The complexity of the Approximate Multiple Pattern Matching Problem for random strings

Frédérique Bassino
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, 99 av. J.-B. Clément, F-93430 Villetaneuse, France `https://lipn.univ-paris13.fr/~bassino/`
bassino@lipn.univ-paris13.fr

Tsinjo Rakotoarimalala
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, 99 av. J.-B. Clément, F-93430 Villetaneuse, France `https://lipn.univ-paris13.fr/~rakotoarimalala/`
rakotoarimalala@lipn.univ-paris13.fr

Andrea Sportiello
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS UMR 7030, 99 av. J.-B. Clément, F-93430 Villetaneuse, France `https://lipn.univ-paris13.fr/~sportiello/`
sportiello@lipn.univ-paris13.fr

Abstract

We describe a multiple string pattern matching algorithm which is well-suited for approximate search and dictionaries composed of words of different lengths. We prove that this algorithm has optimal complexity rate up to a multiplicative constant, for arbitrary dictionaries. This extends to arbitrary dictionaries the classical results of Yao [SIAM J. Comput. 8, 1979], and Chang and Marr [Proc. CPM94, 1994].

2012 ACM Subject Classification Formal languages and automata theory; Design and analysis of algorithms

Keywords and phrases Average-case analysis of algorithms, String Pattern Matching, Computational Complexity bounds.

Digital Object Identifier 10.4230/LIPIcs.AOFA.2020.24

Funding *Andrea Sportiello*: Supported by the French ANR-MOST MetAConC.

1 The problem

1.1 Definition of the problem

Let Σ be an alphabet of s symbols, $\xi = \xi_0 \dots \xi_{n-1} \in \Sigma^n$ a word of n characters (the input *text string*), $D = \{w_1, \dots, w_\ell\}$, $w_i \in \Sigma^*$ a collection of words (the *dictionary*). We say that $w = x_1 \dots x_m$ occurs in ξ with final position j if $w = \xi_{j-m+1}\xi_{j-m+2} \cdots \xi_j$. We say that w occurs in ξ with final position j, with no more than k errors, if the letters $x_1, \dots, x_m$ can be aligned to the letters $\xi_{j-m'}, \dots, \xi_j$ with no more than k errors of insertion, deletion or substitution type, i.e., it has *Levenshtein distance* at most k to the string $\xi_{j-m'} \dots \xi_j$ (see an example in Figure 1). Let $r_m(D)$ be the number of distinct words of length m in D. We call $\boldsymbol{r}(D) = \{r_m(D)\}_{m \geq 1}$ the *content* of D, a notion of crucial importance in this paper.

The *approximate multiple string pattern matching problem* (AMPMP), for the datum (D, ξ, k), is the problem of identifying all the pairs (a, j) such that $w_a \in D$ occurs in ξ with final position j, and no more than k errors (cf. Figure 1). This is a two-fold generalisation of the classical *string pattern matching problem* (PMP), for which the exact search is considered, and the dictionary consists of a single word.

A precise historical account of this problem, and a number of theoretical facts, are presented in Navarro's review [8]. The first seminal works have concerned the PMP. Results

31st International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AOFA 2020).
Editors: Michael Drmota and Clemens Heuberger; Article No. 24; pp. 24:1–24:15
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

```
                 5    10   15   20   25   30   35   40   45   50   55
         ....|....|....|....|....|....|....|....|....|....|....|.
         now.is.the.winter.of.our.discontent.made.glorious.summer
sour     ....................Xour.....................XouX.......
intent   ............inteXX...........Xntent.....................
galore   .........................................glorX..........
therein  .......theSSin..........................................
```

soul: 23, 48; intent: 16, 35; galore: 45; therein: 14.

Figure 1 Typical output of an approximate multiple string pattern matching problem, on an English text (alphabet of 26 characters plus the space symbol .). In this case $k = 2$ and $\boldsymbol{r}(D) = (0, 0, 0, 0, 1, 0, 2, 1, 0, \ldots)$. The symbols D, S and i stand for deletion, substitution and insertion errors, while X corresponds to an insertion or a substitution.

included the design of efficient algorithms (notably Knuth–Morris–Pratt and Boyer–Moore), and have led to the far-reaching definition of the Aho–Corasick automata [1, 3, 7, 11]. In particular, Yao [11] is the first paper that provide rigorous bounds for the complexity of PMP in random texts. To make a long story short, it is argued that an interesting notion of complexity is the asymptotic average fraction of text that needs to be accessed (in particular, at least at this stage, it is *not* the time complexity of the algorithm), and is of order $\ln(m)/m$ for a word of length m. The first works on approximate search, yet again for a single word (APMP), are the description of the appropriate data structure, in [10, 4], and, more relevant to our aims here, the derivation of rigorous complexity bounds in Chang and Marr [5]. Yet again in simplified terms, if we allow for k errors, the complexity result of Yao is deformed into order $[\ln(m)+k]/m$. More recent works have concerned the case of dictionaries composed of several words, all of the same length [9],[1] however, also at the light of unfortunate flaws in previous literature, the rigorous derivation of the average complexity for the MPMP has been missing even in the case of words of the same length, up to our recent paper [2], where it is established that the Yao scaling $\ln(m)/m$ is (roughly) modified into $\max_m \ln(mr_m)/m$ (a more precise expression is given later on). By combining the formula of Chang and Marr for APMP, and our formula for MPMP, it is thus natural to expect that the AMPMP may have a complexity of the order $\max_m[\ln(mr_m) + k]/m$. This paper has the aim of establishing a result in this fashion.

Of course, the present work uses results, ideas and techniques already presented in [2], for the PMPM. A main difference is that in [2] we show that, for any dictionary, a slight modification of an algorithm by Fredriksson and Grabowski [6] is optimal within a constant, while this is not true anymore for approximate search with Levenshtein distance (we expect that it remains optimal for approximate search in which only substitution errors are allowed, although we do not investigate this idea here). As a result, we have to modify this algorithm more substantially, by combining it with the algorithmic strategy presented in Chang and Marr [5], and including one more parameter (to be tuned for optimality). This generalised algorithm is presented in Section 2.2.

Also, a large part of our work in [2] is devoted to the determination of a relatively tight lower bound, while the determination of the upper bound consists of a simple complexity analysis of the Fredriksson–Grabowski algorithm. Here, instead, we will make considerable efforts in order to determine an upper bound for the complexity of our algorithm, which is

[1] This is the reason why, before our paper [2], which deals with dictionaries having words of different length, the forementioned notion of "content" of a dictionary did not appear in the literature.

	exact	approximate
single word	$C_{\text{Yao}} \dfrac{\ln m}{m}$ (Yao)	$C_{\text{CM}} \dfrac{\ln m + k}{m}$ (Chang and Marr)
dictionary	$C_1^{\text{ex}} \dfrac{1}{m_{\min}} + C_2^{\text{ex}} \max\limits_m \dfrac{\ln(smr_m)}{m}$	$\dfrac{C_1 k + C_1'}{m_{\min}} + C_2 \max\limits_m \dfrac{\ln(smr_m)}{m}$

Table 1 Summary of average complexities for exact and approximate search, for a single word or on arbitrary dictionaries. The results are derived from Yao [11], Chang and Marr [5], our previous paper [2], and the present paper, respectively.

the content of Section 2.4, while we will content ourselves of a rather crude lower bound, derived with small effort in Section 1.3 by combining the results of [5] and [2].

1.2 Complexity of pattern matching problems

In our previous paper [2] we have established a lower bound for the (exact search) multiple pattern matching problem, in terms of the size s of the alphabet, and the content $\boldsymbol{r} = \{r_m\}$ of the dictionary, involving the length $m_{\min}$ of the shortest word in the dictionary, and a function $\phi(\boldsymbol{r})$ with the specially simple structure $\phi(\boldsymbol{r}) = \max_m f(m, r_m)$. More precisely, calling $\Phi_{\text{aver}}(\boldsymbol{r})$ (resp. $\Phi_{\max}(\boldsymbol{r})$) the average over random texts, of the average (res. maximum) over dictionaries D of content $\boldsymbol{r}$, of the asymptotic fraction of text characters that need to be accessed, we have

▶ **Theorem 1** (Bassino, Rakotoarimalala and Sportiello, [2]). *Let $s \geq 2$ and $m_{\min} \geq 2$, and define $\kappa_s = 5\sqrt{s}$. For all contents $\boldsymbol{r}$, the complexity of the MPMP on an alphabet of size s satisfies the bounds*

$$\frac{1}{\kappa_s}\left(\phi(\boldsymbol{r}) + \frac{1}{2s\, m_{\min}}\right) \leqslant \Phi_{\text{aver}}(\boldsymbol{r}) \leqslant \Phi_{\max}(\boldsymbol{r}) \leqslant 2\left(\phi(\boldsymbol{r}) + \frac{1}{2s\, m_{\min}}\right), \tag{1}$$

where

$$\phi(\boldsymbol{r}) := \max_m \frac{1}{m} \ln(s\, m\, r_m)\,. \tag{2}$$

Note a relative factor $\ln s$ between the statement of the result above, and its original formulation in [2], due to a slightly different definition of complexity.

As we have anticipated, such a result is in agreement with the result of Yao [11], for dictionaries composed of a single word, which is simply of the form $\ln(m)/m$. Combining this formula with the complexity result for APMP, derived in Chang and Marr [5], it is natural to expect that the AMPMP has a complexity whose functional dependence on k and $\boldsymbol{r}$ is as in Table 1. Indeed, the bottom-right corner of the table is consistent both with the entry above it, and the entry at its left. Furthermore, it is easily seen that, up to redefining the constants, several other natural guesses would have this same functional form in disguise. Let us give some examples of this mechanism. Write $X \gtrless a_{\text{L/U}} Y + b_{\text{L/U}} Z$ as a shortcut for $a_{\text{L}} Y + b_{\text{L}} Z \leqslant X \leqslant a_{\text{U}} Y + b_{\text{U}} Z$. Now, suppose that we establish that $\Phi(\boldsymbol{r}, k) \gtrless a_{\text{L/U}}\,(k+1)/m_{\min} + b_{\text{L/U}}\,\max_m\,(\ln(mr_m) + k)/m$. Then we also have $\Phi(\boldsymbol{r}, k) \gtrless a'_{\text{L/U}}\,(k+1)/m_{\min} + b_{\text{L/U}}\,\max_m \ln(mr_m)/m$, with $a'_{\text{U}} = a_{\text{U}} + b_{\text{U}}$ (and all other constants unchanged). On the other side, if we have $\Phi(\boldsymbol{r}, k) \gtrless a_{\text{L/U}}\,(k+1)/m_{\min} + b_{\text{L/U}}\,\max_m \ln(mr_m)/m$, with $a_{\text{L}} > b_{\text{L}}$, then we also have $\Phi(\boldsymbol{r}, k) \gtrless a_{\text{L/U}}\,(k+1)/m_{\min} + b'_{\text{L/U}}\,\max_m(\ln(mr_m) + k)/m$, with $b'_{\text{L}} = a_{\text{L}} - b_{\text{L}}$.

The precise result that we obtain in this paper is the following:

▶ **Theorem 2.** *For the AMPMP, with k errors and a dictionary D of content $\{r_m\}$, the complexity rate $\Phi(D)$ is bounded in terms of the quantity*

$$\widetilde{\Phi}(D) := \frac{C_1(k+1)}{m_{\min}} + C_2 \max_m \frac{\ln(smr_m)}{m} \tag{3}$$

as

$$\frac{1}{C_1+\kappa_s C_2}\widetilde{\Phi}(D) \leqslant \Phi(D) \leqslant \widetilde{\Phi}(D)\,, \tag{4}$$

with $a = \ln(2s^2/(2s+1))$, $a' = \ln(4s^2-1)$, $\kappa_s = 5\sqrt{s}$ (as in Theorem 1) and

$$C_1 = \frac{a+2a'}{a}\,; \qquad\qquad C_2 = \frac{2(a+2a')}{aa'} = \frac{2}{a'}C_1\,. \tag{5}$$

1.3 The lower bound

Now, let us derive a lower bound of the functional form as in Table 1 for the AMPMP, by combining our results in [2] for the MPMP and the results in [5] for the APMP. Let us first observe a simple fact. Suppose that we have two bounds $A^{\mathrm{LB}}(\boldsymbol{r},k) \leqslant \Phi(\boldsymbol{r},k) \leqslant A^{\mathrm{UB}}(\boldsymbol{r},k)$ and $B^{\mathrm{LB}}(\boldsymbol{r},k) \leqslant \Phi(\boldsymbol{r},k) \leqslant B^{\mathrm{UB}}(\boldsymbol{r},k)$ (with $A^{\mathrm{LB}}(\boldsymbol{r},k)$ and $B^{\mathrm{LB}}(\boldsymbol{r},k)$ positive). Then, for all functions $p(\boldsymbol{r},k)$, valued in $[0,1]$, we have

$$p(\boldsymbol{r},k)A^{\mathrm{LB}}(\boldsymbol{r},k) + (1-p(\boldsymbol{r},k))B^{\mathrm{LB}}(\boldsymbol{r},k) \leqslant \Phi(\boldsymbol{r},k) \leqslant A^{\mathrm{UB}}(\boldsymbol{r},k) + B^{\mathrm{UB}}(\boldsymbol{r},k)\,.$$

We want to exploit this fact by using as bounds $A^{\mathrm{LB/UB}}(\boldsymbol{r},k)$ our previous result for the exact search, and as lower bound $B^{\mathrm{LB}}(\boldsymbol{r},k)$ the simple quantity $(k+1)/m_{\min}$. Then, later on, in Section 2, we will work on the determination of a bound $B^{\mathrm{UB}}(\boldsymbol{r},k)$ which has the appropriate form for our strategy above to apply. Let us discuss why $\Phi(\boldsymbol{r},k) \geq (k+1)/m_{\min}$. We will prove that this quantity is a bound to the minimal density of a *certificate*, over a single word of length $m = m_{\min}$, and text ξ. A certificate, as described in [11], is a subset $I \subseteq \{1,\dots,n\}$ such that, for the given text, the characters $\{\xi_i\}_{i\in I}$ imply that no occurrences of words of the dictionary may be possible, besides the ones which are fully contained in I. Some reflection shows that: (1) for the interesting case $m > k$, the smallest density $|I|/n$ of a certificate is realised on a *negative certificate*, that is, on a text ξ with no occurrences of the word w; (2) the smallest density is realised, for example, by the text $\xi = bbb\cdots b$, and the word $w = aaa\cdots a$; (3) in such a certificate, we must have read at least $k+1$ characters in every interval of size m, otherwise the alignment of w to this portion of text, in which we perform all the substitutions on the disclosed characters, would still be a viable candidate. Note in particular that deletion and insertion errors do not lead to higher lower bounds (although, for large m, they lead to bounds which are only slightly smaller).

As a result, recalling the expression for the lower bound in Theorem 1, by choosing $p(\boldsymbol{r},k)$ to satisfy $\frac{p}{1-p} = \frac{\kappa_s C_2}{C_1}$ we have

$$\Phi(\boldsymbol{r},k) \geq (1-p)\frac{k+1}{m_{\min}} + \frac{p}{\kappa_s}\phi(\boldsymbol{r}) = \frac{p}{\kappa_s C_2}\left(C_1\frac{k+1}{m_{\min}} + C_2\phi(\boldsymbol{r})\right) = \frac{C_1\frac{k+1}{m_{\min}} + C_2\phi(\boldsymbol{r})}{C_1+\kappa_s C_2}\,.$$

This proves the lower bound part of Theorem 2. Note that we could confine all the dependence from $\{r_m\}$ to the function ϕ (in particular, the choice $\frac{p}{1-p} = \frac{\kappa_s C_2}{C_1}$ only depends on the size of the alphabet s).

2 The (q, L) search algorithm, and the upper bound

2.1 Definition of alignment

We define a *partial alignment* α of the word $w = x_1 \dots x_m$ to the portion of text $\xi_{i_1} \dots \xi_{i_2}$, with k errors, and boundary parameters $(\varepsilon, \varepsilon') \in \mathbb{N}$, as the datum $\alpha = (w; i_1, i_2; \varepsilon, \varepsilon'; u)$, where u is a string in $\{C, S_a, D, I_a\}^*$, (these letters stand for *correct*, *substitution*, *deletion* and *insertion*, respectively, and the index a runs from 1 to s). Two integer parameters (for example i_2 and ε') are not independent, as they are deduced (say) from i_1, ε and the length of u. Indeed, say that the string u has m_C symbols C, m_D symbols D, m_S symbols of type S_a (for all a's altogether) and m_I symbols of type I_a, then

$$
\begin{aligned}
k &= m_S + m_D + m_I && \text{(number of errors)} \\
\varepsilon + \varepsilon' &= m - (m_C + m_S + m_D) && \text{(portion of the word on the sides)} \\
i_2 - i_1 + 1 &= m_C + m_S + m_I && \text{(length of the aligned portion of text)}
\end{aligned}
$$

The alignment has the following pattern (with a dash $-$ denoting a skipped character, in the text or in the word):

$$
\begin{array}{rc|ccccccccccc|c}
 & & i_1 & & & & & & & & & & i_2 & \\
\xi_{i_1} \cdots \xi_{i_2} = & \dots & \xi_{i_1} & \dots & w_j & \dots & - & \dots & a & \dots & a & \dots & \xi_{i_2} & \dots \\
u = & & & \dots & C & \dots & D & \dots & S_a & \dots & I_a & \dots & & \\
w = & \underbrace{w_1 \cdots w_\varepsilon}_{\varepsilon} & w_{\varepsilon+1} & \dots & w_j & \dots & w_{j'} & \dots & w_{j''} & \dots & - & \dots & w_{m-\varepsilon'} & \underbrace{w_{m-\varepsilon'+1} \cdots w_m}_{\varepsilon'}
\end{array}
$$

For example, if $w =$ `counteroffers`, in our reference text of Figure 1 we have the alignment $\alpha = (w; i_1, i_2; \varepsilon, \varepsilon'; u) = (w; 14, 24; 3, 1; u)$ with $k = 4$ and $u = CCCCI_{\square}CCS_{\square}S_{\mathtt{o}}I_{\mathtt{u}}C$, as indeed

$$
\begin{array}{rl}
 & \qquad\qquad\quad \underline{i_1} = 14 \qquad\qquad\qquad \underline{i_2} = 24 \\
\xi = & \cdots\mathtt{the.winter.of.our.discontent}\cdots \\
u = & \qquad\qquad\quad C\,C\,C\,C\,I_{\square}\,C\,C\,S_{\square}S_{\mathtt{o}}\,I_{\mathtt{u}}\,C \\
w = & \qquad\quad \underbrace{\mathtt{cou}}_{\varepsilon=3}\mathtt{nter{-}offe{-}r}\underbrace{\mathtt{s}}_{\varepsilon'=1}
\end{array}
$$

This example shows an important feature of this notion: several strings u may correspond to equivalent alignments among the same word and the same portion of text, and with the same offset ε. For example, the three last errors of $u = \cdots S_{\square}S_{\mathtt{o}}I_{\mathtt{u}}C$ can be replaced as in $u' = \cdots S_{\square}I_{\mathtt{o}}S_{\mathtt{u}}C$ or as in $u'' = \cdots I_{\square}S_{\mathtt{o}}S_{\mathtt{u}}C$. As the underlying idea in producing an upper bound from an explicit algorithm is to analyse the algorithm while using the union bound on the possible alignments, it will be useful to recognise classes of equivalent alignments, and, in the bound, 'count' just the classes, instead of the elements (we are more precise on this notion in Section 2.3).

We define a *full alignment* to be likewise a partial alignment, but with $\varepsilon = \varepsilon' = 0$. That is, the goal of any algorithm for the AMPMP is to output the list of (say) positions i_2 of the full alignments among the given text and dictionary. Note that we can always complete a partial alignment with k errors and boundary parameters $(\varepsilon, \varepsilon')$ to a full alignment with no more than $k + \varepsilon + \varepsilon'$ errors, and no less than k errors, by including substitution or insertion errors at the two sides.

We define a *c-block partial alignment* as the generalisation of the notion of partial alignment to the case in which the portion of text consists of c non-adjacent blocks. In this

```
                 | now.is.the.winter.of.our.discontent.made.glorious
                 |                  I□              D
deformed.pattern |               def ormed.pat-ten
                 |                  I□              Sn
                 |               def ormed.pat-tern
```

Figure 2 Typical outcome for the search of the pattern `deformed pattern` in our reference text. In this example $L = 3$ and $q = 12$, the number of full blocks is $c(\alpha) = 2$, and can be aligned to the disclosed portion of the text (denoted by underline) with $k = 3$ errors: one deletion on the first block, one insertion in the second block, and one deletion somewhere in between the two blocks. On the bottom line, another alignment of the same word, in which, instead of inserting the letter `r` in the second block, we have substituted `n` by `r`, still with $k = 3$. These two alignments are sufficiently different to contribute separately to our estimate of the complexity, within our version of the union bound (because the values of ε are different).

case, besides the natural alignment parameters ε, ε', and $i_{1,a}$, $i_{2,a}$, and u_a, for the blocks $a = 1, \dots, c$, we have $c-1$ parameters $\delta_a \in \mathbb{Z}$, associated to the offset between the alignment of the word to the blocks with index a and $a+1$. As a result, in order to extend a c-block partial alignment to a full alignment, we need to perform at least $-\delta_a$ further insertion errors, or $+\delta_a$ further deletion errors, depending on the sign of δ_a, for each of the $c-1$ intervals between the portions of text. That is, any c-block partial alignment α with k errors can be completed to a full alignment with no less than $k + \sum_a |\delta_a|$ errors.

Note that in the following we will *not* need to count all of the possible ways in which these deletions or insertions can be performed, as it may seem natural in a naïve perspective on the use of the union bound. This fact will allow us to efficiently bound the number of possible multi-block partial alignments arising in our algorithm analysis (instead of counting directly the possible full alignments, which would result in a too large bound).

2.2 The algorithm

Here we introduce an algorithm for AMPMP, concentrating on the pertinent notion of complexity, which is the ratio between the number of accesses to the text and the length of the text, and neglecting all implementation issues, and analysis of time complexity.

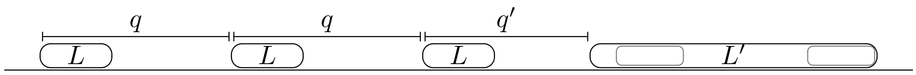
The algorithm is determined by two integers q and L, such that $k+1 \leqslant L < q \leqslant m_{\min} - k$. The emerging inequality $2k+1 < m_{\min}$ is not a limitation, as when this inequality is not satisfied we have to read a fraction $\Theta(1)$ of the text, and in this regime there is no point in showing that some algorithm can reach a complexity which is optimal up to a multiplicatve constant. When $L = 1$, the algorithm coincides with the one described by Fredriksson and Grabowski [6], and already analysed in detail in [2] for the MPMP. When we have a single word of length m, and q has the maximal possible value $q = m - k$, the algorithm coincides with the one used by Chang and Marr [5] for their proof of complexity of the APMP. As we will see in Section 2.5, choosing the optimal values of q and L for a given dictionary D (when the words are of different length) is not a trivial task.

Call the interval $\xi_{bq}\xi_{bq+1} \cdots \xi_{bq+L-1}$ of the text ξ the *b-th block of text*. The text is thus decomposed in a list of blocks of length L, and of intervals between the blocks, of length $q - L$. To every possible full alignment α of the word w to the text, are associated two integers: $c(\alpha)$ is the number of blocks which are fully contained in the alignment, and $b(\alpha)$ is the index of the rightmost of these blocks. Furthermore, we define $c(w)$ as the minimum of $c(\alpha)$ among the possible alignments involving w (indeed, it is either $c(\alpha) = c(w)$ for all α, or $c(\alpha) \in \{c(w), c(w)+1\}$ for all α, and, of course, at fixed q and L, $c(w)$ only depends on the length $|w|$ of the word).

Our algorithm accesses the text in three steps, namely, for every block index $b = 0, 1, \dots, \lceil n/q \rceil - 1$:

- We read all the characters ξ_i of the text, for $bq \leqslant i < bq + L$, that is we read the b-th block;
- We consider the possible c-block partial alignments α (with $c = c(\alpha)$) such that $b(\alpha) = b$, and associated to the intervals of text read so far. If any of these alignments is not excluded or determined positively, we read also the characters ξ_i for $i = bq - 1, bq - 2, \dots$, one by one, in this order, up to when all partial alignments are either excluded, or reach $\varepsilon = 0$. For a given instance of the problem, call $\mathcal{E}_L(b)$ (left-excess at block b) the set of positions of further characters that we need to access by this second step (with indices shifted so that the block starts at 1), and $e_L(b) = |\mathcal{E}_L(b)|$.
- If at the previous step we still have partial alignments which are not excluded, we read also the characters at positions $i = bq + L, bq + L + 1, \dots$, in this order, up to when all partial alignments are either excluded, or completed to a full alignment. Similarly to above, introduce $\mathcal{E}_R(b)$ and $e_R(b) = |\mathcal{E}_R(b)|$ (right-excess at block b).

An example with $c(\alpha) = 2$ is in Figure 2. Note that, at all steps, the pattern of the accessed part of the text consists of some blocks of length L and spacing q, plus one rightmost block with length $L' \geq L$ and spacing $q' \leqslant q$. A typical situation within the second step is as follows (here $c = 5$, $L = 3$, $q = 8$, $L' = 12$ and $q' = 7$):



Call $\mathcal{E}(b) = \mathcal{E}_L(b) \cup \mathcal{E}_R(b)$, and $e(b) = e_L(b) + e_R(b)$. Call Ψ_h^{exact} the average over random texts of the indicator function for the event that $e(b) \geq h$. Clearly, the average complexity rate of our algorithm is bounded by the expression

$$\Phi_{\text{alg}}(D) \leqslant \frac{L + \mathbb{E}(e(b))}{q} = \frac{L + \sum_{h \geq 1} \Psi_h^{\text{exact}}}{q} \,,$$

where the average is taken over random texts, at fixed dictionary. Note that, because of our choice of range for q and L, $c(\alpha) \geq 1$ for all α, and $c(|w|) \geq 1$ for all w.

Let α be a full alignment associated to the block b. Call $\mathcal{E}[\alpha]$ the set of extra positions of the text (besides the blocks) that we need to access in order to determine the alignment α. Then clearly $\mathcal{E}(b) = \bigcup_\alpha \mathcal{E}[\alpha]$.

2.3 Proof strategy for the upper bound

Our proof strategy is to prove that there exists a choice of parameters L and q, with the properties that $q = \Theta(m_{\min})$, $L/q = \Theta(\phi(\boldsymbol{r}(D)))$, and $\mathbb{E}(e(b)) = \Theta(1)$. This last condition is equivalent to the requirement that Ψ_h^{exact} is a summable series, and we will see that indeed the first can be bounded by a geometric series, and the second is rather small. Up to calculating the pertinent multiplicative constants, such a pattern would imply the functional form of the complexity anticipated in Section 1.2.

The idea is that the exact calculation of $\mathbb{E}(e(b))$ or of Ψ_h^{exact}, even at q and L fixed (which is easier than optimising w.r.t. these parameters), is rather difficult, but we can produce a simpler upper bound by:

- For alignments α with $c(\alpha) > 1$, neglect the information coming from the $e(b')$ extra characters that we have accessed at blocks $b' < b$. This allows to separate the analysis on the different blocks of text.

- Naïvely, for different (full) alignments α, we could perform a *union bound*, that is, $e(b) = |\mathcal{E}(b)| = |\bigcup_\alpha \mathcal{E}[\alpha]| \leqslant \sum_\alpha |\mathcal{E}[\alpha]|$, which thus separates the analysis over the different alignments. We will make an improved version of this bound, namely we use this bound, not with full alignments, but rather with "classes of equivalent partial alignments".

As we anticipated, the crucial point is that we count partial alignments instead of full alignments. A further slight improvement of the bound comes from considering these 'classes of equivalent partial alignments', instead of just the partial alignments. These two facts are motivated by the same argument, that we now elucidate.

Consider the two following notions: (1) Each set $A_h(w)$ of partial alignments is partitioned into classes I. (2) There is a subset $\bar{A}_h(w) \subseteq A_h(w)$ of alignments, that we shall call *basic alignments*. Now, suppose that the two following properties hold: (i) $I \cap \bar{A}_h(w) \neq \varnothing$ for all classes I of $A_h(w)$. (ii) For each $\alpha \in I$, there exists a $\bar{\alpha} \in I \cap \bar{A}_h(w)$, such that $\mathcal{E}(\alpha) \subseteq \mathcal{E}(\bar{\alpha})$. In this case it is easily estalished that the bound above can be improved into $e(b) = |\mathcal{E}(b)| = |\bigcup_\alpha \mathcal{E}[\alpha]| \leqslant \sum_{\bar{\alpha}} |\mathcal{E}[\bar{\alpha}]|$, where the sum runs only on basic partial alignments. Thus, calling $\Psi_h := \sum_{w \in D} \sum_{\alpha \in \bar{A}_h(w)} \mathbb{P}(|\mathcal{E}[\bar{\alpha}]| \geq h)$, we have $\Psi_h \geq \Psi_h^{\text{exact}}$.

We propose the following definition of basic alignment. Let α be in $A_h(w)$. In the string u, suppose that we write C_a instead of C, whenever the well-aligned character is a, and D_a when the deleted character is a (this is clearly just a bijective decoration of u). For $\alpha \in \bar{A}_h(w)$, we require that there are no occurrences of $C_a I_a$ as factors of u (as these are equivalent to $I_a C_a$), of $C_a D_a$ (as these are equivalent to $D_a C_a$) and of $I_a D_b$ or $D_b I_a$ (as these are equivalent to C_a or S_a, depending if $a = b$ or not). If α can be obtained from α' by a sequence of these rewriting rules, then α and α' are in the same class I.

It is easy to see that this definition of basic alignment and classes has the defining properties above.

2.4 Evaluation of an upper bound at q and L fixed

Let us call $p_{c,h,\varepsilon'}(w)$ the probability that, for a given word w and parameter ε', there exists an alignment $\alpha \in A_h(w)$, to a text consisting of $c-1$ blocks of length L and one block of length $L+h$, which is visited by the algorithm (that is, it makes at most k errors), that is, in particular,

$$\Psi_h \leqslant \sum_{\varepsilon'=0}^{q-1} p_{c,h,\varepsilon'}(w)\,. \tag{6}$$

We have the important fact

▶ **Proposition 3.**

$$p_{c,h,\varepsilon'}(w) \leqslant \beta s^{-(cL+h)} B_{cL+h+c-1,k} \tag{7}$$

for all ε', *where* $\beta = \frac{(2s-1)L+k}{(2s-1)L-k}$ *and* $B_{L,k} = (2s-1)^k \binom{L+k}{k}$.

The proof of this proposition is slightly complicated, and is presented in Appendix A. Note however that for the special case $c = 1$, and with exactly k errors (instead of at most k errors), the bound $s^{-(L+h)}(2s)^k \binom{L+k}{k}$ can be established trivially. Also note that the bound does not depend on ε', and, in particular, it only depends on $h = |\mathcal{E}_L| + |\mathcal{E}_R|$ for the alignments α at given w and ε', and not separately on the two summands.

We are now ready to evaluate the expressions for the upper bound on the quantity Ψ_h in (6), in light of (7). Call $R_c = \sum_{m\,:\,c(m)=c} r_m = \sum_{m=qc+L-1}^{q(c+1)+L-2} r_m$, and $p_{c,h}$ as q times the

RHS of (7) (that is, an upper bound to $\sum_{\varepsilon'=0}^{q-1} p_{c,h,\varepsilon'}(w)$). We have the bound

$$\sum_h \Psi_h \leqslant \sum_c R_c \sum_h p_{c,h} = \sum_c R_c \sum_h \beta\, q\, s^{-(cL+h)} B_{cL+h+c-1,k}\,. \tag{8}$$

Recalling that

$$\sum_{h\geq 0} s^{-h} \binom{a+k+h}{k} \leqslant \frac{1}{1-\frac{1}{s}\frac{a+k+1}{a+1}} \binom{a+k}{k}\,,$$

(and that $q < m_{\min}$), substituting in (8) gives

$$\begin{aligned}\Phi_{\text{alg}}(D) &\leqslant \frac{1}{q}\left(L + \beta q \sum_c R_c \frac{1}{1-\frac{1}{s}\frac{cL+k+c}{cL+c}} s^{-cL} \binom{cL+c-1+k}{k} (2s-1)^k\right) \\ &\leqslant \frac{1}{q}\left(L + \frac{\beta m_{\min}(2s-1)^k}{1-\frac{1}{s}\frac{L+k}{L}} \sum_c R_c\, s^{-cL} \binom{c(L+1)+k}{k}\right).\end{aligned} \tag{9}$$

We want to prove that

$$\Phi_{\text{alg}}(D) \leqslant \frac{C_1 k + C_1'}{m_{\min}} + C_2 \max_m \frac{\ln(s m r_m)}{m}\,, \tag{10}$$

with suitable constants C_1, C_1' and C_2 (it will turn out at the end that we can set $C_1' = C_1$ and C_1, C_2 to be as in Theorem 2, but at this point it is convenient to let them be three separate variables). This would prove the upper bound part of Theorem 2.

Note that, if $k/m_{\min} \geq 1/C_1$, the upper bound expression (10) is larger than the trivial bound $\Phi_{\text{alg}}(D) \leqslant 1$, and there is nothing to prove. So we can assume that $k/m_{\min} < 1/C_1$.

2.5 Optimisation of q and L

We now have to analyse the expression (9), in order to understand which values of q and L make the bound smaller. The sum over c is the most complicated term. We simplify it by using the fact that, for all $\xi \in \mathbb{R}^+$, $\ln\binom{a+k}{k} \leqslant k\ln(1+\xi) + a\ln(1+\xi^{-1})$, which gives

$$\begin{aligned}T &:= m_{\min}(2s-1)^k \sum_c R_c\, s^{-cL} \binom{c(L+1)+k}{k} \\ &\leqslant \sum_c \frac{1}{c^2} \exp\Big[-c\Big(LA - \frac{1}{c}\left(\ln(R_c m_{\min}) + k\ln((1+\xi)(2s-1))\right) - \frac{\ln c^2}{c} - \ln(1+\xi^{-1})\Big)\Big] \\ &= \sum_c \frac{1}{c^2} \exp\left[-c(LA - \phi'(c) - \ln(1+\xi^{-1}))\right]\,,\end{aligned} \tag{11}$$

where $A = \ln(s\xi/(1+\xi))$, $A' = \ln((1+\xi)(2s-1))$ and

$$\phi'(c) = \frac{\ln(c^2 R_c m_{\min}) + kA'}{c}\,. \tag{12}$$

Ultimately, we want to choose L such that T is bounded by a constant, as its summands over c are bounded by a convergent series. With this goal, let c^* be the value maximising the expression $\phi'(c)$, and ϕ^* the value of the maximum. The sum above is then bounded by

$$\sum_c \frac{1}{c^2} \exp[-c(LA - \phi^* - \ln(1+\xi^{-1}))]\,.$$

For any value of ξ such that $A > 0$ (that is, for $\xi > (s-1)^{-1}$), there exists a positive smallest value of L such that the exponent in the expression above is negative. So we set

$$L^* = \left\lceil \frac{\phi^* + \ln(1+\xi^{-1})}{A} \right\rceil ,$$

(as the choice of ξ is free, we can tune it at the end so that the ratio is an integer), and recognise that the RHS of equation (11), specialised to $L = L^*$, is bounded by $\sum_c \frac{1}{c^2} = \pi^2/6$. Note that

$$\phi^* \geq \phi'(1) \geq kA'$$

so that

$$\frac{L^*}{k} \geq \frac{A'}{A} = \frac{\ln((1+\xi)(2s-1))}{\ln(s\,\xi/(1+\xi))} ,$$

which implies that we can set $\beta = \frac{2s-1+A/A'}{2s-1-A/A'}$, and

$$\frac{1}{1-\frac{1}{s}\frac{L+k}{L}} \leqslant \frac{1}{1-\frac{1}{s}(1+A/A')} = \frac{1}{1-\frac{1}{s}\frac{\ln(s\xi(2s-1))}{\ln((1+\xi)(2s-1))}} .$$

Now, let us choose $q = \left\lfloor \frac{m_{\min}-k}{2} \right\rfloor$, which coincides with the choice of the analogous parameter in Chang and Marr [5]. This is the largest possible value such that $c(w) \geq 1$ for all $w \in D$. With this choice,

$$\frac{1}{q} \leqslant \frac{2}{m_{\min}} \frac{C_1}{C_1 - 1} .$$

Collecting the various factors calculated above, we get that the expression (9) is bounded by

$$\Phi_{\mathrm{alg}}(D) \leqslant \frac{2}{m_{\min}} \frac{C_1}{C_1-1} \left(L^* + \frac{\beta \frac{\pi^2}{6}}{1-\frac{1}{s}(1+A/A')} \right) .$$

We are left with two tasks: choosing suitable values for ξ and C_1 (both of order 1), and recognising that the expression for L^* (and for ϕ^*) can be related to the quantity $\phi(\boldsymbol{r})$ in (2). Let us start from the latter. Note that, as for any $m \geq m_{\min}$

$$\frac{m-k}{q} - 2 \leqslant c(m) \leqslant \frac{m}{q}$$

we can write[2] $m \leqslant m_{\min} c(m) \leqslant s^2 m$, which gives

$$\max_c \frac{1}{c} \ln(c^2 m_{\min} R_c) \leqslant \max_m \frac{m_{\min}}{m} \ln(s^2 m^2 r_m) \leqslant 2 m_{\min} \phi(\boldsymbol{r}) .$$

As, of course $\max_c(X(c)+Y(c)) \leqslant \max_c X(c) + \max_c Y(c)$, we have in particular that

$$\phi^* \leqslant 2 m_{\min}\phi(\boldsymbol{r}) + kA' \qquad\qquad L^* \leqslant \frac{2m_{\min}\phi(\boldsymbol{r}) + kA' + \ln(1+\xi^{-1})}{A} ,$$

[2] Because $s \geqslant 2$, and we anticipate that, under our choice, $C_1 \geq 5$, thus

$$m \leqslant 2(m-k-q) \leqslant m_{\min}\left(\frac{m-k}{q} - 2\right) \leqslant m_{\min} c(m) \leqslant m_{\min}\frac{m}{q} \leqslant 2\left(\frac{C_1}{C_1-1}\right) m \leqslant s^2 m .$$

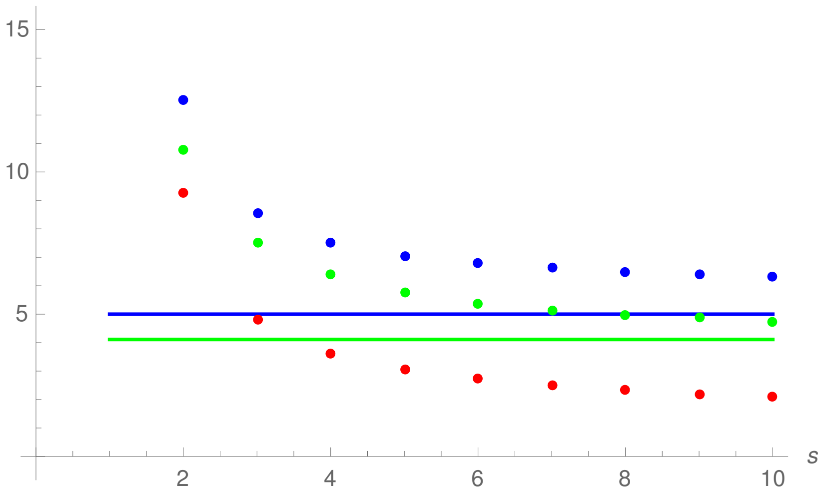


Figure 3 Plot of the constant $C_1(s)$, $C_1'(s)$ and $C_2(s)$, as given by the expressions in (13) (respectively, in blue, green and red). The asymptotic values are 5, $5\pi^2/12$ and 0 respectively.

which thus implies

$$\begin{aligned}\Phi_{\mathrm{alg}}(D) &\leqslant \frac{2}{m_{\min}} \frac{C_1}{C_1-1}\left(\frac{2m_{\min}}{A}\phi(\boldsymbol{r}) + k\frac{A'}{A} + \frac{\beta\frac{\pi^2}{6}}{1-\frac{1}{s}(1+A/A')} + \frac{\ln(1+\xi^{-1})}{A}\right) \\ &= \frac{2C_1}{C_1-1}\left[\frac{A'}{A}\frac{k}{m_{\min}} + \left(\frac{\beta\frac{\pi^2}{6}}{1-\frac{1}{s}(1+\frac{A}{A'})} + \frac{\ln(1+\xi^{-1})}{A}\right)\frac{1}{m_{\min}} + \frac{2}{A}\phi(\boldsymbol{r})\right].\end{aligned}$$

Let us choose $C_1 = 2A'/A + 1$. The expression above simplifies into

$$\Phi_{\mathrm{alg}}(D) \leqslant \frac{C_1 k}{m_{\min}} + \frac{2A'+A}{AA'}\left[\left(\frac{A\beta\frac{\pi^2}{6}}{1-\frac{1}{s}(1+\frac{A}{A'})} + \ln(1+\xi^{-1})\right)\frac{1}{m_{\min}} + 2\phi(\boldsymbol{r})\right],$$

in particular, this justifies the notation C_1, which in the introduction was chosen to denote the coefficient in front of the $\frac{k}{m_{\min}}$ summand. Now we shall choose the optimal value of ξ. The dependence on ξ is mild, provided that we are in the appropriate range $\xi > 1/(s-1)$. The choice of ξ, in turns, determines the ratio between the lower and upper bound, which has the functional form $C_1' + \kappa_s C_2$ (with notations as in the theorem). A choice which is a good trade-off among the three summands in this expression, and for which the analytic expression is relatively simple, is to take $\xi = 2s$. Under this choice we have

$$C_1 = 1 + 2\frac{\ln(4s^2-1)}{\ln(2s^2/(2s+1))}, \qquad C_2 = \frac{4}{\ln(2s^2/(2s+1))} + \frac{2}{\ln(4s^2-1)},$$
$$C_1' = \frac{C_2}{'2}\left[\ln\frac{2s+1}{2s} + \frac{\beta\pi^2}{6}\frac{s\ln(2s^2/(2s+1))\ln(4s^2-1)}{(s-1)\ln(4s^2-1)-\ln(2s^2/(2s+1))}\right]$$

or, in a more compact way, calling $a = A|_{\xi=2s} = \ln(2s^2/(2s+1))$ and $a' = A'|_{\xi=2s} = \ln(4s^2-1)$, and substituting back the value of β,

$$C_1 = \frac{a+2a'}{a}, \qquad C_2 = \frac{a+2a'}{a}\frac{2}{a'}, \tag{13a}$$
$$C_1' = \frac{a+2a'}{a}\left(\frac{\ln s - a}{a'} + \frac{\pi^2}{6}\frac{(2s-1)a'+a}{(2s-1)a'-a}\frac{as}{(s-1)a'-a}\right). \tag{13b}$$

The behaviour in s of these constants is depicted in Figure 3.

It can be verified that, with our choice of ξ, $C_1' < C_1$ for all $s \geq 2$.[3] we can replace C_1' by C_1 in the functional form (10) for the bound on $\Phi_{\mathrm{alg}}(D)$, and thus obtain the statement of Theorem 2. This concludes our proof.

References

1. Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.
2. Frédérique Bassino, Tsinjo Rakotoarimalala, and Andrea Sportiello. The complexity of the Multiple Pattern Matching Problem for random strings. In *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 40–53. SIAM, 2018.
3. Robert S Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.
4. William I. Chang and Eugene L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4-5):327–344, 1994.
5. William I. Chang and Thomas G. Marr. Approximate string matching and local similarity. In *Combinatorial Pattern Matching - CPM 1994*, volume 807 of *Lecture Notes in Computer Science*, pages 259–273. Springer, 1994.
6. Kimmo Fredriksson and Szymon Grabowski. Average-optimal string matching. *Journal of Discrete Algorithms*, 7(4):579–594, 2009.
7. Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.
8. Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1):31–88, 2001.
9. Gonzalo Navarro and Kimmo Fredriksson. Average complexity of exact and approximate multiple string matching. *Theoretical Computer Science*, 321(2-3):283–290, 2004.
10. Peter H. Sellers. The theory and computation of evolutionary distances: pattern recognition. *Journal of Algorithms*, 1(4):359–373, 1980.
11. Andrew Chi-Chih Yao. The complexity of pattern matching for a random string. *SIAM Journal on Computing*, 8(3):368–387, 1979.

[3] One way to see this is by proving that both $C_1(s)$ and $C_1'(s)$ decrease monotonically as functions on the real interval $[2, +\infty[$, that $\lim_{s\to\infty} C_1(s) = 5$, that $C_1(s) > C_1'(s)$ for $s \in \{2, 3, \ldots, 7\}$, and that $C_1'(8) < 5$.

A Proof of Proposition 3

In this section we evaluate an upper bound to $p_{c,h,\varepsilon'}$, which is the probability that, for a given word w with $c(|w|) = c$, the disclosed text composed of $c-1$ intervals of size L and one interval of size $L+h$ corresponds to at least one basic alignment α by making no more than k errors. The statement of the result, equation (14) below, is given in Proposition 3.

Let us introduce the recurring quantity

$$B_{L,k} := (2s-1)^k \binom{L+k}{k}.$$

First, let us analyse the case in which we have a single block, and exactly k errors. For w a word of length m, it is clear that the result depends only on the $m-\varepsilon'$ left-most characters of the word, not on the ε' right-most ones, so we can assume without loss of generality that $\varepsilon' = 0$. Call $H_{L,k}(m)$ the number of different words of length L obtained by transforming the suffixes of w and making exactly k errors. We have

▷ Proposition A.1. For all $L \geq k \geq 1$, $H_{L,k} \leqslant B_{L,k}$.

Proof. Note that the analogous statement with $2s-1$ replaced by $2s$ in $B_{L,k}$ is trivial, as we have exactly $2s$ types of errors (one deletion, s insertions and $s-1$ substitutions), and the counting of their possible positions in the string u is a function of the length of the string, bounded from above by the worst case, associated to all insertion errors.

We can gain the factor $2s-1$ instead of $2s$ by restricting to basic alignments, but this requires a finer analysis involving generating functions. Let us call $f(u,y,z)$ the generating function such that $[u^a y^L z^k] f(u,y,z)$ is the number of basic alignments of length L obtained by transforming a word of length a and making exactly k errors. Calculating $f(u,y,z)$ exactly is a difficult task, and the result would depend on w as a word, not only on $m = |w|$, but we will calculate a simpler upper bound $f_{\mathrm{UB}}(u,y,z)$, which in particular only depends on m. In this context, a generating-function upper bound is an upper bound for partial sums, that is $g \succeq f$ if $\sum_{h=0}^{k} [u^a y^L z^h](g(u,y,z) - f(u,y,z)) \geq 0$ for all L and a. Let us construct f_{UB} by starting from $f_0(u,y,z) := \frac{uy}{1-uy}$, which is the generating function f specialised to $z=0$, and let us introduce the various types of errors one at the time.

The first operation corresponds to allow for *insertion* errors. The restriction to basic alignments, however, brings to a subtlety. For example, starting with a word $w = abcd$, in order to get the alignment $aaabcd$ we can proceed in several ways: $a\boldsymbol{aa}bcd$ or $\boldsymbol{a}a\boldsymbol{a}bcd$ or by $\boldsymbol{aa}abcd$ (bold letters correspond to insertions). Under the notion of basic alignment we avoid to overcount these manifestly equivalent alignments, as of these expressions we would only keep the latter, $\boldsymbol{aa}abcd$, that is, at the left of a letter a we can only insert letters different from a. On the other hand, at the right end of the word one can insert strings consisting of any character of the alphabet.

Calling f_i the generating function in which insertion errors are allowed, we thus get

$$f_i(u,y,z) = \frac{1}{1-syz} f(u,y,z)\big|_{uy \to uy\left(\frac{1}{1-(s-1)z}\right)} = \frac{uy}{(1-syz)(1-uy-(s-1)yz)}.$$

We now introduce deletion errors, which, consistently, we allow only on the characters of the initial string (not on the ones which have just been insterted). Thus, any given original character can be either left as is, or deleted. This gives the generating function $f_{i,d}$, with

$$f_{i,d}(u,y,z) = f_i(u,y,z)\big|_{uy \to uy+uz} = \frac{uz+uy}{(1-syz)(1-uy-uz-(s-1)yz)}.$$

Finally, for substitution errors, again we can either substitute any initial character with one of the $s-1$ other characters of the alphabet, or leave it unchanged, which brings to $f_{i,d,s}$, with

$$f_{i,d,s} = f_{i,d}(u,y,z)|_{uy \to uy+(s-1)uyz} = \frac{u(syz - yz + y + z)}{(1-syz)(1-uz-(s-1)(u+1)yz)} \,.$$

Note that, by this procedure, we have already produced an upper bound, as $f_{i,d,s} \succeq f$ (in the sense defined above). Note also that it is *not* $f_{i,d,s} = f$, because, for example, we have overcounted the equivalent cases in which in a word $w = \cdots aa \cdots$ we have deleted the first *or* the second character.

If the word w is shorter than $L+k$, we may miss some alignments because they do not fit in the text interval. As we are evaluating an upper bound, we can restrict to the case in which w is long enough for this not to happen, and thus sum over all suffixes by just setting $u = 1$, and conclude that $H_{L,k} \leqslant [y^L z^k] f'(1,y,z)$. Thus, in order to conclude, we must show that $[y^L z^k] f'(1,y,z) \leqslant B_{L,k}$. Let us call

$$F_{L,k} = [y^L z^k] \frac{1}{(syz-1)(2syz-2yz+y+z-1)} \,.$$

We can rewrite the inequality above as $H_{L,k} \leqslant F_{L-1,k} + F_{L,k-1} + (s-1)F_{L-1,k-1}$, and thus, if we can prove that $F_{L,k} \leqslant B_{L,k}$, for all pairs of integers $L \geqslant k$, we could conclude in light of the fact that

$$H_{L,k} \leqslant B_{L-1,k} + B_{L,k-1} + (s-1)B_{L-1,k-1} = (2s-1)^k \binom{L+k}{k} - R_{L,k} \,,$$

where $R_{L,k} = (2s-1)^{k-1} \left(2(s-1) \frac{k-1}{L} \binom{L+k-2}{k-1} \right)$ is indeed easily checked to be non-negative for all $L \geqslant k \geqslant 1$.

So, to finish the proof, let us show that $F_{L,k} \leqslant B_{L,k}$. First,

$$\begin{aligned} F_{L,k} &= [y^L z^k] \left(\frac{1}{1-syz} + \frac{2syz-2yz+y+z}{1-2syz+2yz-y-z} \right) \\ &= \delta_{L,k} s^k + F_{L-1,k} + F_{L,k-1} + 2(s-1)F_{L-1,k-1} \,. \end{aligned}$$

Since $L \geqslant k \geqslant 1$, we have $R_{L,k} \geqslant \delta_{L,k} s^k$ for $s \geqslant 2$, and $B_{L,k} \geqslant F_{L,k} \geqslant H_{L,k}$.

To conclude, we just check the boundary conditions in the recursion above for $F_{L,k}$ and $B_{L,k}$, which again are in agreement with the inequality. Indeed we have, for $(L,k) \in \{(0,0),(0,1),(1,0)\}$, $F_{0,0} = B_{0,0} = 1$, $B_{0,1} = 2s-1 \geq 1 = F_{0,1}$ and $B_{1,0} = 4s-2 \geq 3s = F_{1,0}$. ◀

Now we want to deal with the more general case, in which we have more than one block, and we sum over the number of errors up to k. We will prove a more general statement, in which we have c blocks of lengths $L_1, \ldots, L_c$, separated by gaps of lengths $q_1, \ldots, q_{c-1}$, which in particular is so general to allow us to treat in one stroke the case in which we add characters at the left or at the right of the b-th algorithm block.

Similarly to the argument above, in order to produce an upper bound we can set without loss of generality that $\varepsilon' = 0$, all the q_i's are larger than k and that m is larger than $\sum L_i + \sum q_i + k$, as any variant of this would give no more alignments. So, we will call $p_{L_1,\ldots,L_c;k}$ the corresponding quantity, in which the dependence from the q_i's and m has been dropped.

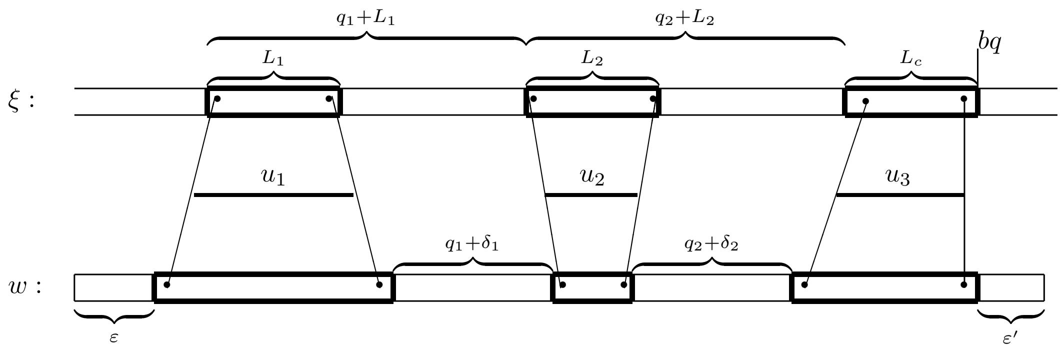


Figure 4 Example of multi-interval alignment analysed for the estimate of $p_{L_1,\dots,L_c;k}$.

For multi-block partial alignments, we have parameters $\delta_1, \dots, \delta_{c-1}$ for the offset among the different consecutive blocks of the partial alignment, and, if we have an offset δ_i in the alignment of two blocks, we have to perform at least $|\delta_i|$ deletions or insertions errors when completing the partial alignment to a full one (cf. figure 4).

Calling $\bar{L} = \sum_{i=1}^{c} L_i$, this leads to the following sum

$$p_{L_1,\dots,L_c;k} \leqslant s^{-\bar{L}} \sum_{t=0}^{k} \sum_{\Delta=0}^{t} \sum_{\substack{k_1,k_2,\dots,k_c\in\mathbb{N} \\ k_1+k_2+\dots+k_c=t-\Delta}} \sum_{\substack{\delta_1,\delta_2,\dots,\delta_{c-1}\in\mathbb{Z} \\ \delta_1+\delta_2+\dots+\delta_{c-1}=\Delta}} B_{L_1,k_1} B_{L_2,k_2} \dots B_{L_c,k_c} \,.$$

From the Vandermonde convolution formula, $\sum_{i=0}^{k} \binom{l_1+i}{i}\binom{l_2+k-i}{k-i} = \binom{l_1+l_2+k+1}{k}$, which implies $\sum_h B_{L_1,h} B_{L_2,k-h} = B_{L_1+L_2+1,k}$, we can simplify the expression above into

$$p_{L_1,\dots,L_c;k} \leqslant s^{-\bar{L}} \sum_{t=0}^{k} \sum_{\Delta=0}^{t} \sum_{\substack{\delta_1,\delta_2,\dots,\delta_{c-1}\in\mathbb{Z} \\ \delta_1+\delta_2+\dots+\delta_{c-1}=\Delta}} B_{\bar{L}+c-1,t-\Delta} \,.$$

The sum over the δ_i's gives

$$\sum_{\substack{\delta_1,\delta_2,\dots,\delta_{c-1}\in\mathbb{Z} \\ \delta_1+\delta_2+\dots+\delta_{c-1}=\Delta}} 1 = [z^{\Delta}] \left(\frac{1+z}{1-z}\right)^{c-1}$$

that is, by recognising that $B_{L,k-h} \leqslant B_{L,k} \left(\frac{k}{(2s-1)L}\right)^h$, we get

$$p_{L_1,\dots,L_c;k} \leqslant s^{-\bar{L}} B_{\bar{L}+c-1,k} \left(\frac{(1+z)^{c-1}}{(1-z)^c}\right)\Bigg|_{z=\frac{k}{(2s-1)(\bar{L}+c-1)}} .$$

This is all we shall say at this level of generality. Now note that, in our patterns, $\bar{L}+c-1 \geq cL$ (and $k \leqslant L$), so that, in this range of parameters,

$$p_{L_1,\dots,L_c;k} \leqslant s^{-\bar{L}} B_{\bar{L}+c-1,k} \left(\frac{(1+z)^{c-1}}{(1-z)^c}\right)\Bigg|_{z=\frac{1}{(2s-1)c}\frac{k}{L}} \leqslant \frac{(2s-1)+\frac{k}{L}}{(2s-1)-\frac{k}{L}} s^{-\bar{L}} B_{\bar{L}+c-1,k} \,. \quad (14)$$