Comparison of LZ77-type Parsings

Dmitry Kosolobov\textsuperscript{a}, Arseny M. Shur\textsuperscript{b}

\textsuperscript{a}University of Helsinki, Helsinki, Finland
\textsuperscript{b}Ural Federal University, Ekaterinburg, Russia

Abstract

We investigate the relations between different variants of the LZ77 parsing existing in the literature. All of them are defined as greedily constructed parsings encoding each phrase by reference to a string occurring earlier in the input. They differ by the phrase encodings: encoded by pairs (length + position of an earlier occurrence) or by triples (length + position of an earlier occurrence + the letter following the earlier occurring part); and they differ by allowing or not allowing overlaps between the phrase and its earlier occurrence.

For a given string of length $n$ over an alphabet of size $\sigma$, denote the numbers of phrases in the parsings allowing (resp., not allowing) overlaps by $z$ (resp., $\hat{z}$), for “pairs”, and by $z_3$ (resp., $\hat{z}_3$), for “triples”. We prove the following bounds and provide series of examples showing that these bounds are tight:

- $z \leq \hat{z} \leq z \cdot O(\log \frac{\log \sigma}{\log z})$ and $z_3 \leq \hat{z}_3 \leq z_3 \cdot O(\log \frac{\log \sigma}{\log z_3})$;
- $\frac{1}{2} \hat{z} < \hat{z}_3 \leq \hat{z}$ and $\frac{1}{2} z < z_3 \leq z$.

Keywords: LZ77, lossless data compression, greedy parsing, non-overlapping phrases

1. Introduction

The Lempel–Ziv parsing [16] (LZ77 for short) is one of the central techniques in the data compression and it plays an important role in stringology and algorithms in general. The literature on LZ77 is full of different variations of the parsing originally described by Lempel and Ziv [16] (curiously, the most popular modern LZ77 modifications differ from the original one\textsuperscript{1}). Some of these LZ77-based parsings lie at the heart of common compressors such as \texttt{gzip}, \texttt{7-zip}, \texttt{pkzip}, \texttt{rar}, etc. and some serve as a basis for compressed indexes on highly repetitive data (e.g., see [4, 10, 11]).

Most LZ77 variations have a noticeable optimality property: they have the least number of phrases among all reference-based parsings with the same fixed-length coding scheme for phrases (for details, see [1, 12] or Lemma 2 below). The analysis in [1] shows that many other popular reference-based methods (including LZ78 [17]) are significantly worse than LZ77 in the worst case. Probably, because of this well-known kind of LZ77 good behavior, many authors often implicitly consider different LZ77 variations as somehow equivalent in terms of the number of produced phrases. Despite the fact that numerous works have been published in the last 40 years on this topic (e.g., see [13] and references therein), to our knowledge, until very recently (see [5]), there were no theoretical comparative studies of this side of LZ77 modifications. We partially close this gap establishing tight bounds on the ratios between the numbers of phrases in several popular LZ77 variations. Note that the comparison of the parsings in terms of the bit size of their variable-length encodings is a different and, as it seems, more challenging problem (see [3, 9]).

We investigate the relations between the most popular variants of the LZ77 parsing that one might find in the existing literature on the subject. All of them are defined as greedily constructed parsings that encode each phrase by reference to a string occurring earlier in the input, but they differ by the format of the phrase encodings and by the constraints imposed on earlier phrase occurrences. We primarily investigate four LZ77 variants that, at a generic step of the left-to-right greedy construction, define the phrase $f$ starting at the current position $i$ as follows:

---

\textsuperscript{1}The original parsing is the LZ\textsubscript{3} parsing defined below.
Theorem 1. For any given string of length $n$ over an alphabet of size $\sigma$, one has \[ z \leq \hat{z} \leq \frac{1}{3} \log n \] 

Theorem 2. For any integers $n > 1$, $\sigma \in [2..n]$, $z \in [\sigma, \log_{\sigma} n]$, there is a string of length $n$ over an alphabet of size $\sigma$ such that the sizes of its LZ and LZ$_3$ parsings are, respectively, $\Theta(z)$ and $\Omega(z \log \frac{n}{\log_{\sigma} n})$. The same result holds for the novel LZ and novel LZ$_3$ parsings.

Note that while the necessity of the condition $z \leq \frac{n}{\log_{\sigma} n}$ is justified by the well-known fact that the size of the LZ/LZ$_3$ parsing of any string of length $n$ over an alphabet of size $\sigma$ is at most $O\left(\frac{n}{\log_{\sigma} n}\right)$ (e.g., see [9]).

Theorems 1 and 2 are the main results of this paper. To complete the picture, we also investigate the relations between the numbers $z, z_3$ and, respectively, $\hat{z}, \hat{z}_3$, proving simple bounds and their tightness in the following theorem.

Theorem 3. For any given string, one has $z < z_3 < \hat{z} < \hat{z}_3$. Moreover, for each $k \geq 1$ and each of the four restrictions $z_3 = z = k$ and $z = 2k - 1$; $\hat{z}_3 = \hat{z} = k$ and $\hat{z} = 2k - 1$ there is a binary string satisfying this restriction.

It is known that a random string of length $n$ has $\Theta(n/\log_{\sigma} n)$ phrases in its Lempel–Ziv parsings (e.g., see [16]). A “reasonably compressible” string has, say, $\Omega(n/\log O(1))$ phrases. For these strings our theorems imply that the sizes of all four considered LZ77 parsings are within $O(\log \log n)$ factor from each other; thus, we partially support the intuition that all these LZ77 variations are similar.

As a side note, we also relate the parsings to their versions in which the length of each phrase is artificially restricted to a given integer $\ell$, and obtain the (not surprising) bounds like $z_{\ell} - \frac{\ell}{2} \leq z \leq z_{\ell}$

The paper is organized as follows. In Section 2 we formalize the definitions of the LZ77 parsings under consideration and introduce some useful tools. In Section 3, the proofs of the main results (Theorems 1 and 2) are given. The remaining results are proved in Section 4. We conclude with some remarks and open problems in Section 5.

2. Preliminaries

A string $s$ of length $n$ over an alphabet $\Sigma$ is a map $\{1, 2, \ldots, n\} \rightarrow \Sigma$, where $n$ is referred to as the length of $s$, denoted by $|s|$. We write $s[i]$ for the $i$th letter of $s$ and $s[i..j]$ for $s[i]s[i+1] \cdots s[j]$. A string $u$ is a substring of $s$ if $u = s[i..j]$ for some $i$ and $j$; the pair $(i, j)$ is not necessarily unique and we say that $i$ specifies an occurrence of $u$ in $s$. A substring $s[1..j]$ (resp., $s[i..n]$) is a prefix (resp. suffix) of $s$. For any $i, j$, the set $\{k \in \mathbb{Z} : i \leq k \leq j\}$ (possibly empty) is denoted by $[i..j]$. A decomposition of a string is its representation as the concatenation of nonempty substrings; writing a decomposition, we separate these substrings by dots. Two strings $u$ and $v$ are called conjugate if $u = xy$ and $v = yx$ for some $x$ and $y$. An integer $p \in [1..|s|]$ is called a period of $s$ if $s[i] = s[i+p]$ for any $i \in [1..|s|−p]$. The following lemma is obvious.
Lemma 1. Suppose that, in a string s, we have \( w = s[i..j] = s[i'..j'] \) and \( i < i' \leq j \); then \( i' - i \) is a period of \( w \).

For a given string \( s \), the \( \text{LZ} \) (resp., \( \text{novLZ} \)) parsing of \( s \) is the decomposition \( s = f_1f_2 \cdots f_r \) built from left to right by the following greedy procedure: if a prefix \( s[1..i-1] = f_1f_2 \cdots f_{p-1} \) is already processed, then the string \( f_p \) (which is called a phrase) is either the letter \( s[i] \) that does not occur in \( s[1..i-1] \) or is the longest string that starts at position \( i \) and has an occurrence at position \( j < i \) (resp., \( j \leq i - |f_p| \)). The \( \text{LZ} \) (resp., \( \text{novLZ} \)) parsing is constructed by an analogous greedy procedure but the phrase \( f_p \) is chosen as the longest string occurring at position \( i \) such that the string \( f_p[1..|f_p|-1] \) has an occurrence at position \( j < i \) (resp., \( j \leq i - |f_p| + 1 \)).

Consider \( s = abababc \). The \( \text{LZ} \), \( \text{novLZ} \), \( \text{LZ}_3 \), and \( \text{novLZ}_3 \) parsings of \( s \) are, respectively, \( a, b, b, a, b, abc \), \( a, b, ab, b, c \), \( a, b, ab, b, c \), and \( a, b, aba, bc \).

Let \( s = t_1t_2 \cdots t_r \) be a decomposition of \( s \) into non-empty strings \( t_1, \ldots, t_r \). We say that \( t_1t_2 \cdots t_r \) is an \( \text{LZ} \)-type (resp., \( \text{novLZ} \)-type) parsing if for each \( i \in [1..r] \), the string \( t_i \) either is a letter or has an occurrence in the string \( s[1..|t_1 \cdots t_{i-1}|] \) (resp., in \( t_1t_2 \cdots t_{i-1} \)). Analogously, we say that \( t_1t_2 \cdots t_r \) is an \( \text{LZ}_3 \)-type (resp., \( \text{novLZ}_3 \)-type) parsing if for each \( i \in [1..r] \), the string \( t_i[1..|t_i|] \) has an occurrence in the string \( s[1..|t_1 \cdots t_{i-1}|-2] \) (resp., in \( t_1t_2 \cdots t_{i-1} \)).

The number of phrases in a parsing is called the size of the parsing. We write \( z \) (resp., \( z, z_3, \hat{z}_3 \)) to denote the size of the \( \text{LZ} \) (resp., \( \text{novLZ} \), \( \text{LZ}_3 \), \( \text{novLZ}_3 \)) parsing of a given string.

Our main tool in the subsequent analysis is the following well-known optimality lemma (e.g., see [1, 12, 14]). We omit the proof as it is straightforward.

Lemma 2. For any given string, the size of its \( \text{LZ} \) (resp., \( \text{novLZ} \), \( \text{LZ}_3 \), \( \text{novLZ}_3 \)) parsing is less than or equal to the size of any \( \text{LZ} \)-type (resp., \( \text{novLZ} \)-type, \( \text{LZ}_3 \)-type, \( \text{novLZ}_3 \)-type) parsing.

3. Relations Between Overlapping and Non-overlapping Parsings

For the proof of Theorem 1, we need the following technical lemma.

Lemma 3. Consider a set of positive numbers \( \{t_1, \ldots, t_r\} \) such that \( t_1 + t_2 + \cdots + t_r \leq n \) for some \( n > 0 \). Then, for any given \( k > 0 \), we have \( \sum_{i=1}^r \log \frac{1}{t_i} \leq r \log \frac{n}{k} \).

Proof. Denote \( \alpha_i = \frac{1}{t_i} \). Note that \( \alpha_1 + \cdots + \alpha_r \leq \frac{n}{k} \).

A well-known corollary of the concavity of the logarithmic curve is that the sum \( \sum_{i=1}^r \log \alpha_i \) is maximized whenever all \( \alpha_i \) are equal and maximal under the condition \( \sum_{i=1}^r \alpha_i \leq \frac{n}{k} \), i.e., \( \alpha_i = \frac{n}{r k} \) for all \( i \in [1..r] \). Hence, the result follows.

Proof of Theorem 1. Let us consider the case of \( z \) and \( \hat{z} \); the proof for \( z_3 \) and \( \hat{z}_3 \) can be reconstructed by analogy.

Let \( s \) be a string of length \( n \) over an alphabet of size \( \sigma \). Since the \( \text{novLZ} \) parsing of \( s \) is an \( \text{LZ} \)-type parsing, \( z \leq \hat{z} \) by Lemma 2. Hence, it suffices to prove that \( \hat{z} \leq z \cdot O(\log \frac{n}{\log \sigma}) \). The idea of the proof is to use the \( \text{LZ} \) parsing \( f_1f_2 \cdots f_r \) of \( s \) to construct a \( \text{novLZ} \)-type parsing of size \( z \cdot O(\log \frac{n}{\log \sigma}) \); then, the required bound follows from Lemma 2.

We construct a new parsing for \( s \) substituting each phrase \( f_i \) with a set of new phrases. If a phrase \( f_i \) has an occurrence in the string \( f_1 \cdots f_{i-1} \), then we do not alter \( f_i \) and include it in the new parsing. Consider a phrase \( f_i \) such that the leftmost occurrence of \( f_i \) in the string \( f_1 \cdots f_{i-1} \) occurs at position \( j \) such that \( |f_1 \cdots f_{i-1} - f_i| + 1 < j \) (i.e., this occurrence of \( f_i \) overlaps with \( f_i \)). Let us choose an arbitrary constant \( \alpha \in (0, 1) \). Denote \( k = \alpha \log_{\sigma} z \). We first discuss how to process the case \( j \leq |f_1 \cdots f_{i-1} - k \) (i.e., when the leftmost occurrence of \( f_i \) is farther than \( k \) letters from \( f_i \)).

By Lemma 1, \( p = |f_1 \cdots f_{i-1} + 1 - j \) is a period of \( f_i \) and \( p \in [k..|f_i|] \). We decompose \( f_i \) as follows: \( f_i = t_1 \cdots t_r \), where \( |t_1| = 2^p \), \( |t_2| = 2^{p-1} \), \( |t_{i-1}| = 2^{p-2} \), and \( t_r \) is a non-empty suffix of \( f_i \) of length \( \leq 2^{p-1} \). Since \( p \) is a period of \( f_i \) and the substring of length \( p \) preceding the phrase \( f_i \) is equal to \( f_i[1..p] \), any string \( t_h \) from the decomposition occurs at \( 2^{h-1} \) positions to the left and, since \( |t_h| = 2^{h-1} \), this occurrence does not overlap \( t_h \). Therefore, we can include the strings \( t_1, \ldots, t_r \) from the decomposition \( f_i = t_1 \cdots t_r \) as phrases in the \( \text{novLZ} \)-type parsing under construction. It is easy to see that \( r = O(\log \frac{|f_i|}{p}) \). Since \( p \geq k \), we obtain \( r = O(\log \frac{|f_i|}{k}) \). Hence, it follows from Lemma 3 that the number of new phrases introduced by all such decompositions is upper bounded by \( O(\sum_{i=1}^r \log \frac{|f_i|}{k}) \leq z \cdot O(\log \frac{n}{k}) \leq z \cdot O(\log \frac{n}{\log \sigma}) \), exactly as required.

Now we process each phrase \( f_i \) whose leftmost occurrence is at position \( j_i > |f_i \cdots f_{i-1} - k \) and overlaps \( f_i \). Denote by \( M \) the set of all such phrases having length greater than \( k \). We first
consider the phrases from \( M \). For each phrase \( f_i \) from \( M \), denote by \( p_i = |f_1 \cdot \cdots \cdot f_{i-1}| + 1 - j_i \), its period induced by the leftmost overlapping occurrence. Let us partition the set \( M \) into the minimal number of buckets \( M_1, \ldots, M_s \) such that, for any two phrases \( f_i \) and \( f_j \) from the same bucket, we have \( f_i[1..p_i] \neq f_j[1..p_j] \) (i.e., their “roots”, which are the prefixes of length equal to their periods, coincide). The crucial observation is that there are only at most \( k \alpha^k \) distinct strings of length at most \( k \) in \( s \) and, therefore, the number \( q \) of buckets is small compared to \( z \).

We decompose the leftmost phrase \( f_i \) of each bucket as \( f_i = t_1 t_2 \cdots t_r \), where \( |t_0| = 2^0 p_i, |t_2| = 2^1 p_i, \ldots, |t_{r-1}| = 2^{r-2} p_i \), and \( t_r \) is a non-empty suffix of \( f_i \) of length \( \leq 2^{r-1} p_i \); as in the above analysis, it is easy to show that each substring \( t_h \) from the decomposition has a non-overlapping left occurrence. We include all substrings \( t_h \) from this decomposition in the parsing under construction and, thus, obtain \( O(\log |f_i|) \) new phrases. The number of phrases introduced by each bucket in this way is upper bounded by \( O(q \log n) \). For \( z \geq \sqrt{n} \), we obtain \( q \log n \leq z^\alpha \log^{\Theta(1)} n \), which is \( o(z) \) because \( \alpha < 1 \). For \( z < \sqrt{n} \), we obtain \( q \leq \frac{n}{z \log z} = n \cdot O(\log \sigma) \); hence, \( q \log n \leq z \log n = z \cdot O(\log \sigma) \).

Now consider a phrase \( f_i \) from \( M \) that was not decomposed yet. Let \( f' \) be the leftmost phrase from the same bucket as \( f_i \) (clearly, \( f' \) and \( f_i \) are different phrases). Denote by \( c \) the maximal integer such that \( c p_i \leq k \) (note that \( c p_i \geq \frac{k}{2} \)). We decompose \( f_i \) as \( f_i = t_0 \cdots t_r \), where \( |t_0| = c p_i, |t_1| = 2^0 c p_i, |t_2| = 2^1 c p_i, \ldots, |t_{r-1}| = 2^{r-2} c p_i \), and \( t_r \) is a non-empty suffix of \( f_i \) of length \( \leq 2^{r-1} c p_i \). It is easy to see that \( t_0 \) is a prefix of \( f' \) since \( |f'| > k \), \( f' \) and \( f_i \) both have the same period \( p_i \), and \( f'[1..p_i] = f_i[1..p_i] \). Further, for each \( h \in [1, r] \), \( t_h \) is a prefix of \( f_i \); and this prefix does not overlap with the substring \( t_h \) from the decomposition \( f_i = t_0 \cdots t_r \). Thus, all the substrings \( t_0, \ldots, t_r \) have earlier non-overlapping occurrences. We include \( t_0, \ldots, t_r \) in our new parsing. Since \( c p_i \geq \frac{k}{2} \), we obtain \( r = O(\log \frac{p_i}{c}) = O(\log \frac{p_i}{k}) \). It follows from Lemma 3 that the number \( \sigma \) of phrases introduced in this way is upper bounded by \( O(\sum_{i=1}^s \log \frac{p_i}{k}) \leq z \cdot O(\log \frac{p_i}{k}) \leq z \cdot O(\log \frac{n}{z \log z}) \). It remains to consider phrases of length at most \( k \).

Since there are \( O(\sigma^k) \) distinct strings of length at most \( k \) in \( s \), there are \( O(\sigma^k) \) phrases of length at most \( k \) whose leftmost occurrences overlap them. We decompose each such “short” self-overlapping phrase into one-letter phrases and, thus, obtain at most \( O(\sigma^k) \) new phrases, which is \( o(z) \), i.e., negligible compared to \( z \).

Finally, summing the numbers of all introduced phrases in the constructed newLZ-type parsing, we obtain \( z \cdot O(\log \frac{n}{z \log z}) \) phrases.

The lower bound \( z \) for \( z \) (respectively, \( z_3 \) for \( z_3 \)) is obviously tight since the overlapping and non-overlapping parsings coincide for any string having no overlaps, and such overlap-free strings of any length exist for any non-unity alphabet (see [15]). Theorem 2 proves the tightness of the upper bound \( z \cdot O(\frac{n}{z \log z}) \) for \( z \) (and of the respective upper bound \( z_3 \cdot O(\frac{n}{z_3 \log z_3}) \) for \( z_3 \)).

Proof of Theorem 2. We describe examples for LZ. The examples for LZ_4 are exactly the same and the analysis is analogous, so we omit the details.

The example for an unlimited alphabet is easy (for simplicity, we assume here that \( n \) is a multiple of \( \sigma \)): the string \( a_1^{n/\sigma} \cdot a_2^{n/\sigma} \cdots a_{\sigma}^{n/\sigma} \), where \( a_1, \ldots, a_{\sigma} \) are distinct letters, satisfies \( z = 2 \sigma \) and \( \hat{z} = \Omega(\sigma \log \frac{n}{\sigma}) = \Omega(\log \frac{n}{z \log z}) \).

We generalize this simple example for alphabets of restricted size \( \sigma \) replacing each letter \( a_i \) with a string of length \( \Theta(\log \sigma) \). Let us describe the strings that serve as replacements. Denote \( d = \lceil \log_{\sigma} z \rceil \). In [2] it was shown that all \( \sigma^d \) possible strings of length \( d \) over an alphabet of size \( \sigma \) can be arranged in a sequence \( v_1, v_2, \ldots, v_{\sigma^d} \) (called a \( \sigma \)-ary Gray code [2, 6]) such that, for any \( i \in [2, \sigma^d] \), the strings \( v_{i-1} \) and \( v_i \) differ in exactly one position. Moreover, we can choose such sequence so that \( v_{1} = b^d \), where \( b \) is an arbitrarily chosen letter from the alphabet. The strings \( u_i = ab^{d-1}v_i \), where \( a \) is a letter that differs from \( b \), serve as the replacements for \( a_i \). The important property of \( u_i \) is that no two distinct strings \( u_i \) and \( u_j \) are conjugates; this follows from the observation that conjugates must contain two occurrences of \( ab^{d-1} \), while the only string \( u_i \) with this property is \( (ab^{d-1})^2 \).

Suppose that \( z \leq 8 \). Since \( \Omega(\sigma \log \frac{n}{z \log z}) = \Omega(\log n) \) in this case (note that \( \sigma \leq z \leq 8 \)), the statement of the theorem can be easily proved using strings like \( a^n \). Now suppose that \( z > 8 \). Let \( k \) be the maximal integer such that \( k \leq z/8 \). Observe that \( k \geq 1 \) and \( \sigma^d = \sigma^{\lceil \log_{\sigma} z \rceil} \geq z > k \). Our example is the following string:

\[
\sigma = u_1^{\lceil \sigma z \rceil} u_2^{\lceil \sigma z \rceil} \cdots u_k^{\lceil \sigma z \rceil},
\]
which consists of $k$ “blocks” $u^1_1 \cdots u^1_{k-2} u^1_{k-1}$. Since $|u_1| = 2d$, the length of $s$ is $\frac{z}{\log_2 n} 2kd \leq n$. We append a number of letters $a$ to the end of $s$ to make the length equal to $n$; such modification does not affect the proof that follows, so, without loss of generality, we assume that $|s| = n$.

Since $z \leq \frac{1}{\log_2 n}$ and $k \leq z/8$, we have $kd \leq (z/8)[\log_2 n] \leq \frac{n}{8 \log_2 n} [\log_2 n] \leq n/4$. Therefore, $[\frac{n}{\log_2 n}] \geq 2$, i.e., each block $u^1_1 \cdots u^1_{k-1}$ consists of at least two copies of $u_1$.

The string $s$ has an LZ-type parsing with at most 4 phrases per block: $a.b.a.b.a.b.a.ab.a.b.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a$. $u_1' = u_2' \cdots u_{k-1}' \cdots u_k'$ is inside $u_1 (\text{resp., } u_k')$ is the shortest common prefix (resp., suffix) of $u_1$ and $u_{k-1}$, and $c_1$ is a character. This shows that the size of the LZ parsing of $s$ is at most $4k \leq \frac{n}{2}$. On the other hand, let us demonstrate that the size of the non-overlapping LZ parsing of $s$ is at least $k \log_2 \left(\frac{n}{\log_2 n}\right)$. Consider, for $i > 1$, the leftmost occurrence of a conjugate of $u_i$ in $s$. It is inside $u_i^j$ or $u_{i-1}^j u_j$ for some $j \leq i$. In the first case, the occurrence is a conjugate of $u_j$, implying $j = i$. In the second case, it is a conjugate of either $u_{i-1}$ or $u_j$ (since $u_{i-1}$ and $u_j$ differ in exactly one position and have the same length 2d). So again $j = i$. This means that the first $u_i$ in the $r$th block contains at least one border between phrases of the novLZ parsing, and such a phrase containing the suffix of this $u_i$ has length at most 2d (if it has length 2d, then it is a conjugate of $u_i$ occurring exactly 2d symbols to the right of its reference occurrence). The next phrase has length at most 4d, then 8d, and so on until the phrase border inside or immediately before the first occurrence of $u_{i+1}$. Thus, we have proved that at least $\log_2 \left(\frac{n}{\log_2 n}\right)$ phrases are needed for each of the $k$ blocks, as required. Therefore, since $\frac{n}{\log_2 n} - 1 < k \leq \frac{n}{2}$, we obtain $\hat{z} = \Omega \left(\frac{n}{\log_2 n}\right)$. Thus, the upper bound for $\hat{z}$ is reached on the string $s$.

4. Further Results

Now we prove Theorem 3.

Proof of Theorem 3. Let us consider $z$ and $z_3$; the analysis of $\hat{z}$ and $\hat{z}_3$ is the same. Let $f_1 f_2 \cdots f_{z_3}$ be the LZ$_3$ parsing of a string $s$. It is immediate from the definitions that $f_1 t_1 f_1' \cdots t_{z_3} f_{z_3}'$, where $t_i = f_i[1..|f_i| - 1]$, and $t'_i = f_i[|f_i|]$, is an LZ$_3$-type parsing of $s$ of size at most $2z_3 - 1$ (we remove empty strings $t_i$ from the parsing). Hence $z < 2z_3$ by Lemma 2.

Further, the LZ$_3$ parsing of $s$ is an LZ$_3$-type parsing of $s$ by definition. Therefore, again by Lemma 2, we obtain $z_3 \leq z$.

Let us show the tightness of the bounds. Let $k \geq 2$ (the case $k = 1$ is trivial). The restriction $z = 2k - 1$, $z_3 = k$ is satisfied by the string

$$aababa \cdots aab^{2k-1}aabb^{2k-2},$$

whose LZ and LZ$_3$ parsings are, respectively,

$$a.a.b.a.b.a.b.a.b.a.a.b^{2k-2}b, a.a.b.a.b.b.a.a.b^{2k-2}b,$$

Next, the equalities $z = z_3 = 2k$ hold for the string

$$abab^{4}abab^{10} \cdots abab^{2k-1}2^{-1},$$

having the following LZ and LZ$_3$ parsings:

$$a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.b.a.$$

Note that if we delete the last 3 $2k-1$ b’s, both parsings of the resulting string will have size $2k - 1$. Therefore, the equality $z = z_3 = k$ can be achieved for any $k$. Further, the string

$$a^2b^2ab^3a^1b^7 \cdots a^{32k-1}b^{2k-1}a^{32k-1}a^2b^2$$

satisfies $\hat{z} = 4k + 3$, $z_3 = 2k + 2$ as its corresponding novLZ and novLZ$_3$ parsings look as follows:

$$a.a.b.a.a.b.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.a.a.a.b.$$

If we delete the last phrase of the novLZ$_3$ parsing, the resulting string will satisfy $\hat{z} = 4k + 1$, $z_3 = 2k + 1$. Therefore, the condition $\hat{z} = 2k - 1$, $z = k$ can be satisfied for any k. Finally, it is easy to see that one has $z_3 = \hat{z} = k$ for the string $(ab)^{2k-1}$. The theorem is proved.

In the end, let us discuss a restricted version of LZ-type parsings that one can often observe in practical applications.

Theorem 4. For $\ell > 0$ and a given string of length $n$, denote by $z_{\ell}$ the size of a smallest (in the number of phrases) LZ-type parsing in which each phrase has length at most $\ell$. Then, we have $z_{\ell} \leq \frac{n}{\ell} \leq z \leq z_{\ell}$. The same bound holds for analogously restricted versions of $z_3$, $\hat{z}$, $z_3$. 

5
Proof. Since the bound $z \leq z_{<\ell}$ obviously follows from Lemma 2, it remains to show that $z_{<\ell} \leq \frac{z}{2} \leq z$. Let $f_1 f_2 \cdots f_z$ be the LZ parsing of $s$. We simply split each phrase $f_i$ of length greater than $\ell$ into $\lceil |f_i|/\ell \rceil$ phrases with lengths at most $\ell$. Obviously, in this way we obtain an LZ-type parsing whose size is at most $z + \frac{z}{2}$, which finishes the proof. The proof for $z_3$, $\tilde{z}$, $\tilde{z}_3$ is analogous.

5. Concluding Remarks

In the literature there is still a lack of information concerning the relations between different measures of compressibility for highly repetitive texts. In this paper we investigated the relations between the most popular versions of LZ77 but, besides LZ77, there are other popular measures. For instance, it is a major open problem to find tight relations between an LZ77 parsing of a given string and the number of runs in its Burrows–Wheeler transform (see [5]). Further, it is known that the size of the smallest grammar of any given string of length $n$ is within $O(\log n)$ factor of the size of the LZ parsing and it is known that this bound is tight to within a factor $O(\log \log n)$ (see [1, 7, 12]); but it is still open whether this bound can be improved to $O(\log \log n)$. The things are not always clear even in the realm of LZ77-alike parsings: for examples, it is still not known whether, as it was conjectured in [10], the so-called LZ-End parsing contains at most $2^z$ phrases. Finally, note a rather unexpected connection between $\tilde{z}$ and the number of distinct factors in the Lyndon decomposition of a string [8].

We refer the reader to [5] and references therein for further discussion on different measures of compressibility and their relations; other compression schemes and results on their relations can also be found in [14].

References

[1] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, A. Shelat, The smallest grammar problem, IEEE Transactions on Information Theory 51 (7) (2005) 2554–2576, doi: 10.1109/TIT.2005.850116.
[2] M. Cohn, Affine m-ary Gray codes, Information and Control 6 (1) (1963) 70–78, doi:10.1016/S0019-9958(63)90119-0.
[3] P. Ferragina, I. Nittto, R. Venturini, On the bit-complexity of Lempel–Ziv compression, SIAM Journal on Computing 42 (4) (2013) 1521–1541, doi: 10.1137/120869511.
[4] T. Gagie, P. Gawrychowski, J. Kärkkäinen, Y. Nekrich, S. J. Puglisi, LZ77-based self-indexing with faster pattern matching, in: LATIN 2014, vol. 8392 of LNCS, Springer, 2014, doi:10.1007/978-3-642-54425-1_63.
[5] T. Gagie, G. Navarro, N. Prezza, Optimal-time text indexing in BWT-runs bounded space, arXiv preprint arXiv:1705.10382 (2017) 1–39.
[6] F. Gray, Pulse code communication, US Patent 2,632,058, 1953.
[7] A. Jež, Approximation of grammar-based compression via recompression, Theoretical Computer Science 592 (2015) 115–134, doi:10.1016/j.tcs.2015.05.027.
[8] J. Kärkkäinen, D. Kempa, Y. Nakashima, S. J. Puglisi, A. M. Shur, On the size of Lempel–Ziv and Lyndon factorizations, in: STACS 2017, vol. 66 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 45:1–45:13, doi:10.4230/LIPIcs.STACS.2017.45, 2017.
[9] D. Kosolobov, Relations between greedy and bit-optimal LZ77 encodings, arXiv preprint arXiv:1707.09789.
[10] S. Kreft, G. Navarro, On compressing and indexing repetitive sequences, Theoretical Computer Science 483 (2013) 115–133, doi:10.1016/j.tcs.2012.02.006.
[11] G. Navarro, Indexing text using the Ziv-Lempel trie, J. Discrete Algorithms 2 (1) (2004) 87–114, doi:10.1016/S1570-8667(03)00066-2.
[12] W. Rytter, Application of Lempel–Ziv factorization to the approximation of grammar-based compression, Theoretical Computer Science 302 (1-3) (2003) 211–222, doi:10.1016/S0304-3975(02)00777-6.
[13] D. Salomon, Data compression: the complete reference, 4th edition, Springer Verlag, 2006.
[14] J. A. Storer, T. G. Szymanski, Data compression via textual substitution, Journal of the ACM 29 (4) (1982) 928–951, doi:10.1145/322344.322346.
[15] A. Thue, Über die gegenseitige Lage gleicher Zeichenreihen, Norske vid. Selsk. Skr. Mat. Nat. Kl. 1 (1912) 1–67.
[16] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343, doi:10.1109/TIT.1977.1055714.
[17] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory 24 (5) (1978) 530–536, doi:10.1109/TIT.1978.1055934.