Tight Bounds for Compressing Substring Samples

Philip Bille
Technical University of Denmark, Lyngby, Denmark

Christian Mikkelsen Fuglsang
Technical University of Denmark, Lyngby, Denmark

Inge Li Gørtz
Technical University of Denmark, Lyngby, Denmark

Abstract

We consider the problem of compressing a set of substrings sampled from a string and analyzing the size of the compression. Given a string S of length n, and integers d and m where $n \geq m \geq 2d > 0$, let $\mathrm{SCS}(S, m, d)$ be the string obtained by sequentially concatenating substrings of length m sampled regularly at intervals of d starting at position 1 in S. We consider the size of the LZ77 parsing of $\mathrm{SCS}(S, m, d)$, in relation to the size of the LZ77 parsing of S. This is motivated by genome sequencing, where the mentioned sampling process is an idealization of the short-read DNA sequencing. We show the following upper bound:

$$|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| \leq |\mathrm{LZ77}(S)| + 2\frac{n-m}{d}.$$

We also give a lower bound showing that this is tight. This improves previous results by Badkobeh et al. [ICTCS 2022], and closes the open problem of whether their bound can be improved.

Another natural question is whether assuming that all letters in S are part of a sample, it is always the case that $|\mathrm{LZ77}(S)| \leq |\mathrm{LZ77}(\mathrm{SCS}(S, m, d))|$. Surprisingly, we show that there is a family of strings such that $|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = |\mathrm{LZ77}(S)| - 1$.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases Compression, Algorithms, Lempel-Ziv

Digital Object Identifier 10.4230/LIPIcs.CPM.2024.9

Funding *Philip Bille*: Supported by the Independent Research Fund Denmark (DFF-9131-00069B and 10.46540/3105-00302B).
Inge Li Gørtz: Supported by the Independent Research Fund Denmark (DFF-9131-00069B and 10.46540/3105-00302B).

1 Introduction

The recent revolution in short-read sequencing technologies has made the acquisition of large genome sequences significantly cheaper and faster. This has led to a drastic increase in the amount of genome data and by extension the need to compress these vast datasets. Numerous ambitious sequencing projects (and existing databases) are currently underway, such as the recent Earth BioGenome Project [30, 31], the 10K Vertebrate Genomes Project [33], and The International Genome Sample Resource (IGSR) [9] built on the foundation of the 1,000 Genomes Project, among many others. These projects aim to create large databases of strings (genomes) that vary only slightly from each other and as a result, contain large repetitions of data.

The vast amount of genome data in these databases makes the importance of compression and fast random access especially apparent. There are several tools that are popular for compressing genome data, some of which are the standard `gzip` and `7zip` compressors. The

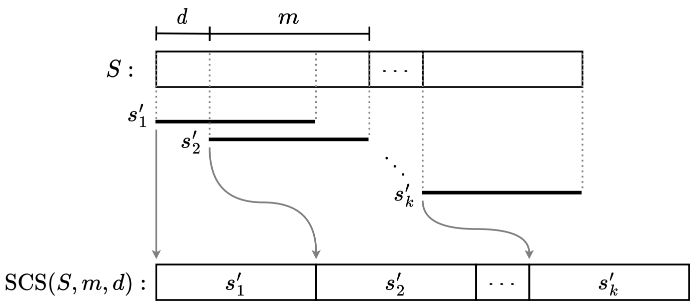

Figure 1 The sampling process for constructing $\mathrm{SCS}(S, m, d)$.

basis of these is Lempel-Ziv (LZ77) parsing [29, 43], typically followed by an entropy encoding. Additionally, there are several read-set specific compressors [1, 5, 7, 16], at least one of which is also a Lempel-Ziv type compressor.

A natural question is to consider the effects the sampling process has on the compressibility of the resulting data. This question was originally posed by Badkobeh et al. [3], and to our knowledge, this is the only instance before us where analysis in this area has been undertaken. In particular, we consider the problem of compressing a set of substrings sampled from a string and analyzing the size of the compression. Specifically, we will analyze the size of the LZ77 parsing (which we define later) as it and variations thereof are widely used in many relevant compressors.

The analysis is achieved by defining an idealized model, which mimics the genome sequencing process. The idealization occurs since we do not consider errors, i.e., insertions, deletions, and substitutions of letters, which are introduced by short-read sequencing technologies. Moreover, we do not consider variations in the distance between or the length of samples across the genome. In practice, these fluctuate for a variety of reasons [8]. However, analysis of these technologies, even in an idealized setting, can give much insight into their effects on compressibility.

We now define the idealized model, as described by Badkobeh et al. [3], which we use throughout the remainder of the paper. Given a string $S = S[1, n] = S[1]S[2]\cdots S[n]$ of length n, and integers d and m where $n \geq m \geq 2d > 0$, let $\mathrm{SCS}(S, m, d)$ be the string obtained by sequentially concatenating substrings of length m sampled regularly at intervals of d starting at position 1 in S. The sampling process and construction of $\mathrm{SCS}(S, m, d)$ is shown in Figure 1.

Formally, we have a total of $k = 1 + \lfloor (n - m)/d \rfloor$ samples, where $\lfloor x \rfloor$ is the largest integer which is smaller than or equal to x. The jth sample s'_j consists of the m consecutive letters in S starting at position $(j - 1)d + 1$. Notice that the last $(n - m \mod d)$ letters in S are not part of any sample due to rounding. The samples are concatenated sequentially to form the string $\mathrm{SCS}(S, m, d) = s'_1 s'_2 \cdots s'_k$.

The connection between this model and genome sequencing is very well described by Badkobeh et al. [3]. In short, $\mathrm{SCS}(S, m, d)$ corresponds to a file of the short-read sequences, which is the typical output of a sequencing experiment (e.g. the FASTQ format). Here, m corresponds to the *read length* and m/d to the *coverage*, i.e., the average number of samples that cover a position in S. The assumption that $m \geq 2d$ corresponds to a coverage of at least 2, which is a relevant case for DNA sequencing.

In this paper, we consider the size of the LZ77 parsing of $\mathrm{SCS}(S, m, d)$, in relation to the size of the LZ77 parsing of the original string S. In the remainder of this section we define the LZ77 parsing of a string, briefly present previous work, and state our results and techniques.

1.1 The LZ77 Parsing

The Lempel-Ziv (LZ77) parsing [29, 43] (also known as the LZ77 factorization) of a string is a fundamental part of data compression [10, 22, 14, 21], and for string processing such as detecting the periodicities of a string [2, 19].

The LZ77 parsing of S partitions S into a sequence of z_S substrings $\mathrm{LZ77}(S) := f_1, f_2, ..., f_{z_S}$ called phrases. The size of the LZ77 parsing of S is $|\mathrm{LZ77}(S)| = z_S$. The phrases are constructed greedily from left to right using the following rules. The ith phrase f_i with starting position p_i is encoded either as (a) the first occurrence of a letter in S, or (b) the longest substring with an occurrence in S before p_i. The compression of S occurs since the phrases of type (b) are encoded as a pair (r_i, l_i), where $r_i > 0$ is the distance from p_i to the beginning of the previous occurrence of f_i, and l_i is the length of f_i. This is the LZ77-variant given by Storer and Szymanski [40], whereas the original definition [43] always added an extra letter to the end of these phrases. Furthermore, we consider the variant of LZ77, where a previous occurrence referenced by a phrase is allowed to overlap with the corresponding phrase.

Naturally, we say that a phrase f_i *covers* an interval $[a, b]$ if and only if every element in $[a, b]$ is contained within the interval $[p_i, p_i + l_i - 1]$, corresponding to the range of f_i in S. Similarly, we say that the phrase *overlaps* the interval when the range contains at least one element in $[a, b]$.

Computing the LZ77 parsing is a very well-studied problem, and there are many algorithms solving it with various trade-offs. The simplest way to construct the LZ77 parsing is to build an index on the input string (e.g. a suffix tree or suffix array), and greedily from left to right find the longest prefix of the current suffix with an occurrence to the left of the current position. There has been lots of previous (and ongoing) research on LZ77 leading to practical and space-efficient computation [6, 13, 15, 23, 24, 18, 26, 20, 34, 37], parallel [38] and external computation [25], online parsing [35, 36, 39, 41], and more [12]. Other practical solutions include the sliding window LZ77 parsing [11, 27, 4], where the previous occurrence of a phrase is restricted to start no more than w letters away from the start position of the phrase, with w as a parameter.

Often these articles include performance metrics obtained experimentally by compressing collections of strings, such as DNA sequences (e.g. [25, 22, 21, 15, 23, 24, 34, 38, 35]), to emphasize the benefits of the corresponding compressor. This demonstrates that an important motivation for improving upon LZ77 factorization is among others to improve the storage of genome databases.

1.2 Previous Work

We will now consider previous results and the techniques that have been used. As mentioned earlier, Badkobeh et al. [3] originally posed the question of the effects of the sampling process on compressibility. They have shown that $|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| \leq m - d + 2|\mathrm{LZ77}(S)| + (2n - m)/d$.

The techniques they employ in their proof involve partitioning $\mathrm{SCS}(S, m, d)$ into several smaller intervals and examining the number of phrases incurred by each individually. More precisely, they consider the intervals given by the first $m - d$ letters followed by the last

d letters in each sample. In the former, they show that the first $m-d$ letters in the first sample trivially incur at most $m-d$ phrases and that the first $m-d$ letters in the remaining samples incur at most one phrase each. The last d letters in each sample are analyzed by defining a rather involved projection of the individual phrases in $\mathrm{LZ77}(S)$ onto $\mathrm{SCS}(S,m,d)$, and showing a bound on how many phrases are incurred by the projected intervals.

They conclude their paper by posing the open question of whether their upper bound can be improved.

1.3 Our Results

In this paper, we answer the question asked by Badkobeh et al. [3] in the affirmative and give tight upper and lower bounds improving upon theirs. More precisely, we show the following upper bound.

▶ **Theorem 1.** *Let S be a string of length n, then for all integers d and m where $n \geq m \geq 2d > 0$:*

$$|\mathrm{LZ77}(\mathrm{SCS}(S,m,d))| \leq |\mathrm{LZ77}(S)| + 2\frac{n-m}{d}\ .$$

Intuitively, the upper bound given in Theorem 1 states that there is no overhead for the first sample and that the remaining samples have an overhead of two phrases each. This bound is strictly better than that given by Badkobeh et al. [3]. In particular, consider what happens when $m = n$, i.e., when we only have a single sample. In this case, their upper bound is $(n - d + 2|\mathrm{LZ77}(S)| + n/d)$, whereas ours is $|\mathrm{LZ77}(S)|$. The latter is tight, since $\mathrm{SCS}(S,n,d) = S$ regardless of the choice of d.

The techniques we use in the proof of Theorem 1 are similar to those used by Badkobeh et al. [3], in the sense that we also partition $\mathrm{SCS}(S,m,d)$ into several smaller intervals which we consider individually. The primary differences are that we tightly analyze the phrases incurred by the entire first sample, and we do not define a projection for analyzing the last d letters of the remaining samples. Instead, we categorize the substrings of phrases incurred by dividing S during the sampling process. We show that these substrings incur at most one phrase each in $\mathrm{LZ77}(\mathrm{SCS}(S,m,d))$, and use this to give a bound on the total number of phrases. This new approach allowed us to significantly improve the upper bound.

Furthermore, we show that our upper bound is tight for any choice of $d \geq 3$.

▶ **Theorem 2.** *Let d and m be integers, where $d \geq 3$ and $m \geq 2d$. Then for all integers $n \geq m$ there exists a string S of length n such that:*

$$|\mathrm{LZ77}(\mathrm{SCS}(S,m,d))| = |\mathrm{LZ77}(S)| + 2\left\lfloor\frac{n-m}{d}\right\rfloor\ .$$

The primary difference between this and the upper bound is the floor after division. This is necessary here since we are referring to an exact number of phrases. We obtain Theorem 2 by constructing a string S of arbitrary length $n \geq m$ based on parameters d and m, and analyzing the compressibility of $\mathrm{SCS}(S,m,d)$ and the constructed string.

Another natural question is whether $|\mathrm{LZ77}(S)| \leq |\mathrm{LZ77}(\mathrm{SCS}(S,m,d))|$ is always the case, assuming that all letters in S are part of a sample, i.e., $n \geq m$ and $n \equiv m \pmod{d}$. The latter assumption is important since it is otherwise trivial to disprove (e.g. let S be m repetitions of a followed by a single b). According to our knowledge, this question has not been considered in detail until now. We show that there exists a family of strings where this is not the case, leading to the following surprising result.

▶ **Theorem 3.** *Let d and m be integers, where $d \geq 2$, $m \geq 2d$, and $m \equiv 0 \pmod{d}$. Then, there exists a string S of length $n \geq m$ where $n \equiv m \pmod{d}$ such that:*

$$|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = |\mathrm{LZ77}(S)| - 1 .$$

We obtain Theorem 3 by constructing a string S of length $n = 3m - d$, and analysing the compressibility after sampling, in a similar fashion to the proof of Theorem 2.

In the following sections, we provide proof of our results. We prove the upper bound in Section 2, the corresponding lower bound showing that this is tight in Section 3, and the theorem on improved compressibility in Section 4. Finally, we finish the paper with concluding remarks and future work in Section 5.

2 Upper Bound

Let S be the given string of length n and let $S' := \mathrm{SCS}(S, m, d)$. We assume w.l.o.g. that every letter in S is part of some sample, i.e., $n \geq m$ and $n \equiv m \pmod{d}$, and partition each sample into two substrings such that $s'_j = u_j s_j$, where $|u_j| = m - d$ and $|s_j| = d$. Thus, $S' = u_1 s_1 u_2 s_2 \cdots u_k s_k$. In order to prove Theorem 1, we partition S' by considering the following three cases separately:

1. the first sample $s'_1 = u_1 s_1$,
2. the first $m - d$ letters u_j in every sample s'_j for $2 \leq j \leq k$, and
3. the last d letters s_j in every sample s'_j for $2 \leq j \leq k$.

This partitions S' into non-overlapping intervals. Similarly, we use this interpretation to write the given string equivalently as $S = u_1 s_1 s_2 \cdots s_k$, and define $z_S := |\mathrm{LZ77}(S)|$ and $z_{S'} := |\mathrm{LZ77}(S')|$ which will be useful when showing these cases.

Any LZ77 parsing contains exactly the same number of phrase starting positions as phrases. Therefore, by bounding the number of starting positions of phrases in the above intervals, we bound the total number of phrases in the LZ77 parsing of S'. As a property of LZ77 there are several substrings which incur at most one starting position of a phrase in the compression of S'. These substrings are categorized in Lemma 4 and Lemma 5, and we use these several times throughout the proof. This strategy is similar to that used by Badkobeh et al. [3], but we show a tighter bound.

▶ **Lemma 4.** *Let T be a string, and P be a substring of T, i.e., $P = T[i, j]$, where $1 \leq i \leq j \leq |T|$. If P has a previous occurrence in T starting at position $i' < i$, then the LZ77 parsing of T contains at most one starting position of a phrase in the interval $[i, j]$.*

Proof. Assume the LZ77 parsing contains more than one starting position in the interval $[i, j]$. Then the first of these phrases has a starting position p and length l, where $i \leq p \leq p+l-1 < j$. Since P has an occurrence in T at position $i' < i$, the substring $T[p, j]$ with length $j-p+1 > l$ also has an occurrence at $i' + p - i < p$. This contradicts the property that every phrase starting at position p covers the *longest* substring with an occurrence in T before p. ◀

▶ **Lemma 5.** *Let T be a string, and $f_1 f_2 \cdots f_{z_T}$ be the phrases in the LZ77 parsing of T. Then for every phrase f_i with starting position p_i which is not the first occurrence of a letter, and for every pair of integers (v, w) where $1 \leq v \leq w \leq l_i$, substring $f_i[v, w]$ has an occurrence in T before $p_i + v$.*

Proof. Consider phrase f_i. This is either the first occurrence of a letter, in which case the lemma trivially holds, or it has an occurrence in T at position $p_i - r_i$. Therefore, the substring $f_i[v, w]$ for any pair (v, w) where $1 \leq v \leq w \leq l_i$ must also have an occurrence in T at position $p_i - r_i + v < p_i + v$. ◀

As mentioned, we partition S' into several intervals. We consider these cases in the following paragraphs, whereafter we collect the results to give the final bounds.

Case 1. Consider the first sample s'_1. By definition, this is the substring sampled by the letters in the interval $[1, m]$ in S. We denote the interval as X and the number of phrases overlapping X in the LZ77 parsing of S as z_X. Intuitively, the LZ77 parsing of S' contains exactly z_X starting positions of phrases in that same interval. This is illustrated in Figure 2.

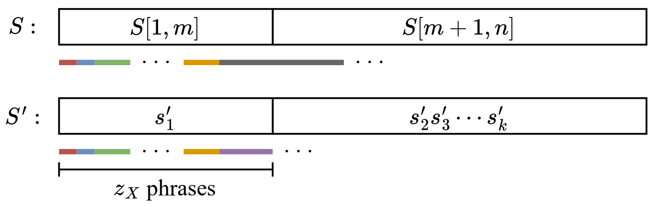

Figure 2 Intuition of the first z_X phrases in LZ77(S') compared to those in LZ77(S).

Formally, we consider the phrases f_i for $1 \le i < z_X$ in LZ77(S). Since f_i only depends on the previous letters and $S[1, p_i + l_i - 1] = S'[1, p_i + l_i - 1]$, this phrase is exactly the same as the ith phrase in LZ77(S'). This is the case for every phrase except f_{z_X}. However, by Lemma 5 the substring $S[p_{z_X}, m] = S'[p_{z_X}, m]$ has a previous occurrence, and by Lemma 4 this incurs at most one starting position of a phrase, and the proof follows.

Case 2. Consider the first $m - d$ letters u_j in samples s'_j for every j, where $2 \le j \le k$. By definition of the jth sample, u_j is a repeat of the last $m - d$ letters in s'_{j-1}. This is illustrated in Figure 3.

Following directly from Lemma 4, these repeating intervals contain at most one starting position of a phrase and therefore contribute at most $k - 1 = (n - m)/d$ to the total number of phrases.

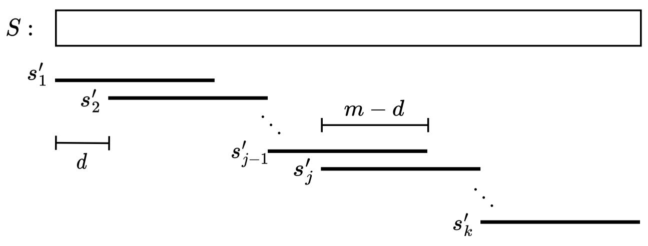

Figure 3 Illustration of the overlap between samples.

Case 3. Finally, consider the last d letters s_j in samples s'_j for every j, where $2 \le j \le k$. The positions of these in S' are illustrated in Figure 4.

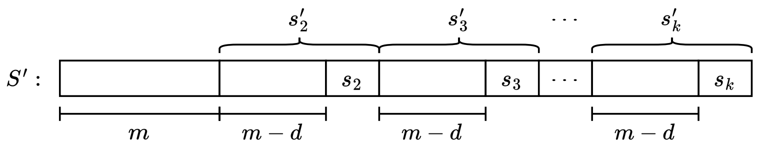

Figure 4 Illustration of the last d letters s_j in sample s'_j, for each j where $2 \le j \le k$.

Consider the phrases in LZ77(S) which encode these substrings, i.e., the phrases overlapping the interval $[m+1, n]$. There are at most $(z_S - z_X + 1)$ such phrases, where z_X is the number of phrases overlapping the interval $[1, m]$. This is trivially shown since phrase intervals are disjoint, and at most one phrase f_{z_X} might overlap both intervals in S. This is also illustrated in Figure 5.

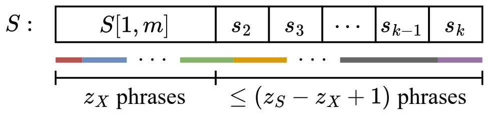

Figure 5 The number of phrases in LZ77(S) overlapping the interval $[m+1, n]$.

We denote these phrases as $f_{z_X}, f_{z_X+1}, ..., f_{z_S}$, and consider their overlap with each sample s_j. In particular, each phrase f_i which overlaps s_j either (i) has zero endpoints in s_j, (ii) ends in s_j, (iii) starts in s_j, or (iv) has both endpoints in s_j.

These cases are illustrated in Figure 6.

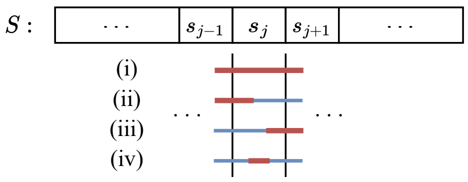

Figure 6 All cases where a phrase overlaps substring s_j. The phrase considered is marked in red.

We partition each phrase f_i into the largest substring for each s_j which does not cross the border from s_{j-1} to s_j, or from s_j to s_{j+1}. I.e., the substring in S given by the intersection between the intervals of f_i and s_j. This ensures that the substrings exist in S'. These substrings are collectively denoted as *phrase parts*, and exactly partition the last d letters of every sample except s'_1. This is illustrated in Figure 7. Notice that by definition the length of each phrase part is at most d.

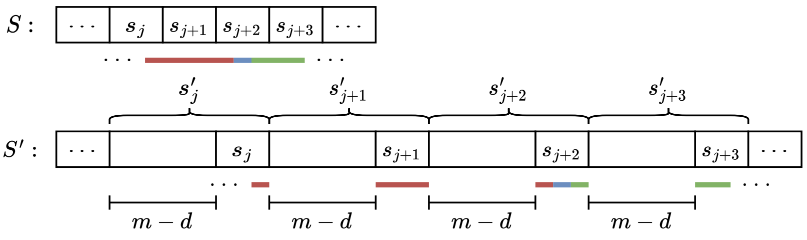

Figure 7 Example of phrase parts constructed from phrases.

▶ **Lemma 6.** *A phrase part incurs at most one phrase starting position in the LZ77 parsing of S'.*

Proof. Consider phrase part P constructed from phrase f_i and substring s_j. The lemma trivially holds if f_i is the first occurrence of a letter. Therefore, we assume w.l.o.g. that f_i has a previous occurrence in S. It then follows directly from Lemma 5 that P also has a previous occurrence in S. We will show that a previous occurrence of P also exists in S' and that by Lemma 4 this incurs at most one phrase starting position. There are three cases that should be considered for the previous occurrence of P in S. It either starts in (a) u_1, (b) some $s_{j'}$ where $j' < j$, or (c) s_j. In case (a) the occurrence must either end in u_1 or s_1, since the length is at most d. This occurrence is therefore contained within s'_1, and must also be present in S'. Similarly, in case (b) the occurrence crosses at most one border from $s_{j'}$ to $s_{j'+1}$ due to the length being at most d. Since the length of each sample is at least $2d$, the sample $s'_{j'+1}$ must end with the substring $s_{j'}s_{j'+1}$, and since $j' + 1 \leq j$ a previous occurrence of P must also exist in S'. Lastly, in case (c), the occurrence must also end in s_j since the phrase part would otherwise overlap with the border from s_j to s_{j+1}. Such a previous occurrence clearly also exists in S'. Thus, a phrase part always has a previous occurrence in S', and the proof follows. ◀

In order to determine a bound on the number of phrase starting positions, we therefore only have to count how many phrase parts are present in S in the interval $[m + 1, n]$.

There can be at most one phrase part of type (i) or (ii) for each substring s_j. This is shown by a simple contradiction. Assume there is more than one phrase part of either type in s_j. This would imply that more than one phrase starts before s_j and crosses the border from s_{j-1} to s_j. However, this contradicts the requirement that phrases do not overlap. Thus, there are at most $k - 1 = (n - m)/d$ such parts. Similarly, we will at most have $(z_S - z_X)$ phrase parts of type (iii) or (iv). There cannot be any more, since phrase f_{z_X} does not have its starting position in the interval $[m + 1, n]$ and a phrase would otherwise need to have two starting positions which is not possible. Therefore, there are at most $(z_S - z_X + (n - m)/d)$ phrase parts partitioning the interval $[m + 1, n]$ in S, and as shown in Lemma 6 these incur at most one phrase each in the LZ77 parsing of S'.

Putting it together

In summary, we have shown that case 1 incurs at most z_X phrases, case 2 at most $(n - m)/d$, and finally case 3 at most $(z_S - z_X + (n - m)/d)$. Therefore:

$$z_{S'} \leq z_X + \frac{n-m}{d} + z_S - z_X + \frac{n-m}{d} = z_S + 2\frac{n-m}{d}.$$

This concludes the proof of Theorem 1.

3 Lower Bound

Given integers $d \geq 3$, $m \geq 2d$, and $k > 0$, we construct the string S of length $m + (k - 1)d$ over the alphabet $\Sigma = \{\lambda_0, \lambda_1, ..., \lambda_k\}$. In particular, we construct S from several substrings such that $S := u_1 s_1 s_2 \cdots s_k$, where $|u_1| = m - d$, and $|s_j| = d$ for all $1 \leq j \leq k$. This is exactly the interpretation used during the proof of Theorem 1. We define u_1 as $m - d$ consecutive repetitions of letter λ_0, such that $u_1 := \lambda_0 \cdots \lambda_0$. Similarly, we define each s_j as a single letter λ_{j-1} followed by $d - 1$ repetitions of letter λ_j, such that $s_j := \lambda_{j-1}\lambda_j \cdots \lambda_j$. This construction is shown in Figure 8:

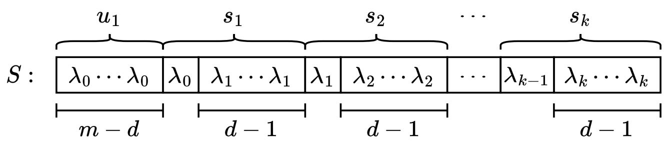
u_1
s_1
s_2
$\cdots$
s_k
$S:$
$\lambda_0 \cdots \lambda_0$
λ_0
$\lambda_1 \cdots \lambda_1$
λ_1
$\lambda_2 \cdots \lambda_2$
$\cdots$
λ_{k-1}
$\lambda_k \cdots \lambda_k$
$m - d$
$d - 1$
$d - 1$
$d - 1$

Figure 8 Construction of S in the proof of Theorem 2.

As seen above $S = \lambda_0^{m-d+1}\lambda_1^d \cdots \lambda_{k-1}^d \lambda_k^{d-1}$. Due to the constraints on m and d, the length of each of the repetitions is at least 2 and therefore requires exactly two phrases to encode. Thus, the total number of phrases in the LZ77 parsing of S is $z_S = 2k + 2$.

We now consider how many phrases are required to encode $S' := \mathrm{SCS}(S, m, d)$. We prove this by induction showing that every sample is encoded by exactly four phrases, and these phrases never cross the border between two samples. We consider the first sample s'_1 in Figure 9.

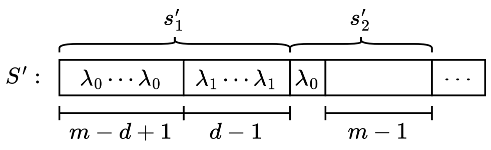

Figure 9 The structure of the first sample in S'.

The first sample s'_1 consists of $m - d + 1$ repetitions of λ_0 followed by $d - 1$ repetitions of λ_1. Since the repetitions have length at least 2, these are encoded by exactly four phrases in total. Therefore, we only have to argue that the last phrase does not overlap s'_2. The first letter in s'_2 is always λ_0 since the start position of this sample is $d + 1$, which is within the first $m - d + 1$ repetitions of λ_0, since $m \geq 2d$. Hence, this is the first time λ_0 follows λ_1, and thus the last phrase in s'_1 does not overlap s'_2.

We now assume the hypothesis for every sample prior to the jth sample, seen in Figure 10.

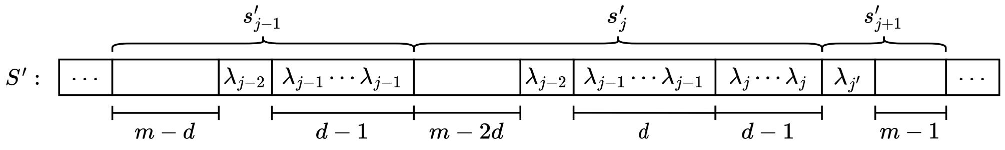

Figure 10 The structure of the jth sample in S'.

By the induction hypothesis, the last phrase encoding s'_{j-1} does not overlap any letters in s'_j. Therefore, the first phrase encoding s'_j starts with the first letter in s'_j. We show that s'_j is encoded by exactly four phrases which also do not cross the border to s'_{j+1}.

(i) The first phrase covers exactly the first $m - d$ letters in s'_j.
(ii) The second phrase has length 1, covering the last occurrence of λ_{j-1} in s'_j.
(iii) The third phrase has length 1, covering the first occurrence of λ_j.
(iv) The fourth phrase covers exactly the last $d - 2$ repetitions of λ_j in s'_j.

As mentioned during the proof of Theorem 1, the first $m - d$ letters u_j in sample s'_j where $j > 1$, is a repeat of the last $m - d$ letters in s'_{j-1}. By the induction hypothesis and Lemma 4, this implies that there can be at most one phrase overlapping this interval. Since every sample has length $m \geq 2d$, the jth sample ends with the substring $s_{j-1}s_j$ as seen in Figure 10. Therefore, if the first phrase covered more than $m - d$ letters, it would also have to cover d repetitions of λ_{j-1} following letter λ_{j-2}. However, this is the first time we encounter d repetitions of λ_{j-1} in a row. This shows (i). The last occurrence of λ_{j-1} is also covered by exactly one phrase since the phrase would otherwise also have to cover the first occurrence of λ_j which is not possible. This shows (ii). It is trivial to show (iii) and (iv), since it is the first occurrence of λ_j. In the latter case, the phrase does not overlap s'_{j+1}, since that sample begins with some $\lambda_{j'}$ where $j' \neq j$.

The induction proof does not consider the last sample, however, in this case, the last phrase clearly ends at the end of s'_k. Therefore, we have shown that each sample is encoded by exactly four phrases. Thus, the total number of phrases in the LZ77 parsing of S' is $z_{S'} = 4k = z_S + 2(k-1)$.

Theorem 2 is therefore shown for some parameter k. However it is also possible to construct a string S of any length $n \geq m$ by letting $k := \lfloor (n-m)/d \rfloor + 1$ and padding $(n - m \mod d)$ repetitions of λ_k to the end of S in the definition stated in Figure 8. This yields exactly the same number of phrases for encoding S, and since the last $(n - m \mod d)$ letters are not part of any sample, S' remains unchanged. Thus, we have shown the equality:

$$|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = |\mathrm{LZ77}(S)| + 2(k-1) = |\mathrm{LZ77}(S)| + 2\left\lfloor \frac{n-m}{d} \right\rfloor.$$

This concludes the proof of Theorem 2.

4 Improved Compressibility

Given integers $d \geq 2$ and $m \geq 2d$, where $m \equiv 0 \pmod d$, we construct a string S of length $n = 3m - d$ over alphabet Σ consisting of exactly two letters a and b, i.e., $\Sigma = \{a, b\}$. Notice that $n \geq m$ and the required property of $n \equiv m \pmod d$ holds, since $m \equiv 0 \pmod d$ implies that $3m - d \equiv m \pmod d$. We construct S by concatenating several repetitions of letters from Σ, as shown in Figure 11. The phrases in the LZ77 parsing of S are shown in Figure 12.

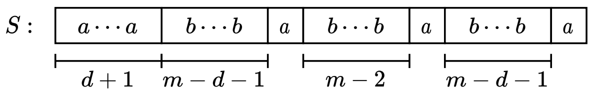

Figure 11 Construction of S in the proof of Theorem 3.

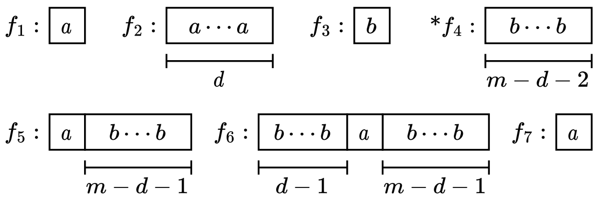

Figure 12 The phrases in the LZ77 parsing of S. Phrase $*f_4$ is only present when $m > d + 2$.

Notably, the first repetition of letter b in Figure 11 has two cases, and requires either one or two phrases to encode depending on whether $m = d + 2$ or $m > d + 2$, respectively. This is only relevant when $d = 2$ and $m = 4$, since we otherwise always fulfill $m > d + 2$. Therefore, the number of phrases in the LZ77 parsing of S following this construction is:

$$|\mathrm{LZ77}(S)| = \begin{cases} 6 & \text{if } d = 2 \text{ and } m = 4, \\ 7 & \text{otherwise.} \end{cases}$$

We now consider the number of phrases in the LZ77 parsing of $S' := \mathrm{SCS}(S, m, d)$. There is a total of $k = \lfloor (3m - d - m)/d \rfloor + 1 = 2m/d$ samples contributing to S'. The first sample s'_1 consists of $d + 1$ repetitions of letter a followed by $m - d - 1$ repetitions of letter b, since these are the first m letters in S. This sample is shown in Figure 13.

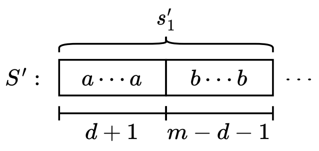

Figure 13 The first sample of S'.

The following $m/d-1$ samples together follow a pattern of a single a followed by $m-d-1$ repetitions of b. This pattern occurs since the jth sample ends with $(j-1)d-1$ repetitions of letter b, and the $(j+1)$th sample starts with $m-jd$ repetitions of letter b, resulting in a total of $m-d-1$ repetitions of b. This repeating pattern is illustrated in Figure 14.

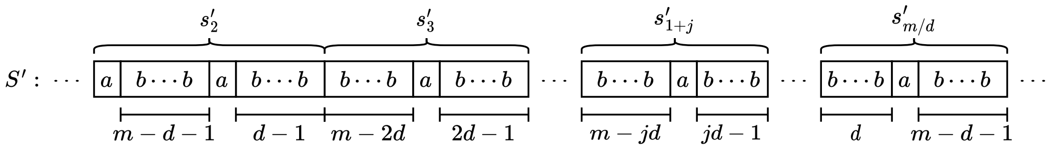

Figure 14 The 2nd to the (m/d)th samples of S'. These form a repeating pattern of a single letter a followed by $m-d-1$ repetitions of letter b.

The $(m/d+1)$th sample is shown in Figure 15. This is the sample starting at position m in S and also starts with the pattern of a single a followed by $m-d-1$ repetitions of b. The number of times this pattern occurs is therefore once for the first sample, m/d times for the following $m/d-1$ samples, and once for the $(m/d+1)$th sample, totaling $m/d+2$ times.

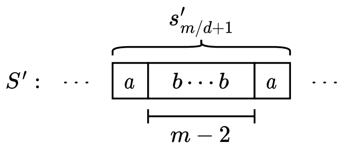

Figure 15 The $(m/d+1)$th samples of S'.

Finally, the remaining $m/d-1$ samples consist of a similar repeating pattern with $m-d-1$ repetitions of b followed by a single a. The pattern in this case is repeated only m/d times, i.e., two times less than previously. This is illustrated in Figure 16.

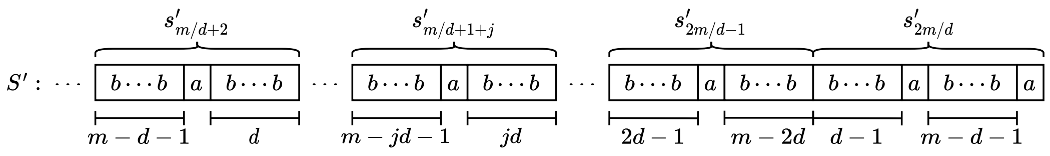

Figure 16 The $(m/d+2)$th to the $(2m/d)$th samples of S'. These form a repeating pattern of $m-d-1$ repetitions of letter b followed by a single letter a.

The complete structure of S' is shown in Figure 17. We have adjusted the second pattern slightly, to make it more similar to the first pattern. The phrases in the LZ77 parsing of S' are shown in Figure 18.

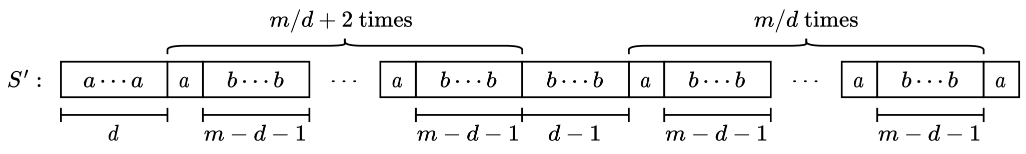

Figure 17 The structure of S' in the proof of Theorem 3.

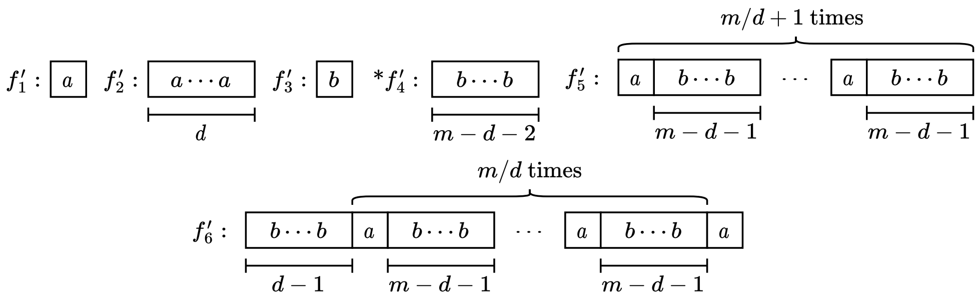

Figure 18 The phrases in the LZ77 parsing of S'. Phrase $*f'_4$ is only present when $m > d + 2$.

Notably, the LZ77 parsing of S' takes advantage of the pattern where a single letter a is followed by a repetition of letter b. This is especially relevant for phrases f'_5 and f'_6. Again, the first repetition of the letter b has the same two cases exactly as described previously, requiring either one or two phrases. Therefore, the size of the LZ77 parsing of S' is:

$$|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = \begin{cases} 5 & \text{if } d = 2 \text{ and } m = 4, \\ 6 & \text{otherwise.} \end{cases}$$

This is always exactly one phrase less than the number of phrases in the LZ77 parsing of S. Thus, we have shown the following equality:

$$|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = |\mathrm{LZ77}(S)| - 1$$

This concludes the proof of Theorem 3.

5 Concluding Remarks

We have considered the problem of compressing a set of substrings sampled from a string and analyzing the size of the compression. We have shown that $|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| \leq |\mathrm{LZ77}(S)| + 2(n - m)/d$ and that this upper bound is tight. Likewise, we have shown that there exists a family of strings where the compressibility after the sampling process improves by exactly one phrase, i.e., where $|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| = |\mathrm{LZ77}(S)| - 1$.

There are several directions that future work could take. A natural question is to derive bounds for other compression algorithms, such as Relative Lempel-Ziv [17, 22], LZ78 [44], Re-Pair [28], or other well-known context-free grammar compressors [32, 42]. As an extension of the original motivation it is also relevant to consider what happens when we change some of the idealizations made to the model (e.g. errors during sampling as introduced by short-read sequencing technologies, or a generalization of sample lengths and/or positions). Finally, an interesting question is whether there exists a string where the size of the LZ77 parsing of the concatenated samples improves by more than one phrase, i.e., is the best you can do the equality in Theorem 3, or does there exist an instance where $|\mathrm{LZ77}(\mathrm{SCS}(S, m, d))| < |\mathrm{LZ77}(S)| - 1$?

References

1 Sultan Al Yami and Chun-Hsi Huang. LFastqC: A lossless non-reference-based FASTQ compressor. *PLoS One*, 14(11):e0224806, 2019.

2 Golnaz Badkobeh, Maxime Crochemore, and Chalita Toopsuwan. Computing the maximal-exponent repeats of an overlap-free string in linear time. In *Proc. SPIRE*, pages 61–72, 2012.

3 Golnaz Badkobeh, Sara Giuliani, Zsuzsanna Lipták, and Simon J. Puglisi. On compressing collections of substring samples. In *Proc. 23rd ICTCS*, pages 136–147, 2022.

4 Philip Bille, Patrick Hagge Cording, Johannes Fischer, and Inge Li Gørtz. Lempel-Ziv compression in a sliding window. In *Proc. 28th CPM*, volume 78, pages 15:1–15:11, 2017.

5 Shubham Chandak, Kedar Tatwawadi, Idoia Ochoa, Mikel Hernaez, and Tsachy Weissman. SPRING: a next-generation compressor for FASTQ data. *Bioinformatics*, 35(15):2674–2676, 2018.

6 Maxime Crochemore, Lucian Ilie, and William F. Smyth. A simple algorithm for computing the Lempel Ziv factorization. In *Proc. DCC*, pages 482–488, 2008.

7 Sebastian Deorowicz. FQSqueezer: k-mer-based compression of sequencing data. *Sci. Rep.*, 10(1):578, 2020.

8 Robert Ekblom, Linnéa Smeds, and Hans Ellegren. Patterns of sequencing coverage bias revealed by ultra-deep sequencing of vertebrate mitochondria. *BMC Genomics*, 15:467, 2014.

9 Susan Fairley, Ernesto Lowy-Gallego, Emily Perry, and Paul Flicek. The International Genome Sample Resource (IGSR) collection of open human genomic variation resources. *Nucleic Acids Res.*, 48(D1):D941–D947, 2019.

10 Paolo Ferragina and Giovanni Manzini. On compressing the textual web. In *Proc. WSDM*, pages 391–400, 2010.

11 Edward R. Fiala and Daniel H. Greene. Data compression with finite windows. *Commun. ACM*, 32(4):490–505, 1989.

12 Johannes Fischer, Travis Gagie, Paweł Gawrychowski, and Tomasz Kociumaka. Approximating LZ77 via small-space multiple-pattern matching. In *Proc. ESA*, pages 533–544, 2015.

13 Johannes Fischer, Tomohiro I, and Dominik Köppl. Lempel Ziv computation in small space (LZ-CISS). In *Proc. CPM*, pages 172–184, 2015.

14 Travis Gagie and Paweł Gawrychowski. Grammar-based compression in a streaming model. In *Proc. LATA*, pages 273–284, 2010.

15 Keisuke Goto and Hideo Bannai. Space efficient linear time Lempel-Ziv factorization for small alphabets. In *Proc. DCC*, pages 163–172, 2014.

16 Christopher Hoobin, Trey Kind, Christina Boucher, and Simon J. Puglisi. Fast and efficient compression of high-throughput sequencing reads. In *Proc. 6th ACM-BCB*, pages 325–334, 2015.

17 Christopher Hoobin, Simon J. Puglisi, and Justin Zobel. Relative Lempel-Ziv factorization for efficient storage and retrieval of web collections. *Proc. VLDB Endow.*, 5(3):265–273, 2011.

18 Dominik Kempa and Simon J. Puglisi. Lempel-Ziv factorization: Simple, fast, practical. In *Proc. ALENEX*, pages 103–112, 2013.

19 Roman Kolpakov and Gregory Kucherov. Finding approximate repetitions under Hamming distance. *Theor. Comput. Sci.*, 303(1):135–156, 2003.

20 Dmitry Kosolobov. Faster lightweight Lempel-Ziv parsing. In *Proc. MFCS*, pages 432–444, 2015.

21 Sebastian Kreft and Gonzalo Navarro. Lz77-like compression with fast random access. In *Proc. DCC*, pages 239–248, 2010.

22 Shanika Kuruppu, Simon J. Puglisi, and Justin Zobel. Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. In *Proc. SPIRE*, pages 201–206, 2010.

23 Juha Kärkkäinen, Dominik Kempa, and Simon J. Puglisi. Lightweight Lempel-Ziv parsing. In *Proc. SEA*, pages 139–150, 2013.

24 Juha Kärkkäinen, Dominik Kempa, and Simon J. Puglisi. Linear time Lempel-Ziv factorization: Simple, fast, small. In *Proc. CPM*, pages 189–200, 2013.

25 Juha Kärkkäinen, Dominik Kempa, and Simon J. Puglisi. Lempel-Ziv parsing in external memory. In *Proc. DCC*, pages 153–162, 2014.

26 Dominik Köppl and Kunihiko Sadakane. Lempel-Ziv computation in compressed space (LZ-CICS). In *Proc. DCC*, pages 3–12, 2016.

27 Niklas Jesper Larsson. Extended application of suffix trees to data compression. In *Proc. DCC*, pages 190–199, 1996.

28 Niklas Jesper Larsson and Alistair Moffat. Off-line dictionary-based compression. *Proc. IEEE*, 88(11):1722–1732, 2000.

29 Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Trans. Inform. Theory*, 22(1):75–81, 1976.

30 Harris A. Lewin et al. Earth BioGenome Project: Sequencing life for the future of life. *Proc. Natl. Acad. Sci. U.S.A*, 115(17):4325–4333, 2018.

31 Harris A. Lewin et al. The earth biogenome project 2020: Starting the clock. *Proc. Natl. Acad. Sci. U.S.A*, 119(4), 2022.

32 Craig G. Nevill-Manning and Ian H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *JAIR*, 7:67–82, 1997.

33 Genome 10K Community of Scientists. Genome 10K: A Proposal to Obtain Whole-Genome Sequence for 10 000 Vertebrate Species. *J. Hered.*, 100(6):659–674, 2009.

34 Enno Ohlebusch and Simon Gog. Lempel-Ziv factorization revisited. In *Proc. CPM*, pages 15–26, 2011.

35 Daisuke Okanohara and Kunihiko Sadakane. An online algorithm for finding the longest previous factors. In *Proc. ESA*, pages 696–707, 2008.

36 Alberto Policriti and Nicola Prezza. Fast online Lempel-Ziv factorization in compressed space. In *Proc. SPIRE*, pages 13–20, 2015.

37 Alberto Policriti and Nicola Prezza. Computing LZ77 in run-compressed space. In *Proc. DCC*, pages 23–32, 2016.

38 Julian Shun and Fuyao Zhao. Practical parallel Lempel-Ziv factorization. In *Proc. DCC*, pages 123–132, 2013.

39 Tatiana Starikovskaya. Computing lempel-ziv factorization online. In *Proc. MFCS*, pages 789–799, 2012.

40 James Andrew Storer and Thomas Gregory Szymanski. Data compression via textual substitution. *J. ACM*, 29(4):928–951, 1982.

41 Jun'ichi Yamamoto, Tomohiro I, Hideo Bannai, Shunsuke Inenaga, and Masayuki Takeda. Faster compact on-line Lempel-Ziv factorization. In *Proc. 31st STACS*, volume 25, pages 675–686, 2014.

42 En-Hui Yang and John C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform — Part one: Without context models. *IEEE Trans. Inform. Theory*, 46(3):755–777, 2000.

43 Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory*, 23(3):337–343, 1977.

44 Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory*, 24(5):530–536, 1978.