On Compressing Collections of Substring Samples

Golnaz Badkobeh[1], Sara Giuliani[2,*], Zsuzsanna Lipták[2] and Simon J. Puglisi[3]

[1]Goldsmiths, University of London, UK
[2]University of Verona, Italy
[4]University of Helsinki, Finland

Abstract

Given a string $X = X[1..n]$ of length n, and integers m and d, such that $n > m \geq 2d > 0$, we consider the problem of compressing the string S formed by concatenating the substrings of X of length m starting at positions $i \equiv 1 \pmod{d}$. In particular, we provide an upper bound of $(2n - m)/d + 2z + (m - d)$ on the size of the Lempel-Ziv (LZ77) parsing of S, where z is the size of the parsing of X. We also show that a related bound holds regardless of the order in which the substrings are concatenated in the formation of S. If X is viewed as a genome sequence, the above substring sampling process corresponds to an idealized model of short read DNA sequencing.

Keywords

Lempel-Ziv, LZ77, data compression, strings, repetitiveness, combinatorics on words, parsing, short reads

1. Introduction

We consider the problem of compressing a set of substrings sampled from a string. In particular, given a string X of length n and two integer parameters d and m, $m \geq 2d$, we call a *sample* of X a substring of X of length m starting at any position $1 + i \cdot d$ in X, where $i \geq 0$. We refer to the samples as $S_i = X[(i-1) \cdot d + 1..(i-1) \cdot d + m]$, for $i \geq 1$, and to the concatenation as $S = S_1 S_2 \cdots S_r$, where $r = \lfloor (n-m)/d \rfloor + 1$. Note that if $\frac{n-m}{d}$ is not an integer, then the final few characters of X will not be part of any sample.

If X is viewed as a genome sequence, the above substring sampling process corresponds to an idealized model of short read DNA sequencing. In the language of genome sequencing, m is the *read length* and m/d is the so-called *coverage* (the number of samples that cover a given position in X, on average). Our assumption that $m \geq 2d$ corresponds to a coverage of at least 2, which is the relevant case for DNA sequencing. S represents a file of short read sequences — the typical output of a sequencing experiment. Our sampling process idealizes short read sequencing in at least two ways. Firstly, short read sequencing may produce strings of slightly different length and may introduce errors — insertions, deletions, and substitutions of letters — to the sampled strings, albeit with fairly low probability for present-day short read technology

Proceedings of the 23rd Italian Conference on Theoretical Computer Science, Rome, Italy, September 7-9, 2022
*Corresponding author.
g.badkobeh@gold.ac.uk (G. Badkobeh); sara.giuliani_01@univr.it (S. Giuliani); zsuzsanna.liptak@univr.it (Zs. Lipták); simon.puglisi@helsinki.fi (Simon J. Puglisi)

($p < 0.01$ for Illumina short reads, for example). Secondly, in short read sequencing, coverage is not completely uniform, fluctuating across the genome for a variety of reasons (see, e.g. [1]).

The extraordinarily wide adoption of high-throughput sequencing in medical and evolutionary biology over the last decade has made short read data sets abundant. These data sets are also very large. For example, a typical human sequencing experiment might run at 20x coverage on a underlying genome of size $n = 3 \cdot 10^9$ nucleotides. The resulting read set in FASTQ format (a standard file format, which stores one ASCII-encoded short read sequence per line—akin to our S string) is then 60 gigabytes in size. The de facto standard in most labs (and large institutions such as, e.g., the NCBI) is to compress such files with the gzip all-purpose file compressor, which usually leads to a factor four reduction in size[1], or 15GB in our human sequencing example. A gzip'd large read set is thus, alas, still relatively large.

With the rapid growth in, and the need to store, short read data sets, specialized compressors that exploit properties inherent to such data sets will become paramount. Several read set specific compressors have now been developed (see, e.g., [2, 3, 4, 5]). None are yet in wide use. However, to our knowledge, no careful analysis of the compressibility of short read data sets—even in an idealized setting such as that described above—has been undertaken. This article addresses that need. In particular, we derive a non-trivial upper bound on the size of the LZ parsing [6, 7] of S, the concatenation of sampled substrings, in terms of n, m, d and the size of the LZ parsing of X. We also show a different upper bound that holds regardless of the order in which the samples are concatenated to form S.

In the next section we set notation and basic concepts and results used throughout. In Sections 3 and 4 we establish the above mentioned bounds, and in Section 5 we gauge the tightness of these bounds experimentally. We then sketch several directions for future work.

2. Preliminaries

Throughout we consider a string $X = X[1..n] = X[1]X[2]\dots X[n]$ of $|X| = n$ symbols drawn from an ordered alphabet Σ of size $|\Sigma| = \sigma$. For $i = 1, \dots, n$, we write $X[i..n]$ to denote the *suffix* of X of length $n - i + 1$, that is, $X[i..n] = X[i]X[i+1]\dots X[n]$. We will often refer to suffix $X[i..n]$ simply as "suffix i" or the "ith suffix". We write $X[1..i] = X[1]\dots X[i]$ to denote the *prefix* of X of length i. We write $X[i..j]$ to represent the *substring* (or *factor*) $X[i]X[i+1]\dots X[j]$ of X that starts at position i and ends at position j. If $i > j$, then $X[i..j] = \epsilon$, where ϵ is the empty string. Let $\mathrm{lcp}(i, j)$ denote the length of the longest common prefix of suffix i and suffix j.

For example, in the string $X = \mathsf{zzzzzipzip}$, $\mathrm{lcp}(2, 5) = 1 = |\mathsf{z}|$, and $\mathrm{lcp}(5, 8) = 3 = |\mathsf{zip}|$. For technical reasons we define $\mathrm{lcp}(i, 0) = \mathrm{lcp}(0, i) = 0$ for all i.

Definition 1 (LZ Factorization). *The LZ factorization of X is a factorization $X = f_1 f_2 \dots f_{z_X}$ of X into z_X* phrases *such that each phrase f_i (a substring of X) is either*

1. *a letter that does not occur in $f_1 \cdots f_{i-1}$, or*

[1] This can be loosely interpreted as gzip, a sliding-window dictionary compressor with window size too small to capture any large dispersed repeated substrings present in the file, essentially reducing the space used by each DNA letter from the 8 bits used in the plain ASCII encoding to the 2 bits that a flat minimal binary code for four letters would use.

2. *the longest substring that occurs at least twice in* $f_1 \cdots f_i$.

Thus, the LZ factorization of our example string $X = \mathsf{zzzzzipzip}$ produces the following $z_X = 5$ factors, $f_1 = \mathsf{z}$, $f_2 = \mathsf{zzzz}$, $f_3 = \mathsf{i}$, $f_4 = \mathsf{p}$, $f_5 = \mathsf{zip}$.

We denote with $\ell_i = |f_i|$ the length of phrase f_i, and with s_i the starting position of phrase f_i in X, i.e., $s_i = 1 + \sum_{j<i} \ell_j$. Finally, we denote with p_i the position of the first occurrence of substring f_i in X and we call substring $X[p_i..p_i + \ell_i - 1]$ the *source* of f_i. Note that phrases and sources are allowed to overlap, as is the case with f_2 in our example.

The following is a known upper bound on the number of phrases in the parsing.

Theorem 1 ([8], Theorem 5.20). *Let X be a string of length n over Σ, with $|\Sigma| = \sigma$. Then $z_X \leq Z$, where*

$$Z = \frac{n - (\sigma/(\sigma-1))}{\log_\sigma n - \log_\sigma \log_\sigma n - (1/(\sigma-1))}.$$

We will return to the above bound later in the paper, but for now we note that it is asymptotically tight. In particular, an appropriate prefix of the de Bruijn sequence contains at least $n/\lceil \log_\sigma n \rceil$ LZ phrases [8].

We close this section with another well-known property of the LZ parsing we make use of in the proof of our main theorem.

Fact 1. *Let T be a string, and $1 < i \leq j \leq |T|$. If $t = T[i..j]$ has an occurrence before i, i.e., if there exists an $i' < i$ s.t. $T[i'..i' + |t| - 1] = t$, then the interval $[i, j]$ can contain at most one starting position of a phrase.*

3. Upper Bound on the Number of LZ Phrases of the Concatenation of Samples

In this section, we show an upper bound on the number of LZ phrases of string S formed by concatenating samples from string X.

Formally, given a string X of length n and two integer parameters m, d, such that $m \geq 2d$, let $S = S_1 S_2 \cdots S_r$, where $r = \lfloor (n-m)/d \rfloor + 1$, and, for $i \in [1..r]$, $S_i = X[(i-1) \cdot d + 1..(i-1) \cdot d + m]$.

Let us write, for $i \geq 1$, $S_i = v_i x_i$, where $|v_i| = m - d$ and $|x_i| = d$. Then X and S can be written as follows, where $|u| < d$:

$$X = v_1 x_1 x_2 \cdots x_r u, \tag{1}$$

$$S = v_1 x_1 v_2 x_2 v_3 x_3 \cdots v_r x_r. \tag{2}$$

We will now count the number of phrases z_S of the LZ parsing of string S. Note that this is equivalent to counting the number of starting positions of phrases.

Lemma 1. *Let b_i and e_i denote the beginning resp. ending position of the v_i's in the factorization of S given in (2). Then, for $i > 1$, the interval $[b_i, e_i]$ can contain at most one starting position of a phrase.*

Proof. Since two consecutive samples S_i and S_{i+1} overlap by $m-d$ characters, it follows that, for all $i > 1$, $S_i = y_i v_{i+1}$, where y_i is the d-length prefix of S_i. Therefore, S contains the square $v_i v_i$ for every $i > 1$, and b_i is the start of the second occurrence of v_i in this square. By Fact 1, therefore, $[b_i, e_i]$ can contain at most one starting position. □

Next we will count the number of starting positions in the x_i's. For this, we consider phrases f of the LZ parsing of X, and count how many starting positions these induce in S in the worst case. We first need the definition of the projection of a substring of X on S (see also Figure 1).

Definition 2. *Let $1 \leq b \leq e \leq n$. We define the* projection of substring $X[b,e]$ on S *as a collection of substrings in S as follows: For x_ι, let b_ι and e_ι denote the starting resp. ending positions of x_ι in S w.r.t. the factorization given in (1). Then*

$$Proj_S([b,e]) = \begin{cases} (\bigcup_{\iota=i}^{j}[\iota m - d, \iota m]) \cup [(i-2)\cdot(m-d)+b, (i-1)m] \cup \\ [mj+m-d+1, j(m-d)+e] & \text{if } m-d < b \\ [b,e] & \text{if } b, e \leq m-d \\ [b, m-d] \cup Proj_S[m-d+1, e] & \text{if } b \leq m-d < e. \end{cases}$$

Informally, a projection in S of a substring U of X is the collections of substrings of S coinciding with regions of U, namely substrings of S which U consists of, w.r.t. the factorization given in (1). The first case (see Fig. 1) gives the general situation, when U starts after the prefix v_1 of X, which is covered only by the first sample S_1. The second case is when U lies fully within v_1, while the third case is when U starts within v_1 and ends after it.

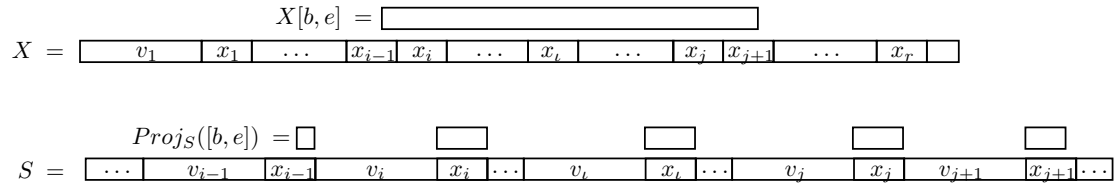

Figure 1: The projection of the substring $X[b,e]$ on the string S is shown. The number of substrings of X contained in the collection of substrings produced by the projection is the number of x_i's intersected by $X[b,e]$ in X, and they are all contained in the corresponding x_i's in S.

Definition 3. *Let f be a phrase of the LZ parsing of X, with starting position s and length ℓ. Define $g(f)$ as the number of starting positions in $Proj_S([s, s+\ell-1])$.*

Lemma 2. *Let $m \geq 2d$. Let f be a phrase of the LZ factorization of X, with starting position $s > m-d$. Then $g(f) \leq \frac{|f|}{d} + 2$.*

Proof. Let $\ell = |f|$, $k = \lceil \frac{\ell}{d} \rceil$ and $f = X[s..s+\ell-1] = x' x_i x_{i+1} \cdots x_j x''$, where x' and x'' are a proper suffix of x_{i-1} respectively a proper prefix of x_{j+1}, both possibly empty. We will show that each of these substrings is charged with at most one phrase starting position. From Fact 1, the claim follows.

By construction of S, each of the substrings $x', x_i, x_{i+1}, \ldots, x_j, x''$ will appear contiguously in S. The number of these substrings is either k or $k+1$. Additionally, a v substring separates the projections in S of each pair of contiguous aforementioned substrings of X.

Consider now the source f' of f occurring in X in some position $s' < s$. Then $X[s..s+\ell-1] = X[s'..s'+\ell-1] = u' x_{i'} x_{i'+1} \cdots x_{j'} u''$, where u' and u'' are a proper suffix of $x_{i'-1}$ respectively a proper prefix of $x_{j'+1}$, both possibly empty. As for f, the projection of f' is also split in S in such a way that the projection of each pair of mentioned substrings of f' is separated by a v substring in S.

Notice that, since $m \geq 2d$, each x_b in X has an occurrence in S also as suffix of $v_{b'+1}$, occurring immediately after x_{b+1} in S. This means that, even if the projections of f and f' in S will be split asynchronously, each of the substrings $x', x_i, x_{i+1}, \cdots, x_j, x''$ in the projection of f has already occurred as the concatenation of a suffix of some $v_{b'}$ intersecting the projection of f' and the contiguous $x_{b'+1}$.

There are at most $k+1$ factors $x', x_i, x_{i+1}, \ldots, x_j, x''$, and each of them has a previous occurrence in S. Therefore, by Fact 1, there will be at most $k+1$ phrase starting positions for f. See Fig. 2 for an illustration. □

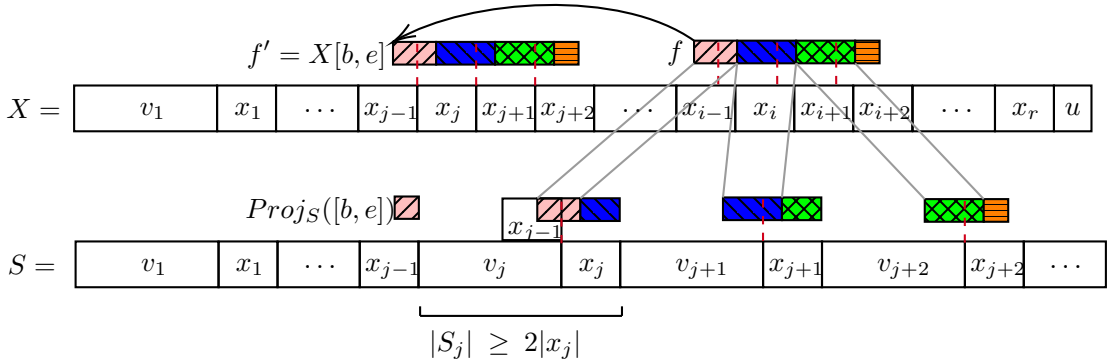

Figure 2: The original string X on top, and the concatenation S of the r samples of X below are shown. In particular, a phrase f and its corresponding source f' in X are represented with distinct colors for each of the x_i segments of X intersected by the phrase. Finally, the projection $Proj_S(f')$ of the source is shown. It is clear from the figure that, in S', the samples intersecting some string in the projection of f' fully contain at least one substring of the projection of f.

Lemma 3. *With respect to the factorization of S given in (2), the number of phrase starting positions in x_i's is at most $\frac{n}{d} + 2z_X$.*

Proof. The sum of the lengths of all phrases $f_1, \ldots, f_{z_X}$ in the LZ factorization of X is the length n of the string X. Therefore, we can bound the total contribution to z_S of X as follows:

$$\sum_{j=1}^{z_X} g(f_j) \leq \sum_{j=1}^{z_X} \left(\frac{|f_j|}{d} + 2\right) = \frac{n}{d} + 2z_X.$$

□

Theorem 2. *Let z_X be the number of phrases of the LZ parsing of X, and z_S the number of phrases of the LZ parsing of S. Then $z_S \leq \frac{2n-m}{d} + 2z_X + m - d$.*

Proof. We show the maximum number of phrase starting positions in S summing up the contribution of the $(m-d)$-length prefix of S, and of the remaining x_i's (Lemma 3) and v_i's (Lemma 1) substrings.

$$\begin{aligned} z_S &= \text{number of starting positions in } v_1 + \text{number of starting positions in } x_i\text{'s} \\ &\quad + \text{number of starting positions in } v_i\text{'s, for } i \geq 2 \\ &\leq (m-d) + \frac{n}{d} + 2z_X + \frac{n-m}{d} = \frac{2n-m}{d} + 2z_X + m - d. \end{aligned}$$

□

Theorem 2 essentially says that the number of LZ phrases of S is at most twice the number of samples plus twice the number of LZ phrases of X. Note that the important parameter which determines the number of samples is d, while m influences only the number of samples in which a given position occurs (i.e. m/d is the so-called coverage).

4. Upper Bound for Arbitrary Concatenation Order

This section examines the effect that the ordering of the samples in the concatenation has on the number of phrases.

As before, we are given a string X of length n and two integer parameters m and d, such that $m \geq 2d$. Let $r = \lfloor (n-m)/d \rfloor + 1$, and, for $i \in [1..r]$, $S_i = X[(i-1) \cdot d + 1..(i-1) \cdot d + m]$.

We will show that, regardless of the order in which the samples S_i of X are concatenated, the number of phrases in the LZ factorization of that concatenation is at most $n + \frac{2(n-m)}{d}$. We make this argument precise below.

Theorem 3. *Let S' be the concatenation of the samples of X in any order, then the number of phrases is $z_{S'} \leq n + \frac{2(n-m)}{d}$.*

Proof. Fix i. We will show that the number of phrase starting positions in sample S_i where it appears in S' is at most $2 + |u_i|$, where u_i is a specific substring of S_i. Let $k = \max\{i' < i \mid S_{i'} \text{ appears before } S_i \text{ in } S'\}$, and let w_i be the longest suffix of S_k which is also a prefix of S_i (i.e., w_i is the maximum overlap). Note that w_i may be empty. Similarly, let $k' = \min\{i' > i \mid S_{i'} \text{ appears before } S_i \text{ in } S'\}$, and let w'_i be the longest prefix of $S_{k'}$ which is also a suffix of S_i, again possibly empty. If $|w_i| + |w'_i| \geq m = |S_i|$, then set $u_i = \epsilon$. Otherwise, S_i can be written as $S_i = w_i u_i w'_i$, for some u_i.

This u_i is a substring of X that has not so far been covered by any sample in S', and this is because of the definition of k and k'. We call u_i the *new part of S_i*; in particular, if $u_i = \epsilon$, then the new part of S_i is empty.

By Fact 1, if w_i is non-empty, then at most one phrase starting position is contained within w_i. Similarly, if w'_i is non-empty, at most one starting position is contained within w'_i. The number of starting positions within the new part u_i can be trivially upper bounded by $|u_i|$.

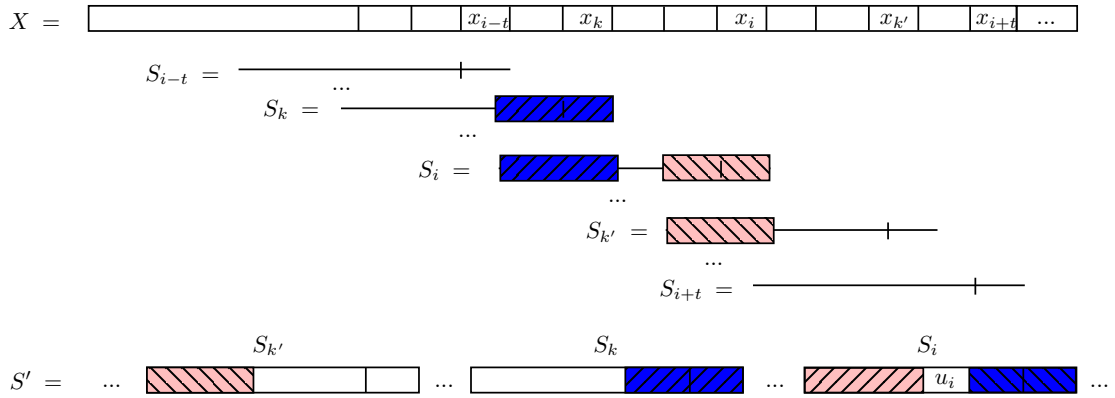

Figure 3: The contribution of the samples of X to $z_{S'}$, where S' is the concatenation of the samples in an arbitrary order. The colored substrings are the prefixes and suffixes with multiple occurrences in S' of some sample. Each sample consists of a prefix and a suffix (possibly empty) that has already occurred in S', and a substring in between them (i.e., labelled u_i, in white) that is not assumed to necessarily occur previously in S'.

Summing over all samples S_i, we thus get

$$z_{S'} \leq \sum_{i=1}^{r}(2 + |u_i|) = 2r + \sum_{i=1}^{r}|u_i| = 2r + n,$$

with the last equality using the fact that every position of X occurs exactly once in some u_i. □

Theorem 3 essentially says that the number of LZ phrases of S', i.e. of the concatenation of the samples in an arbitrary order, is at most the length of X plus twice the number of samples.

5. Experimental Results

In order to gauge the tightness of our bounds, we computed the number of phrases in the conventional LZ factorization (defined in Section 2) of the concatenation of the samples taken from each of the texts in Table 1. We did this for a range of d and m parameters, and concatenated the samples both in string order and after a random shuffling.

5.1. Data

Our test data, which contains files of varying repetitiveness is shown in Table 1.

By nature, viruses contain very little recurrent genetic heritage, and so a viral genome represents a real-world non-repetitive string. We used a genome of SARS-CoV2 taken from the COVID-19 Data Portal [9] of length 29 836, that is factorized in 4 373 LZ phrases. We also performed experiments on extreme cases between which real-life genomes lie: Fibonacci words and random words.

Data	Description	n	z	n/z
SARS-CoV2	Taken from the COVID-19 Data Portal [9]	29 835	4373	6.8
50 SARS-CoV2	Concatenation of 50 virus genomes taken from the COVID-19 Data Portal [9]	1 490 134	5421	275
Fibonacci word	Fibonacci word of order 22	28 657	22	1302.6
Random	Word over an alphabet of size 4 built with python function `random.choices()`	29 835	4575	6.5

Table 1
Data used in the experiments. The table shows the length (n) and the number of phrases (z) of each text used for the experiments.

Let $s_0 = \mathtt{b}$, $s_1 = \mathtt{a}$, the Fibonacci word of order $i+1$ is the binary word $s_{i+1} = s_i s_{i-1}$. In other words, the Fibonacci word of order $i+1$ is the concatenation of the Fibonacci words of the two previous orders.

By construction, Fibonacci words have a very high degree of repetition, resulting in very few phrases with respect to their length. Random words, on the other hand, could be considered, instead, not repetitive at all. To perform a comparison with the virus genome, we chose a random word of the same length of the genome, and the Fibonacci word of length 28 657, the closest to the length of the genome. The strings are factorized in 4 575 respectively 22 phrases.

We also performed experiments with larger data. In particular, we counted the number of phrases of the concatenation of 50 SARS-CoV2 genomes, which, due to the high similarity of the individual genomes, constitutes a highly repetitive dataset.

5.2. Experiments

We are interested in simulating the coverage provided by the most used technologies in genome sequencing. The coverage is given by the number of samples that cover a given position in the text. In the context of genome sequencing, a reasonable coverage is given by large m and small d, in order to have the coverage as large as possible with reasonable sample lengths.

We perform the experiments on each of the aforementioned texts with $d = 1, 2, 4, 8, 16, 32, 64$, and m from 50 to 1 000, in increments of 10. We counted the number of phrases for $m > d$.

Figure 4, on top, displays the number of phrases for concatenations of samples in string order of a Fibonacci word (top-left corner), and of a random string of a similar length (top-right corner). Below, we show the counts for an arbitrary ordered concatenation of the samples of the same data. In all figures, the number of phrases for some selected m's are shown in distinct colours, in gray for all other m's. The corresponding coloured line shows our bounds for each m and d pair. Finally, the colored rounded points show the existing bound given by Kärkkäinen [8], see Theorem 1.

The analogous comparison is shown in Figure 5 between a single SARS-CoV2 genome and the concatenation of 50 genomes.

The overlapping of the bound lines in all plots reflects the small impact of the sampling length m in the bound for fixed d.

When the samples are concatenated in string order (on the top of both Figure 4 and 5), the number of phrases in the original string has a big influence on the bound. We can see that the bound lines in the Fibonacci word's plot are closer to the actual counts than in the other plots, where the number of phrases of the original strings is higher. On the other hand, because of the different nature of the bound for the samples in string order versus random order, we can see that the latter bound has a similar trend for the three strings of similar length. In this case, the length and the number of samples of the original string determine the bound rather than its number of phrases.

Comparing the bound given in this paper (lines) to the one reported in Theorem [8] by Kärkkäinen (rounded points), when repetitive strings are considered, our bound is not worse than the existing one for the string order concatenation, while it gets more loose with the concatenation in arbitrary order. See the plots for Fibonacci words and the concatenation of 50 genomes of SARS-CoV-2. On the other hand, for non repetitive strings, namely random strings and the single viral genome in the plots, the existing bound is often better than ours.

Data	$d = 1$	$d = 2$	$d = 4$	$d = 8$	$d = 16$	$d = 32$	$d = 64$
Fibonacci							
max m	1000	980	990	970	960	960	940
phrases	1336	1321	1308	1021	1007	910	603
bound	57 357	29 189	15 111	8049	4510.1	2733.1	1800.8
Fibonacci shuffled							
max m	1000	990	990	990	630	970	980
phrases	25 164	13 179	6802	3505	1802	942	521
bound	83 971	56 324	42 490.5	35 573.75	32 160.4	30 387.4	29 521.906 25
SARS-CoV2							
max m	80	150	50	70	50	90	210
phrases	44 438	27 823	14 880	8945	6759	5575	4973
bound	68 417	38 655	23 697.5	16 258.25	12 506.38	10 665.93	9821.09
SARS-CoV2 shuffled							
max m	200	260	350	540	780	680	760
phrases	45 725	27 898	17 538	11 620	7969	6142	5238
bound	89 108	59 412	44 574	37 185	33 468	31 658.25	30 744.63

Table 2
A summary of relevant results for the Fibonacci word and the single SARS-CoV2 genome. For each sampling frequency d, we report the value of m for which the maximum number of phrases in the concatenation of the samples is produced in both string order and after the random shuffling. We further show the counts and the value of our bound for the mentioned m.

In Table 2 we show, for fixed d, the maximum number of phrases in S, varying m. For strings X that are already very repetitive (Fibonacci word), the longer the string, the higher the number of phrases. On the other hand, for strings that are not that repetitive (single SARS-CoV2 genome), introducing long repetitions may decrease the number of phrase starting positions in contrast to having a shorter string with very short repetitions. The concatenation of 50

SARS-CoV2 genomes lies in the middle.

(a) Fibonacci word of length 28 657 (b) Random word of length 29 835

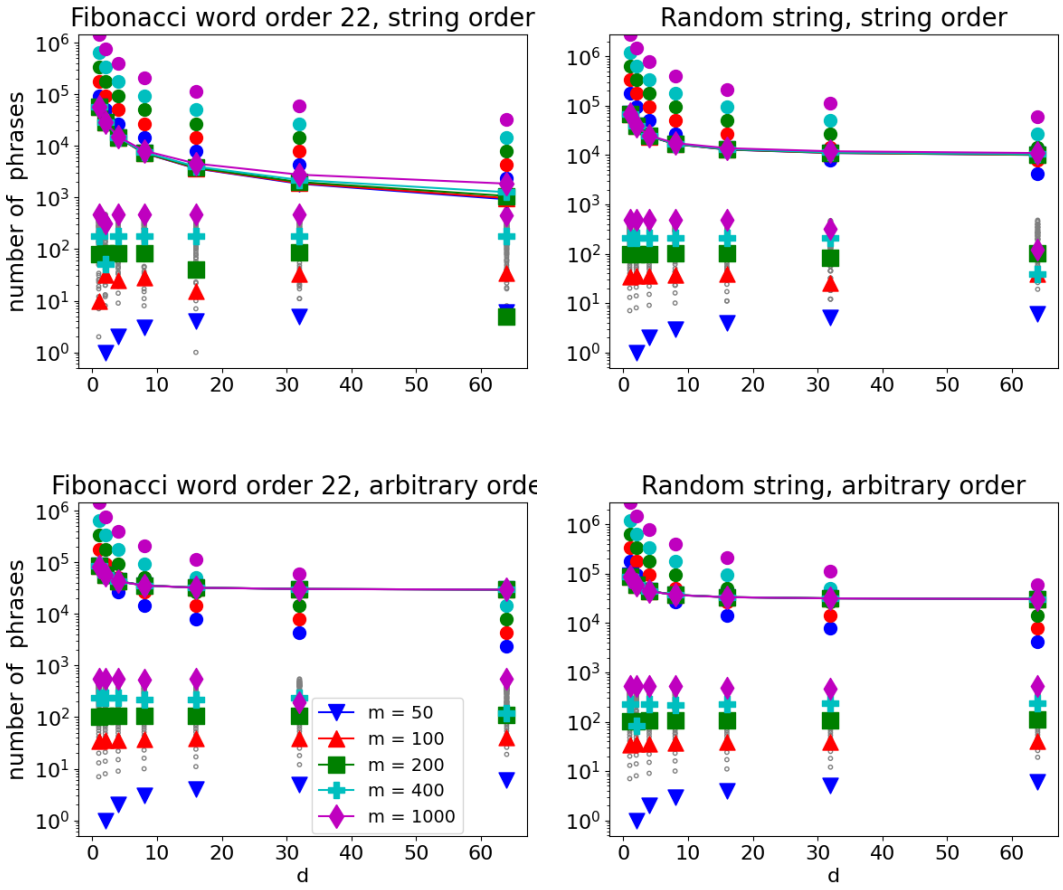

Figure 4: The number of phrases in log scale for concatenation of samples of a Fibonacci word (left column) and a random word (right column), both in string order (on top) and in an arbitrary chosen order (on bottom). The counts are shown for each concatenation of m-length samples at every d position. Colored markers indicate the number of phrases for $m = 50, 100, 200, 400, 1000$, while we use grey points to mark all others m's. The bounds shown in Theorem 2 and 3 for the concatenation in string order respectively random order are shown with colored and shaped lines accordingly to the corresponding m, d pair, while the coloured rounded points indicate the bound given by Kärkkäinen [8], see Theorem 1.

6. Concluding Remarks

This paper has explored the compressibility of collections of substrings sampled from a string. In particular, given a string $X = X[1..n]$ of length n with an LZ parsing size of z_X, and integers m and d, such that $n > m \geq 2d > 0$, we have shown that the size of the LZ parsing of the string

(a) SARS-CoV2 genome of length 29 835

(b) Concatenation of 50 SARS-CoV2 genome, of total length 1 490 134

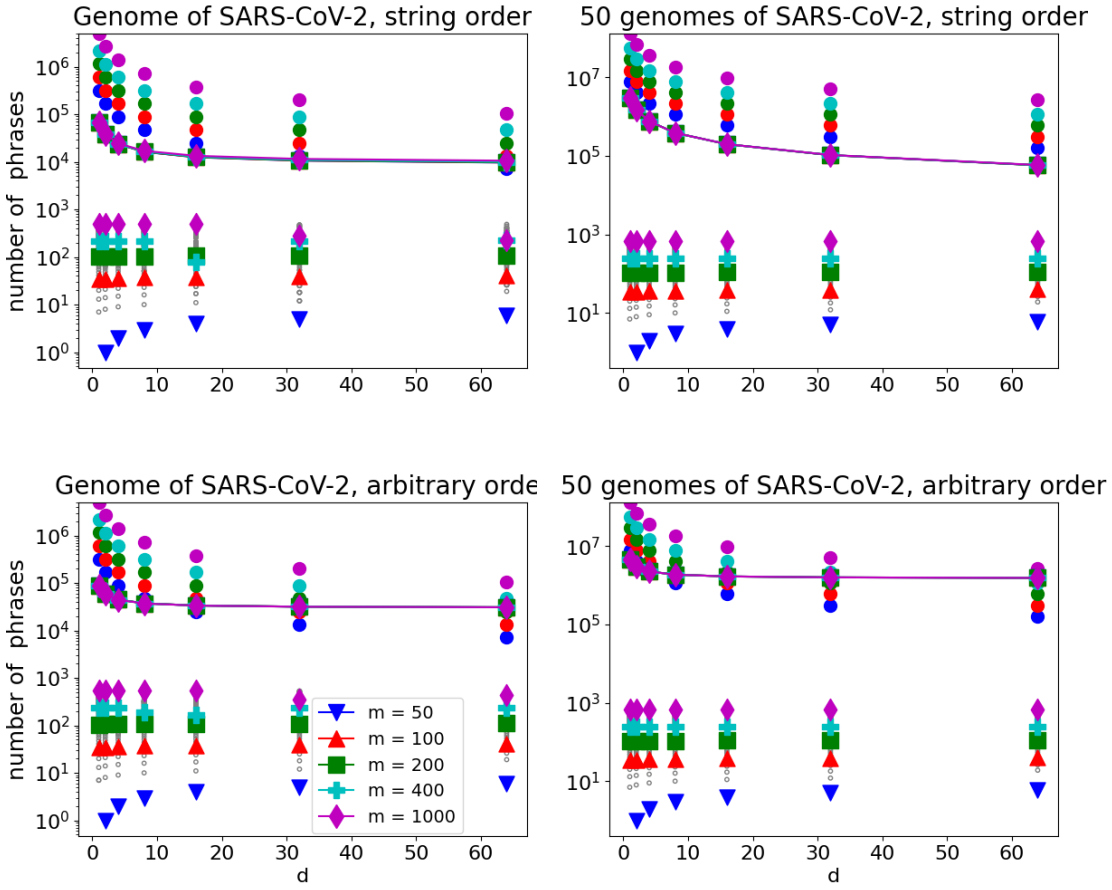

Figure 5: The number of phrases in log scale for concatenation of samples of a single SARS-CoV2 genome (left column) and of a concatenation of 50 SARS-CoV2 genomes (right column), both in string order (on top) and in any arbitrary chosen order (on bottom). The counts are shown for each concatenation of m-length samples at every d position. Colored markers indicate the number of phrases for $m = 50, 100, 200, 400, 1000$, while we use grey points to mark all others m's. The bounds shown in Theorem 2 and 3 for the concatenation in string order respectively random order are shown with colored and shaped lines accordingly to the corresponding m, d pair, while the coloured rounded points indicate the bound given by Kärkkäinen [8], see Theorem 1.

S formed by concatenating the substrings of X of length m starting at positions $i \equiv 1 \pmod{d}$, is bounded by $(2n-m)/d+2z+(m-d)$. We also proved a bound for the case of any arbitrary chosen order of the samples in the concatenation.

There are several avenues future work could take. The most immediate perhaps is the question of whether the bounds given in Theorem 2 and Theorem 3 can be improved. Another direction is to derive bounds for other compression measures, such as the size of the smallest context-free

grammar for S [10], the number of runs in its Burrows-Wheeler transform [11], the size of its smallest string attractor [12], or other measures of compressibility discussed in Navarro's recent survey [13].

Along the lines of our original motivation from DNA sequencing, how are the bounds affected by the introduction of some probability of error to the sample substrings, such as those errors introduced in collections of strings produced by short-read sequencing technologies?

Finally, note that we have assumed $d \leq m/2$, which corresponds to the interesting case in practice. However, it may be interesting from a theoretical perspective to also examine the case where $d > m/2$.

References

[1] R. Ekblom, L. Smeds, H. Ellegren, Patterns of sequencing coverage bias revealed by ultra-deep sequencing of vertebrate mitochondria, BMC Genomics 15 (2014) 467.

[2] S. Al Yami, C.-H. Huang, LFastqC: A lossless non-reference-based FASTQ compressor, PLoS One 14 (2019) e0224806.

[3] S. Chandak, K. Tatwawadi, I. Ochoa, M. Hernaez, T. Weissman, SPRING: a next-generation compressor for FASTQ data, Bioinformatics 35 (2019) 2674–2676.

[4] S. Deorowicz, FQSqueezer: k-mer-based compression of sequencing data, Scientific Reports 10 (2020) 1–9.

[5] C. Hoobin, T. Kind, C. Boucher, S. J. Puglisi, Fast and efficient compression of high-throughput sequencing reads, in: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, 2015, pp. 325–334.

[6] A. Lempel, J. Ziv, On the complexity of finite sequences, IEEE Trans. Inf. Theory 22 (1976) 75–81.

[7] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Trans. Inf. Theory 23 (1977) 337–343.

[8] J. Kärkkäinen, Repetition-Based Text Indexes, Ph.D. thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 1999.

[9] P. W. Harrison, R. Lopez, N. Rahman, S. G. Allen, R. Aslam, N. Buso, C. Cummins, Y. Fathy, E. Felix, et al., The COVID-19 Data Portal: accelerating SARS-CoV-2 and COVID-19 research through rapid open access data sharing, Nucleic Acids Research 49 (2021) W619–W623.

[10] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, A. Shelat, The smallest grammar problem, IEEE Trans. Inf. Theory 51 (2005) 2554–2576.

[11] G. Manzini, An analysis of the Burrows-Wheeler transform, J. ACM 48 (2001) 407–430.

[12] D. Kempa, N. Prezza, At the roots of dictionary compression: string attractors, in: Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC), ACM, 2018, pp. 827–840.

[13] G. Navarro, Indexing highly repetitive string collections, part I: repetitiveness measures, ACM Comput. Surv. 54 (2021) 29:1–29:31.