSA}[\mathtt{a}_x[i-1], 1..x]$ *has length* $\mathtt{d}_x[i]$*, if* $i > 1$ *and such suffix is non-empty, otherwise* $\mathtt{d}_x[i] = x + 1$.

▶ **Lemma 5** ([10, 20])**.** *Let* $\mathsf{MSA}[1..r, 1..c]$ *be a gapless MSA over integer alphabet* Σ *of size* $O(r)$*. Given the positional Burrows–Wheeler transform for position* $x \in [1..c-1]$*, that is, arrays* $\mathtt{a}_x[1..r]$ *and* $\mathtt{d}_x[1..r]$*, arrays* $\mathtt{a}_{x+1}[1..r]$ *and* $\mathtt{d}_{x+1}[1..r]$ *can be computed in* $O(r)$ *time.*

Finally, one last data structure that we use in this paper indexes an integer array to later find out the minimum value inside a given range of array positions.

▶ **Lemma 6** (Semi-dynamic range minimum query data structure [8, Lemma 7])**.** *There exists a data structure that maintains an integer array* $\mathtt{I}[1..c]$ *and supports: the append query, which adds a new element to the end of* $\mathtt{I}$ *and increments* c*, in* $O(1)$ *amortized time; and the Range Minimum Query, which for any given* $x, y \in [1..c]$ *with* $x \leqslant y$*, computes a position* $k \in [1..c]$ *such that* $\mathtt{I}[k] = \min\{Q[\ell] : x \leqslant \ell \leqslant y\}$*, in* $O(1)$ *time.*

4 Solutions

In Section 4.1 we remark useful properties of gapless MSAs. Then, in Sections 4.2 and 4.3 we provide linear-time solutions to min-U-cardinality and min-L-size for MSAs under this setting.

4.1 Basic properties and non-trivial settings

Before investigating under which setting problems min-U-cardinality and min-L-size are trivial, consider the following properties of MSAs without gaps that we will exploit in this section.

▶ **Observation 7** (Monotonicity of left extensions [22])**.** *Given* $\mathsf{MSA}[1..r, 1..c]$*, for any* x, y *with* $1 \leqslant x \leqslant y \leqslant c$ *we say that* $[x..y]$ *is a* left extension *of suffix* $\mathsf{MSA}[1..r, y+1..c]$*. If the MSA is* gapless*, the following monotonicity property holds for any left extension* $[x..y]$ *of* $\mathsf{MSA}[1..r, y+1..c]$*:*

$$r \geqslant \mathrm{H}([x'..y]) \geqslant \mathrm{H}([x..y]) \qquad \forall x' < x.$$

▶ **Observation 8.** *Given a* gapless $\mathsf{MSA}[1..r, 1..c]$*, the (gap-aware) size of any segment* $[x..y]$ *is equal to the height times the length of the segment, in symbols* $\mathrm{N}_\varepsilon([x..y]) = \mathrm{N}([x..y]) = |x..y| \cdot \mathrm{H}([x..y])$*. Indeed, if the MSA does not contain the gap symbol then* $\mathrm{spell}(\mathsf{MSA}[i, x..y]) = \mathsf{MSA}[i, x..y]$ *for all* $i \in [1..r]$ *and all strings spelled by segment* $[x..y]$ *have length* $y - x + 1$*.*

The simplest version of min-L-size occurs when the MSA contains no gap symbol and there is no lower bound on the segment length, that is, when $L = 1$. Intuitively, this setting presents the same trivial solution as in the founder reconstruction problem (see [25, Theorem 1]), since transforming each column into its own segment achieves the best possible compression. Let $[x]$ denote the singleton range $[x..x] = \{x\}$.

▶ **Theorem 9.** *An optimal solution to min-*1*-size for a* gapless $\mathsf{MSA}[1..r, 1..c]$ *is the trivial segmentation* $S^{|||} = [1], [2], \ldots, [c]$*.*

Proof. Consider any segment $[x..y]$ with $|[x..y]| > 1$. Then $[x..y-1]$ and $[y]$ are also valid segments, and since the MSA is gapless

$$\begin{aligned} \mathrm{N}[x..y] &= |[x..y-1]| \cdot \mathrm{H}([x..y]) + 1 \cdot \mathrm{H}([x..y]) && \text{(Observation 8)} \\ &\geqslant |[x..y-1]| \cdot \mathrm{H}([x..y-1]) + |[y]| \cdot \mathrm{H}([y]) && \text{(Observation 7)} \\ &= \mathrm{N}([x..y-1]) + \mathrm{N}([y]). \end{aligned}$$

Thus breaking down segments longer than 1 never increases the size and the (gap-aware) size of $S^{|||}$ is less than or equal to the size of any other segmentation. ◀

However, the trivial solution is not valid if $L > 1$ or it can be suboptimal if the MSA contains gaps: in the former setting, greedily picking segments of length exactly L can be suboptimal, and in the latter case each empty string ε contributes 1 to the total size of the segmentation so the parts of the MSA with long runs of gaps need to be carefully considered. See Figure 2.

Regarding the problem of minimizing the cardinality of the resulting segmentation, min-U-card behaves differently than the other segmentation problems, as it does not admit a trivial optimal segmentation even if there is no upper bound on the segment length (i.e. $U = c$) and the MSA does not have any gap symbol.

▶ **Observation 10.** *Trivial segmentations* $S^{|||} = [1], [2], \ldots, [c]$ *and* $S^{\equiv} = [1..c]$ *are not optimal with respect to min-c-card for some* gapless $\mathsf{MSA}[1..r, 1..c]$*: on one hand, in regions of high similarity longer segments can be preferable as they can spell very few strings; on the other hand, shorter segments generate less strings in regions of high diversity. See the example* $\mathsf{MSA}[1..10, 1..4]$ *in Figure 3.*

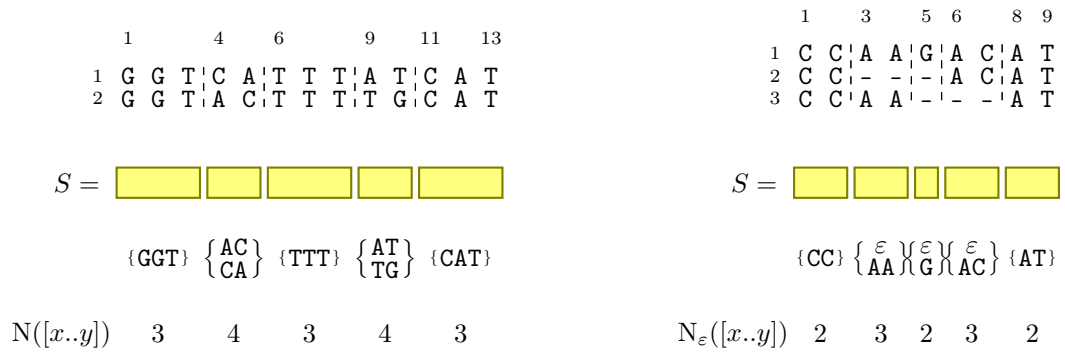

Figure 2 On the left, a gapless MSA where the optimal solution to min-2-size (i.e. the lower bound on the minimum segment length is 2) has size 17 and is non-trivial. On the right, an MSA with gaps for which the optimal solution to min-1-size is non-trivial and has gap-aware size 12.

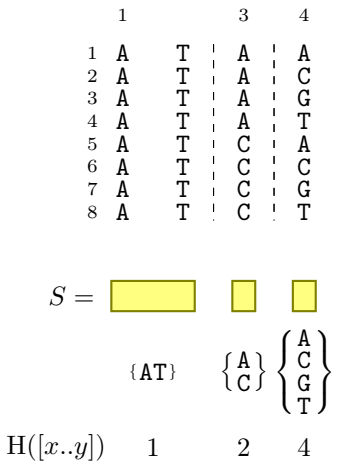

Figure 3 Example of a gapless MSA$[1..8, 1..4]$ for which the trivial segmentations $S^{\equiv} = [1..c] = [1..4]$ and $S^{|||} = [1], [2], [3], [4]$ are not a solution to min-4-size, since their cardinality is equal to 8. Instead, $S = [1..2], [3], [4]$ (shown) is such that $\mathrm{m}(S) = 7$.

4.2 Minimizing the cardinality

Given $\mathsf{MSA}[1..r, 1..c]$ and $U \in [1..c]$, for any $y \in [1..c]$ we define m_y as the size of an optimal solution of min-U-cardinality on instance $\mathsf{MSA}[1..r, 1..y]$ respecting segment upper bound U. Then, the following recursion holds, since the cardinality of a segmentation S is the sum of the individual cardinalities of its segments:

$$\mathrm{m}_0 = 0,$$
$$\mathrm{m}_y = \min_{x \in [0..y-1]} \Big(\mathrm{m}_x + \mathrm{H}([x+1..y]) \Big) \qquad \forall y \in [1..c], \tag{1}$$

and it is easy to see that m_c is equal to the cardinality $\mathrm{m}(S)$ of an optimal segmentation S.

Norri et al. [22] recognized that, at least for gapless MSAs, all changes in segment height can be described compactly and in a range fashion by fixing an end column y and considering values $\mathrm{H}([x..y])$ for $x \in [1..y]$. This concept was restated and extended to MSAs with gaps by Rizzo et al. [24] into the *extensions* of the MSA suffix $\mathsf{MSA}[1..r, y+1..c]$ that are *meaningful*. Consider the following definition alongside Figure 4.

▶ **Definition 11** (Meaningful left extensions [22, 24]). *Given* $\mathsf{MSA}[1..r, 1..c]$*, let* $L, U \in [1..c]$ *be some given lower bound and upper bound on the segment length*[1]*, with* $L \leqslant U$*. For any* $y \in [1..c]$ *we denote with* $\mathcal{L}_y = \ell_{y,1}, \ell_{y,2}, \ldots, \ell_{y,d_y}$ *the* meaningful left extensions *of* $\mathsf{MSA}[1..r, y+1..c]$*, meaning the strictly decreasing sequence of all positions smaller than or equal to* y *such that:*

1. $y - U + 1 \leqslant \ell_{y,d_y} < \cdots < \ell_{y,2} < \ell_{y,1} = y - L + 1$*, so that* $\mathcal{L}_y$ *captures the left extensions of* $\mathsf{MSA}[1..r, y+1..c]$ *of length at least* L *and at most* U*; and*
2. $\mathrm{H}([\ell_{y,j}..y]) \neq \mathrm{H}([\ell_{y,j}+1..y])$ *for* $2 \leqslant j \leqslant d_y$*, so that each* $\ell_{y,j}$ *marks a column where the height of the left extension changes.*

If $y < L$*, i.e.* $\mathsf{MSA}[1..r, y+1..c]$ *has no left extension of size at least* L*, we define* $\mathcal{L}_y = ()$ *and* $d_y = 0$*. Otherwise, for completeness, we define* $\ell_{y,d_y+1} = \max(0, y - U)$ *(see how this is used later in Equation* (2)*).*

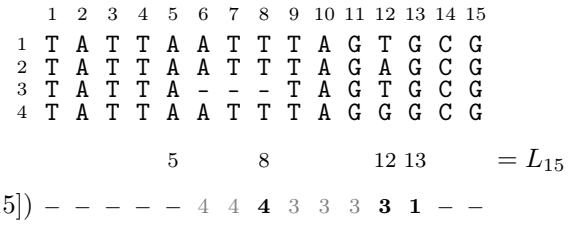

Figure 4 Example of $\mathsf{MSA}[1..4, 1..15]$ and of meaningful left extensions $\mathcal{L}_{15} = 13, 12, 8$ (represented from right to left in the figure), given lower bound $L = 3$ and upper bound $U = 10$.

The meaningful left extensions $\mathcal{L}_y$ are indeed a compact description of how the height of $[x..y]$ changes when fixing y and moving x: all segments $[x..y]$ with $x \in [\ell_{y,j+1}+1..\ell_{y,j}]$ have height $\mathrm{H}([\ell_{y,j}..y])$; if the monotonicity of left extensions (Observation 7) holds, then $d_y \leqslant r$,

[1] For simplicity, the definition accepts both a lower bound and an upper bound on the segment length, even though min-U-cardinality and min-L-size require only one of the two. If either is unspecified, we simply assume that $L = 1$ or $U = c$ accordingly, that is, there is no lower or upper bound.

that is, $|\mathcal{L}_y|$ is at most r for each $y \in [1..c]$, and the cumulative count of all meaningful left extensions and their height values is $O(rc)$. Regardless of the number of meaningful left extensions, Equation (1) can be rewritten as

$$\mathrm{m}_y = \min_{j \in [1..d_y]} \Big(\mathrm{H}([\ell_{y,j}..y]) + \min_{x \in [\ell_{y,j+1}+1..\ell_{y,j}]} \mathrm{m}_{x-1} \Big), \tag{2}$$

where m_{x-1} is used due to the different meaning of x in Equation (2) and Definition 11. Given pairs $(\ell_{y,j}, \mathrm{H}([\ell_{y,j}..y]))$ in input for each $y \in [1..c]$, we can compute values m_y in a dynamic programming fashion and we can solve operation $\min_{x \in [\ell_{y,j+1}+1..\ell_{y,j}]} \mathrm{m}_{x-1} = \min_{x \in [\ell_{y,j+1}..\ell_{y,j}-1]} \mathrm{m}_x$ by indexing the array of values m_y for Range Minimum Queries with Lemma 6, as shown in Algorithm 1.

Algorithm 1 Segmentation of $\mathsf{MSA}[1..r, 1..c]$ with segment-length upper bound U minimizing the cardinality. Note that bound U is implicitly used in value $\ell_{y,d_y+1} = \max(0, y - U)$ (see Definition 11 and Equation (2)).

input : integers $r, c \in \mathbb{N}$,
segment length upper bound $U \in [1..c]$, and
meaningful left extensions $(\ell_{y,j}, h_{y,j})$ for $y \in [1..c]$, $j \in [1..d_y]$
output : minimum cardinality of a segmentation $S_1, \ldots, S_k$ such that $|S_i| \leqslant U$ for $i \in [1..k]$

```
Create array m[0..c] holding values in [0..rc];
Preprocess m for semi-dynamic Range Minimum Queries;
m[0] ← 0;
for y ← 1 to c do
    m[y] ← +∞;
    for j ← 1 to d_y do
        x ← RangeMinQuery(m, [ℓ_{y,j+1}..ℓ_{y,j} − 1]);
        m[y] ← min(m[y], h_{y,j} + m[x]);
    end
    Update m for Range Minimum Queries over incremented range [1..y];
end
return m[c]
```

Finally, for gapless MSAs the meaningful left extensions can be computed efficiently with a modification of the positional Burrows–Wheeler Transform (Definition 4). Norri et al. correctly realized that the pBWT for position y already describes $\mathcal{L}_y$ and all changes in height [22, Lemma 4] and modified the pBWT to obtain values $(\ell_{y,j}, \mathrm{H}([\ell_{y,j}..y]))$ in $O(r)$ time per each column y.

▶ **Lemma 12** ([22, Lemmas 5 and 6]). *Let $\mathsf{MSA}[1..r, 1..c]$ be a gapless multiple sequence alignment over alphabet Σ, with $|\Sigma| \in O(r)$. The meaningful left extensions $\mathcal{L}_y = \ell_{y,1}, \ldots, \ell_{y,d_y}$ and their corresponding segment height values $\mathrm{H}([\ell_{y,j}..y])$ for $j \in [1..d_y]$ can be computed for $y = 1, \ldots, c$ in $O(rc)$ time and in $O(r + c)$ working space. Specifically, the values are computed column-wise in a streaming fashion, from $y = 1$ to $y = c$, and the space bound does not include storing the computed values.*

By interleaving the computation from Lemma 12 and Algorithm 1 we obtain a linear-time solution to min-U-cardinality in the case with no gap.

▶ **Theorem 13.** *Let* $\mathsf{MSA}[1..r, 1..c]$ *be a gapless multiple sequence alignment over alphabet* Σ*, with* $|\Sigma| \in O(r)$*, and* $U \in [1..c]$ *be an upper bound on the maximum segment length. We can solve* min-U-cardinality *(Problem 2) in* $O(rc)$ *time and* $O(c + r)$ *space by finding the optimal segmentation* S *respecting* U *and minimizing the cardinality* $\mathrm{m}(S)$*.*

Proof. The final algorithm is provided by combining Algorithm 1, that utilizes the meaningful left extensions $\mathcal{L}_y$ and their height values from $y = 1$ to $y = c$, with Lemma 12, that computes and can provide pairs $(\ell_{y,j}, \mathrm{H}([\ell_{y,j}..y]))$ in the same order. The correctness of Algorithm 1 follows from Equation (2). Since the MSA is gapless, the total number of iterations of the inner for-loop in Algorithm 1 is $O(rc)$ due to the monotonicity of left extensions (Observation 7), and by using the Range Minimum Query data structure from Lemma 6 on array `m` we obtain the stated time and space complexity. Algorithm 1 can be augmented with standard backtracking techniques to obtain the actual segmentation, as shown by Rizzo et al. [24, Section 4.5]. ◀

4.3 Minimizing the size

Similarly to Section 4.2, given $\mathsf{MSA}[1..r, 1..c]$ and $L \in [1..c]$, for any $y \in [1..c]$ we define N_y as the size of an optimal solution of min-L-size on instance $\mathsf{MSA}[1..r, 1..y]$ respecting lower bound L. Then the following recursion holds, since the gap-aware size of a segmentation S is the sum of the individual gap-aware segment sizes:

$$\mathrm{N}_0 = 0,$$
$$\mathrm{N}_y = \min_{x \in [0..y-1]} \Big(\mathrm{N}_x + \mathrm{N}_\varepsilon([x+1..y]) \Big) \qquad y \in [1..c], \tag{3}$$

and it is easy to see that N_c is equal to the gap-aware size of an optimal segmentation.

We now concentrate on gapless MSAs. Recall that, in such setting, segment size and gap-aware segment size coincide, and they are equal to the length of the segment times its height (Observation 8). Also, consider how the meaningful left extensions (Definition 11) are a compact description of all possible segment heights, as discussed in Section 4.2. Then, in the gapless setting, Equation (3) for $y \in [1..c]$ can be rewritten as

$$\begin{aligned}
\mathrm{N}_y &= \min_{x \in [0..y-1]} \Big(\mathrm{N}_x + \mathrm{H}([x+1..y]) \cdot (y - x) \Big) && \text{(Observation 8)} \\
&= \min_{x \in [0..y-1]} \Big(\mathrm{N}_x - \mathrm{H}([x+1..y]) \cdot x + \mathrm{H}([x+1..y]) \cdot y \Big) \\
&= \min_{j \in [1..d_y]} \Big(\mathrm{H}([\ell_{y,j}..y]) \cdot y + \min_{x \in [\ell_{y,j+1}..\ell_{y,j}-1]} \Big(\mathrm{N}_x - \mathrm{H}([\ell_{y,j}..y]) \cdot x \Big) \Big), && (4)
\end{aligned}$$

where the last step holds because the meaningful left extensions $\mathcal{L}_y$ partition by construction range $[1..y - L + 1]$ according to the segment height $\mathrm{H}([x..y])$ for variable x, or in other words, $\mathrm{H}([x+1..y]) = \mathrm{H}([\ell_{y,j}..y])$ for all $x + 1 \in [\ell_{y,j+1} + 1..\ell_{y,j}]$, $j \in [1..d_y]$. Equation (4) is suitable for an efficient dynamic programming approach, shown in Algorithm 2:

- value $\mathrm{H}([\ell_{y,j}..y]) \cdot y$ inside the outer min operator depends on y and on the height considered (i.e. the meaningful left extension), as in Equation (2) from Section 4.2;
- the addends in the internal min operation depend on x and on the segment height $\mathrm{H}([\ell_{y,j}..y]) \in [1..r]$, and since the number of possible height values is $O(r)$ we can store and maintain these recursive values for each column $x \in [1..y-1]$ and each height $h \in [1..r]$.

Algorithm 2 Segmentation of $\mathsf{MSA}[1..r, 1..c]$ with segment-length lower bound L minimizing the cardinality. Note that bound L is implicitly used in value $\ell_{y,1} = y - L + 1$ (see Definition 11).

input : integers $r, c \in \mathbb{N}$,
segment length lower bound $L \in [1..c]$, and
meaningful left extensions $(\ell_{y,j}, h_{y,j})$ for $y \in [1..c]$, $j \in [1..d_y]$
output : minimum size of a segmentation $S_1, \ldots, S_k$ such that $|S_i| \leqslant U$ for $i \in [1..k]$
Initialize array $\mathtt{N}[1..r]$ with values in $[0..rc]$;
Initialize matrix $\mathtt{M}[1..r, 0..c]$ with values in $[0..rc]$;
Preprocess the rows of $\mathtt{M}$ for semi-dynamic Range Minimum Queries;
for $h \leftarrow 1$ **to** r **do**
  $\mathtt{M}[h, 0] \leftarrow 0$;
end
for $y \leftarrow 1$ **to** c **do**
  $\mathtt{N}[y] \leftarrow +\infty$;
  for $j \leftarrow 1$ **to** d_y **do**
    $x \leftarrow \text{RangeMinQuery}\big(\mathtt{M}[h_{y,j}], [\ell_{y,j+1}..\ell_{y,j} - 1]\big)$;
    $\mathtt{N}[y] \leftarrow \min\Big(\mathtt{N}[y], \quad h_{y,j} \cdot y + \mathtt{M}[h_{y,j}, x]\Big)$;
  end
  for $h \leftarrow 1$ **to** r **do**
    $\mathtt{M}[h, y] \leftarrow \mathtt{N}[y] - h \cdot y$;
    Update the Range Minimum Query data structure of row h to cover
     incremented range $\mathtt{M}[h, 1..y]$;
  end
end
return $\mathtt{N}[c]$

▶ **Theorem 14.** *Given a* gapless $\mathsf{MSA}[1..r, 1..c]$ *and a lower bound* $L \in [1..c]$ *on the minimum segment length, we can solve* min-L-size *(Problem 3) in* $O(rc)$ *time and* $O(rc)$ *space by finding the optimal segmentation* S *respecting* L *and minimizing the size* $\mathrm{N}(S)$.

Proof. The final algorithm takes the output of Lemma 12, pairs $(\ell_{y,j}, h_{y,j})$ for $y \in [1..c]$ and $j \in [1..d_y]$, and directly runs Algorithm 2, whose correctness follows from Equation (4). The total number of iterations of the two inner for-loops is $O(rc)$, since the number of meaningful left extensions is $O(rc)$ (Observation 7) and the possible height values are in the range $[1..r]$. The stated time and space complexity follows by applying Lemmas 6 and 12. Similarly to Algorithm 1, Algorithm 2 can be augmented with standard backtracking techniques to obtain the actual segmentation, as shown in [24, Section 4.5]. ◀

5 Future work

This initial study on EDS construction opens many interesting directions for further study:

Gaps as symbols yield upper bounds Even though Equations (2) and (3) (but not Equation (4)) hold for MSAs with gaps, Theorems 13 and 14 work only on MSAs without gaps. A straightforward way to extend them is to treat them as normal symbols in the alphabet. After segmentation, the gap symbols can be removed to obtain the final EDS. With all the quality measures we proposed, this works as an upper bound: for example, consider the height of a segment that contains `A-` and `-A`; height is reduced by one after gaps are removed. Thus, the removal of gaps cannot increase the height, and it is of interest to study experimentally how much the measures change after gap removal.

Gaps as symbols yield exact solutions? For height optimization, it seems possible to characterize the class of MSAs with gaps where the approach above yields an optimal solution, and not just an upper bound. Such characterization should be possible through forbidding local sub-optimal alignments like `A-` and `-A`. This raises several natural questions: What is the exact form of the characterization? How to efficiently detect if a given MSA is part of this class? Do all optimal alignments under some column-wise scoring function belong to this class? Are there efficient algorithms to convert an MSA into one in this class without sacrificing the alignment score?

Tailored algorithms with gaps The presented Algorithms 1 and 2 do not find the optimal segmentation of arbitrary MSAs, so it is natural to ask if there are efficient algorithms to directly optimize the studied measures on general MSAs. Alternatively, there may be slight adjustments to the measures that are amenable to efficient algorithms: in a different context, Rizzo et al. [24] introduced the metric of prefix-aware height, which was optimized in linear time, and also optimized in linear time the metric of maximum segment length.

References

1 Mai Alzamel, Lorraine A. K. Ayad, Giulia Bernardini, Roberto Grossi, Costas S. Iliopoulos, Nadia Pisanti, Solon P. Pissis, and Giovanna Rosone. Comparing degenerate strings. *Fundam. Informaticae*, 175(1-4):41–58, 2020. doi:10.3233/FI-2020-1947.

2 Rocco Ascone, Giulia Bernardini, Alessio Conte, Massimo Equi, Estéban Gabory, Roberto Grossi, and Nadia Pisanti. A unifying taxonomy of pattern matching in degenerate strings and founder graphs. In Solon P. Pissis and Wing-Kin Sung, editors, *24th International Conference on Algorithms in Bioinformatics, WABI*, volume 312 of *LIPIcs*, pages 14:1–14:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.WABI.2024.14.

3 Giulia Bernardini, Estéban Gabory, Solon P. Pissis, Leen Stougie, Michelle Sweering, and Wiktor Zuba. Elastic-degenerate string matching with 1 error or mismatch. *Theory Comput. Syst.*, 68(5):1442–1467, 2024. doi:10.1007/S00224-024-10194-8.

4 Giulia Bernardini, Pawel Gawrychowski, Nadia Pisanti, Solon P. Pissis, and Giovanna Rosone. Elastic-degenerate string matching via fast matrix multiplication. *SIAM J. Comput.*, 51(3):549–576, 2022. doi:10.1137/20M1368033.

5 Giulia Bernardini, Nadia Pisanti, Solon P. Pissis, and Giovanna Rosone. Pattern matching on elastic-degenerate text with errors. In Gabriele Fici, Marinella Sciortino, and Rossano Venturini, editors, *String Processing and Information Retrieval - 24th International Symposium, SPIRE 2017, Palermo, Italy, September 26-29, 2017, Proceedings*, volume 10508 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 2017. doi:10.1007/978-3-319-67428-5_7.

6 Giulia Bernardini, Nadia Pisanti, Solon P. Pissis, and Giovanna Rosone. Approximate pattern matching on elastic-degenerate text. *Theor. Comput. Sci.*, 812:109–122, 2020. doi:10.1016/J.TCS.2019.08.012.

7 Jorge Avila Cartes, Paola Bonizzoni, Simone Ciccolella, Gianluca Della Vedova, and Luca Denti. Pangeblocks: customized construction of pangenome graphs via maximal blocks. *BMC Bioinform.*, 25(1):344, 2024. doi:10.1186/S12859-024-05958-5.

8 Bastien Cazaux, Dmitry Kosolobov, Veli Mäkinen, and Tuukka Norri. Linear time maximum segmentation problems in column stream model. In Nieves R. Brisaboa and Simon J. Puglisi, editors, *String Processing and Information Retrieval - 26th International Symposium, SPIRE 2019, Segovia, Spain, October 7-9, 2019, Proceedings*, volume 11811 of *Lecture Notes in Computer Science*, pages 322–336. Springer, 2019. doi:10.1007/978-3-030-32686-9_23.

9 The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, October 2016. doi:10.1093/bib/bbw089.

10 Richard Durbin. Efficient haplotype matching and storage using the positional burrows-wheeler transform (PBWT). *Bioinform.*, 30(9):1266–1272, 2014. doi:10.1093/BIOINFORMATICS/BTU014.

11 Massimo Equi, Veli Mäkinen, and Alexandru I. Tomescu. Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless SETH fails. *Theor. Comput. Sci.*, 975:114128, 2023. doi:10.1016/J.TCS.2023.114128.

12 Massimo Equi, Veli Mäkinen, Alexandru I. Tomescu, and Roberto Grossi. On the complexity of string matching for graphs. *ACM Trans. Algorithms*, 19(3):21:1–21:25, 2023. doi:10.1145/3588334.

13 Massimo Equi, Tuukka Norri, Jarno Alanko, Bastien Cazaux, Alexandru I. Tomescu, and Veli Mäkinen. Algorithms and complexity on indexing founder graphs. *Algorithmica*, 85(6):1586–1623, 2023. doi:10.1007/S00453-022-01007-W.

14 Esteban Gabory, Moses Njagi Mwaniki, Nadia Pisanti, Solon P. Pissis, Jakub Radoszewski, Michelle Sweering, and Wiktor Zuba. Pangenome comparison via ed strings. *Frontiers Bioinform.*, 4, 2024. doi:10.3389/fbinf.2024.1397036.

15 Estéban Gabory, Njagi Moses Mwaniki, Nadia Pisanti, Solon P. Pissis, Jakub Radoszewski, Michelle Sweering, and Wiktor Zuba. Comparing elastic-degenerate strings: Algorithms, lower bounds, and applications. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPIcs*, pages 11:1–11:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CPM.2023.11.

16 Estéban Gabory, Njagi Moses Mwaniki, Nadia Pisanti, Solon P. Pissis, Jakub Radoszewski, Michelle Sweering, and Wiktor Zuba. Elastic-degenerate string comparison. *Inf. Comput.*, 304:105296, 2025. doi:10.1016/J.IC.2025.105296.

17 Roberto Grossi, Costas S. Iliopoulos, Chang Liu, Nadia Pisanti, Solon P. Pissis, Ahmad Retha, Giovanna Rosone, Fatima Vayani, and Luca Versari. On-line pattern matching on similar texts. In *Proc. CPM 2017*, volume 78 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.CPM.2017.9.

18 Costas S. Iliopoulos, Ritu Kundu, and Solon P. Pissis. Efficient pattern matching in elastic-degenerate strings. *Inf. Comput.*, 279:104616, 2021. doi:10.1016/J.IC.2020.104616.

19 David Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2):322–336, April 1978. doi:10.1145/322063.322075.

20 Veli Mäkinen and Tuukka Norri. Applying the positional Burrows–Wheeler transform to all-pairs Hamming distance. *Inf. Process. Lett.*, 146:17–19, 2019. doi:10.1016/J.IPL.2019.02.003.

21 Njagi Moses Mwaniki and Nadia Pisanti. Optimal sequence alignment to ED-strings. In Mukul S. Bansal, Zhipeng Cai, and Serghei Mangul, editors, *Bioinformatics Research and Applications - 18th International Symposium, ISBRA 2022, Haifa, Israel, November 14-17, 2022, Proceedings*, volume 13760 of *Lecture Notes in Computer Science*, pages 204–216. Springer, 2022. doi:10.1007/978-3-031-23198-8_19.

22 Tuukka Norri, Bastien Cazaux, Dmitry Kosolobov, and Veli Mäkinen. Linear time minimum segmentation enables scalable founder reconstruction. *Algorithms Mol. Biol.*, 14(1):12:1–12:15, 2019. doi:10.1186/S13015-019-0147-6.

23 Cédric Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput. Biol.*, 3(8), 2007. doi:10.1371/JOURNAL.PCBI.0030123.

24 Nicola Rizzo, Massimo Equi, Tuukka Norri, and Veli Mäkinen. Elastic founder graphs improved and enhanced. *Theor. Comput. Sci.*, 982:114269, 2024. doi:10.1016/J.TCS.2023.114269.

25 Esko Ukkonen. Finding founder sequences from a set of recombinants. In Roderic Guigó and Dan Gusfield, editors, *Proceedings of 2nd International Workshop on Algorithms in Bioinformatics WABI*, volume 2452 of *Lecture Notes in Computer Science*, pages 277–286. Springer, 2002. doi:10.1007/3-540-45784-4_21.