On the Longest Common Subsequence between two random correlated strings

Pedro Costa
Advisor: João Ribeiro

1 Introduction

The longest common subsequence (LCS) of two strings is one of the main problems in combinatorial pattern matching. It forms the basis of several algorithms for automatic identification of the differences between two versions of a file [NHM20], useful in version control systems such as Git. There are also relevant applications in data compression [BAFA16], since recognising recurring subsequences becomes extremely useful for compression. In computational biology, while studying the evolution of long molecules such as proteins or nucleic acids, it is common practice to try to find large correspondences between them, which translates to finding a longest common subsequence. The Sequence Alignment problem asks for an optimal way of arranging the sequences of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences, and LCS plays an important role in many algorithms for this problem [KOH12].

Definition 1. *We say s is a subsequence of t if we can obtain s by deleting zero or more characters from t, denoted by $s \subseteq t$.*

Definition 2. *The LCS of two strings s and t is defined as the longest subsequence a common to s and t.*

A famous open problem [CS75] related to LCS is the expected length of the LCS of two random strings of length n uniformly distributed over an alphabet of size k, denoted by $EL_n^{(k)}$. We have

$$EL_n^{(k)} = \frac{1}{k^{2n}} \sum_{|s|=|t|=n} LCS(s,t).$$

It is known that $EL_n^{(k)}$ is asymptotically proportional to n and the Chvátal–Sankoff problem asks about the constant of proportionality γ_k, defined as the limit of $EL_n^{(k)}/n$ as $n \mapsto \infty$. For instance, if an alignment or common subsequence of two given sequences such as DNA molecules is relatively larger than $EL_n^{(k)}$, we may infer that it is more than a coincidence and that the result should be studied further.

In practical DNA-based data storage scenarios we often want to reconstruct a string x from some of its traces [YGM17, OAC+18], where each trace is obtained by deleting each bit in the original string with a given deletion probability d. The goal is to reconstruct x while minimizing the number of traces required [BKKM04, CGMR20], however in this work we will try to gain some understanding on the structure of these traces. For example, how close are two independent traces? It seems like a good way to measure distance in this setting is precisely by using the longest common subsequence. Due to its high regularity, we choose to work with the alternating string 101010 . . . but it would be interesting to extend this analysis to other initial strings.

Hence, in this work we consider a setting where two strings Y_1, Y_2 are randomly generated from an initial binary string $X = 101010\dots$ by deleting each bit of X independently with probability d, thus maintaining some degree of correlation. We investigate properties of the LCS between Y_1 and Y_2 and explore its relation to the Chvátal – Sankoff constant γ_2.

2 Preliminaries

We begin with some useful results regarding the Chvátal-Sankoff constants.

Definition 3. *A sequence $(a_n)_{n\geq 1}$ is called superadditive if it satisfies the inequality*

$$a_{n+m} \geq a_n + a_m$$

for all m, n.

Lemma 1. *$(EL_n^{(k)})_{n\geq 1}$ is superadditive.*

Proof. We want to show that $EL_{n+m}^k \geq EL_n^k + EL_m^k$ for any m, n. For any pair of random uniformly generated strings a_i $(i = 1, 2)$ of length $n + m$, we can decompose each a_i as $b_i + c_i$ with $|b_i| = n$, $|c_i| = m$ both being randomly uniformly generated as well$(i = 1, 2)$. Then, LCS(a_1,a_2) $\geq$ LCS(b_1,b_2)+LCS(c_1,c_2) because any common substring of both c_i can be concatenated to any common substring of both b_i to form a valid common substring to both a_i.

Since this works for any pair of strings, it also holds for the expected value of their LCS. $\square$

Lemma 2 (Fekete's Lemma [Fek23]). *Let $(a_n)_{n\in\mathbb{N}}$ be a superadditive sequence. Then*

$$\lim_{n\mapsto\infty} \frac{a_n}{n} \tag{1}$$

exists and is equal to $\sup_{n\in\mathbb{N}} \frac{a_n}{n}$.

It follows from Lemma 1 and Lemma 2 that

$$\gamma_k = \lim_{n\mapsto\infty} \frac{EL_n^{(k)}}{n} = \sup_{n\mapsto\infty} \frac{EL_n^{(k)}}{n}$$

exists. These limiting values γ_k are the Chvátal–Sankoff constants, named after Václav Chvátal and David Sankoff, who began investigating them in the mid-1970s [CS75]. The exact value of these proportions is not known even for $k = 2$, and a number of papers have proved upper and lower bounds on them. The best bounds yet were achieved by [Lue09], yielding $0.788071 \leq \gamma_2 \leq 0.826280$, and non-rigorous numerical evidence suggests that $\gamma_2 \sim 0.811$. It is also known that $1 \leq \gamma_k\sqrt{k} \leq e$ and $\lim_{k\mapsto\infty} \gamma_k\sqrt{k} = 2$.

2.1 Warm up problem

In this work we will study binary strings and the result of the deletion of some bits of those strings. After deleting one bit from a string s with r runs[1] there are exactly r possible distinct outcomes. It would be natural to wonder about the reverse process: starting from a given string s, how many strings t are there such that s could be obtained from t after a one bit deletion? What about a k bit deletion? In other words, given $s \in \{0,1\}^n$ and $k \in \mathbb{N}$, consider

$$A_{s,k} = \{t \in \{0,1\}^{n+k} \colon s \subseteq t\}.$$

[1]A run is a sequence of identical bits which cannot be made longer. For example, string 110001 has 3 runs.

Theorem 1. *Let $k \in \mathbb{N}$ and $s \in \{0,1\}^n$. Then $|A_{s,k}| = \sum_{i=0}^{i=k} \binom{n+k}{i}$.*

Proof. We are asked to count the number of binary strings of length $n+k$ containing s as a substring. We will instead count the number of such strings which do not contain s as a substring. For any $t \in \{0,1\}^{n+k}$, consider the longest prefix s' of s such that $s' \subseteq t$ and assume $s' \neq s$. Furthermore, consider the leftmost occurence of s' in t (i.e., the leftmost occurence of $s'[0]$, the leftmost occurence of $s'[1]$ after that, etc). This leftmost occurence of s' can be represented by the set S_t of indices of t picked by the greedy algorithm described to find it. For example, if $s = 101$ and $t = 11000$ then $s' = 10$ and $S_t = \{0, 2\}$.

However, S_t uniquely defines the string t! Any set S of at most $n-1$ distinct indices of t produces exactly one string t such that $S_t = S$. So there are exactly

$$2^{n+k} - \sum_{i=0}^{i=n-1} \binom{n+k}{i} = \sum_{i=0}^{i=k} \binom{n+k}{i}$$

strings in $A_{s,k}$. $\square$

Remark 1. *When $k = 1$, $|A_{s,k}| = n + 2$. This formula also holds when $k = 0$.*

3 Longest Common Subsequence between two random correlated strings

Consider the alternating string $X = 101010\ldots$ of length n (n is even) and a random process which deletes each bit of X independently with probability d to generate strings Y_1 and Y_2, independently from one another. What can be said about the LCS of Y_1 and Y_2? For instance, what is its expected value? Are there any nice concentration bounds?

3.1 Concentration bounds

The following lemma will be useful:

Lemma 3 (Multiplicative Chernoff bound [MU05, Theorems 4.4 and 4.5])**.** *Suppose $X_1, \ldots, X_n$ are independent Bernoulli random variables. Let X denote their sum and let $\mu = E[X]$ denote the expected value of X. Then,*

$$\mathbb{P}[X \geq (1+\delta)\mu)] \leq e^{-\delta^2\mu/3}, \qquad 0 < \delta \leq 1,$$
$$\mathbb{P}[X \leq (1-\delta)\mu)] \leq e^{-\delta^2\mu/2}, \qquad 0 < \delta < 1.$$

Let Z be the string consisting of bits of X that show up in both Y_1 and Y_2. Then,

$$LCS(Y_1, Y_2) \geq |Z|.$$

We have $E[|Z|] = (1-d)^2 n$. Moreover, $|Z| = \sum_{i=1}^{n} Z_i$ with the Z_i's iid Bernoulli with success probability $(1-d)^2$. Then, Lemma 3 with $\delta \in \Theta(\log n/\sqrt{n})$ tells us that

$$\mathbb{P}[|Z| \geq (1-d)^2 n - O(\sqrt{n}\log n)] \geq 1 - e^{-\Theta((\log n)^2)}$$

On the other hand,

$$LCS(Y_1, Y_2) \leq |Y_1|.$$

Similarly, $|Y_1|$ is the sum of n iid Bernoulli with success probability $1-d$. From Lemma 3,

$$\mathbb{P}[|Z| \leq (1-d)n + O(\sqrt{n}\log n)] \geq 1 - e^{-\Theta((\log n)^2)}.$$

So, with high probability, we get

$$(1-d)^2 - o_n(1) \leq \frac{LCS(Y_1, Y_2)}{n} \leq (1-d) + o_n(1).$$

Now let U_1, Z_1 be the number of one's and zero's of Y_1, respectively. Define U_2, Z_2 analogously and $L = \max(\min(U_1, U_2), \min(Z_1, Z_2))$. Then,

$$LCS(Y_1, Y_2) \geq L.$$

All U_i, Z_i are i.i.d., $U_1 \sim Bin(1-d, n/2)$ and $E[U_1] = (1-d)n/2$. Let $a \in \mathbb{R}_{\geq 0}$. We have

$$\mathbb{P}[L \leq a] = \mathbb{P}[\min(U_1, U_2) \leq a]^2 \leq 4\mathbb{P}[U_1 \leq a]^2.$$

Repeating the same argument, with high probability we get

$$\frac{1-d}{2} - o_n(1) \leq \frac{LCS(Y_1, Y_2)}{n}$$

The following Theorem sums up the results presented in this section.

Theorem 2. *With high probability,*

$$max(\frac{1-d}{2}, (1-d)^2) - o_n(1) \leq \frac{LCS(Y_1, Y_2)}{n} \leq (1-d) + o_n(1).$$

3.2 On the expected value

Denote by $E(n,d)$ the expected value of $LCS(Y_1, Y_2)$. If we fix the length n of the initial string X, $E(n,d)$ is a polinomial of degree at most $2n$ on the interval $[0,1]$. This can be seen by explicitly computing the expected value conditioned on the possible outcomes for Y_1 and Y_2. Furthermore, $E(n,0) = n$ and $E(n,1) = 0$. Normalizing by n, it may be of interest to study the behaviour of $E(n,d)/n$ as $n \mapsto \infty$. One trivial upper bound is the function $1-d$, since $LCS(Y_1, Y_2)$ is always at most the length of the smallest of the two strings and each of them have an expected length of $n(1-d)$.

Lemma 4. $(E(n,d))_{n \in \mathbb{N}}$ *is a superadditive sequence.*

Proof. Analogous to the proof of Lemma 1. □

By Lemma 2, since $(E(n,d))_{n \in \mathbb{N}}$ is superadditive, $\lim_{n \mapsto \infty} E(n,d)/n$ exists and is equal to its supremum. Let $E(d) = \lim_{n \mapsto \infty} E(n,d)/n$, $d \in [0,1]$. Considering $E(2,d) = (1-d)^2$, this yields

$$\left(1-d\right)^2 \leq E(d) \leq 1-d.$$

Lemma 5. $E(n,d)$ *is non increasing for* $d \in [0,1]$.

Proof. Let $b \leq c$ and $n \in \mathbb{N}$. We can think of deleting each bit in the original string with probability c as a two step process: first delete the bit with probability b and then, in case it was not deleted, delete it with probability $1 - \frac{1-c}{1-b}$.
This allows us to think of $E(n,c)$ as the result of deleting each bit with probability b, resulting in an expected value for $LCS(Y_1, Y_2)$ of $E(n,b)$, and then possibily deleting some extra bits. This cannot improve on the LCS, thus $E(n,b) \geq E(n,c)$. □

Remark 2. *Lemma 5 implies that* $E(d)$ *itself is non increasing.*

Remark 3. *Lemma 5 does not depend on the initial string* X *chosen.*

4 Lower Bounds via Markov Chains

In [Dan94] and [PD94], Dančík and Paterson present a finite automaton that models an algorithm which finds a common subsequence on two infinite strings (tapes). These are generated in a different setup than ours, over an alphabet of size k with each character having probability $1/k$ of appearing in each entry of the strings. By analysing the associated Markov chain, a bound on the expected length of the LCS is found.

These automatons alternatively read from each one of the two unbounded tapes. Baeza-Yates, Gavaldá, Navarro, and Scheihing [BGNS99] extend this idea by modifying them be able to read at the same time from both strings, allowing the possibility of applying some symmetry rules which reduce the number of states. The underlying approach in our work is based on this work to obtain lower bounds for $E(d)$, $d \in [0,1]$.

If X is taken to be infinitely long, we can think of Y_i $(i = 1, 2)$ as being continuously generated according to the following process: for each bit j in X, from left to right, add j to Y_i with probability $1 - d$, otherwise delete it and move to the next bit. Given that we just added bit j to Y_i, the probability p_1 that the next bit in Y_i is not the same as j is

$$p_1 = (1-d) + d^2(1-d) + d^4(1-d) + ... = \frac{1}{1+d}. \tag{2}$$

Analogously, the probability p_0 that the next bit in Y_i is the same as j is $d/(1+d)$.

Our algorithm continuously builds upon an existing common subsequence of Y_1 and Y_2, starting with the empty string. Each state $s \in S$ of the automaton is represented by two strings $[u, v]$ which are the letters not yet used in each tape. When reading a new character simultaneously from Y_1 and Y_2, the automaton greedily tries to improve on the existing common subsequence, according to some rule to be defined[2] representing the greedy choice of the algorithm, which will be explored later. Since this choice is made solely based on the letters not used yet, all information about the past has been lost. This means we obtain a lower bound since there could be a common subsequence between Y_1 and Y_2 longer than the one obtained by this algorithm. However, this greatly simplifies the analysis of the problem.

Denoting the empty string by λ, we have to distinguish between two scenarios. If we are on state $[0, \lambda]$ (this means one of the tapes has a yet to be used 0 bit and the other one used its last bit in a previous matching), by (2) the transition probabilities depend on whether the last bit we received on each tape was a 0 or a 1. For that reason, we use λ_i $(i = 1, 2)$ to represent a tape which has no unused bits, and used an i bit in its last matching. We can reduce the number of states due to some symmetries:

- If $u, v \in \{0, 1\}^n \cup \{\lambda_0, \lambda_1\}$, then the states $[u, v]$ and $[v, u]$ are identical, and thus merged.
- The state $[\lambda_0, \lambda_1]$ does not exist, otherwise there would have been an illegal 01 match.

[2] In [BGNS99], several possibilities to generate the next state in a transition were considered. The most successful one was: given a state $s = [u, v]$ and a pair of letters $[x, y]$, the next state is given by $s' = [u', v']$ such that $ux = \omega_1 u'$ and $vy = \omega_2 v'$ where ω_1 and ω_2 are the strings that maximise

$$\frac{LCS(\omega_1, \omega_2)}{|\omega_1| + |\omega_2|}.$$

4.1 A simple Markov chain

As an example we start by defining a seven state Markov chain where we consider only the states $[u, v]$ where u and v do not contain more than one bit each. Our state space is then

$$S = \{[\lambda_0, \lambda_0], [\lambda_1, \lambda_1], [1, \lambda_1], [1, \lambda_0], [0, \lambda_1], [0, \lambda_0], [0, 1]\}$$

and the full automaton diagram is represented in Figure 1. For clarity purposes, the weights of

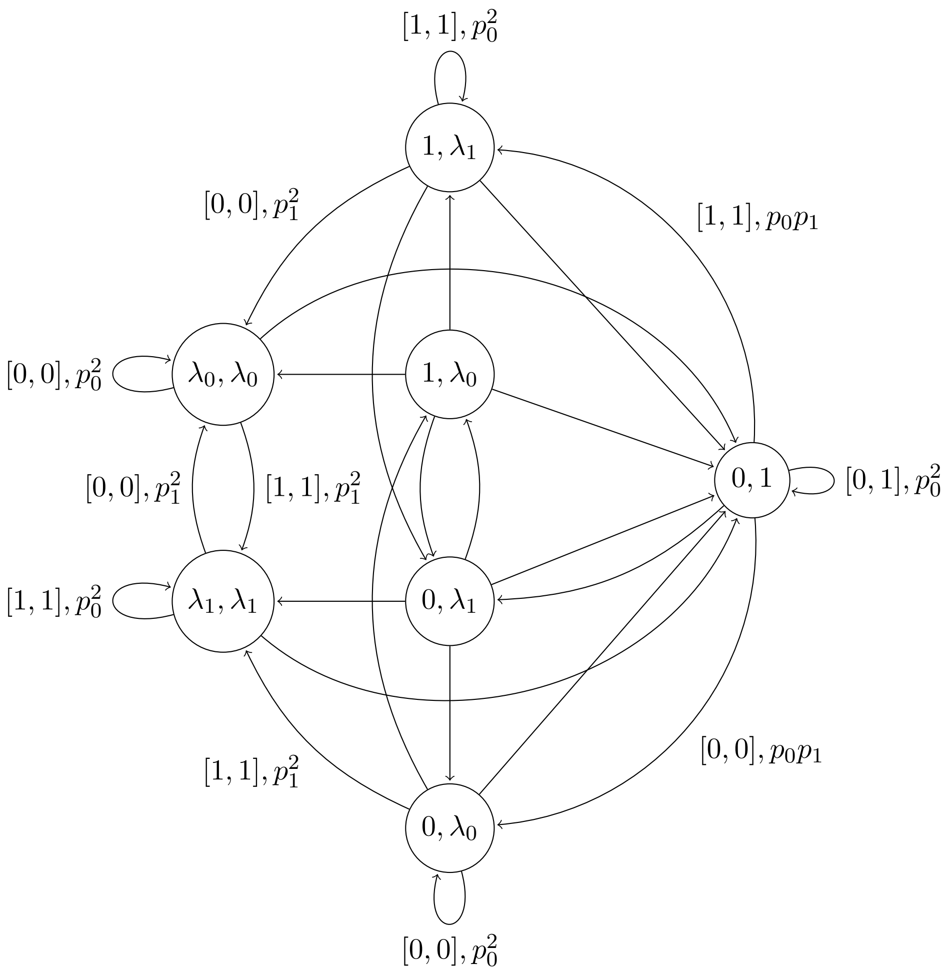

Figure 1: Seven-state automaton with some labelled edges.

some edges were omitted from Figure 1. The full transition matrix is Equation (3).

$$P = \begin{bmatrix} p_0^2 & p_1^2 & 0 & 0 & 0 & 0 & 2p_0p_1 \\ p_1^2 & p_0^2 & 0 & 0 & 0 & 0 & 2p_0p_1 \\ p_1^2 & 0 & p_0^2 & 0 & p_0p_1 & 0 & p_0p_1 \\ p_0p_1 & 0 & p_0p_1 & 0 & p_1^2 & 0 & p_0^2 \\ 0 & p_0p_1 & 0 & p_1^2 & 0 & p_0p_1 & p_0^2 \\ 0 & p_1^2 & 0 & p_0p_1 & 0 & p_0^2 & p_0p_1 \\ 0 & 0 & p_0p_1 & 0 & p_1^2 & p_0p_1 & p_0^2 \end{bmatrix} \tag{3}$$

Note that when we are in state $[0, 1]$ and we receive characters $[1, 0]$, with probability p_1^2, an arbitrary choice is made to match the ones and leave the new zero for a future match, leading to state $[0, \lambda_1]$.

As expected, this choice has no impact on the final results obtained with the automaton.
Figure 1 is a strongly connected Markov chain. In the limit, the probability of being in a given state converges to the solution of

$$P^T\pi = \pi, \quad \sum_{i=1}^{7} \pi_i = 1.$$

After computing these probabilities, we can obtain a lower bound on $E(d)$ based on the expected length of the output formed in each transition

$$c_{avg} = \sum_{s\in S} \pi_s c_s, \tag{4}$$

where c_s is the expected contribution of state s to the output. Using *Mathematica* software [Cos24], these were computed obtaining

$$c_{avg} = \frac{1 + 3d + 4d^2 + d^3}{1 + 4d + 6d^2 + 2d^3}.$$

Finally, with high probability this process will run at least $n(1 - d) - O(\sqrt{n}logn)$ times, so in order to get a lower bound for $E(d)$ we need to multiply c_{avg} by $(1 - d)$. This gives us the final lower bound of

$$f(d) = \frac{(1 - d)(1 + 3d + 4d^2 + d^3)}{1 + 4d + 6d^2 + 2d^3}. \tag{5}$$

Figure 2 is a visual representation of the results obtained so far for bounding $E(d)$. The red line is the trivial upper bound. The coloured curves are the exact values of $E(n, d)/n$ for $n = 2, 4, ..., 12$ and the black curve is the new lower bound.

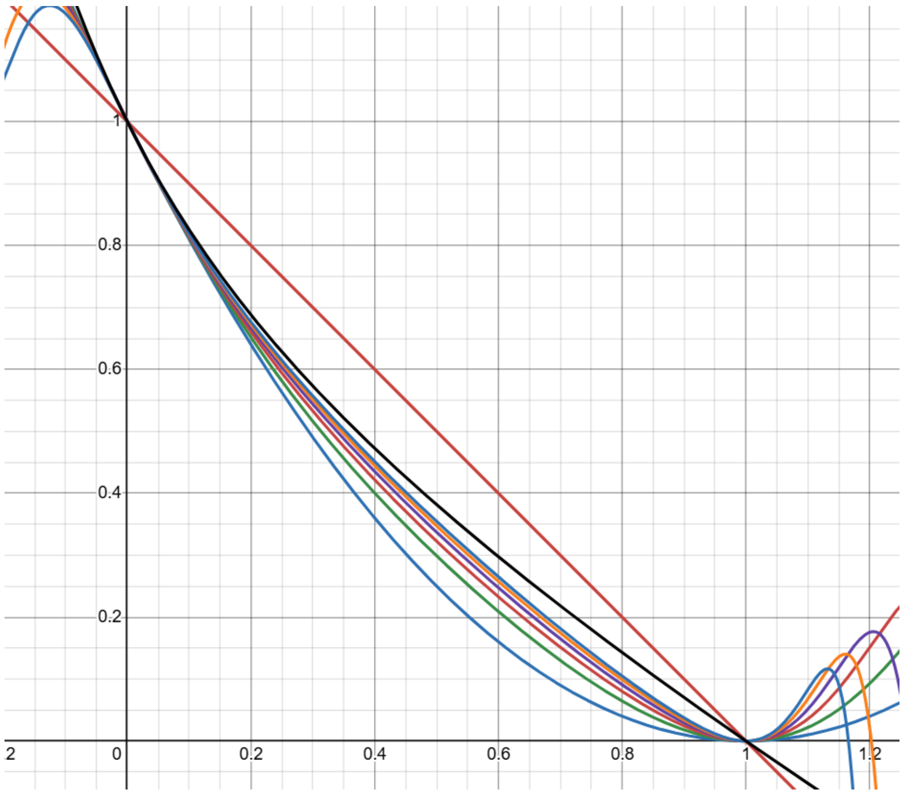

Figure 2: Upper bound (red line) and lower bounds obtained. The seven state Markov chain yielded the lower bound in black.

4.2 Computing larger Markov chains

This approach can be extended to build larger Markov chains with more states and produce better lower bounds. First we fix a parameter h representing the maximum memory each state can hold from each string. The idea is to start with a set of states $S = \{[\lambda_0, \lambda_0]\}$ and expand those states, obtaining S'. Expanding a state s means to concatenate all possible pairs of bits to s and then greedily choosing which characters to match and which to keep, obtaining 4 new (non necessarily distinct) states. We impose the condition that the new states will have at most h characters per string, dropping one or two characters if needed. Then we repeat the process iteratively from S', generating new states and transitions until no new states are found. This is bound to happen due to our parameter h.

This was carried out in *C++* [Cos24] with $h = 1, 2, 3, 4$ and the resulting Markov Chain inputted in the same Mathematica notebook as in Section 4.1. The lower bounds obtained are shown below:

h	Number of states	**Lower bound**
1	7	$\frac{(1-d)(1+3d+4d^2+d^3)}{1+4d+6d^2+2d^3}$
2	18	$\frac{(1-d)(1+6d+17d^2+27d^3+22d^4+9d^5+d^6)}{(1+7d+22d^2+37d^3+32d^4+14d^5+2d^6)}$
3	39	$\frac{(1-d)(1+10d+47d^2+135d^3+257d^4+335d^5+304d^6+186d^7+72d^8+15d^9+d^{10})}{(1+11d+56d^2+171d^3+341d^4+461d^5+430d^6+270d^7+108d^8+24d^9+2d^{10})}$
4	78	$(1-d)\times$Ratio of two degree 16 polynomials

These new lower bounds are increasingly better, however they do not improve much upon what had already been achieved in Figure 2. The gain from each new value of h decreases quickly, implying that this algorithmic approach would not benefit much from more computational power.

5 Monte Carlo simulations and Conclusion

In this section we aim to conduct numerical experiments to approximate the value of $E(d) := \lim_{n\mapsto\infty} E(n,d)/n$ for different values of d, so as to find out whether our lower bounds came close to the true value of $E(d)$.

For each $d \in \{0.1, 0.2, ..., 0.9\}$, 2000 pairs of strings (Y_1, Y_2) were generated with a fixed n (ranging from 1000 to 5000) and the average LCS between them was computed. The results were as follows:

d	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
n	1000	1500	1500	1500	1500	1500	2000	2500	5000
$\widetilde{E}(d)$	0.842	0.720	0.612	0.513	0.421	0.331	0.246	0.162	0.0805
$\frac{\widetilde{E}(d)}{1-d}$	0.936	0.900	0.874	0.855	0.842	0.8275	0.820	0.810	0.805

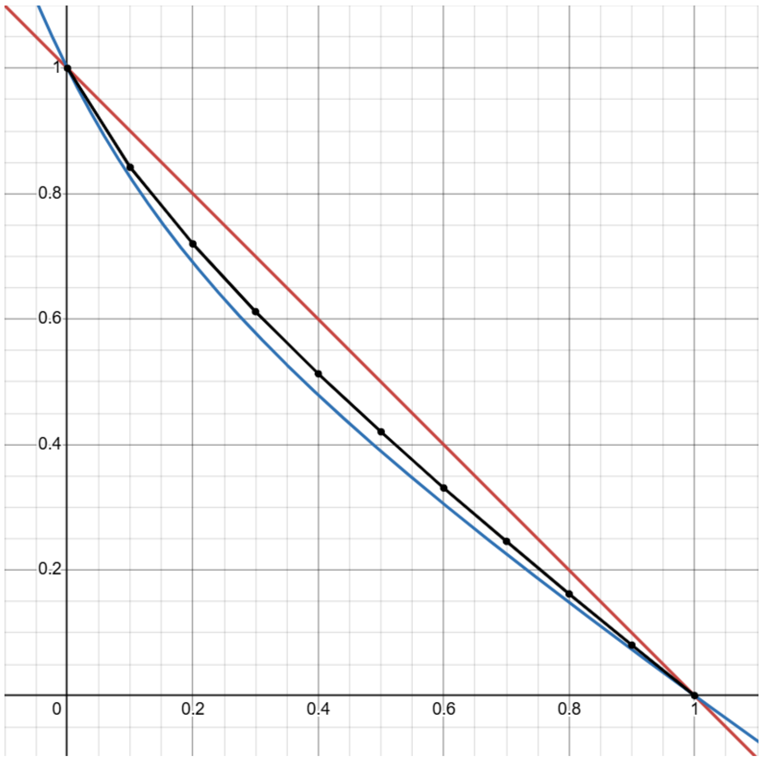

Figure 3: Results of the Monte Carlo simulations. The blue line corresponds to the lower bound given by the 78 state Markov Chain.

The results seem to suggest that the true values of $E(d)$, $d \in [0,1]$, lie closer to our lower bounds than to the trivial $(1-d)$ upper bound. It would be of interest to explore some of the upper bound techniques used in the classical Chvátal-Sankoff setting with two independent random strings to shorten the gap in Figure 3.

One interesting thing to notice is that according to our simulations, there appears to be a limiting value $D \in [0.7, 1]$ such that

$$E(d) \geq 0.811(1-d) \Leftrightarrow d < D$$

Keeping in mind that $\gamma_2 \sim 0.811$, this would suggest that for values of d smaller than D, the strings Y_1 and Y_2 are indeed correlated since their expected LCS is larger than if they were randomly uniformly chosen as strings of length $n(1-d)$. On the contrary, in case $D < 1$ there would be values of d which make (Y_1, Y_2) even less correlated than in the uniform case.
Even setting aside the approximation $\gamma_2 \sim 0.811$, our best lower bound guarantees that for $d < 0.297$, $E(d) > 0.826280(1-d) \geq \gamma_2(1-d)$ and thus Y_1 and Y_2 are correlated.

Conjecture 1. *$D = 1$ and $\lim_{d \mapsto 1} \frac{E(d)}{1-d} = \gamma_2$.*

References

[BAFA16] Richard Beal, Tazin Afrin, Aliya Farheen, and Donald Adjeroh. A new algorithm for "the LCS problem" with application in compressing genome resequencing data. *BMC Genomics*, 17 Suppl 4(Suppl 4):544, August 2016.

[BGNS99] Ricardo A Baeza-Yates, Ricard Gavaldá, Gonzalo Navarro, and Rodrigo Scheihing. Bounding the expected length of longest common subsequences and forests. *Theory of Computing Systems*, 32:435–452, 1999.

[BKKM04] Tuundefinedkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, page 910–918, USA, 2004. Society for Industrial and Applied Mathematics.

[CGMR20] Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and João Ribeiro. Coded trace reconstruction. *IEEE Transactions on Information Theory*, 66(10):6084–6103, 2020.

[Cos24] Pedro Bezerra Costa. Automated Markov Chains, July 2024. https://github.com/Pbezz/Bachelors-Thesis.git.

[CS75] Vaclàv Chvátal and David Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12(2):306–315, 1975.

[Dan94] Vladimír Dancík. *Expected length of longest common subsequences.* PhD thesis, University of Warwick, 1994. Unpublished.

[Fek23] M. Fekete. Über die verteilung der wurzeln bei gewissen algebraischen gleichungen mit ganzzahligen koeffizienten. *Mathematische Zeitschrift*, 17(1):228–249, Dec 1923.

[KOH12] Arabi Keshk, Mohammed Ossman, and Lamiaa Hussein. Fast longest common subsequences for bioinformatics dynamic programming. *International Journal of Computer Applications*, 57:12–18, 11 2012.

[Lue09] George S. Lueker. Improved bounds on the average length of longest common subsequences. *J. ACM*, 56(3), may 2009.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[NHM20] Yusuf Sulistyo Nugroho, Hideaki Hata, and Kenichi Matsumoto. How different are different diff algorithms in git? *Empirical Software Engineering*, 25(1):790–823, Jan 2020.

[OAC$^+$18] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z. Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, Christopher N. Takahashi, Sharon Newman, Hsing-Yeh Parker, Cyrus Rashtchian, Kendall Stewart, Gagan Gupta, Robert Carlson, John Mulligan, Douglas Carmean, Georg Seelig, Luis Ceze, and Karin Strauss. Random access in large-scale dna data storage. *Nature Biotechnology*, 36(3):242–248, Mar 2018.

[PD94] Mike Paterson and Vlado Dančík. Longest common subsequences. In Igor Prívara, Branislav Rovan, and Peter Ruzička, editors, *Mathematical Foundations of Computer Science 1994*, pages 127–142, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[YGM17] S. M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and error-free dna-based data storage. *Scientific Reports*, 7(1):5011, Jul 2017.