

Article

Computable Bounds and Monte Carlo Estimates of the Expected Edit Distance [†]

Gianfranco Bilardi and Michele Schimd * 

Department of Information Engineering, University of Padova, 35131 Padova, Italy; bilardi@dei.unipd.it

* Correspondence: schimdmi@dei.unipd.it

[†] This paper is an extended version of our paper published in String Processing and Information Retrieval, SPIRE 2019, Segovia, Spain, 7–9 October 2019.

Abstract

The edit distance is a metric of dissimilarity between strings, widely applied in computational biology, speech recognition, and machine learning. Let $e_k(n)$ denote the average edit distance between random, independent strings of n characters from an alphabet of size k . This paper is concerned with the efficient computation of $\alpha_k(n) = e_k(n)/n$ as well as of $\alpha_k = \lim_{n \rightarrow \infty} \alpha_k(n)$. It is shown that $\alpha_k(n) - \Theta(\sqrt{\log n/n}) \leq \alpha_k \leq \alpha_k(n)$ and that α_k is computable. An algorithm for the exact computation of $\alpha_k(n)$ is presented with running time $T = \mathcal{O}(n^2 k \min(3^n, k^n))$ thus, of limited practical use. An analysis of Monte Carlo estimates, based on McDiarmid's inequality, shows how $\alpha_k(n)$ can be evaluated with good accuracy and high confidence level, for rather large values of n . In particular, 99.9% confidence intervals of width approximately 10^{-2} are obtained for α_k . An efficiently computable lower bound β_k^* to α_k is derived, with $\lim_{k \rightarrow \infty} \beta_k^* = 1$. For k greater than a few dozens, significant bounds on α_k can be obtained faster via β_k^* than by statistical methods. The above techniques yield improvements on most previous numerical estimates as well as results for alphabet sizes and string lengths not reported before.

Keywords: combinatorics in computer science; algorithms on strings; Monte Carlo methods; rate of convergence, degree of approximation

MSC: 68R05; 68W32; 65C05; 41A25

1. Introduction

Measuring dissimilarity between strings is a fundamental problem in computer science, with applications in computational biology, speech recognition, machine learning, and other fields. One commonly used metric is the edit distance (or Levenshtein distance), defined as the minimum number of substitutions, deletions, and insertions necessary to transform one string into the other.

It is natural to ask what is the expected distance between two randomly generated strings, as the string size grows; knowledge of the asymptotic behavior has proved useful in computational biology [1] and in nearest neighbor search [2], to mention a few examples.

In computational biology, the question often arises whether two strings (e.g., two DNA reads) are noisy copies of the same source or of non-overlapping sources. In several cases of interest, the source is modeled as a sequence of independent and identically distributed symbols (see, e.g., [1,3], with reference to DNA) and the noise is modeled with substitutions, insertions, and deletions (a good approximation for technologies like PacBio



Academic Editor: Rao Mikkilineni

Received: 30 May 2026

Revised: 9 July 2026

Accepted: 11 July 2026

Published: 14 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

and MinION [4]). Then, the statistical inference may be based on a comparison of the distance between the observed strings with either the expected distance between a string and a noisy copy of itself, or the expected distance between two random strings.

Even for uniform and independent strings, the study of the expected edit distance appears to be challenging and little work has been reported on the problem. In contrast, the closely related problem of computing the expected length of the longest common subsequence has been extensively studied, since the seminal work by Chvátal and Sankoff [5].

Using Fekete's lemma, it can be shown that both metrics tend to grow linearly with the string size n [6]. Specifically, let $e_k(n)$ denote the expected edit distance between two random, independent strings of length n on a k -ary alphabet; then $\alpha_k(n) = e_k(n)/n$ approaches (from above) a limit $\alpha_k \in [0, 1]$. Similarly, let $\ell_k(n)$ denote the expected length of the longest common subsequence; then $\gamma_k(n) = \ell_k(n)/n$ approaches (from below) a limit $\gamma_k \in [0, 1]$. The γ_k 's are known as the Chvátal-Sankoff constants. The efficient computation of the exact values of α_k and γ_k is an open problem. This paper establishes the computability of α_k , for any k , and proposes methods for estimating and bounding α_k , also reporting numerical results for various alphabet sizes k .

From the perspective of computational complexity, we remark that, for the problem of computing $\alpha_k(n)$, given input n (for a fixed k), only algorithms that run in doubly exponential time and use exponential space are currently known, including those presented in this paper. Observe that the input size is $\log_2 n$, the number of bits needed to specify the problem input n . Similar statements hold for the computation of $\gamma_k(n)$. Therefore, at the state of the art, we can place these problems in the complexity class $\text{EXPSPACE} \subseteq 2\text{-EXPTIME}$, but not in EXPTIME and, a fortiori, not in $\text{PSPACE} \subseteq \text{EXPTIME}$. Technically, these are traditionally defined as classes of decision problems, hence our statements strictly apply to suitable decision versions of computing the constants of interest. Specifically, the decision version of computing α_k could take as inputs a (positive) integer n and an integer r such that $0 \leq r \leq \lceil k \log_2 n + \log_2 n \rceil$ and output the r -th digit of $A_k(n) = k^n n \alpha_k(n)$. (Observe that $0 \leq A_k(n) \leq k^n n$, whence $A_k(n)$ can be expressed with $\lceil n \log_2 k + \log_2 n \rceil$ bits.) Analogous considerations can be made for the problem of computing the ν most significant bits of α_k (or γ_k). Here, the input size is $\log_2 \nu$ and the time becomes triple exponential in the input size. Whether these problems inherently exhibit high complexity or they can be solved efficiently by exploiting a not yet uncovered deeper structure remains to be seen.

1.1. Related Work

The guaranteed, deterministic bounds presented in this paper were first published in [7]. A preliminary version of this article was published in [8] and cited in the subsequent work by Rosenfeld [9] discussed below. Details on additions and improvements of this article with respect to the conference version [7] are given in Section 1.3.

There is limited literature directly pursuing bounds and estimates for α_k . It is also interesting to review results on γ_k : on the one hand, bounds to γ_k give bounds to α_k ; on the other hand, techniques for analyzing γ_k can be adapted for analyzing α_k .

The only published estimates of α_k can be found in [1] which gives $\alpha_4 \approx 0.518$ for the quaternary alphabet and $\alpha_2 \approx 0.29$ for the binary alphabet. Estimates of γ_k are given by Bundschuh [10], in particular $\gamma_2 \approx 0.8126$ and $\gamma_4 \approx 0.6544$. A similar value is reported in [11], which gives $\gamma_2 \approx 0.8124$. Estimates of γ_k by sampling are given by [12]; their conjecture that $\gamma_2 > 0.82$ appears to be at odds with the estimate in [10]. In [13], the conjecture $\gamma_2 \approx 0.8122$ is proposed. They also derive a closed form for the limit constant when only one string is random and the other is a *periodic* string containing all symbols of Σ_k .

The best published analytical lower bounds to α_k are $\alpha_4 \geq 0.3383$ for a quaternary alphabet and $\alpha_2 \geq 0.1578$ for a binary alphabet [1]. Recently, Rosenfeld [9] published the first systematic study of upper bounds to α_k . Using an adaptation of the technique developed by Lueker for γ_k [14], they established $\alpha_2 \leq 0.315514$ and $\alpha_4 \leq 0.56578$ (as well as bounds for different values of k). The best known analytical lower and upper bounds to γ_2 are given by [14] and improved by [15], who obtained $0.792665 \leq \gamma_2 \leq 0.82628$. For larger alphabets, the best lower bounds are given by [15] and the best upper bounds are given by [11], including $0.6143 \leq \gamma_4 \leq 0.7082$. From known relations between the edit distance and the length of the longest common subsequence, it follows that $1 - \gamma_k \leq \alpha_k \leq 2(1 - \gamma_k)$. Thus, upper and lower bounds to α_k can be respectively obtained from lower and upper bounds to γ_k . From $\gamma_2 \leq 0.8263$ of [14], we obtain $\alpha_2 \geq 0.1737$, which is tighter than the bound given in [1]. Instead $\gamma_4 \leq 0.7082$ of [11] yields $\alpha_4 \geq 0.2918$, which is weaker than the bound $\alpha_4 \geq 0.3383$ [1]. From the weaker relation $(1 - \gamma_2)/2 \leq \alpha_2$, ref. [2] obtained the looser bound $\alpha_2 \geq 0.0869$. In this paper, we derive improved bounds, for both α_2 and α_4 , as well as bounds on α_k , for values of k not fully addressed by earlier literature. Some of our techniques resemble those used in [16] for estimating γ_k . Table 1 shows lower bounds to α_k for various values of k based on this work and on that of previous authors. Lower bounds from [11,14] are obtained from upper bounds to γ_k , which we have translated into $\alpha_k \geq 1 - \gamma_k$. Dančík reported values of the bound only for $k \leq 15$. Lueker reported only the numerical upper bound to γ_2 ; his computational approach is interesting and sophisticated, but its time and space are exponential with k . Experimenting with (a minor adaptation of) the software provided by the author, we have not been able to compute γ_3 within reasonable time. We have obtained the values reported in the Ganguly et al. column by numerically solving their equations (in [1], only the values for $k = 2$ and $k = 4$ were reported). The equations underlying the results in the rightmost column of Table 1 are developed in Section 6, together with a rigorous analysis of their numerical solution.

To assess the tightness of bounds to γ_k , several authors have investigated the rate of convergence of $\gamma_k(n)$ to γ_k . The bound $0 \leq \gamma_k - \gamma_k(n) \leq \mathcal{O}(\sqrt{\log n/n})$ has been obtained by [17] and, with a smaller constant, by [18]. Lueker [14] introduced a sequence of upper bounds γ_k^h converging to γ_k and satisfying $0 \leq \gamma_k^h - \gamma_k \leq \mathcal{O}((\log h/h)^{1/3})$, where the time complexity and the space complexity of computing γ_k^h increase exponentially with h . Observing that h is in turn exponential in the number ν of desired bits for γ_k and that ν is exponential in the input size $\lceil \log_2 \nu \rceil$, we see that computation time is a triple exponential. No study of the rate of convergence of $\alpha_k(n)$ to α_k has been published. In this paper, we show that $0 \leq \alpha_k(n) - \alpha_k \leq \mathcal{O}(\sqrt{\log n/n})$, exploiting a framework developed in [18].

Recently, Tiskin [19] has established that γ_2 is an algebraic number, introducing novel ideas, which may open new perspectives on the analysis of γ_k and α_k , for any k .

Table 1. Comparison of lower bounds to α_k obtained in this paper and in previous work. Best known bounds are highlighted in bold face.

k	Dančík	Lueker	Ganguly et al.	This Work
2	0.162377	0.17372	0.157761	0.170552
3	0.234197	-	0.265028	0.283660
4	0.291764	-	0.338322	0.359783
5	0.335572	-	0.392040	0.415173
6	0.370684	-	0.433508	0.457766
7	0.399816	-	0.466732	0.491836
8	0.424593	-	0.494136	0.519901
16	-	-	0.616273	0.644758
32	-	-	0.708537	0.738677

1.2. Paper Contributions and Organization

The notation and definitions used throughout this paper are given in Section 2. In Section 3, an upper bound $\alpha_k(n) - \alpha_k \leq Q(n)$ is derived, for each $k \geq 2$, where $Q(n) = \Theta(\sqrt{\log n/n})$ is a precisely specified function (independent of k). This implies $\alpha_k \in [\alpha_k(n) - Q(n), \alpha_k(n)]$, where the interval can be made arbitrarily small by choosing a suitably large n . One corollary is the computability of the real number α_k , for each $k \geq 2$. Unfortunately, the algorithm underlying the computability proof is of little practical use, since the only known method to exactly compute $\alpha_k(n)$ is by direct application of its definition, resulting in $\mathcal{O}(n^2 k^{2n})$ time. Even after some improvement presented in Section 5, the upper bound $\alpha_k \leq \alpha_k(n)$ is practically computable only for small values of k and n . Moreover, for the feasible values of n , $Q(n)$ is too large for the lower bound $\alpha_k \geq \alpha_k(n) - Q(n)$ to be useful. These considerations motivate the exploration of alternate approaches.

In Section 4, an analysis, based on McDiarmid's inequality, is developed for Monte Carlo estimates of $\alpha_k(n)$ obtained from the edit distance of a sample of N pairs of strings. The analysis yields the radius Δ of confidence intervals for $\alpha_k(n)$, in terms of n , N , and the desired confidence level λ . The (sequential) time to obtain an estimate can be approximated as $T \approx \tau_{ed} \frac{n}{\Delta^2} \ln^2\left(\frac{1}{1-\lambda}\right)$, where τ_{ed} is of the order of 5 ns, on a typical state of the art processor core. Rather large values of n can then be dealt with. As an indication, a $\lambda = 0.999$ confidence interval of radius $\Delta = 0.67 \times 10^{-3}$ is obtained for $\alpha_k(2^{15})$ in about 43 min. The corresponding confidence interval for α_k has radius $\Delta + \frac{Q(2^{15})}{2} = 0.00068 + 0.01320 = 0.01388$.

In Section 5, upper bounds to α_k by exact computation of $\alpha_k(n)$ for small values of n are obtained, by introducing an $\mathcal{O}(n^2(3k)^n)$ time algorithm that, while still exponential in n , is (asymptotically and practically) faster than the straightforward, $\mathcal{O}(n^2(k^2)^n)$ time, algorithm. When k is of the order of a few dozens, only very small values of n are feasible and $\alpha_k(n)$ does not differ appreciably from the quantity $1 - \frac{1}{k}$, which satisfies $\alpha_k(n) \leq 1 - \frac{1}{k}$, as it can be easily shown by allowing only substitutions (cf. Hamming distance).

In Section 6, a lower bound $\alpha_k \geq \beta_k^*$ is established, for each $k \geq 2$. A counting argument provides a lower bound to the number of string pairs with distance at least βn ; an asymptotic analysis provides conditions on β under which the contribution to $\alpha_k(n)$ of the remaining string pairs vanishes with n . A careful study leads to a numerical algorithm to compute β_k^* , the supremum of the β 's satisfying such conditions, with any desired accuracy, ϵ . Since, as shown in Section 6, $\lim_{k \rightarrow \infty} \beta_k^* = 1$, the interval $[\beta_k^*, 1 - \frac{1}{k}]$, which contains α_k , has size vanishing with increasing k . For k large enough, it becomes a subset of a confidence interval obtained with comparable computational effort. As an example, $\beta_{240}^* \approx 0.999984 \leq \alpha_{240} \leq 1 - 2^{-40} \approx 0.999999$, placing α_{240} in an interval of size smaller than 0.16×10^{-4} . To achieve $Q(n) \leq 0.16 \times 10^{-4}$ requires $n \geq 10^{11}$. On a single core, computing the edit distance for just one pair of strings of length $n = 10^{11}$ would take time $T \approx 5 \times 10^{-9} 10^{22} \text{ s} \approx 1.6 \times 10^6$ years, whereas computing β_{240}^* took just 14 milliseconds, using a straightforward, non-optimized implementation.

By applying the above methodologies, we numerically derive guaranteed as well as statistical estimates for specific α_k 's and $\alpha_k(n)$'s. In particular, Table 2 summarizes our numerical results for various alphabet sizes. For each k , the table reports an interval that provably contains α_k and a (narrower) interval that contains α_k with confidence 0.999. For the confidence interval, more details are provided in Section 4, while for the guaranteed bounds, details are given in Section 5 and Section 6.

In Section 7, we wonder about the asymptotic behavior of α_k , with respect to k . We propose and motivate the conjecture that $\lim_{k \rightarrow \infty} (1 - \alpha_k)k = c_\alpha$ for some constant $c_\alpha \geq 1$. Numerical evidence indicates that perhaps $3 \leq c_\alpha \leq 4$.

Finally, Section 8 presents conclusions and further directions of investigation.

Table 2. Summary of numerical results obtained applying the methodologies presented in this paper. For various sizes k , the table shows an interval guaranteed to contain α_k and an interval containing α_k with confidence 99.9%. The fact that the latter is considerably narrower than the former is due to the difficulty of deriving tight guaranteed bounds.

k	Guaranteed Interval	99.9% Confidence Interval
2	[0.17055, 0.36932]	[0.26108, 0.28884]
3	[0.28366, 0.53426]	[0.40144, 0.42920]
4	[0.35978, 0.63182]	[0.49031, 0.51807]
5	[0.41517, 0.70197]	[0.55289, 0.58066]
6	[0.45776, 0.75149]	[0.60002, 0.62778]
7	[0.49183, 0.79031]	[0.63701, 0.66477]
8	[0.51990, 0.81166]	[0.66694, 0.69470]
16	[0.64475, 0.89554]	[0.79198, 0.81974]
32	[0.73867, 0.96588]	[0.87230, 0.90007]

1.3. Summary of Contributions

We now schematically summarize the key contributions of this paper. Those marked by a “*” represent novelties with respect to an earlier, conference version [7] of this work.

- (*) An analysis of the rate of convergence of $\alpha_k(n)$ to α_k .
- (*) A proof that α_k is a computable real number.
- The coalesced dynamic programming algorithm, which can be exploited to obtain the exact value of $\alpha_k(n)$, unfortunately practical only for small values of n .
- (*) An analysis of the statistical estimates for α_k , achieving largely improved confidence intervals (over analogous results in [7]), with a different approach based on McDiarmid’s inequality.
- A combinatorially derived lower bound on α_k .
- (*) An extensive proof of correctness of the algorithm proposed to compute the lower bound on α_k , with any desired precision.
- (*) A conjecture on the behavior of α_k for large k .

2. Preliminaries

In this section, we introduce the notation adopted throughout the paper and present some preliminary definitions and results used in various parts of the work.

2.1. Notation and Definitions

Let Σ_k be a finite alphabet of size $k \geq 2$ and let $n \geq 1$ be an integer; a *string* x is a sequence of symbols $x[1]x[2] \dots x[n]$ where $x[i] \in \Sigma_k$; n is called the *length* (or *size*) of x , also denoted by $|x|$. Σ_k^n is the set of all strings of length n .

2.1.1. Edit Distance

We consider the following *edit operations* on a string x : the *match* of $x[i]$, the *substitution* of $x[i]$ with a different symbol $b \in \Sigma_k \setminus \{x[i]\}$, the *deletion* of $x[i]$, and the *insertion* of $b \in \Sigma_k$ in position $j = 0, \dots, n$ (insertion in j means b goes after $x[j]$ or at the beginning if $j = 0$); an *edit script* is a sequence of edit operations. With each type of edit operation is associated a cost; throughout this paper, matches have cost 0 and other operations have cost 1. The cost of a script is the sum of the costs of its operations. The *edit distance* between x and y , $d_E(x, y)$, is the minimum cost of any script transforming x into y . It is easy to see that $||x| - |y|| \leq d_E(x, y) \leq \max(|x|, |y|)$.

2.1.2. Simple Scripts

We can view a string as a sequence of *cells*, each containing a symbol from Σ_k , and consider edit operations as acting on such cells: a deletion destroys a cell, a substitution changes the content of a cell, and an insertion creates a new cell with some content in it (matches leave cells untouched). We will say that a script is *simple* if it performs at most one edit operation on each cell. It is easy to see that, if a script transforming x into y is not simple, then there is a script with fewer operations which achieves the same transformation. In fact, if a cell is eventually deleted, any operation performed on it prior to its deletion can be safely removed from the script; if a cell is inserted, any subsequent substitution can be removed, appropriately selecting the content of the initial insertion; and multiple substitutions on a cell that is retained can be either replaced by just one appropriate substitution or removed altogether. Thus, a script of minimum cost is necessarily simple, so that, to determine $d_E(x, y)$, we can restrict our attention to simple scripts.

2.1.3. Scripts and Alignments

Given an edit script transforming x into y , consider those cells of x that are retained in y , possibly with a different content. Since the relative order of two such cells is the same in x and in y , the positions occupied by such cells in x and y form an alignment, in the sense defined next.

An *alignment* $(\mathcal{I}, \mathcal{J})$ between x and y is a pair of increasing integer sequences of the same length s

$$\begin{aligned}\mathcal{I} &= (i_1, \dots, i_s) & 1 \leq i_1 < i_2 < \dots < i_s \leq |x|, \\ \mathcal{J} &= (j_1, \dots, j_s) & 1 \leq j_1 < j_2 < \dots < j_s \leq |y|.\end{aligned}$$

The positions i_ℓ in x and j_ℓ in y are said to be aligned in $(\mathcal{I}, \mathcal{J})$. To each script $\mathcal{S}$, there corresponds a unique alignment $a(\mathcal{S}) = (\mathcal{I}, \mathcal{J})$, where s equals the number of cells of x that are retained in y and, for every $\ell = 1, 2, \dots, s$, the cell in position i_ℓ of x has moved to position j_ℓ of y . If $\mathcal{S}$ is simple, then: (i) for aligned positions i_ℓ and j_ℓ , $y[j_\ell]$ is substituted with or matched to $x[i_\ell]$ depending on whether $y[j_\ell] \neq x[i_\ell]$ or not; (ii) for positions $i \notin \mathcal{I}$, $x[i]$ is deleted, and (iii) for positions $j \notin \mathcal{J}$, $y[j]$ is inserted.

Next, we prove a simple lemma, which will be useful both in Section 3, to cast edit distance within the framework of [18], and in Section 6, to develop a counting argument leading to a lower bound on α_k .

Lemma 1. *With the preceding notation, if $\mathcal{S}$ is a simple script to transform x into y , with $|x| = |y| = n$, and $(\mathcal{I}, \mathcal{J}) = a(\mathcal{S})$ is the corresponding alignment, its cost is*

$$\text{cost}(x, y, \mathcal{S}) = 2(n - s) + \sum_{\ell=1}^s (x[i_\ell] \neq y[j_\ell]). \quad (1)$$

Proof. The script performs $(n - s)$ deletions, $(n - s)$ insertions, and $\sum_{\ell=1}^s (x[i_\ell] \neq y[j_\ell])$ substitutions. $\square$

We may observe that (for given x and y) simple scripts corresponding to the same alignment differ only with respect to the order in which the edit operations are applied. Such an order does not affect the final result, since in a simple script different operations act on different cells. Thus, the number of distinct simple scripts with the same alignment is the factorial of their cost (i.e., of the number of edit operations), given by Equation (1).

2.1.4. Random Strings and the Limit Constant

A *random string* of length n , $X = X[1]X[2] \dots X[n]$, is a sequence of *random symbols* $X[i]$ generated according to some distribution over Σ_k . We will assume that the $X[i]$'s are uniformly and independently sampled from Σ_k or, equivalently, that $\Pr[X = x] = k^{-n}$ for every $x \in \Sigma_k^n$. We define the *eccentricity* $\text{ecc}(x)$ of a string x as its expected distance from a random string $Y \in \Sigma_k^n$:

$$\text{ecc}(x) = k^{-n} \sum_{y \in \Sigma_k^n} d_E(x, y). \quad (2)$$

The expected edit distance between two random, independent strings of Σ_k^n is:

$$\begin{aligned} e_k(n) &= k^{-2n} \sum_{x \in \Sigma_k^n} \sum_{y \in \Sigma_k^n} d_E(x, y) \\ &= k^{-n} \sum_{x \in \Sigma_k^n} \text{ecc}(x). \end{aligned} \quad (3)$$

Let $\alpha_k(n) = e_k(n)/n$; it can be shown (Fekete's lemma from ergodic theory; see, e.g., Lemma 1.2.1 in [6]) that there exists a real number $\alpha_k \in [0, 1]$, such that

$$\lim_{n \rightarrow \infty} \alpha_k(n) = \alpha_k. \quad (4)$$

The main objective of this paper is to derive estimates and bounds to α_k .

2.1.5. Rate of Convergence to the Limit Constant

In the outlined context, it is of interest to develop upper bounds, as functions of n , to the quantity

$$q_k(n) = \alpha_k(n) - \alpha_k,$$

which we will refer to as the *rate of convergence*, following a terminology widely used for analogous quantities in the context of the longest common subsequence (e.g., [17,18]).

2.2. Computing the Edit Distance

The edit distance and the length of the longest common subsequence (LCS) can be computed by a *dynamic programming* algorithm. Given two strings, x of length n and y of length m , their edit distance $d_E(x, y)$ is obtained as the entry $M_{n,m}$ of an $(n+1) \times (m+1)$ matrix $\mathbf{M}$, computed according to the following recurrence:

$$\begin{aligned} M_{i,0} &= i && \text{for } i = 0, \dots, n \\ M_{0,j} &= j && \text{for } j = 0, \dots, m \\ M_{i,j} &= \min(M_{i-1,j-1} + \xi_{i,j}, M_{i-1,j} + 1, M_{i,j-1} + 1) && \text{for } i > 0 \text{ and } j > 0 \end{aligned} \quad (5)$$

where $\xi_{i,j} = 0$ if $x[i] = y[j]$ and $\xi_{i,j} = 1$ otherwise. (A similar algorithm computes the length of the LCS. Recurrence (5) becomes $M_{i,0} = 0$, $M_{0,j} = 0$, and $M_{i,j} = \max(M_{i-1,j-1} + (1 - \xi_{i,j}), M_{i-1,j}, M_{i,j-1})$.) This algorithm takes $\mathcal{O}(nm)$ time and space. An edit script transforming x into y can be obtained *backtracking* on $\mathbf{M}$, along a *path* from cell (n, m) to cell $(0, 0)$. For both edit distance and LCS, the approach by [20], exploiting the method of the Four Russians, reduces the time to $\mathcal{O}(\frac{n^2}{\log n})$, assuming $n \geq m$. Although asymptotically faster, the algorithm in [20] is seldomly used. Other approaches are usually preferred, such as the one proposed by [21], which reduces the n^2 bound by a

factor proportional to the machine word size, implementing Recurrence (5) via bit-wise operations.

The space complexity of the basic dynamic programming algorithm can be reduced to $\mathcal{O}(\min(n, m))$. Assuming, w.l.o.g., that $n \geq m$, simply proceed row-wise storing only the last complete row. While this approach is not directly amenable to constructing scripts by backtracking, a more sophisticated divide and conquer version due to Hirschberg [22] yields the edit distance and an edit script in quadratic time and linear space. In [23], authors applied the Four Russians method to Hirschberg's algorithm, improving its running time by a logarithmic factor.

It is known that both the edit distance and the length of the LCS cannot be computed in time $\mathcal{O}(n^{2-\epsilon})$, unless the *Strong Exponential Time Hypothesis* (SETH) is false ([24,25]).

Approximate computation of the edit distance has been extensively studied. Ukkonen [26] presents a banded algorithm that computes an approximation within a factor $\mathcal{O}(n^{1-\epsilon})$ in time $\mathcal{O}(n^{1+\epsilon})$. Interestingly, this algorithm computes the exact distance whenever such distance is $\mathcal{O}(n^\epsilon)$ (although, it may output the exact distance also for higher values). In [27] authors give an algorithm that computes the exact distance in time $\mathcal{O}(n + d^2)$, where d is the distance itself. Thus, sub-quadratic time can be achieved when the distance is sub-linear. More recently, an $(\log n)^{\mathcal{O}(1/\epsilon)}$ approximation, computable in time $\mathcal{O}(n^{1+\epsilon})$, was proposed by [28], and a constant approximation algorithm with running time $\mathcal{O}(n^{1+5/7})$ was proposed by [29,30]. The work by [31] gives a reduction from approximate length of the longest common subsequence to approximate edit distance, proving that the algorithm in [29] can also be used to approximate the length of the LCS.

In order to compute upper bounds to α_k , we propose an algorithm related to the approaches developed by [14,32]. In these works, portions of the dynamic programming matrix are associated to the states of a finite state machine. Our algorithm conceptually simulates all possible executions of a machine similar to the one defined in [32].

3. Rate of Convergence and Computability of α_k

For each n , $\alpha_k(n)$ is a rational number, which can be computed, according to Equation (3), by exhaustively enumerating all pairs (x, y) of strings in Σ_k^n and accumulating the corresponding values $d_E(x, y)$, which can be obtained with any algorithm for the exact edit distance. On the other hand, the limit constant α_k is known to exist as a real number, whose rationality remains an open problem. In this section, we show that, for every k , this number is computable, according to the following (standard) definition.

Definition 1 (Computability of a real number). *A real number ζ is computable if there exists an algorithm that, given as input a rational number $\epsilon > 0$, outputs a rational number ζ_ϵ , such that $|\zeta - \zeta_\epsilon| < \epsilon$.*

As shown in [33], the subadditivity of a sequence of rational numbers, while sufficient to guarantee the existence of a limit (Fekete's Lemma, [34]), is not sufficient to guarantee its computability which, if present, requires an additional argument. For α_k , such an argument can be provided along the following steps:

- Prove that, for some computable function $b_k(n)$, we have

$$\alpha_k(n) - \alpha_k \leq b_k(n) \quad \forall n \geq 1. \quad (6)$$

- Show that there is an algorithm which, given a rational number $\epsilon > 0$, computes an integer $n_\epsilon \geq 1$ such that

$$b_k(n) < \epsilon \quad \forall n \geq n_\epsilon. \quad (7)$$

- Let $\alpha_{k,\epsilon} = \alpha_k(n_\epsilon)$ and observe that

$$0 < \alpha_{k,\epsilon} - \alpha_k < b_k(n_\epsilon) < \epsilon, \quad (8)$$

thus complying with Definition 1 (with $\zeta = \alpha_k$ and $\zeta_\epsilon = \alpha_{k,\epsilon}$), since n_ϵ is computable from ϵ and $\alpha_{k,\epsilon} = \alpha_k(n_\epsilon)$ is computable from n_ϵ .

To obtain a bound $b_k(n)$ such that Equation (6) is satisfied, we show how the edit distance problem can be cast within a framework developed in [18], to analyze the limit average behavior of certain functions of random string pairs on a finite alphabet Σ . These functions are formally defined next.

Definition 2. Let $S : \Sigma \times \Sigma \rightarrow R_0^+$ be a symmetric ($S(b, a) = S(a, b)$), non-negative, real function and let δ be a real number. The score of a pair of strings $x, y \in \Sigma^n$, with respect to a given alignment $(\mathcal{I}, \mathcal{J})$, is defined as

$$S(x, y; \mathcal{I}, \mathcal{J}) = \sum_{\ell=1}^s S(x[i_\ell], y[j_\ell]) + \delta(n - s).$$

The score of the string pair is the maximum score over all possible alignments:

$$S(x, y) = \max_{(\mathcal{I}, \mathcal{J})} S(x, y; \mathcal{I}, \mathcal{J}).$$

Next, we express the edit distance in terms of a suitable score.

Proposition 1. For $a, b \in \Sigma_k$ let

$$S(a, b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{otherwise} \end{cases}$$

and let $\delta = -1$. Then, for $x, y \in \Sigma_k^n$,

$$d_E(x, y) = n - S(x, y). \quad (9)$$

Proof. Considering a given alignment $(\mathcal{I}, \mathcal{J})$, we can write

$$\begin{aligned} n - S(x, y; \mathcal{I}, \mathcal{J}) &= n - \sum_{\ell=1}^s S(x[i_\ell], y[j_\ell]) - (-1)(n - s) \\ &= 2(n - s) + \left(s - \sum_{\ell=1}^s S(x[i_\ell], y[j_\ell]) \right) \\ &= 2(n - s) + \sum_{\ell=1}^s (x[i_\ell] \neq y[j_\ell]). \end{aligned}$$

Recalling Lemma 1, the quantity in the last row can be recognized as the cost of the simple scripts $\mathcal{S}$ transforming x into y , with alignment $a(\mathcal{S}) = (\mathcal{I}, \mathcal{J})$. Further recalling that optimal scripts are simple, we conclude that maximizing the score with respect to the alignment minimizes the cost of the edit script, whence the claimed Equality (9). $\square$

The preceding proposition enables the application of the following far reaching result to the analysis of the average edit distance.

Theorem 1 ([18]). *Let X and Y be random strings in Σ^n , whose symbols are all mutually independent and equally distributed. Let $l(n) = \frac{1}{n} E_{X,Y} S(X, Y)$, where E denotes the expectation operator, and let $l = \lim_{n \rightarrow \infty} l(n)$. Then*

$$l - l(n) \leq A \sqrt{\frac{2}{n-1} \left(\frac{n+1}{n-1} + \ln(n-1) \right)} + \frac{F}{n-1} := Q_{A,F}(n). \quad (10)$$

where $A = \max_{a,b \in \Sigma} S(a, b)$ and $F = \max_{a,b,c \in \Sigma} |S(a, b) - S(a, c)|$.

The preceding theorem makes no assumption on the probability distribution of the symbols. Moreover, $Q_{A,F}(n)$ is independent of such distribution, although the quantities $l(n)$ and l are not. Here, we assume the uniform distribution, upon which we have based the definition of $\alpha_k(n)$. As a corollary of Theorem 1 and Proposition 1, we obtain the computability of α_k .

Theorem 2. *For any integer $k > 0$, the limit constant α_k defined in Equation (4) satisfies the bound*

$$\alpha_k(n) - \alpha_k \leq \sqrt{\frac{2}{n-1} \left(\frac{n+1}{n-1} + \ln(n-1) \right)} + \frac{1}{n-1} := Q(n), \quad (11)$$

for $n \geq 2$. Therefore, α_k is a computable real number.

Proof. It is an exercise to see that, for the score function $S(a, b)$ of Proposition 1, $A = F = 1$, for any k . Correspondingly, we have $Q(n) = Q_{1,1}(n)$. We also observe that, with $\Sigma = \Sigma_k$, using Equation (9), we have

$$\begin{aligned} \alpha_k(n) &= \frac{1}{n} E_{X,Y} d_E(X, Y) = 1 - \frac{1}{n} E_{X,Y} S(X, Y) = 1 - l_k(n) \\ \alpha_k &= 1 - l_k. \end{aligned}$$

Thus, Equation (11) follows from Equation (10), with $Q(n) = Q_{1,1}(n)$, considering that $\alpha_k(n) - \alpha_k = (1 - l_k(n)) - (1 - l_k) = l_k - l_k(n)$. Finally, it is straightforward to prove that there is an integer $\bar{n}$ such that $Q(n)$ is strictly decreasing for $n \geq \bar{n}$ and that $n_\epsilon = \min\{n \geq \bar{n} : Q(n) < \epsilon\}$ is a computable function of the rational number ϵ . $\square$

Interestingly, $Q(n)$, hence n_ϵ , is independent both of the alphabet size, k , and of the probabilities of symbols in the alphabet Σ . However, computing $\alpha_k(n_\epsilon)$, to obtain a deterministically guaranteed ϵ -approximation of α_k , will require work increasing with k , at least with the currently known approaches that are sensitive to the number k^n of strings of length n , as we will see in the coming sections. Moreover, n_ϵ increases more than quadratically with $1/\epsilon$ (see Equation (11)), making the approach completely impractical. More specifically, to obtain ν bits of α_k we need $\epsilon = 2^{-\nu}$, so that $n_\epsilon \geq 1/\epsilon^2 = 4^\nu$ and $k^{n_\epsilon} \geq k^{4^\nu}$, which is a triple exponential in the problem size $\log_2 \nu$.

Remark 1. *Although, both for simplicity and for homogeneity with other sections, we have stated Theorem 2 for the case of uniform distribution on the alphabet symbols, the arguments that have been developed are easily adapted to establish the computability for the analog of α_k , under any distribution. In fact, as already observed, $Q_{A,F}(n)$ is independent of the distribution and the quantities analog to the $l(n)$'s are trivially (although inefficiently) computable by summing the distances $d_E(x, y)$ over all pairs $x, y \in \Sigma_k^n$, duly weighted by their probabilities.*

We point out that, in the same spirit of this section, the computability of γ_k can be derived from any of the rate-of-convergence bounds given by [14,17,18].

4. Monte Carlo Estimates of α_k

In this section, motivated by the difficulty of the exact computation, we develop an analysis of Monte Carlo estimates of $\alpha_k(n)$, by sampling, and translate them into estimates of α_k , using Theorem 2. We will see how $\alpha_k(n)$ can be estimated with high confidence and good accuracy for values of n up to a quarter million, with less than one core-hour of computation. For α_k , we achieve an error of the order of 10^{-2} .

Intuitively, for fixed k and n , we expect the estimate error to be proportional to $1/\sqrt{N}$, where N is the number of samples (string pairs), and to the standard deviation $S_k(n)/n$ of the single sample $d_E(x, y)/n$. Experimentally, this standard deviation appears to decrease a bit faster than $1/\sqrt{n}$. (We do not report here on this experimental observation systematically; however, estimates of $S_4(n)$ for some values of n can be found in Table 3.) with the implication that, to obtain the same error on $\alpha_k(n)$, fewer samples suffice for larger n . However, the behavior of the standard deviation does not appear easy to establish analytically. Fortunately, the edit distance function has the property that, if only one position is modified in just one of the input strings, the (absolute value of the) variation of the distance is at most 1. This property enables the use of McDiarmid’s inequality to bound from above the probability that $d_E(x, y)/n$ deviates from the mean by more than a certain amount, by a negative exponential in the square of that amount. This paves the way to the desired analysis. In fact, McDiarmid’s inequality can be applied directly to the average over N samples, dealing in a uniform way with the “averaging” effect within a single pair of strings and across multiple pairs.

Table 3. Estimates $\tilde{e}_4(n, N)$ of the average edit distance and $\tilde{\alpha}_4(n, N) = \frac{1}{n}\tilde{e}_4(n, N)$ of the average distance per symbol $\alpha_4(n)$, for various string lengths n , based on $N=2^{39}/n^2$ samples. The last column shows the confidence intervals for $\alpha_4(n)$ corresponding to confidence level 99.9%. $\tilde{S}_4(n, N)$ is the sample standard deviation of the single-pair distance.

n	N	$\Delta_{99.9\%}(n, N)$	$\tilde{e}_4(n, N)$	$\tilde{S}_4(n, N)$	$\tilde{\alpha}_4(n, N)$	99.9% Conf. Int.
2^8	2^{23}	0.59×10^{-4}	138.10	3.838	0.53946	[0.53940, 0.53953]
2^9	2^{21}	0.84×10^{-4}	272.10	4.920	0.53144	[0.53135, 0.53153]
2^{10}	2^{19}	0.12×10^{-3}	538.77	6.307	0.52614	[0.52602, 0.52626]
2^{11}	2^{17}	0.17×10^{-3}	1070.4	8.146	0.52263	[0.52246, 0.52280]
2^{12}	2^{15}	0.24×10^{-3}	2131.5	10.56	0.52039	[0.52015, 0.52063]
2^{13}	2^{13}	0.34×10^{-3}	4250.9	13.62	0.51891	[0.51857, 0.51925]
2^{14}	2^{11}	0.48×10^{-3}	8487.0	17.71	0.51801	[0.51753, 0.51849]
2^{15}	2^9	0.67×10^{-3}	16,954	24.98	0.51739	[0.51671, 0.51807]
2^{16}	2^7	0.95×10^{-3}	33,884	29.12	0.51704	[0.51608, 0.51799]
2^{17}	2^5	0.13×10^{-2}	67,734	38.85	0.51677	[0.51542, 0.51812]
2^{18}	2^3	0.19×10^{-2}	135,450	62.94	0.51670	[0.51479, 0.51861]

Preview of results

To help navigate this section, we preview its results, with a preliminary interpretation.

In Section 4.1, we present Monte Carlo estimates of $\alpha_k(n)$ for which we develop confidence intervals based on McDiarmid’s inequality. The results for the quaternary alphabet (column $\tilde{\alpha}_4(n, N)$ of Table 3) show that these values converge from above to α_k . We observe that the differences $\tilde{\alpha}_4(n, N) - \tilde{\alpha}_4(2n, N)$ decrease with n , suggesting that the estimates become more precise as n increases, even if the confidence intervals become wider, due to the smaller sample size N , manageable within a given computational budget.

The dependence of $\alpha_k(n)$ upon k is investigated Table 4. Here, all confidence intervals have the same width, since the same values of n and N are used for different k 's.

Table 4. Estimates $\tilde{e}_k(n, N)$ of the average edit distance and $\tilde{\alpha}_k(n, N) = \frac{1}{n}\tilde{e}_k(n, N)$ of the average distance per symbol $\alpha_k(n)$, for various alphabet sizes k , based on $N = 2^9$ samples of pairs of strings with length $n = 2^{15}$. The confidence radius is $\Delta_{99.9\%}(n, N) = \Delta_{99.9\%}(2^{15}, 2^9) \approx 0.67 \times 10^{-3}$; the corresponding confidence intervals for $\alpha_k(n)$ are given in the last column. $\tilde{S}_k(n, N)$ is the sample standard deviation of the single-pair distance.

k	$\tilde{e}_k(n, N)$	$\tilde{S}_k(n, N)$	$\tilde{\alpha}_k(n, N)$	99.9% Conf. Int.
2	9442.6	26.04	0.28817	[0.28749, 0.28884]
3	14,042	24.88	0.42852	[0.42784, 0.42920]
4	16,954	24.98	0.51739	[0.51671, 0.51807]
5	19,005	22.78	0.57998	[0.57930, 0.58066]
6	20,549	22.00	0.62710	[0.62642, 0.62778]
7	21,761	21.05	0.66409	[0.66341, 0.66477]
8	22,742	20.15	0.69402	[0.69334, 0.69470]
16	26,839	16.38	0.81906	[0.81838, 0.81974]
32	29,471	14.10	0.89939	[0.89871, 0.90007]

In Section 4.2, we derive estimates of α_k (the limit of $\alpha_k(n)$) and corresponding confidence intervals. These estimates for $k = 4$ (column $\hat{\alpha}_k(n, N)$ in Table 5) are obtained as the center point of a confidence interval depending on both statistical interval and the bound $Q(n)$ on $\alpha_k(n) - \alpha_k$. Interestingly, the width of such interval is largely due to $Q(n)$. Furthermore, the bound appears to be loose, since the values $\hat{\alpha}_k(n, N)$ converge to α_k from below, whereas the $\tilde{\alpha}_k(n, N)$ converge from above. Table 6 presents estimates of α_k for various k , along with confidence intervals (all of equal width, as the same n and N have been fixed for all k 's). The comparison between statistical estimates of $\alpha_k(n)$ and confidence intervals for α_k is also visually illustrated in Figure 1.

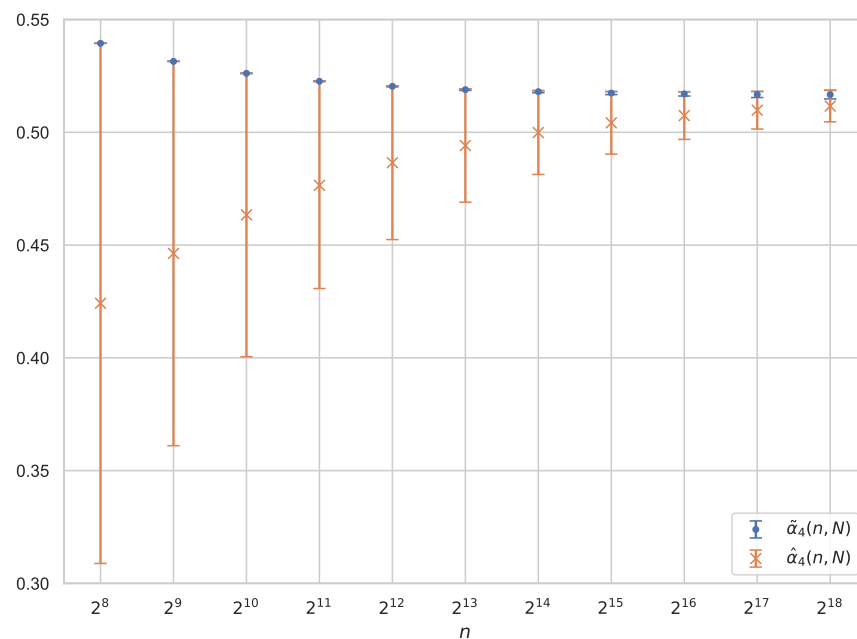


Figure 1. Comparison between estimates $\tilde{\alpha}_4(n, N)$ of $\alpha_k(n)$ (blue dots) and estimates $\hat{\alpha}_4(n, N)$ of α_4 (orange crosses). For each value, the confidence interval is shown.

Table 5. Estimates $\hat{\alpha}_4(n, N)$ of α_4 with confidence intervals for various string lengths n , based on $N = 2^{39}/n^2$ samples (so that each estimate takes approximately the same computation time). $I_4(n, N, \Delta)$ is the corresponding 99.9% confidence interval for α_4 . The estimates improve with growing n , as witnessed by the decrease of the interval radius $R_{0.999}(n, N)$.

n	N	$R_{0.999}(n, N)$	$\hat{\alpha}_4(n, N)$	$I_4(n, N, \Delta)$
2^8	2^{23}	0.11534	0.42418	[0.30884, 0.53953]
2^9	2^{21}	0.08523	0.44629	[0.36105, 0.53153]
2^{10}	2^{19}	0.06287	0.46338	[0.40051, 0.52626]
2^{11}	2^{17}	0.04631	0.47649	[0.43075, 0.52280]
2^{12}	2^{15}	0.03409	0.48654	[0.45244, 0.52063]
2^{13}	2^{13}	0.02512	0.49413	[0.46900, 0.51925]
2^{14}	2^{11}	0.01858	0.49990	[0.48132, 0.51849]
2^{15}	2^9	0.01388	0.50419	[0.49031, 0.51807]
2^{16}	2^7	0.01056	0.50742	[0.49685, 0.51799]
2^{17}	2^5	0.00833	0.50979	[0.50145, 0.51812]
2^{18}	2^3	0.00698	0.51163	[0.50465, 0.51861]

4.1. $\alpha_k(n)$

Given N random and independent pairs of strings $(X_1, Y_1), \dots, (X_N, Y_N)$ from Σ_k^n , we consider the random variable

$$\tilde{\alpha}_k(n, N) = \frac{1}{n} \tilde{e}_k(n, N) = \frac{1}{nN} \sum_{i=1}^N d_E(X_i, Y_i). \quad (12)$$

Clearly, $E[\tilde{\alpha}_k(n, N)] = \alpha_k(n)$. To assess the quality of $\tilde{\alpha}_k(n, N)$ as an estimate for $\alpha_k(n)$, we resort to McDiarmid's inequality, briefly reviewed next.

Proposition 2 ([35]). Let $\mathbf{Z} = (Z_1, \dots, Z_{2m})$ be a vector of $2m$ independent random variables. Let $f(\mathbf{Z})$ be a real function and let $B > 0$ be a real constant such that, if $\mathbf{Z}$ and $\mathbf{Z}'$ differ in at most one component, then $|f(\mathbf{Z}) - f(\mathbf{Z}')| \leq B$. Then, for every $\Delta \geq 0$,

$$\Pr[|f(\mathbf{Z}) - E[f(\mathbf{Z})]| > \Delta] \leq \exp\left(-\frac{\Delta^2}{mB^2}\right). \quad (13)$$

In the present context, based on the previous proposition, we can formulate confidence intervals for $\alpha_k(n)$, as follows.

Proposition 3 (Confidence intervals for $\alpha_k(n)$). For any $\Delta \geq 0$, the (random) interval $[\tilde{\alpha}_k(n, N) - \Delta, \tilde{\alpha}_k(n, N) + \Delta]$ is a confidence interval for the parameter $\alpha_k(n)$, with confidence level $1 - 2 \exp(-Nn\Delta^2)$, that is:

$$\Pr[\tilde{\alpha}_k(n, N) - \Delta \leq \alpha_k(n) \leq \tilde{\alpha}_k(n, N) + \Delta] \geq 1 - 2 \exp(-Nn\Delta^2). \quad (14)$$

Proof. We apply Proposition 2 to the function f given by the rightmost term in Equation (12), with $\mathbf{Z}$ being the concatenation the $2N$ strings x_i 's and y_i 's, each comprising n variables (over Σ_k), so that $m = Nn$. We can set $B = \frac{1}{Nn}$, since changing one string position changes the sum of the N edit distances by at most 1, and the quantity $1/(Nn)$ times the sum by at most $1/(Nn)$. Therefore, considering that $mB^2 = (Nn)(1/(Nn)^2) = 1/(Nn)$, we can write:

$$\Pr[\tilde{\alpha}_k(n, N) - \alpha_k(n) > \Delta] \leq \exp(-Nn\Delta^2). \quad (15)$$

Symmetrically, it can be shown that

$$\Pr[\tilde{\alpha}_k(n, N) - \alpha_k(n) < \Delta] \leq \exp(-Nn\Delta^2). \quad (16)$$

Combining Inequalities (15) and (16), after simple algebra, yields Inequality (14). $\square$

Remark 2. Propositions 2 and 3 only require that the symbols of the random strings are statistically independent, not necessarily with the same distribution. The numerical results reported next refer to the special case where all symbols are uniformly distributed, which underlies the definition of $\alpha_k(n)$ and $e_k(n)$. However, the approach could be straightforwardly applied to other, possibly position-dependent, distributions. Interestingly, for given n and N , the width of the confidence interval, for a given confidence level (e.g., $\Delta_{99.9\%}(n, N)$ in Table 3), is independent of the distribution.

Numerical Results

Table 3 reports estimates based on Monte Carlo estimates, within the framework of Proposition 3. The alphabet size is $k = 4$, a case of special interest in DNA analysis (e.g., [1,36]). For the string length n , the values considered are the powers of two from $2^8 = 256$ to $2^{18} = 262,144$. For each n , the number of samples N has been chosen as $N = 2^{39}/n^2$. This choice (roughly) equalizes the amount of (sequential) computation time devoted to each n , when the edit distance for a sample pair is computed by a quadratic algorithm, say, in time $T_{ed}(n) = \tau_{ed}n^2$, for some constant τ_{ed} , making the overall time for N samples $T(n, N) = \tau_{ed}Nn^2$. In our experiments, this becomes $T(n, 2^{39}/n^2) = \tau_{ed}2^{39} \approx 2560 \text{ s} \approx 43 \text{ min}$, where we measured $\tau_{ed} \approx 1.25 \times 2^{-28} \text{ s} \approx 4.66 \text{ ns}$, on a state-of-the-art processor core. From Inequality (14), straightforward manipulations show that, if the target is a confidence level λ , then the radius Δ of the confidence interval becomes:

$$\Delta_\lambda(n, N) = \sqrt{\frac{1}{Nn} \ln\left(\frac{2}{1-\lambda}\right)}. \quad (17)$$

Choosing $\lambda = 0.999$ and recalling that, in our experiments, we have set $N = 2^{39}/n^2$, the above formula becomes

$$\Delta_{99.9\%}(n, 2^{39}/n^2) = \sqrt{2^{-39}n \ln(2000)} = 2^{-20}3.90\sqrt{n}. \quad (18)$$

Thus, for $n = 2^8$, we have $\Delta_{99.9\%} = 0.59 \times 10^{-4}$. For $n = 2^{18}$, we have $\Delta_{99.9\%} = 0.19 \times 10^{-2}$. The values of $\Delta_{99.9\%}$ have been used, together with the experimental values of $\tilde{\alpha}_4(n, N)$, to obtain the confidence intervals reported in the last column of Table 3, as $[\tilde{\alpha}_4(n, N) - \Delta_{99.9\%}(n, N), \tilde{\alpha}_4(n, N) + \Delta_{99.9\%}(n, N)]$.

Table 4 reports estimates of $\alpha_k(n)$ for various alphabet sizes k . The confidence intervals shown in the last column of the table are based on the confidence level $\lambda = 0.999$. The estimates are obtained from numerical experiments involving $N = 2^9$ random pairs of strings of length $n = 2^{15}$. Since the radius of the confidence interval only depends on λ , n , and N (but not k), the same value $\Delta_{99.9\%}(2^{15}, 2^9) \approx 0.67 \times 10^{-3}$ applies to all k 's. Notice that the standard deviation $\tilde{S}_k(n, N)$ tends to decrease with k ; this makes intuitive sense since, for fixed n , as k increases, the probability that every symbol that appears in one string is distinct from every symbol that appears in the other string approaches 1, so that almost all pairs of strings have distance n , hence the variance is negligible.

4.2. α_k

The following proposition provides confidence intervals for α_k , when the quantity $\tilde{\alpha}_k(n, N) - Q(n)/2$ is used as an estimator, where $Q(n)$ bounds the rate of convergence, according to Equation (11).

Proposition 4 (Confidence intervals for α_k). *For any $\Delta \geq 0$, the (random) interval centered at $\tilde{\alpha}_k(n, N) - Q(n)/2$ and of radius $\Delta + Q(n)/2$, i.e.,*

$$I_k(n, N, \Delta) = [(\tilde{\alpha}_k(n, N) - Q(n)/2) - (\Delta + Q(n)/2), (\tilde{\alpha}_k(n, N) - Q(n)/2) + (\Delta + Q(n)/2)]$$

is a confidence interval for α_k , with confidence level $1 - 2 \exp(-Nn\Delta^2)$, i.e.,

$$\Pr[\alpha_k \in I_k(n, N, \Delta)] \geq 1 - 2 \exp(-Nn\Delta^2). \quad (19)$$

Proof. By Theorem 2 and the fact that $\alpha_k < \alpha_k(n)$, we have that $\alpha_k(n) - Q(n) \leq \alpha_k \leq \alpha_k(n)$. By Inequality (14), this implies

$$\Pr[\tilde{\alpha}_k(n, N) - Q(n) - \Delta \leq \alpha_k \leq \tilde{\alpha}_k(n, N) + \Delta] \geq 1 - 2 \exp(-Nn\Delta^2).$$

To arrive at Inequality (19), it remains to observe that the interval within the probability is just a rewriting of $I_k(n, N, \Delta)$. $\square$

From Proposition 4, straightforward manipulations show that, if the target confidence level is λ , then the radius $R_\lambda(n, N)$ of the confidence interval becomes:

$$R_\lambda(n, N) = \frac{Q(n)}{2} + \sqrt{\frac{1}{Nn} \ln\left(\frac{2}{1-\lambda}\right)}. \quad (20)$$

The first term arises because $\alpha_k(n)$ is a (deterministically bounded) approximation to α_k and the second term because $\tilde{\alpha}_k(n, N)$ is a (statistically bounded) approximation to $\alpha_k(n)$. To get a sense of the relative weight of the two terms contributing to the radius, we observe that, from Equation (11), we can approximate the first term as $\frac{Q(n)}{2} \approx \sqrt{\frac{\ln n}{2n}}$. Therefore, even using just one sample pair ($N = 1$), this term dominates the second one as soon as $n > \left(\frac{2}{1-\lambda}\right)^2$. For $\lambda = 0.999$, we get $n > 4 \times 10^6$.

Remark 3. It is interesting to observe that the size $2R_\lambda(n, N)$ of the confidence interval is independent of the alphabet size k . The same applies to the computational work, $T(n, N) = \tau_{ed}Nn^2 + l.o.t.$, to obtain the statistical estimate of $\alpha_k(n)$ with a given confidence.

Our estimate of α_k is the center of the interval $I_k(n, N, \Delta)$ as defined in Proposition 4

$$\hat{\alpha}_k(n, N, \Delta) = \tilde{\alpha}_k(n, N) - Q(n)/2. \quad (21)$$

Table 5 reports these estimates for $k = 4$, based on the values $\tilde{\alpha}_4(n, N)$ reported in Table 3 and on Proposition 4 for confidence level $\lambda = 0.999$.

Table 6 reports estimates of α_k , for various values of k , all based on numerical experiments with $N = 2^9$ samples of string pairs of size $n = 2^{15}$. The corresponding confidence interval $I_k(n, N, \Delta)$ is obtained at confidence level $\lambda = 0.999$. All intervals have radius $R_{0.999}(2^{15}, 2^9) \approx 1.4 \times 10^{-2}$.

Table 6. Estimates of α_k for various alphabet sizes k , based on $N = 2^9$ samples of pairs of string with length $n = 2^{15}$. The radius of the interval $I_k(n, N, \Delta)$ is based on a $\lambda = 0.999$ confidence level and is $R_{0.999}(2^{15}, 2^9) \approx 1.4 \times 10^{-2}$, for all values of k .

k	$\hat{\alpha}_k(n, N)$	$I_k(n, N, \Delta)$
2	0.27496	[0.26108, 0.28884]
3	0.41532	[0.40144, 0.42920]
4	0.50419	[0.49031, 0.51807]
5	0.56678	[0.55289, 0.58066]
6	0.61390	[0.60002, 0.62778]
7	0.65089	[0.63701, 0.66477]
8	0.68082	[0.66694, 0.69459]
16	0.80586	[0.79198, 0.81974]
32	0.88619	[0.87230, 0.90007]

5. Upper Bounds for α_k

In this section, we present methods to derive upper bounds to α_k based on the exact computation of $\alpha_k(n) = e_k(n)/n$ for some n , and on the relation $\alpha_k \leq \alpha_k(n)$, valid for all $n \geq 1$. The computation of $e_k(n)$ can be reduced to that of the eccentricity, as in Equation (3) repeated here for convenience:

$$e_k(n) = k^{-n} \sum_{x \in \Sigma_k^n} \text{ecc}(x). \quad (22)$$

If $\text{ecc}(x)$ is computed according to Equation (2) and the distance $d_E(x, y)$ is computed by the $\mathcal{O}(n^2)$ -time dynamic programming algorithm for each of the k^n strings $y \in \Sigma_k^n$, then the overall computation time is $\mathcal{O}(n^2 k^n)$ for $\text{ecc}(x)$ and $\mathcal{O}(n^2 k^{2n})$ for $e_k(n)$, since the eccentricity of each of the k^n strings $x \in \Sigma_k^n$ is needed in Equation (22). Below, we propose a more efficient algorithm to speed up the computation of $\text{ecc}(x)$ and, in turn, that of $e_k(n)$, achieving time $\mathcal{O}(n^2 \min(k, 3)^n k^n) = \mathcal{O}(n^2 3^n k^n)$. We also show how to exploit some symmetries of $\text{ecc}(x)$ in order to limit the computation of the eccentricity needed to obtain $e_k(n)$ to a suitable subset of Σ_k^n .

5.1. The Coalesced Dynamic Programming Algorithm for Eccentricity

Let $\mathbf{M}(x, y)$ be the matrix produced by the dynamic programming algorithm (reviewed in Section 2.2) to compute $d_E(x, y)$, with $x, y \in \Sigma_k^n$. We develop a strategy to coalesce the computations of $\mathbf{M}(x, y)$ for different $y \in \Sigma_k^n$, while keeping x fixed. To this end, we choose to generate the entries of $\mathbf{M}(x, y)$, according to Equation (5), in column-major order. Clearly, column j is fully determined by x and by the prefix of y of length j . Define now the *column multiset* $\mathcal{C}_j$ containing column j (i.e., the last one) of $\mathbf{M}(x, y[1] \dots y[j])$ for each string $y[1] \dots y[j] \in \Sigma_k^j$. The multiset $\mathcal{C}_j$ is a function of (just) x , although, for simplicity, the dependence upon x is not reflected in our notation. Clearly, each column in $\mathcal{C}_{j-1}$ generates k columns in $\mathcal{C}_j$, one for each symbol of Σ_k ; therefore the cardinality of $\mathcal{C}_j$ is $|\mathcal{C}_j| = k^j$. However, several columns may be equal to each other, so that the number of distinct such columns can be much smaller. In fact, we will show that this number is upper bounded by 3^n , which is smaller than k^j for $j > \frac{\log_2 3}{\log_2 k} n$. This circumstance can be exploited to save both space and computation, by representing $\mathcal{C}_j$ as a set of records each containing a distinct column and its multiplicity. Intuitively, we are coalescing the computation of the dynamic programming matrices corresponding to different strings, when such matrices happen to have the same j -th column.

The *Coalesced Dynamic Programming* (CDP) algorithm described next (referring also to the line numbers of Algorithm 1), constructs the sequence of multisets $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_n$.

A column multiset $\mathcal{C}$ will be represented as a set of pairs $(\mathbf{c}, \mu(\mathbf{c}))$, one for each distinct member $\mathbf{c}$, with $\mu(\mathbf{c})$ being the multiplicity of $\mathbf{c}$ in $\mathcal{C}$. The eccentricity of x is obtained (lines 13–17) as the weighted average of the n -th elements of all columns in $\mathcal{C}_n$:

$$\text{ecc}(x) = k^{-n} \sum_{\mathbf{c} \in \mathcal{C}_n} \mu(\mathbf{c}) \mathbf{c}[n]. \quad (23)$$

As can be seen from Equation (5), multiset $\mathcal{C}_0$ contains the column $(0, 1, \dots, n)$, with multiplicity 1 (line 3). For $j = 1, \dots, n$, $\mathcal{C}_j$ is obtained by scanning all $\mathbf{c} \in \mathcal{C}_{j-1}$ (line 6) and all $b \in \Sigma_k$ (line 7), and by

- computing the j -th column $\mathbf{c}'$ resulting from Equation (5) when the $(j-1)$ -st column is $\mathbf{c}$ and $\xi_{i,j} = 0$ if $x[i] = b$ or else $\xi_{i,j} = 1$ (call to $\text{NEXTCOLUMN}(x, \mathbf{c}, j, b)$, line 8);
- inserting $\mu(\mathbf{c})$ copies of $\mathbf{c}'$ in $\mathcal{C}_j$, by either creating a new pair $(\mathbf{c}', \mu(\mathbf{c}))$ when $\mathbf{c}'$ is not present in the multiset or by incrementing its multiplicity by $\mu(\mathbf{c})$ otherwise (call to $\text{INSERT}(\mathcal{C}_j, (\mathbf{c}', \mu(\mathbf{c})))$, line 9).

The correctness of the CDP algorithm is pretty straightforward to establish. A few observations are however necessary in order to describe and analyze a data structure that can efficiently implement, in our specific context, multisets with the insertion operation. The key property is that, for $j = 0, 1, \dots, n$, the column of $\mathbf{M}(x, y)$ with index j satisfies the conditions (a) $M_{0,j} = j$ and (b) $(M_{i,j} - M_{i-1,j}) \in \{-1, 0, 1\}$, for $i = 1, \dots, n$. Using this property, the set of distinct columns that belong to the multiset $\mathcal{C}_j$ can be represented as a ternary tree where each arc has a label from the set $\{-1, 0, 1\}$ and a column $(M_{0,j}, M_{1,j}, \dots, M_{n,j})$ is mapped to a leaf v such that the n arcs in the path from the root to v have labels $(M_{1,j} - M_{0,j}), \dots, (M_{n,j} - M_{n-1,j})$. Each leaf stores the multiplicity of the corresponding column. The size of the tree for $\mathcal{C}_j$ is $\mathcal{O}(\min(3^n, k^j))$, since there are at most 3^n columns satisfying the constraints and k^j k -ary strings that contribute (not necessarily distinct) columns. Hence, the body of the loop, whose iteration range is defined in lines 4, 6, and 7, is executed $nk\mathcal{O}(\min(3^n, k^j))$ times. Considering that one call to $\text{NEXTCOLUMN}()$ as well as one call to $\text{INSERT}()$ can be easily performed in $\mathcal{O}(n)$ time, we can summarize the previous discussion as follows, where we also consider that, at any given time, the algorithm only needs to store two consecutive column multisets.

Algorithm 1 Coalesced dynamic programming algorithm to compute $\text{ecc}(x)$

```

1: procedure ECCENTRICITY( $x$ )
2:    $n \leftarrow |x|$ 
3:    $\mathcal{C}_0 \leftarrow \{((0, 1, \dots, n), 1)\}$ 
4:   for  $j \leftarrow 1$  to  $n$  do
5:      $\mathcal{C}_j \leftarrow \emptyset$ 
6:     for  $(\mathbf{c}, \mu(\mathbf{c})) \in \mathcal{C}_{j-1}$  do
7:       for  $b \in \Sigma_k$  do
8:          $\mathbf{c}' \leftarrow \text{NEXTCOLUMN}(x, \mathbf{c}, j, b)$ 
9:          $\text{INSERT}(\mathcal{C}_j, (\mathbf{c}', \mu(\mathbf{c})))$ 
10:      end for
11:    end for
12:  end for
13:   $e \leftarrow 0$ 
14:  for  $\mathbf{c} \in \mathcal{C}_n$  do
15:     $e \leftarrow e + \mu(\mathbf{c}) * \mathbf{c}[n]$ 
16:  end for
17:  return  $e/k^n$ 
18: end procedure

```

Proposition 5. The CDP algorithm computes the eccentricity $\text{ecc}(x)$ of a string x of length n over a k -ary alphabet in time $T = \mathcal{O}(n^2 k \min(3^n, k^n))$ and space $S = \mathcal{O}(\min(3^n, k^n))$. Correspondingly, the expected distance $e_k(n)$ can be computed in time $T = \mathcal{O}(n^2 k^{n+1} \min(3^n, k^n))$ and space $S = \mathcal{O}(\min(3^n, k^n))$.

Remark 4. The CDP algorithm can be easily generalized to handle the case where $\text{ecc}(x)$ is defined with respect to a random string y whose symbols are independently, but not necessarily uniformly, distributed. Moreover, the distribution does not need to be stationary. In other words, the probability of the string $y[1] \dots y[n]$ has the form $\prod_{j=1}^n p_j(y[j])$. Essentially, rather than maintaining the multiplicity of each column $\mu(\mathbf{c})$, we maintain its probability $\pi(\mathbf{c})$. The rules to obtain C_j from C_{j-1} are straightforward. The eccentricity is obtained as $\sum_{\mathbf{c} \in \mathcal{C}_n} \pi(\mathbf{c}) \mathbf{c}[n]$. This generalized version of the CDP algorithm naturally enables the computation of the expected edit distance between two random strings x and y whose symbols are all mutually independent. The computational bounds remain those stated in Proposition 5.

5.2. Exploiting Symmetries of $\text{ecc}(x)$ in the Computation of $e_k(n)$

The edit distance enjoys some useful symmetries, which can be easily derived from the definition. One is that, if we let $x^R = x[n] \dots x[1]$ denote the reverse of the string $x = x[1] \dots x[n]$, then $d_E(x, y) = d_E(x^R, y^R)$. Another one is that if $\pi : \Sigma_k \rightarrow \Sigma_k$ is a permutation of the alphabet symbols and $\pi(x)$ denotes the string $\pi(x[1]) \dots \pi(x[n])$, then $d_E(x, y) = d_E(\pi(x), \pi(y))$. The following is a simple, but useful corollary of these properties.

Proposition 6. For any $x \in \Sigma_k^n$, we have $\text{ecc}(x^R) = \text{ecc}(x)$. Furthermore, for any permutation π of Σ_k , we have $\text{ecc}(\pi(x)) = \text{ecc}(x)$.

It is useful to define the equivalence class of x as the set of strings that have the same eccentricity as x , due to Proposition 6, and denote by $\nu(x)$ the cardinality of such set. If $\mathcal{R}_{k,n} \subseteq \Sigma_k^n$ contains exactly one (representative) member for each equivalence class, then Equation (22) can be rewritten as

$$e_k(n) = k^{-n} \sum_{x \in \mathcal{R}_{k,n}} \nu(x) \text{ecc}(x). \quad (24)$$

Computing $e_k(n)$ according to Equation (24) enables one to reduce the number of strings for which the eccentricity has to be computed (via the CDP algorithm) by a factor slightly smaller than $(2k!)$, with a practically appreciable reduction in computation time.

The strategy outlined in this section has been implemented in C++ and run on a 32 core IBM Power7 server. For several alphabet sizes k , we have considered values of n up to a maximum value n_k^{ub} , under the constraint that the running time would not exceed one week. The resulting values $e_k(n_k^{ub})$ are presented in Table 7. For the quaternary alphabet, we obtain $\alpha_4 \leq 0.6318$, which is rather loose because based on a small string length, namely $n_4^{ub} = 15$. The limitation on the string length is obviously due to the high complexity of the algorithm. In contrast, the statistical estimate presented in Section 4 is based on much longer strings and represents a more accurate approximation of α_4 , although the estimate comes with a confidence interval, rather than with a deterministic guarantee.

We mention that the CDP algorithm could perhaps be improved, by a constant factor related to the machine word-size, with the bit-vector approach presented in [21], or by a factor logarithmic with n , with the Four Russians method in [20]. Considering the exponential nature of the CDP algorithm, however, these approaches are unlikely to yield substantial improvements of the deterministic upper bound and we have not pursued them.

Table 7. Values $\alpha_k(n)$ computed using the Coalesced Dynamic Programming algorithm for various alphabets. The string lengths n_k^{ub} 's have been chosen so that the total time needed to compute $\alpha_k(n)$ is roughly the same for each k (approximately one week on a machine with 32 cores).

k	n_k^{ub}	$\alpha_k(n_k^{ub})$
2	24	0.36932
3	17	0.53426
4	15	0.63182
5	13	0.70197
6	12	0.75149
7	11	0.79031
8	11	0.81166
16	10	0.89554
32	6	0.96588

6. Lower Bounds for α_k

In this section, we establish a lower bound β_k^* to α_k , for each $k \geq 2$. We first characterize β_k^* analytically, as the supremum of a suitably defined set of real numbers, and then provide an efficient algorithm to compute β_k^* , within any desired approximation. We focus on the expression of $e_k(n)$ in terms of eccentricities, given in the second line of Equation (3). We derive lower bounds to $\text{ecc}(x)$ by ignoring the contribution of the strings inside the ball of radius r centered at x and by setting to $r + 1$ the contribution of the string outside the same ball. The objective is to determine the largest value $r_k^*(n)$ of r for which (it can be shown that) the ball of radius r contains a fraction of Σ_k^n that vanishes with n ; then $r_k^*(n)/n$ will converge to a lower bound to α_k . Below, we formalize this idea and show that we can choose $r_k^*(n) = \beta_k n$ for suitable values of β_k independent of n ; this establishes that $\alpha_k \geq \beta_k^*$, where β_k^* is the supremum of such values. It is shown that $\lim_{k \rightarrow \infty} \beta_k^* = 1$ and $\beta_k^* \leq \alpha_k \leq 1 - \frac{1}{k}$, whence, for k large enough, β_k^* provides an increasingly accurate estimate of α_k . Thus, we turn our attention to translating the analytical characterization of β_k^* into an efficient numerical algorithm for its computation, a translation which is not completely straightforward. Finally, we present the numerical values of our lower bound for a sample of alphabet sizes.

6.1. Lower Bounds to $\text{ecc}(x)$ from Upper Bounds to Ball Size

In this subsection, we derive lower bounds to $\text{ecc}(x)$ based on upper bounds to the size of the ball of radius r centered at x . Such bounds hold for every string x , but depend only upon the length n of x . They will be used to compute lower bounds to α_k .

Definition 3. For a string $x \in \Sigma_k^n$, the ball of radius r centered at x is defined as the set of strings having edit distance at most r from x :

$$B_{k,r}(x) = \{y \in \Sigma_k^n : d_E(x, y) \leq r\}.$$

Similarly, the shell of radius r centered at x is defined as the set of strings having edit distance exactly r from x :

$$S_{k,r}(x) = \{y \in \Sigma_k^n : d_E(x, y) = r\}.$$

The next lemma shows how an upper bound to the ball size can provide a lower bound to the eccentricity, thus motivating the derivation of such an upper bound.

Lemma 2. Let $u_{k,r}(x) \geq |B_{k,r}(x)|$; then for every $r^* = 0, 1, \dots, n$:

$$\text{ecc}(x) \geq r^*(1 - k^{-n}u_{k,r^*}(x)). \quad (25)$$

Proof. By partitioning Σ_k^n into shells centered at x , we can rewrite (2) as

$$\begin{aligned} \text{ecc}(x) &= k^{-n} \sum_{r=0}^{r^*} r |S_{k,r}(x)| + k^{-n} \sum_{r=r^*+1}^n r |S_{k,r}(x)| \\ &\geq k^{-n}(r^* + 1) \sum_{r=r^*+1}^n |S_{k,r}(x)| \\ &= k^{-n}(r^* + 1)(|B_{k,n}(x)| - |B_{k,r^*}(x)|) \\ &> r^*(1 - k^{-n}|B_{k,r^*}(x)|) \\ &\geq r^*(1 - k^{-n}u_{k,r^*}(x)), \end{aligned}$$

where, in the last two steps, the relationships $|B_{k,n}(x)| = k^n$ and $|B_{k,r^*}(x)| \leq u_{k,r^*}(x)$ have been utilized. $\square$

The bound $u_{k,r^*}(x)$ we derive below depends only upon the length n of x so that it can be written as $u_{k,r^*}(n)$, with a harmless overloading of notation. Then, simple manipulations of Equation (3) show that

$$\alpha_k(n) = \frac{e_k(n)}{n} \geq \frac{r^*}{n}(1 - k^{-n}u_{k,r^*}(n)); \quad (26)$$

$$\alpha_k = \lim_{n \rightarrow \infty} \alpha_k(n) \geq \lim_{n \rightarrow \infty} \frac{r^*}{n}(1 - k^{-n}u_{k,r^*}(n)). \quad (27)$$

We will show that, for suitable values of β_k , letting $r^* = \beta_k n$, the quantity $k^{-n}u_{k,\beta_k n}(n)$ converges to 0 whence, by Equation (27), $\alpha_k \geq \beta_k$.

6.2. Upper Bounds on Ball Size

To apply Lemma 2, we need an upper bound to $|B_{k,r}(x)|$. The next proposition develops such an upper bound by (i) showing that every string $y \in B_{k,r}(x)$ can be obtained from x by applying a script of certain type with cost r or $r - 1$ and (ii) counting such scripts. In general, the upper bound will not be tight, because the count may include multiple scripts that produce y from x .

Proposition 7. For any $x \in \Sigma_k^n$ and for any $r = 1, \dots, n$

$$|B_{k,r}(x)| \leq (k-1)^r \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d+1}{r-2d} \left(\frac{k}{(k-1)^2} \right)^d. \quad (28)$$

Proof. We call *canonical simple script* (CSS) a simple script (see Section 2.1) where all deletions precede all substitutions, the latter precede all insertions and, within each type of operation, cells are processed from left to right. For any script transforming x into y , there is a CSS of non greater cost which achieves the same transformation. Therefore, if $d_E(x, y) = r$, then there is a CSS of cost r which, applied to x , produces y . Each CSS of cost $r \in \{0, 1, \dots, n\}$ can be constructed by a sequence of choices, as specified below (shown within square brackets is the number of possible choices):

- $d \in \{0, 1, \dots, \lfloor r/2 \rfloor\}$
- d positions to delete from x $\left[\binom{n}{d}\right]$
- $(r - 2d)$ of the remaining $(n - d)$ positions to be substituted $\left[\binom{n-d}{r-2d}\right]$

- d positions to insert in y $\left[\binom{n}{d}\right]$
- the symbols in the substitutions $[(k-1)^{r-2d}]$
- the symbols in the insertions $[k^d]$

Straightforwardly, the number of CSSs of cost r is

$$s_{k,r} = \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d}{r-2d} (k-1)^{r-2d} k^d. \quad (29)$$

Next, we prove that any $y \in B_{k,r}(x)$ can be obtained from x via a simple script of cost $r-1$ or r . Let $r' = d_E(x, y) \leq r$. When $r' = r$, an optimal script of cost r' is also a simple script of the same cost. Hence, the canonical version of such optimal script can be used to obtain y from x . The same reasoning applies to the case $r' = r-1$. Finally, for $r' < r-1$, consider an optimal CCS of cost r' that transforms x into y . By augmenting this script with $\lfloor (r-r')/2 \rfloor$ pairs of deletions and insertions, each pair acting on a matched position, we obtain a simple script of cost r , if $r-r'$ is even, or of cost $r-1$ if $r-r'$ is odd. The prescribed augmentation is always possible since the number of matches is at least $n-r' \geq r-r' \geq (r-r')/2$.

The thesis is then established by the following chain of inequalities:

$$\begin{aligned} |B_{k,r}(x)| &\leq s_{k,r} + s_{k,r-1} \\ &\leq \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d}{r-2d} (k-1)^{r-2d} k^d + \sum_{d=0}^{\lfloor (r-1)/2 \rfloor} \binom{n}{d}^2 \binom{n-d}{r-1-2d} (k-1)^{r-1-2d} k^d \\ &\leq \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d}{r-2d} (k-1)^{r-2d} k^d + \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d}{r-1-2d} (k-1)^{r-2d} k^d \\ &= \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d+1}{r-2d} (k-1)^{r-2d} k^d, \end{aligned}$$

where we have made use of the identity

$$\binom{n-d}{r-2d} + \binom{n-d}{r-1-2d} = \binom{n-d+1}{r-2d}.$$

□

6.3. Asymptotic Behavior of Ball Size and Bounds for α_k

The next results show that the right hand side of Inequality (28), divided by k^n , is bounded by a sum of exponential functions whose exponents all vanish with n , when the ball radius is set to $\beta_k n$, with β_k satisfying certain conditions (depending upon k). Intuitively, this means that, except for a vanishing fraction, all strings in Σ_k^n lie outside of the ball $B_{k,\beta_k n}(x)$ whence, by Equation (27), $\alpha_k \geq \beta_k$.

Definition 4. Let $H(x)$, with $0 \leq x \leq 1$, denote the binary entropy function

$$H(x) = -x \log_2 x - (1-x) \log_2 (1-x),$$

and let $H'(x) = \frac{dH}{dx} = \log_2 \left(\frac{1-x}{x} \right)$ and $H''(x) = \frac{d^2H}{dx^2} = -\frac{\log_2 e}{x(1-x)}$ denote its first and second derivatives.

Definition 5. For $\beta \in [0, 1]$ and $\delta \in [0, \beta/2]$, we define the function

$$g_k(\beta, \delta) = (\beta - 2\delta) \log_2(k - 1) - (1 - \delta) \log_2 k + 2H(\delta) + (1 - \delta)H\left(\frac{\beta - 2\delta}{1 - \delta}\right). \quad (30)$$

Lemma 3. Let $u_{k,r}(n)$ be given by the right hand side of (28) and $g_k(\beta, \delta)$ be given by (30). For every $\beta \in [0, 1]$,

$$k^{-n}u_{k,\beta n}(n) \leq (n + 1) \sum_{d=0}^{\lfloor \beta n/2 \rfloor} 2^{ng_k(\beta, \frac{d}{n})}. \quad (31)$$

Proof. Using the relation

$$\binom{n-d+1}{r-2d} = \frac{n-d+1}{n-r+d+1} \binom{n-d}{r-2d} \leq (n+1) \binom{n-d}{r-2d},$$

the bound $\binom{n}{k} \leq 2^{nH(k/n)}$ (see, e.g., Equation (5.31) in [34]), and defining $\beta = r/n$, we get

$$\begin{aligned} k^{-n}u_{k,r}(n) &\leq k^{-n}(k-1)^r \sum_{d=0}^{\lfloor r/2 \rfloor} \binom{n}{d}^2 \binom{n-d+1}{r-2d} \left(\frac{k}{(k-1)^2}\right)^d \\ &\leq (n+1) \sum_{d=0}^{\lfloor r/2 \rfloor} 2^{2nH(\frac{d}{n}) + (n-d)H(\frac{r-2d}{n-d}) + (r-2d)\log_2(k-1) + (d-n)\log_2 k} \\ &= (n+1) \sum_{d=0}^{\lfloor \beta n/2 \rfloor} 2^{ng_k(\beta, \frac{d}{n})}. \end{aligned}$$

□

Theorem 3. For integer $k \geq 2$ and real $\beta \in [0, 1]$, define the real function

$$G_k(\beta) = \max_{0 \leq \delta \leq \beta/2} g_k(\beta, \delta) \quad (32)$$

and the set of real numbers

$$A_k = \{\beta \in [0, 1] : G_k(\beta) < 0\}. \quad (33)$$

Then,

$$\alpha_k \geq \beta_k^* := \sup A_k.$$

Proof. First, we observe that the definition of $G_k(\beta)$ is well posed; in fact, for any fixed $\beta \in [0, 1]$, the function $g_k(\beta, \delta)$ is bounded and continuous with respect to δ , hence it attains a maximum value in the compact set $0 \leq \delta \leq \beta/2$ (by Weierstrass Theorem).

Second, we observe that A_k is not empty, since $G_k(0) < 0$. In fact, when $\beta = 0$, the condition $\delta \in [0, \beta/2]$ is satisfied only by $\delta = 0$, and $g_k(0, 0) = -\log_2 k < 0$, for any $k \geq 2$. Finally, since $A_k \subseteq [0, 1]$, then $\sup A_k \leq 1$.

For $\beta \in A_k$, letting $f(n) = (n + 1) \left(\left\lfloor \frac{\beta n}{2} \right\rfloor + 1 \right)$, we see from Lemma 3 that

$$k^{-n}u_{k,\beta n}(n) \leq (n + 1) \sum_{d=0}^{\lfloor \beta n/2 \rfloor} 2^{ng_k(\beta, \frac{d}{n})} \leq f(n)2^{nG_k(\beta)},$$

where we have used the relation $g_k\left(\beta, \frac{d}{n}\right) \leq G_k(\beta)$. The latter follows from the definition of $G_k(\beta)$ and the fact that, in each of the $\lfloor \frac{\beta n}{2} \rfloor + 1$ terms of the summation, $0 \leq \frac{d}{n} \leq \beta/2$. Taking now the limit in (27) with $r^* = \beta n$ yields:

$$\alpha_k \geq \lim_{n \rightarrow \infty} \beta \left(1 - f(n)2^{nG_k(\beta)}\right) = \beta,$$

as $f(n) = \mathcal{O}(n^2)$ and $2^{nG_k(\beta)}$ is a negative exponential. In conclusion, since α_k is no smaller than any member of A_k , it is also no smaller than $\beta_k^* = \sup A_k$. $\square$

As a first application of Theorem 3, we obtain an analytical lower bound to each α_k . This bound is generally not the best that can be obtained numerically from the theorem, but does provide some insight. In particular, it shows that, as k grows, both β_k^* and α_k approach 1.

Proposition 8. *Let the constant M be defined as*

$$M = \max_{\substack{0 \leq \beta \leq 1 \\ 0 \leq \delta \leq \beta/2}} \left(2H(\delta) + (1 - \delta)H\left(\frac{\beta - 2\delta}{1 - \delta}\right)\right) \approx 2.52$$

Then, for any $k \geq 3$, we have

$$\alpha_k \geq \hat{\beta}_k = 1 - \frac{M}{\log_2(k - 1)}.$$

Two obvious corollaries are that $\lim_{k \rightarrow \infty} \beta_k^ = 1$ and $\lim_{k \rightarrow \infty} \alpha_k = 1$.*

Proof. For $k < 7$, $\hat{\beta}_k < 0$, thus, $\alpha_k \geq \hat{\beta}_k$ is trivially satisfied (recall that $\alpha_k \geq 0$). Hence, we assume, for the remainder of the proof, $k \geq 7$ so that $\hat{\beta}_k \geq 0$, leading to a well posed definition of M , since it involves a maximum of a bounded, continuous function over a compact domain. We need to show that $g_k(\hat{\beta}_k, \delta) < 0$ for any $\delta \in [0, \hat{\beta}_k/2]$. By plugging the definition of $\hat{\beta}_k$ in (30), after simple manipulations, we obtain

$$\begin{aligned} g_k(\hat{\beta}_k, \delta) = & -[\log_2 k - \log_2(k - 1)] - [\delta(2\log_2(k - 1) - \log_2 k)] \\ & - \left[M - \left(2H(\delta) + (1 - \delta)H\left(\frac{\hat{\beta}_k - 2\delta}{1 - \delta}\right) \right) \right] < 0. \end{aligned}$$

It is straightforward to check that the terms within each of the first two pairs of square brackets are positive for every $k \geq 3$, while the expression within the third pair of square brackets is non-negative (by the definition of M). Finally, we clearly have $\hat{\beta}_k \leq \beta_k^* \leq \alpha_k < 1$, hence the stated limits are implied by the fact that $\lim_{k \rightarrow \infty} \hat{\beta}_k = 1$. $\square$

The corollary $\lim_{k \rightarrow \infty} \alpha_k = 1$ also follows from the result $\lim_{k \rightarrow \infty} \gamma_k = 0$ (Theorem 1 in [5]), together with the relationship $1 - \gamma_k \leq \alpha_k$, already mentioned in the introduction.

6.4. Numerical Computation of the Lower Bound

In this subsection, we develop numerical procedures, both to decide whether a specific (rational) number β qualifies as a lower bound to α_k , according to Theorem 3 (i.e., whether $g_k(\beta, \delta) < 0$ for every $\delta \in [0, \beta/2]$ or, equivalently, $G_k(\beta) < 0$) and to obtain the lower bound that subsumes all the β 's (that is, $\beta_k^* = \sup A_k$). The procedures presented below are based on some properties of the functions $g_k(\beta, \delta)$ and $G_k(\beta)$, which will be established, along the following lines.

- We show analytically that $g_k(\beta, \delta)$, when viewed as a function of δ , for some fixed value of β , achieves its maximum at a unique point in its domain. By a bisection-like procedure, driven by the sign of the derivative $\frac{\partial g_k}{\partial \delta}$, lower and upper bounds to such maximum, $G_k(\beta)$, can be computed with any desired accuracy.
- We then develop a partial procedure that returns $\text{sign}(G_k(\beta))$, when $G_k(\beta) \neq 0$, and does not halt otherwise.
- Finally, we show, analytically, that $G_k(\beta)$ is an increasing function taking both negative and positive values, so that the point $\beta_k^* = \sup A_k$ is the only root of the equation $G_k(\beta) = 0$ and can be (arbitrarily) approximated by a bisection-like procedure, driven by the sign of $G_k(\beta)$.

6.4.1. Computing $G_k(\beta)$

For simplicity, in this subsection, we adopt an idealized *infinite precision model*, where we assume that (i) the (real) numbers arising throughout the computation are represented with infinite precision and (ii) the results of the four basic arithmetic operations, of comparisons, and of logarithms are computed exactly. We will discuss how to deal with the somewhat subtle issues of finite precision in the next subsection.

We begin by identifying an interval that contains $G_k(\beta)$ and then show how this interval can be made arbitrarily small.

Proposition 9. Let $0 \leq \beta \leq 1$ and let $0 < \delta^l < \delta^r < \beta/2$ be such that $0 < \frac{\partial g_k}{\partial \delta}(\beta, \delta^l) < \infty$ and $-\infty < \frac{\partial g_k}{\partial \delta}(\beta, \delta^r) < 0$. Let

$$y(\delta^l, \delta^r) = \frac{\frac{\partial g_k}{\partial \delta}(\beta, \delta^l) \frac{\partial g_k}{\partial \delta}(\beta, \delta^r) (\delta^l - \delta^r) + \frac{\partial g_k}{\partial \delta}(\beta, \delta^l) g_k(\beta, \delta^r) - \frac{\partial g_k}{\partial \delta}(\beta, \delta^r) g_k(\beta, \delta^l)}{\frac{\partial g_k}{\partial \delta}(\beta, \delta^l) - \frac{\partial g_k}{\partial \delta}(\beta, \delta^r)}$$

be the ordinate of the intersection of the two straight lines tangent to the curve $g_k(\beta, \delta)$ (for fixed β) at $(\delta^l, g_k(\beta, \delta^l))$ and $(\delta^r, g_k(\beta, \delta^r))$, respectively. Then

$$G_k(\beta) \in [\max(g_k(\beta, \delta^l), g_k(\beta, \delta^r)), y(\delta^l, \delta^r)]. \quad (34)$$

To prove the above proposition, we will need the following lemma, which highlights some useful properties of $\frac{\partial g_k}{\partial \delta}$.

Lemma 4. Let $0 \leq \beta \leq 1$. Then, as δ increases from 0 to $\beta/2$, the derivative $\frac{\partial g_k}{\partial \delta}$ decreases from $+\infty$ to $-\infty$ and vanishes at a unique point, $\zeta_k(\beta)$, where $g_k(\beta, \zeta_k(\beta)) = G_k(\beta)$.

Proof. From Equation (30), basic calculus operations yield

$$\frac{\partial g_k}{\partial \delta} = \log_2 \frac{k}{(k-1)^2} + 2H'(\delta) - H\left(\frac{\beta-2\delta}{1-\delta}\right) - \frac{2-\beta}{1-\delta} H'\left(\frac{\beta-2\delta}{1-\delta}\right). \quad (35)$$

Considering that $\lim_{x \rightarrow 0^+} H'(x) = +\infty$ (see Definition 4), we have that $\lim_{\delta \rightarrow 0^+} \frac{\partial g_k}{\partial \delta} = +\infty$ (due to the second term), while $\lim_{\delta \rightarrow \beta/2^-} \frac{\partial g_k}{\partial \delta} = -\infty$ (due to the fourth term). Taking one more derivative, after some cancellation of terms and simple rearrangements, we have

$$\frac{\partial^2 g_k}{\partial^2 \delta} = 2H''(\delta) + \frac{(2-\beta)^2}{(1-\delta)^3} H''\left(\frac{\beta-2\delta}{1-\delta}\right) < 0, \quad (36)$$

where the last inequality follows from the fact that $H''(x) < 0$, for any $0 \leq x \leq 1$ (see Definition 4). From $\frac{\partial^2 g_k}{\partial^2 \delta} < 0$, we have that $\frac{\partial g_k}{\partial \delta}$ is strictly decreasing and, considering that $\frac{\partial g_k}{\partial \delta}(\beta, 0^+) = +\infty$ and $\frac{\partial g_k}{\partial \delta}(\beta, \beta/2^-) = -\infty$, we conclude that $\frac{\partial g_k}{\partial \delta}$ takes each real

value exactly once. Let then $\zeta_k(\beta)$ be the (unique) point where $\frac{\partial g_k}{\partial \delta}(\beta, \zeta_k(\beta)) = 0$. It is straightforward to argue that this is the unique point of maximum of $g_k(\beta, \delta)$ (with respect to δ , for fixed β). Then, according to the definition of G_k (see Theorem 3), $G_k(\beta) = g_k(\beta, \zeta_k(\beta))$. $\square$

Proof. (of Proposition 9) To better follow this proof, the reader may refer to the graphical illustration provided in Figure 2. The lower bound to $G_k(\beta)$, in Equation (34), trivially follows from the fact that $G_k(\beta)$ is the maximum value of $g_k(\beta, \delta)$, for $0 \leq \delta \leq \beta/2$. To establish the upper bound, let $x(\delta^l, \delta^r)$ be the abscissa of the intersection of the tangents (gray lines in Figure 2) considered in the statement (these tangents do intersect, since they have different slopes). Due to the downward convexity ($\frac{\partial^2 g_k}{\partial \delta^2} < 0$) of $g_k(\beta, \delta)$, for any $\delta \in [\delta^l, x(\delta^l, \delta^r)]$, the graph of $g_k(\beta, \delta)$ lies below the tangent at (β, δ^l) . Symmetrically, for any $\delta \in [x(\delta^l, \delta^r), \delta^r]$, the graph of $g_k(\beta, \delta)$ lies below the tangent at (β, δ^r) . Hence, for any $\delta \in [\delta^l, \delta^r]$, the graph of $g_k(\beta, \delta)$ lies below the ordinate $y(\delta^l, \delta^r)$ of the intersection of the two tangents. In particular, $G_k(\beta) = g_k(\beta, \zeta_k(\beta)) \leq y(\delta^l, \delta^r)$, since $\zeta_k(\beta) \in [\delta^l, \delta^r]$. $\square$

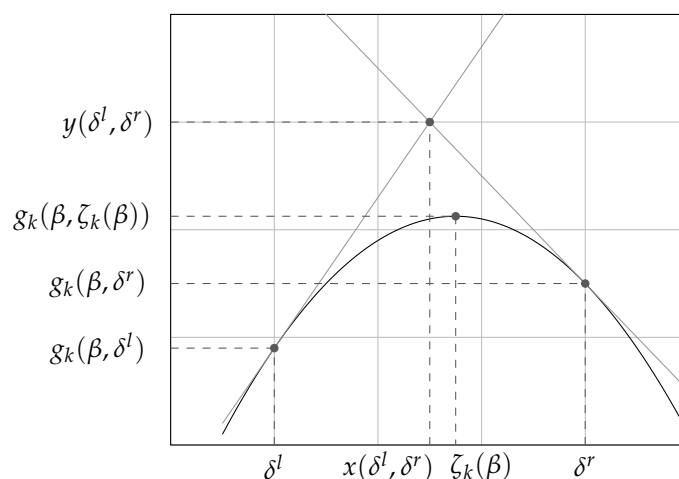


Figure 2. Illustration of the proof of Proposition 9.

Proposition 10 (Computability of G_k , in the infinite precision model). *There is a procedure (described in the proof) which, given as inputs an integer $k \geq 2$ and two real values $\beta \in [0, 1]$ and $\epsilon > 0$, outputs an interval $[G', G'']$ such that $G_k(\beta) \in [G', G'']$ and $G'' - G' < \epsilon$.*

Proof. The proposed procedure distinguishes 2 cases.

Case 1: $\frac{\partial g_k}{\partial \delta}(\beta, \beta/4) = 0$. Here, $\beta/4 = \arg \max_{\delta} g_k(\beta, \delta)$, whence $G_k(\beta) = g_k(\beta, \beta/4)$. The procedure outputs $G' = G'' = g_k(\beta, \beta/4)$, clearly satisfying the requirements in the statement.

Case 2: $\frac{\partial g_k}{\partial \delta}(\beta, \beta/4) \neq 0$. Here the procedure includes two phases. In a first phase, a bisection process determines two points, δ_0^l and δ_0^r , which satisfy the assumptions of Proposition 9. (For simplicity, the dependence of δ_0^l and δ_0^r upon k and β is not made explicit in the notation.) In a second phase, the interval $[\delta_0^l, \delta_0^r]$ is iteratively bisected, until the interval appearing in Equation (34) has size smaller than ϵ . In both phases, the bisection is driven by the sign of $\frac{\partial g_k}{\partial \delta}$. The first phase includes two subcases:

Subcase 2a: $\frac{\partial g_k}{\partial \delta}(\beta, \beta/4) > 0$. We define the sequence $\mu_j = (1 - 2^{-j})(\beta/2)$, for $j \geq 1$. Letting $h = \min\{j \geq 2 : \frac{\partial g_k}{\partial \delta}(\beta, \mu_j) < 0\}$, we set $[\delta_0^l, \delta_0^r] = [\mu_{h-1}, \mu_h]$.

Subcase 2b: $\frac{\partial g_k}{\partial \delta}(\beta, \beta/4) < 0$. We define the sequence $\nu_j = 2^{-j}(\beta/2)$, for $j \geq 1$. Letting $h = \min\{j \geq 2 : \frac{\partial g_k}{\partial \delta}(\beta, \nu_j) > 0\}$, we set $[\delta_0^l, \delta_0^r] = [\nu_h, \nu_{h-1}]$.

Since $\lim_{j \rightarrow \infty} \mu_j = \beta/2$ and $\lim_{j \rightarrow \infty} \nu_j = 0$, the interval $[\delta_0^l, \delta_0^r]$ is well defined, in either subcase. Its endpoints can be computed by iteratively testing the condition on the derivative for $j = 1, 2, \dots$, till it is satisfied.

In both subcases, $\zeta_k(\beta) \in [\delta_0^l, \delta_0^r]$. We can then construct a sequence of intervals, each half the size of the preceding one and containing $\zeta_k(\beta)$, as follows.

For $i = 1, 2, \dots$ do:

1. $c_i = (\delta_{i-1}^l + \delta_{i-1}^r)/2$.
2. If $\frac{\partial g_k}{\partial \delta}(\beta, c_i) = 0$, then set $G' = G'' = g_k(\beta, c_i)$ and exit.
3. If $\frac{\partial g_k}{\partial \delta}(\beta, c_i) > 0$, then let $[\delta_i^l, \delta_i^r] = [c_i, \delta_{i-1}^r]$.
4. If $\frac{\partial g_k}{\partial \delta}(\beta, c_i) < 0$, then let $[\delta_i^l, \delta_i^r] = [\delta_{i-1}^l, c_i]$.
5. Set $G' = \max(g_k(\beta, \delta_i^l), g_k(\beta, \delta_i^r))$ and $G'' = y(\delta_i^l, \delta_i^r)$, as defined in Proposition 9. If $G'' - G' < \epsilon$, then exit.

It is straightforward to show that, for any $\eta > 0$, $\log_2(1/\eta)$ bisection iterations (counting those of both phases) are sufficient to guarantee $\delta^r - \delta^l \leq \eta$, hence to determine $\zeta_k(\beta)$ with accuracy $\eta > 0$.

It remains to show that the above for loop is eventually exited. If the loop is exited at step 2, then we are done. Otherwise, as we will argue, $G'' - G'$ vanishes with $\delta^r - \delta^l$ so that, for some i , $G'' - G' < \epsilon$, hence the loop is eventually exited, at step 5. Toward this conclusion, we can observe that

$$\begin{aligned} y(\delta^l, \delta^r) - g_k(\beta, \delta^l) &\leq \left| \frac{\partial g_k}{\partial \delta}(\beta, \delta^l) \right| (x(\delta^l, \delta^r) - \delta^l), \\ y(\delta^l, \delta^r) - g_k(\beta, \delta^r) &\leq \left| \frac{\partial g_k}{\partial \delta}(\beta, \delta^r) \right| (\delta^r - x(\delta^l, \delta^r)), \end{aligned}$$

where $x(\delta^l, \delta^r)$ is the abscissa of the intersection of the tangents. As $\delta^r - \delta^l$ approaches 0, we have that

$$\begin{aligned} \lim \delta^l &= \lim \delta^r = \lim x(\delta^l, \delta^r) = \zeta_k(\beta), \\ \lim \frac{\partial g_k}{\partial \delta}(\beta, \delta^l) &= \lim \frac{\partial g_k}{\partial \delta}(\beta, \delta^r) = 0, \\ \lim G'' - G' &= \lim y(\delta^l, \delta^r) - \max(g_k(\beta, \delta^l), g_k(\beta, \delta^r)) = 0. \end{aligned}$$

□

6.4.2. Computing $\text{sign}(G_k(\beta))$

We now consider the more realistic model where only rational numbers are represented, and only arithmetic operations and comparisons with rational inputs can be computed exactly. If $f(x)$ is a real function of the real variable x , we will restrict our attention to rational inputs and be interested in computing arbitrary approximations of $f(x)$. More specifically, our target is an *approximation algorithm* $\mathcal{A}_f(x, \epsilon)$ of the rational inputs x and $\epsilon > 0$, whose output $\tilde{f}(x, \epsilon)$ satisfies the relationship $|\tilde{f}(x, \epsilon) - f(x)| < \epsilon$. Thus, $f(x)$ is a computable real number, in the sense of Definition 1. These notions naturally extend to functions of several variables.

Approximation algorithms are well known for $\log_2 x$, such as the iterative method of [37]. For evaluating a finite arithmetic expression, a tedious but straightforward analysis of error propagation will determine the required accuracy of each intermediate calculation capable of guaranteeing the desired accuracy for the target result. Therefore, approximation algorithms can be derived for $g_k(\beta, \delta)$ and $\frac{\partial g_k}{\partial \delta}(\beta, \delta)$.

One issue we need to deal with is that an algorithm to compute $f(x)$ with any desired accuracy does not automatically translate into an algorithm to systematically determine the

sign (positive, negative, or zero), of $f(x)$, denoted $\text{sign}(f(x))$. In fact, while $\tilde{f}(x, \epsilon) \geq +\epsilon$ implies $f(x) > 0$ and $\tilde{f}(x, \epsilon) \leq -\epsilon$ implies $f(x) < 0$, in the remaining case, $|\tilde{f}(x, \epsilon)| < \epsilon$, nothing can be inferred about $\text{sign}(f(x))$. On the one hand, when $f(x) = 0$, this “undeterminate” case is bound to occur, for any $\epsilon > 0$. On the other hand, when $f(x) \neq 0$, this case will not occur if $\epsilon \leq |f(x)|/2$, since $|\tilde{f}(x, \epsilon) - f(x)| < \epsilon \leq |f(x)|/2$ implies $|\tilde{f}(x, \epsilon)| \geq |f(x)|/2 \geq \epsilon$.

The preceding observations suggest the following procedure $\mathcal{S}(\mathcal{A}_f, x)$ which, building on an algorithm $\mathcal{A}_f$ that computes an approximation $\tilde{f}(x, \epsilon)$ to $f(x)$, will halt and output $\text{sign}(f(x)) \in \{-, +\}$, when $f(x) \neq 0$ and will not halt when $f(x) = 0$. Given a monotonically vanishing, computable sequence $\epsilon_1, \epsilon_2, \dots$ (e.g., $\epsilon_i = 2^{-i}$), for $i = 1, 2, \dots$, the procedure $\mathcal{S}$ computes $\tilde{f}_i = \tilde{f}(x, \epsilon_i)$ by a call to $\mathcal{A}_f(x, \epsilon_i)$; if $\tilde{f}_i \geq \epsilon_i$ or $\tilde{f}_i \leq -\epsilon_i$, then it returns $\text{sign}(\tilde{f}_i)$ and halts.

The bisection procedure has to be modified so that it will not get stuck in the attempt of evaluating the sign of a zero. Assuming that f has a unique zero in the interval $[a, b]$, say, with $f(a) < 0$ and $f(b) > 0$, a point c where to split the interval can be found as follows. Let $c' = (2a + b)/3$ and $c'' = (a + 2b)/3$ the points that *trisection* the interval. Interleave the executions of the calls $\mathcal{S}(\mathcal{A}_f, c')$ and $\mathcal{S}(\mathcal{A}_f, c'')$ until the termination of one of them, an event guaranteed to occur since at least one between $f(c')$ and $f(c'')$ differs from zero. Let c be the argument for which the execution has terminated, whence $\text{sign}(f(c))$ has been determined. The refined interval is chosen to be $[a, c]$ when $f(c) > 0$ and $[c, b]$ when $f(c) < 0$. In all cases, the interval size shrinks at least by $2/3$.

With the tools we have introduced, we can now tackle the finite-precision computation of $\text{sign}(G_k(\beta))$, a quantity that will play a key role in the computation of β_k^* discussed in the next subsection.

Proposition 11 (Partial computability of $\text{sign}(G_k(\beta))$, in the finite precision model). *There is a procedure (described in the proof) which, given as inputs an integer $k \geq 2$ and two rational values $\beta \in [0, 1]$ and $\epsilon > 0$, outputs $\text{sign}(G_k(\beta))$ and halts, if $G_k(\beta) \neq 0$, and does not halt, if $G_k(\beta) = 0$.*

Proof. We obtain the procedure for $\text{sign}(G_k(\beta))$ by adapting the procedure to approximate $G_k(\beta)$ presented in the proof of Proposition 10. To lighten the notation, throughout this proof, we let $g(\delta) = g_k(\beta, \delta)$ and $g'(\delta) = \frac{\partial g_k}{\partial \delta}(\beta, \delta)$.

Interleave the executions of the calls $\mathcal{S}(\mathcal{A}_{g'}, \beta/4)$ and $\mathcal{S}(\mathcal{A}_{g'}, \beta/8)$ until termination of one of them, an event guaranteed to occur since at least one between $g'(\beta/4)$ and $g'(\beta/8)$ differs from zero. Let δ_0 be the argument for which the execution has terminated. If $g'(\delta_0) > 0$, then search for a μ_h such that $g'(\mu_h) < 0$ and let $[\delta_0^l, \delta_0^r] = [\delta_0, \mu_h]$. Else $g'(\delta_0) < 0$, search for a ν_h such that $g'(\nu_h) > 0$ and let $[\delta_0^l, \delta_0^r] = [\nu_h, \delta_0]$. The search has to interleave the evaluation of two consecutive points in the sequence, to avoid the potential non termination of the execution, which can occur at most at one point.

The sequence $[\delta_i^l, \delta_i^r]$, for $i = 1, 2, \dots$, of successive refinements of $[\delta_0^l, \delta_0^r]$ is then constructed, choosing the splitting point c_i by trisection. Let ϵ_i be the error bound such that the call to $\mathcal{A}_{g'}(c_i, \epsilon_i)$ has enabled the determination of the sign of $g'(c_i)$. Let also $\eta_i = \min(2^{-i}, \epsilon_i)$ and compute η_i -approximations $\tilde{g}(\delta_i^l, \eta_i)$, $\tilde{g}(\delta_i^r, \eta_i)$, and $\tilde{y}(\delta_i^l, \delta_i^r, \eta_i)$. The conditions that enable determining $\text{sign}(G_k(\beta))$ are as follows:

- If $\tilde{g}(\delta_i^l, \eta_i) \geq \eta_i$ or $\tilde{g}(\delta_i^r, \eta_i) \geq \eta_i$, then return $\text{sign}(G_k(\beta)) = +$ and halt.
- If $\tilde{y}(\delta_i^l, \delta_i^r, \eta_i) \leq -\eta_i$, then return $\text{sign}(G_k(\beta)) = -$ and halt.

Indeed, in the first case, the condition implies that $g(\delta_i^l) = g_k(\beta, \delta_i^l) > 0$ or $g(\delta_i^r) = g_k(\beta, \delta_i^r) > 0$, which in turn implies $G_k(\beta) > 0$. Conversely, if $G_k(\beta) > 0$, for i large enough,

both $g(\delta_i^l)$ and $g_k(\delta_i^r)$, which are monotonically non decreasing with i , become positive. Since η_i vanishes with i , it will eventually become sufficiently small to satisfy the condition.

Symmetrically, in the second case, the condition implies that $G_k(\beta) < 0$. Conversely, if $G_k(\beta) < 0$, for i large enough, $y(\delta_i^l, \delta_i^r)$, which decreases with i , becomes negative. Again, η_i will eventually become sufficiently small to satisfy the condition.

Finally, we observe that if $G_k(\beta) = 0$, then neither condition will ever hold and the procedure will not halt. $\square$

6.4.3. Computing $\beta_k^* = \sup A_k$

In this subsection, we will see that the set A_k is an interval, closed on the left and open on the right: specifically, $A_k = [0, \beta_k^*)$. On the one hand, this property of A_k is not surprising, since if β' is a lower bound to α_k and $\beta < \beta'$, then β is a lower bound too. On the other hand, the property does require a proof, since membership in A_k is a sufficient, but not necessary condition for β to be a lower bound, a scenario compatible with A_k having “holes”, that is, with $G_k(\beta)$ taking both negative and positive values to the left of β_k^* . However, we will argue that A_k has no such holes, since $G_k(\beta)$ has a unique zero in $[0, 1]$ which, by Equation (33), is our target, $\beta_k^* = \sup A_k$. We will show this zero to be computable.

Proposition 12. *For every $k \geq 2$, in the interval $0 \leq \beta \leq 1$, the function $G_k(\beta)$ is increasing and has a unique zero, which equals $\beta_k^* = \sup A_k$.*

Proof. From Lemma 4, we have that $G_k(\beta) = g_k(\beta, \zeta_k(\beta))$, where $\zeta_k(\beta)$ is the unique solution of the equation (in δ) $\frac{\partial g_k}{\partial \delta}(\beta, \delta) = 0$, so that

$$\frac{\partial g_k}{\partial \delta}(\beta, \zeta_k(\beta)) = 0. \quad (37)$$

By the chain rule for the total derivative and the above relationship, we get

$$\begin{aligned} \frac{\partial G_k}{\partial \beta} &= \frac{\partial g_k}{\partial \beta}(\beta, \zeta_k(\beta)) \frac{\partial \beta}{\partial \beta} + \frac{\partial g_k}{\partial \delta}(\beta, \zeta_k(\beta)) \frac{\partial \zeta_k(\beta)}{\partial \beta} = \frac{\partial g_k}{\partial \beta}(\beta, \zeta_k(\beta)) \\ &= \log_2(k-1) + H' \left(\frac{\beta - 2\zeta_k(\beta)}{1 - \zeta_k(\beta)} \right), \end{aligned}$$

where the last step follows from Equation (30). After several but simple manipulations and letting $\zeta = \zeta_k(\beta)$, the expression just derived together with Equations (37) and (35) yield the relationship

$$\left(\frac{2-\beta}{1-\zeta} \right) \frac{\partial G_k}{\partial \beta} = \log_2 \frac{k}{k-1} + 2H'(\zeta) - H \left(\frac{\beta - 2\zeta}{1-\zeta} \right) + \left(1 - \frac{\beta - 2\zeta}{1-\zeta} \right) \log_2(k-1). \quad (38)$$

We will argue that the right hand side is positive, for any $\beta \in [0, 1]$ and any $\zeta \in [0, \beta/2]$. As easily seen, the only negative term in the right hand side of Equation (38) is $-H \left(\frac{\beta - 2\zeta}{1-\zeta} \right) \geq -1$. A case analysis shows that this negative term is offset by one or more of the remaining terms, yielding a positive sum.

Case 1: $\zeta < 1/(1 + \sqrt{2})$. Considering that H' is monotonically decreasing (see Definition 4), we have

$$2H'(\zeta) > 2H' \left(\frac{1}{1 + \sqrt{2}} \right) = 2 \log_2 \frac{1 - 1/(1 + \sqrt{2})}{1/(1 + \sqrt{2})} = 2 \log_2 \sqrt{2} = 1.$$

Case 2: $\zeta \geq 1/(1 + \sqrt{2})$ and $k \geq 4$. We observe that $\left(1 - \frac{\beta - 2\zeta}{1 - \zeta}\right)$ is decreasing with β and increasing with ζ , whereas $\log_2(k - 1)$ is increasing with k , whence

$$\left(1 - \frac{\beta - 2\zeta}{1 - \zeta}\right) \log_2(k - 1) > \left(1 - \frac{1 - 2/(1 + \sqrt{2})}{1 - 1/(1 + \sqrt{2})}\right) \log_2(3) > \frac{1}{\sqrt{2}} 1.58 > 1.11$$

Case 3: $\zeta \geq 1/(1 + \sqrt{2})$ and $k = 2$. The right hand side of Equation (38) becomes

$$1 + 2H'(\zeta) - H\left(\frac{\beta - 2\zeta}{1 - \zeta}\right),$$

which is always positive. In fact, if $\zeta < 1/2$, then $2H'(\zeta) > 0$ and $1 - H\left(\frac{\beta - 2\zeta}{1 - \zeta}\right) \geq 0$. On the other hand, if $\zeta = 1/2$, then $2H'(\zeta) = 0$; furthermore, it must be $\beta = 1$ so that $H\left(\frac{\beta - 2\zeta}{1 - \zeta}\right) = H(0) = 0$.

Case 4: $\zeta \geq 1/(1 + \sqrt{2})$ and $k = 3$. Considering that (i) $H'(\zeta) \geq 0$ (for $0 \leq \zeta \leq 1/2$), (ii) $H(x) \leq 1$ (for any $0 \leq x \leq 1$), and (iii) $\left(1 - \frac{\beta - 2\zeta}{1 - \zeta}\right) > \frac{1}{\sqrt{2}}$ (as seen in Case 2), Equation (38) implies

$$\left(\frac{2 - \beta}{1 - \zeta}\right) \frac{\partial G_k}{\partial \beta} \geq \log_2 \frac{3}{2} - 1 + \frac{1}{\sqrt{2}} > 0.58 - 1 + 0.70 = 0.28 > 0. \quad (39)$$

Having established that $G_k(\beta)$ has a positive derivative, hence it is increasing, in the interval $[0, 1]$, we now argue the existence of a (unique) zero of $G_k(\beta)$, by showing that $G_k(0) < 0$ and $G_k(1) > 0$, for every $k \geq 2$.

At $\beta = 0$, we simply observe that, by definition (i.e., Equation (32)), we have $G_k(0) = \max_{0 \leq \delta \leq 1} g_k(0, \delta) = g_k(0, 0) = -\log_2 k < 0$, for every $k \geq 2$.

At $\beta = 1$, we observe that $G_k(1) = \max_{0 \leq \delta \leq 1/2} g_k(1, \delta) \geq g_k(1, 1/k)$ and show that $g_k(1, 1/k) > 0$, for every $k \geq 2$. We consider the following chain of relationships where, starting from Equation (30), (i) we have dropped the contribution $(1 - \delta) \log_2 \frac{1}{1 - \delta}$ to $H(\delta)$ and the second entropy term, which are both non-negative; (ii) we have plugged $\beta = 1$ and $\delta = \frac{1}{k}$; and (iii) we have performed simple algebraic manipulations, also making use of the inequality $\log_2(1 + x) \leq (\log_2 e)x$:

$$\begin{aligned} g_k(1, 1/k) &\geq \left(1 - 2\frac{1}{k}\right) \log_2(k - 1) - \left(1 - \frac{1}{k}\right) \log_2 k + 2\frac{1}{k} \log_2 k \\ &= \log_2 \frac{k - 1}{k} + \frac{2}{k} \log_2 \frac{k}{k - 1} + \frac{1}{k} \log_2 k \\ &\geq -(\log_2 e) \left(1 - \frac{2}{k}\right) \frac{1}{k} + \frac{1}{k} \log_2 k \\ &= \frac{1}{k} \left[\log_2 k - (\log_2 e) \left(1 - \frac{2}{k}\right) \right]. \end{aligned}$$

For $k = 2$, the expression within the square bracket evaluates to 1. For $k \geq 3$, it is easy to see that the square bracket is positive, as $\log_2 k - (\log_2 e)(1 - \frac{2}{k}) > \log_2 3 - \log_2 e > 0$. In conclusion, for every $k \geq 2$, we have $g_k(1, \frac{1}{k}) > 0$, as claimed.

Finally, given that $G_k(\beta)$ is increasing with β , its unique zero is the supremum β_k^* of A_k (cf. Equation (33)). $\square$

We can now present a simple numerical algorithm to approximate β_k^* , from below, with any desired accuracy.

Proposition 13. (Computability of β_k^* , in the finite precision model.) There is a procedure (outlined in the proof) which, on inputs $k \geq 2$ and $\epsilon > 0$, outputs a number $\tilde{\beta}_k^* \in [\beta_k^* - \epsilon, \beta_k^*]$. Clearly, $\tilde{\beta}_k^* \leq \alpha_k$.

Proof. In light of Proposition 12, it is straightforward to develop a trisection procedure that starts with the interval $[0, 1]$, is driven by the sign of $G_k(\beta)$ computed according to Proposition 11, stops after $\lceil \log_{3/2} \frac{1}{\epsilon} \rceil$ iterations, and outputs the left endpoint of the current interval containing β_k^* , which has size smaller than $(3/2)^{-\lceil \log_{3/2} \frac{1}{\epsilon} \rceil} \leq \epsilon$. $\square$

Table 8. Lower bound $\tilde{\beta}_k^*$ to α_k , for various values of k . With reference to Proposition 13, $\epsilon = 10^{-8}$, hence the five digits to the right of the decimal point are guaranteed to be the same as those of β_k^* . The simple $1 - 1/k$ upper bound to α_k shows how, for large k , β_k^* is quite close to α_k .

k	$\tilde{\beta}_k^*$	$1 - 1/k$
2	0.17055	0.50000
3	0.28366	0.66667
4	0.35978	0.75000
5	0.41517	0.80000
6	0.45776	0.83333
7	0.49183	0.85714
8	0.51990	0.87500
16	0.64475	0.93750
32	0.73867	0.96875
2^{10}	0.94359	0.99902
2^{20}	0.99686	0.99999
2^{30}	0.99978	0.99999
2^{40}	0.99998	0.99999

Table 8 reports the lower bound $\tilde{\beta}_k^*$ (approximating β_k^* from below), for a set of values of k . The program we have used is based on a direct implementation of the bisection version of the relevant procedures. For the level of accuracy of the reported results, the precision of standard floating point arithmetic turns out to be sufficient. The execution is rather fast: even the result for $k = 2^{40}$ took about 15 milliseconds of core-time to compute, even using a non-optimized, straightforward implementation of the bisection algorithm. As a term of comparison, the table also shows the value of the simple upper bound $\alpha_k \leq 1 - \frac{1}{k}$ (from Hamming distance). Considering that, from Proposition 8, $\lim_{k \rightarrow \infty} (\alpha_k - \beta_k) = 0$, we see how, as k increases, β_k^* provides an increasingly better approximation (from below) to α_k . To guarantee the same level of approximation via the lower bound $\alpha_k \geq \alpha_k(n) - Q(n)$, increasing values of n are required, as k increases. Eventually, computing β_k^* becomes less expensive than estimating $\alpha_k(n)$.

7. A Conjecture on the Asymptotic Behavior of α_k

Unlike the previous sections of this paper, the present one is rather speculative and is meant to take some rather preliminary steps toward the understanding of the asymptotic behavior of α_k with k . To focus the exploration, we formulate a specific conjecture, namely that $\lim_{k \rightarrow \infty} (1 - \alpha_k)k = c_\alpha$ for some constant $c_\alpha \geq 1$. Essentially, we are conjecturing that the limit is finite. The bound $c_\alpha \geq 1$ follows from $\alpha_k \leq 1 - \frac{1}{k}$ (the expected Hamming distance per symbol). The discussion below provides context and motivation for the conjecture, and contemplates the more specific hypothesis that $3 \leq c_\alpha \leq 4$.

In the context of the LCS problem, [38] have proven the conjecture, proposed by [39], that $\lim_{k \rightarrow \infty} \gamma_k \sqrt{k} = 2$. As a corollary, we have that, for large k , $1 - \alpha_k \geq \gamma_k \approx \frac{2}{\sqrt{k}}$. However, our intuition is that, although for large n and k we can expect an LCS of length

approximately $\frac{2n}{\sqrt{k}}$, it would be too costly, in terms of insertions and deletions, to align more than $\mathcal{O}(\frac{n}{k})$ matches, so that, ultimately, for large k , we expect $\alpha_k \approx 1 - \frac{c_\alpha}{k}$.

We tentatively investigate the conjecture numerically, by considering Monte Carlo estimates of the quantity $c_{\alpha,k}(n) = (1 - \alpha_k(n))k$. Unfortunately, obtaining such estimates with the required precision presents some challenges.

Since we expect values of $c_{\alpha,k}(n)$ not much larger than 1, we need the error on $(1 - \alpha_k(n))$, hence on $\alpha_k(n)$, to be a fraction of $1/k$, a constraint that becomes increasingly stringent as k increases. In order to guarantee a sufficient upper bound on the error, based on Proposition 3 and its corollary Equation (17), the necessary values of n and N quickly become prohibitive, as k increases. However, we suspect that these errors bounds do become rather loose, for large k . More specifically, we hypothesize that, for $x, y \in \Sigma_k^n$ with large k , the standard deviation of $d_E(x, y)/n$ can be approximated as $\sqrt{\frac{1}{kn}}$. This is based on the intuition that, for large k , the edit distance behaves similarly to the Hamming distance $d_H(x, y)$, whose standard deviation can be easily determined to be $\sqrt{n\frac{1}{k}(1 - \frac{1}{k})} \approx \sqrt{\frac{n}{k}}$. We have tested this hypothesis by experimentally estimating the standard deviation of $d_E(x, y)/n$, for $n = 2^{18}$ and $k = 2^{14}$, based on $N = 80$ independent pairs of random strings. The estimated value turned out to be 0.14×10^{-4} , well in line with our assumption that the standard deviation is approximately $\sqrt{\frac{1}{kn}} = \sqrt{\frac{1}{2^{14}2^{18}}} = 2^{-16} \approx 0.15 \times 10^{-4}$.

Table 9 shows estimates of $c_{\alpha,k}(n)$ for $n = 2^{17}, 2^{18}, 2^{19}, 2^{20}$ and $k = 2^7, 2^8, \dots, 2^{20}$. Highlighted in bold face are the entries (33 in number) for which the hypothesized statistical error $k\sqrt{\frac{1}{kn}} = \sqrt{\frac{k}{n}}$ on $c_{\alpha,k}(n)$ is at most $\frac{1}{4}$, that is, $k \leq \frac{n}{16}$. We can observe that, where the hypothesized error is small enough, $3 \leq c_{\alpha,k}(n) \leq 4$. We remark that, although to cope with the quadratic complexity of the edit distance computation we have considered only one pair of random strings for each (k, n) entry in Table 9, the consistency of the numerical value of $c_{\alpha,k}(n)$ across the 33 bold entries, obtained from statistically independently samples, does lend some support to the sub-conjecture $3 \leq c_\alpha \leq 4$.

Of course, how well $c_{\alpha,k}(n)$ approximates $c_{\alpha,k}$ depends on how well $\alpha_k(n)$ approximates α_k . The quality of the latter approximation increases for large k , where the the bound provided by Theorem 2 becomes loose. In fact, putting together various results, we have that

$$1 - \frac{M}{\log_2(k-1)} \leq \alpha_k \leq \alpha_k(n) \leq 1 - \frac{1}{k}. \quad (40)$$

We can see that the difference between the last and the first term, hence the difference $\alpha_k(n) - \alpha_k$ between the intermediate terms, vanishes when k diverges. The quantitative impact on the difference $c_{\alpha,k} - c_{\alpha,k}(n)$ remains to be seen, but the relative stability of the bold entries in each column of Table 9 seems compatible with assuming a small impact. Of course, this section remains in the realm of conjectures, which will hopefully provide some motivation for rigorous analysis that may confirm as well as refute them.

Table 9. Estimates of $c_{\alpha,k}(n) = (1 - \alpha_k(n))k$, each based on a single random pair ($N = 1$) generated independently of the other pairs. In bold face are shown the entries of the table where the (hypothesized) standard deviation of the error $\sqrt{k/n}$ lies in the interval $[0.011, 0.250]$.

	n			
k	2^{17}	2^{18}	2^{19}	2^{20}
2^7	3.553	3.552	3.566	3.581
2^8	3.617	3.608	3.629	3.635
2^9	3.531	3.656	3.658	3.678
2^{10}	3.570	3.625	3.654	3.669
2^{11}	3.406	3.516	3.668	3.652

Table 9. *Cont.*

k	2^{17}	2^{18}	n	2^{19}	2^{20}
2^{12}	3.250	3.656		3.625	3.672
2^{13}	3.000	3.500		3.500	3.703
2^{14}	3.125	3.250		3.563	3.672
2^{15}	3.000	3.500		3.750	3.313
2^{16}	3.500	2.750		3.250	3.125
2^{17}	3.000	2.000		3.000	3.000
2^{18}	4.000	4.000		2.000	2.500
2^{19}	0.000	2.000		2.000	1.500
2^{20}	0.000	0.000		0.000	0.000

8. Conclusions and Further Questions

In this paper, we have explored ways to compute Monte Carlo estimates, upper bounds, and lower bounds to the asymptotic constant characterizing the expected edit distance between random, independent strings. We have presented the theoretical basis for various approaches and used them to obtain numerical results for some alphabet sizes k , which improve over previously known values [1]. However, there is still a significant gap between upper and lower bounds that can be actually computed in a reasonable time. Below, we outline a number of open questions worthy of further investigation.

The approaches proposed here can be extended to the study of other statistical properties of the edit distance for a given length n , e.g., the standard deviation, which has been widely studied in the context of the longest common subsequence. Ultimately, a characterization of the full distribution would be desirable.

The exact rate of convergence $q_k(n) = \alpha_k(n) - \alpha_k$, or even just its asymptotic behavior, remains to be determined. In particular, we are not aware of any significant lower bound to $q_k(n)$ to be compared with the upper bound $Q(n)$. Moreover, this upper bound is oblivious to k , whereas the rate of convergence is affected by k , as indicated by Equation (40).

From Proposition 8 and a straightforward analysis of “Hamming scripts”, which use only matches and substitutions, we know that $\frac{M}{\log_2 k} \leq 1 - \alpha_k \leq \frac{1}{k}$, (where $M \approx 2.52$); what is the exact asymptotic behavior of $1 - \alpha_k$? We have conjectured that, asymptotically, $1 - \alpha_k$ approaches c_α/k , for some constant $c_\alpha \geq 1$; but whether the conjecture holds, and if so for which value of c_α , remains to be seen.

Of mathematical interest, is the question whether α_k is a rational or an algebraic number. The answer could depend upon k . The recent work of [19] on the algebraic nature of γ_2 indicates that this line of investigation may lead to uncovering deep combinatorial properties that can shed light on various aspects of the subject, well beyond mere mathematical curiosity.

The analysis of the statistical error could be somewhat improved, if the experimental evidence that the standard deviation of $d_E(x, y)$ is $o(\sqrt{n})$ were corroborated analytically. The dependence upon k could also play a role here.

The complexity of computing, say, the most significant h bits of $\alpha_k(n)$ remains a wide open question. The only known lower bound is $\Omega(\log k + \log n + h)$, based on the input and output size. It is a far cry from the current upper bound, which is doubly exponential in $\log n$. However, a deeper understanding of the distribution and symmetries of the edit distance is likely to be required before the computation time of $\alpha_k(n)$ can be significantly improved.

The lower bounds presented in Section 6 are based on upper bounds to the number of optimal scripts of a given cost r . Our counting argument could be refined to take into

account some properties of optimal edit scripts. For example, in an optimal script, an insertion cannot immediately precede or follow a deletion (since the same result could be achieved by just one substitution). Furthermore, it would be easy to show that, for many pairs of strings, multiple scripts are counted by our argument. Part of the difficulty with improving script counting comes from the analytical tractability of the resulting combinatorial expressions, which would be far more complicated than Equation (28).

Another potential weakness of our lower bound is that it is derived from a lower bound to $\text{ecc}(x)$ that must hold for every $x \in \Sigma_k^n$. If, as n goes to infinity, the fraction of strings with eccentricity significantly smaller than the average were to remain sufficiently high, then the approach would be inherently incapable of yielding tight bounds. On the other hand, preliminary efforts seem to indicate that characterizing the strings with minimum eccentricity is not straightforward. In contrast, it is a relatively simple exercise to prove that the strings of maximum eccentricity are those where all positions contain the same symbol and that their eccentricity equals $(1 - \frac{1}{k})n$.

In terms of applications, it would be interesting to explore the role of statistical properties of the edit distance in string alignment and other key problems in DNA processing and molecular biology. One motivation is provided by the error profile of reads coming from third generation sequencers (e.g., PacBio), where sequencing errors can be modeled as edit operations. In this context, it would be important to generalize the analysis to non-uniform string distributions, whether defined analytically or from empirical data, such as the distribution of substrings from the human DNA. As briefly discussed in Sections 4 and 5, the approach we have presented for Monte Carlo estimates and for exact estimates via the Coalesced Dynamic Programming algorithm can easily handle arbitrary symbol distributions, as long as the symbols are statistically independent. The extension of the lower bounds of Section 6 appears less straightforward.

Also of great interest would be to analyze the expected distance between noisy copies of two independent strings, as well as between a string and a noisy copy of itself, when the noise can be modeled in terms of edit operations.

While it is natural to begin investigating questions in the simplest scenarios, which for the present investigation is represented by strings with independent symbols, in several application contexts, the assumption of independence is not satisfied. Hence, future work should go in the direction of removing this assumption. However, while the experimental evidence may clearly indicate that some sequences exhibit a non trivial degree of intersymbol dependence, for example because they are compressible, such dependence has not been satisfactorily captured by well defined stochastic models. Therefore, it is likely that, in the near future, dependent models will be studied more because they have a well understood structure (say, Markov models) than because they are known to capture natural phenomena. Finding the right models, perhaps based on variants of transformers, is probably the main key step, to enable a better connection between theory and applications.

Author Contributions: Conceptualization, G.B. and M.S.; methodology, G.B. and M.S.; software, M.S.; validation, G.B. and M.S.; formal analysis, G.B. and M.S.; writing—original draft preparation, G.B. and M.S.; writing—review and editing, G.B. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the European Union through MUR “National Center for HPC, Big Data, and Quantum Computing”, CN00000013 (PNRR); by MUR, the Italian Ministry of University and Research, under PRIN Project n. 2022TS4Y3N - EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data; by the University of Padova, Project CPGA³ Parallel and Hierarchical Computing: Architectures, Algorithms, and Applications; and by IBM under a Shared University Research Award.

Data Availability Statement: Software and data can be found on the following GitLab repository <https://gitlab.com/skimmy/average-edit-distance>

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ganguly, S.; Mossel, E.; Racz, M.Z. Sequence assembly from corrupted shotgun reads. *arXiv* **2016**, arXiv:1601.07086.
2. Rubinfeld, A. Hardness of Approximate Nearest Neighbor Search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*; Association for Computing Machinery: New York, NY, USA, 2018; STOC 2018, pp. 1260–1268. <https://doi.org/10.1145/3188745.3188916>.
3. Comin, M.; Schindl, M. Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns. *BMC Bioinform.* **2014**, *15*, S1. <https://doi.org/10.1186/1471-2105-15-S9-S1>.
4. Weirather, J.L.; de Cesare, M.; Wang, Y.; Piazza, P.; Sebastiano, V.; Wang, X.J.; Buck, D.; Au, K.F. Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research* **2017**, *6*, 100. <https://doi.org/10.12688/f1000research.10571.2>.
5. Chvátal, V.; Sankoff, D. Longest common subsequences of two random sequences. *J. Appl. Probab.* **1975**, *12*, 306–315. <https://doi.org/10.2307/3212444>.
6. Steele, J.M. *Probability Theory and Combinatorial Optimization*; SIAM: Philadelphia, PA, USA 1997. <https://doi.org/10.1137/1.9781611970029>.
7. Schindl, M.; Bilardi, G. Bounds and estimates on the average edit distance. In *Proceedings of the String Processing and Information Retrieval*; Brisaboa, N.R., Puglisi, S.J., Eds.; Springer: Cham, Switzerland, 2019; pp. 91–106. https://doi.org/10.1007/978-3-030-32686-9_7.
8. Bilardi, G.; Schindl, M. Computable Bounds and Monte Carlo Estimates of the Expected Edit Distance. *arXiv* **2022**, arXiv:2211.07644.
9. Rosenfeld, M. Upper bounds on the average edit distance between two random strings. *arXiv* **2024**, arXiv:2407.18113. <https://doi.org/10.48550/arXiv.2407.18113>.
10. Bundschuh, R. High precision simulations of the longest common subsequence problem. *Eur. Phys. J. B-Condens. Matter Complex Syst.* **2001**, *22*, 533–541. <https://doi.org/10.1007/s100510170102>.
11. Dancík, V. Expected Length of Longest Common Subsequences. Ph.D. Thesis, University of Warwick, Coventry, UK, 1994.
12. Ning, K.; Choi, K.P. Systematic assessment of the expected length, variance and distribution of Longest Common Subsequences. *arXiv* **2013**, arXiv:1306.4253.
13. Bukh, B.; Cox, C. Periodic words, common subsequences and frogs. *Ann. Appl. Probab.* **2022**, *32*, 1295–1332. <https://doi.org/10.1214/21-AAP1709>.
14. Lueker, G.S. Improved Bounds on the Average Length of Longest Common Subsequences. *J. ACM* **2009**, *56*, 17:1–17:38. <https://doi.org/10.1145/1516512.1516519>.
15. Heineman, G.T.; Miller, C.; Reichman, D.; Salls, A.; Sárközy, G.; Soiffer, D. Improved Lower Bounds on the Expected Length of Longest Common Subsequences. *arXiv* **2024**, arXiv:2407.10925.
16. Baeza-Yates, R.A.; Gavalda, R.; Navarro, G.; Scheihing, R. Bounding the Expected Length of Longest Common Subsequences and Forests. *Theory Comput. Syst.* **1999**, *32*, 435–452. <https://doi.org/10.1007/s002240000125>.
17. Alexander, K.S. The Rate of Convergence of the Mean Length of the Longest Common Subsequence. *Ann. Appl. Probab.* **1994**, *4*, 1074–1082. <https://doi.org/10.1214/aoap/1177004903>.
18. Lember, J.; Matzinger, H.; Torres, F. The rate of the convergence of the mean score in random sequence comparison. *Ann. Appl. Probab.* **2012**, *22*, 1046–1058. <https://doi.org/10.1214/11-AAP778>.
19. Tiskin, A. The Chvátal–Sankoff problem: Understanding random string comparison through stochastic processes. In *Proceedings of the Representation Theory, Dynamical Systems, Combinatorial Methods, Part XXXIV. Zapiski Nauchnykh Seminarov POMI*; Vershik, A.M., Vassiliev, N.N., Eds.; Springer: Berlin/Heidelberg, Germany, 2022; pp. 191–224.
20. Masek, W.J.; Paterson, M.S. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.* **1980**, *20*, 18–31. [https://doi.org/10.1016/0022-0000\(80\)90002-1](https://doi.org/10.1016/0022-0000(80)90002-1).
21. Myers, G. A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming. *J. ACM* **1999**, *46*, 395–415. <https://doi.org/10.1145/316542.316550>.
22. Hirschberg, D.S. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Commun. ACM* **1975**, *18*, 341–343. <https://doi.org/10.1145/360825.360861>.
23. Kundeti, V.; Rajasekaran, S. Extending the four Russian algorithm to compute the edit script in linear space. In *Proceedings of the International Conference on Computational Science*; Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 893–902. https://doi.org/10.1007/978-3-540-69384-0_94.

24. Abboud, A.; Backurs, A.; Williams, V.V. Tight Hardness Results for LCS and Other Sequence Similarity Measures. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science*; IEEE: Piscataway, NJ, USA, 2015; pp. 59–78. <https://doi.org/10.1109/FOCS.2015.14>.
25. Backurs, A.; Indyk, P. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (Unless SETH is False). In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery; Association for Computing Machinery: New York, NY, USA, 2015; STOC '15, pp. 51–58. <https://doi.org/10.1145/2746539.2746612>.
26. Ukkonen, E. Algorithms for approximate string matching. *Inf. Control* **1985**, *64*, 100–118. [https://doi.org/10.1016/S0019-9958\(85\)80046-2](https://doi.org/10.1016/S0019-9958(85)80046-2).
27. Landau, G.M.; Myers, E.W.; Schmidt, J.P. Incremental String Comparison. *SIAM J. Comput.* **1998**, *27*, 557–582. <https://doi.org/10.1137/S0097539794264810>.
28. Andoni, A.; Krauthgamer, R.; Onak, K. Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*; IEEE: Piscataway, NJ, USA, 2010; pp. 377–386. <https://doi.org/10.1109/FOCS.2010.43>.
29. Chakraborty, D.; Das, D.; Goldenberg, E.; Koucky, M.; Saks, M. Approximating Edit Distance within Constant Factor in Truly Sub-Quadratic Time. In *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science*; IEEE: Piscataway, NJ, USA, 2018; pp. 979–990. <https://doi.org/10.1109/FOCS.2018.00096>.
30. Chakraborty, D.; Das, D.; Goldenberg, E.; Koucky, M.; Saks, M. Approximating edit distance within constant factor in truly sub-quadratic time. *J. ACM* **2020**, *67*, 36. <https://doi.org/10.1145/3422823>.
31. Rubinstein, A.; Song, Z. Reducing approximate Longest Common Subsequence to approximate Edit Distance. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*; SIAM: Philadelphia, PA, USA 2020; pp. 1591–1600. <https://doi.org/10.1137/1.9781611975994.98>.
32. Calvo-Zaragoza, J.; Oncina, J.; de la Higuera, C. Computing the Expected Edit Distance from a String to a Probabilistic Finite-State Automaton. *Int. J. Found. Comput. Sci.* **2017**, *28*, 603–621. <https://doi.org/10.1142/S0129054117400093>.
33. Boche, H.; Böck, Y.; Deppe, C. On the Effectiveness of Fekete's Lemma in Information Theory. In *Proceedings of the 2020 IEEE Information Theory Workshop (ITW)*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5. <https://doi.org/10.1109/ITW46852.2021.9457634>.
34. Spencer, J. *Asymptopia*; American Mathematical Society: Providence, RI, USA, 2014. <https://doi.org/10.1090/stml/071>.
35. McDiarmid, C. On the method of bounded differences. *Surv. Comb.* **1989**, *141*, 148–188. <https://doi.org/10.1017/cbo9781107359949.008>.
36. Bhardwaj, V.; Pevzner, P.A.; Rashtchian, C.; Safonova, Y. Trace Reconstruction Problems in Computational Biology. *IEEE Trans. Inf. Theory* **2021**, *67*, 3295–3314. <https://doi.org/10.1109/TIT.2020.3030569>.
37. Majithia, J.C.; Levan, D. A note on base-2 logarithm computations. *Proc. IEEE* **1973**, *61*, 1519–1520. <https://doi.org/10.1109/PROC.1973.9318>.
38. Kiwi, M.; Loebl, M.; Jiří, M. Expected length of the longest common subsequence for large alphabets. *Adv. Math.* **2005**, *18*, 480–498. <https://doi.org/10.1016/j.aim.2004.10.012>.
39. Sankoff, D.; Kruskal, J. Common Subsequences and Monotone Subsequences. In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*; Addison-Wesley Publishing Company: Reading, MA, USA, 1983; pp. 363–365.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.