

# Optimal LZ-End Parsing Is Hard

**Hideo Bannai** ✉ 

M&D Data Science Center, Tokyo Medical and Dental University, Japan

**Mitsuru Funakoshi** ✉ 


Department of Informatics, Kyushu University, Fukuoka, Japan  
Japan Society for the Promotion of Science, Tokyo, Japan

**Kazuhiro Kurita** ✉ 

Nagoya University, Japan

**Yuto Nakashima** ✉ 

Department of Informatics, Kyushu University, Fukuoka, Japan

**Kazuhisa Seto** ✉ 

Faculty of Information Science and Technology, Hokkaido University, Sapporo, Japan

**Takeaki Uno** ✉

National Institute of Informatics, Tokyo, Japan

---

## Abstract

LZ-End is a variant of the well-known Lempel-Ziv parsing family such that each phrase of the parsing has a previous occurrence, with the additional constraint that the previous occurrence must end at the end of a previous phrase. LZ-End was initially proposed as a greedy parsing, where each phrase is determined greedily from left to right, as the longest factor that satisfies the above constraint [Kreft & Navarro, 2010]. In this work, we consider an optimal LZ-End parsing that has the minimum number of phrases in such parsings. We show that a decision version of computing the optimal LZ-End parsing is NP-complete by showing a reduction from the vertex cover problem. Moreover, we give a MAX-SAT formulation for the optimal LZ-End parsing adapting an approach for computing various NP-hard repetitiveness measures recently presented by [Bannai et al., 2022]. We also consider the approximation ratio of the size of greedy LZ-End parsing to the size of the optimal LZ-End parsing, and give a lower bound of the ratio which asymptotically approaches 2.

**2012 ACM Subject Classification** Theory of computation → Data compression

**Keywords and phrases** Data Compression, LZ-End, Repetitiveness measures

**Digital Object Identifier** 10.4230/LIPIcs.CPM.2023.3

**Related Version** *Previous Version:* <https://arxiv.org/abs/2302.02586>

**Funding** This work was partially supported by JSPS KAKENHI Grant Numbers JP20H05964 (for YN).

*Hideo Bannai:* JSPS KAKENHI Grant Number JP20H04141.

*Mitsuru Funakoshi:* JSPS KAKENHI Grant Number JP20J21147.

*Kazuhiro Kurita:* JSPS KAKENHI Grant Number JP21K17812, JP22H03549, JP21H05861, and JST ACT-X Grant Number JPMJAX2105.

*Yuto Nakashima:* JSPS KAKENHI Grant Number JP21K17705, JP23H04386, and JST ACT-X Grant Number JPMJAX200K.

*Kazuhisa Seto:* JSPS KAKENHI Grant Number JP21H05839.

**Acknowledgements** We would like to thank Dominik Köppl for discussion. We also gratefully acknowledge the comments of anonymous reviewers for improving the manuscript.



© Hideo Bannai, Mitsuru Funakoshi, Kazuhiro Kurita, Yuto Nakashima, Kazuhisa Seto, and Takeaki Uno;

licensed under Creative Commons License CC-BY 4.0

34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023).

Editors: Laurent Bulteau and Zsuzsanna Lipták; Article No. 3; pp. 3:1–3:11

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

In the context of lossless data compression, various repetitiveness measures – especially those based on dictionary compression algorithms – and relations between them have recently received much attention (see the excellent survey by Navarro [12, 13]). One of the most fundamental and well-known measures is the LZ77 parsing [15], in which a string is parsed into  $z$  phrases such that each phrase is a single symbol, or the longest substring which has a previous occurrence. LZ-End [9, 10] is a variant of LZ77 parsing with the added constraint that a previous occurrence of the phrase must end at the end of a previous phrase. More formally, the LZ-End parsing is a factorization  $q_1, \dots, q_{z_e}$  of a given string that can be greedily obtained from left to right: each phrase  $q_i$  is either (1) a symbol that is the leftmost occurrence of the symbol or (2) the longest prefix of the remaining suffix  $q_i \cdots q_{z_e}$  that is a suffix of  $q_1 \cdots q_j$  for some  $j < i$ . It is known that LZ-End parsing can be computed in linear time [6], and there exists a space-efficient algorithm [5].

While there is no known data structure of  $O(z)$  size that provides efficient random access to arbitrary positions in the string, it was recently shown that  $\tilde{O}(1)$  time access could be achieved with  $O(z_e)$  space [8]. Furthermore, concerning the difference between  $z$  and  $z_e$ , an upper bound of  $z_e = O(z \log^2(n/z))$  was shown [8], where  $n$  is the length of the (uncompressed) string. On the other hand, there is an obvious bound of  $z_e = \Omega(z \log n)$  for the unary string, since a previous occurrence of an LZ-End phrase cannot be self-referencing, i.e., overlap with itself, while an LZ77 phrase can. Notice that  $z \leq z_{no} \leq z_e$  holds for any string, where  $z_{no}$  is the number of phrases in the LZ77 parsing that does not allow self-referencing. A family of strings such that the ratio  $z_e/z_{no}$  asymptotically approaches 2 (for large alphabet [10], for binary alphabet [4]) is known, and it is conjectured that  $z_e \leq 2z_{no}$  holds for any strings [10].

While the phrases in the parsings described above are chosen greedily (i.e., longest), we can consider variants which do not impose such constraint, e.g., in an *LZ-End-like* parsing, each phrase  $q_i$  is either (1) a symbol that is the leftmost occurrence of the symbol or (2) a (not necessary longest) prefix of the remaining suffix which is a suffix of  $q_1 \cdots q_j$  for some  $j < i$ . We refer to an LZ-End-like parsing with the smallest number  $z_{end}$  of phrases, an *optimal* LZ-End parsing [12], and call the original, the *greedy* LZ-End parsing.<sup>1</sup> Thus  $z \leq z_{no} \leq z_{end} \leq z_e$  holds.

Interestingly,  $z_{end} \leq g$  holds, where  $g$  is the size of the smallest context free grammar that derives (only) the string, while a similar relation between  $z_e$  and  $g$  does not seem to be known [12].

This brings us to two natural and important questions about the measure  $z_{end}$ :

- How efficiently can we compute  $z_{end}$ ?
- How much smaller can  $z_{end}$  be compared to  $z_e$ ?

In this work, we answer a part of the above questions. Namely:

1. We prove the NP-hardness of computing  $z_{end}$ .
2. We present an algorithm for exact computation by MAX-SAT.
3. We give a lower bound of the maximum value of the ratio  $z_e/z_{end}$ .

In Section 3, we give the hardness result. Our reduction is from the vertex cover problem: finding a minimum set  $U$  of vertices such that every edge in a graph is incident to some vertex in  $U$ . In Section 4, we show a MAX-SAT formulation for computing the optimal

---

<sup>1</sup> Notice that we do not need the distinction for LZ77, since the greedy LZ77 parsing is also an optimal LZ77-like parsing.

LZ-End parsing that follows an approach by Bannai et al. that allows computing NP-hard repetitiveness measures using MAX-SAT solvers [1]. In Section 5, we consider the ratio  $z_e/z_{end}$ . We give a family of binary strings such that the ratio asymptotically approaches 2. Note that we can easily modify this result to a larger alphabet. Since  $(z_e/z_{end}) \leq (z_e/z_{no})$ , the bound is tight, assuming that the conjecture by Kreft and Navarro [10] holds.

## Related work

The LZ77 and LZ78 are original members of the LZ family [15, 16]. It is well-known that the (greedy) LZ77 parsing produces the optimal version of the parsing [11]. The LZ78 parsing satisfies that each phrase can be represented as a concatenation of a previous phrase and a symbol. The NP-hardness of computing the optimal version of the LZ78 variant was shown [2]. This hardness result is also given by a reduction from the vertex cover problem. However, our construction of the reduction for the LZ-End differs from that for the LZ78 since these parsings have very different structures. The smallest string attractor [7] is one of the most fundamental repetitiveness measures. It is also known that computing the smallest string attractor of a given string is NP-hard [7]. The hardness result was proven by a reduction from the set-cover problem.

## 2 Preliminaries

### 2.1 Strings

Let  $\Sigma$  be an *alphabet*. An element of  $\Sigma^*$  is called a *string*. The length of a string  $w$  is denoted by  $|w|$ . The empty string  $\varepsilon$  is the string of length 0. Let  $\Sigma^+$  be the set of non-empty strings, i.e.,  $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ . For any strings  $x$  and  $y$ ,  $x \cdot y$  denotes the concatenation of two strings. We will sometimes abbreviate “ $\cdot$ ” (i.e.,  $x \cdot y = xy$ ). For a string  $w = xyz$ ,  $x$ ,  $y$  and  $z$  are called a *prefix*, *substring*, and *suffix* of  $w$ , respectively. They are called a *proper prefix*, a *proper substring*, and a *proper suffix* of  $w$  if  $x \neq w$ ,  $y \neq w$ , and  $z \neq w$ , respectively. The  $i$ -th symbol of a string  $w$  is denoted by  $w[i]$ , where  $1 \leq i \leq |w|$ . For a string  $w$  and two integers  $1 \leq i \leq j \leq |w|$ , let  $w[i..j]$  denote the substring of  $w$  that begins at position  $i$  and ends at position  $j$ . For convenience, let  $w[i..j] = \varepsilon$  when  $i > j$ . We will sometimes use  $w[i..j]$  to denote  $w[i..j - 1]$ . For any string  $w$ , let  $w^1 = w$  and let  $w^k = ww^{k-1}$  for any integer  $k \geq 2$ , i.e.,  $w^k$  is the  $k$ -times repetition of  $w$ .

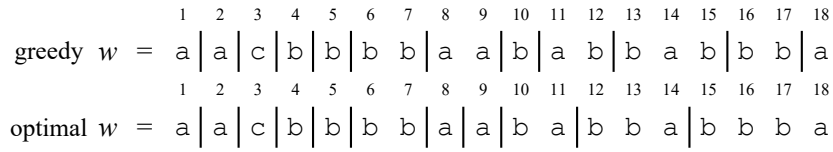
### 2.2 LZ-End parsing

We give a definition of the LZ-End parsing, which is a variant of the Lempel-Ziv family.

► **Definition 1** ((Greedy) LZ-End parsing). *The LZ-End parsing of a string  $w$  is the parsing  $LZEnd(w) = q_1, \dots, q_{z_e}$  of  $w$  such that  $q_i$  is either a symbol that is the leftmost occurrence of the symbol or the longest prefix of  $q_i \cdots q_{z_e}$  that occurs as a suffix of  $q_1 \cdots q_j$  for some  $j < i$ , which we call a source of the phrase.*

We refer to each  $q_i$  as a *phrase*. This definition, used in [8], is slightly different from the original version [9, 10] where a symbol is added to each phrase. The results in this paper hold for the original version as well (which we will show in the full version of the paper), but here we use this definition for simplicity. In this paper, we consider a more general version of the LZ-End parsing: a parsing  $q_1, \dots, q_{z_{end}}$  of a string  $w$  such that  $q_i$  is a (not necessary longest) suffix of  $q_1 \cdots q_j$  for some  $j < i$ . We call such a parsing with a minimum number  $z_{end}$  of phrases an *optimal LZ-End parsing* of  $w$ . We give an example of the greedy LZ-End parsing and the optimal LZ-End parsing in Figure 1.

### 3:4 Optimal LZ-End Parsing Is Hard



■ **Figure 1** Let  $w = \text{aacbbbbbbaababbabbba}$ . The greedy LZ-End parsing  $LZEnd(w)$  of  $w$  is illustrated in the upper part of the figure. For the phrase at position 10, a longer substring  $w[10..11] = \text{ba}$  has another previous occurrence at position 7, but there is no phrase that ends at position 8, and any longer substring does not have a previous occurrence. Therefore, the phrase starting at position 10 is  $\text{b}$ . The lower part of the figure shows an optimal LZ-End parsing (which is smaller than the greedy one) on the same string. Each phrase has a previous occurrence that ends at the end of some LZ-End phrase. The size of the greedy parsing is 12 and the size of the optimal parsing is 11.

## 2.3 Graphs

Let  $G = (V, E)$  be a graph with the set of vertices  $V$  and the set of edges  $E$ . An edge  $e = \{u, v\}$  is called an *incident edge of  $u$* . We denote the set of incident edges of  $v$  as  $\Gamma_G(v)$  and drop the subscript whenever it is clear from context. For an edge  $e = \{u, v\}$ , vertices  $u$  and  $v$  are the *end points* of  $e$ . For a subset of vertices  $U \subseteq V$ ,  $U$  is a *vertex cover* if for any  $e \in E$ , at least one end point of  $e$  is contained in  $U$ . Let  $\tau_G$  be the size of the minimum vertex cover of  $G$  (i.e.,  $\tau_G$  denotes the *vertex cover number* of  $G$ ). Notice that computing  $\tau_G$  is NP-complete [3].

## 2.4 Maximum Satisfiability (MAX-SAT) problem

Let  $\{x_1, \dots, x_n\}$  be a set of literals and  $C$  be a conjunctive normal form (CNF) formula. Each variable in  $C$  is assigned a Boolean value (i.e., true or false). The goal of the *Satisfiability (SAT)* problem is to compute an assignment of variables that satisfies all clauses of  $C$ . The *Maximum Satisfiability (MAX-SAT)* problem is a variant of SAT, in which there are two types of clauses: hard clauses and soft clauses. A solution for MAX-SAT is a truth assignment of the variables such that all hard clauses are satisfied, and the number of soft clauses that are satisfied is maximized.

## 3 NP-hardness of computing the optimal LZ-End parsing

In this section, we consider the problem of computing the optimal LZ-End parsing of a given string. A decision version of the problem is given as follows.

► **Problem 2** (Decision version of computing the optimal LZ-End parsing (OptLE)). *Given a string  $w$  and an integer  $k$ , decide whether there exists an LZ-End parsing of size  $k$  or less.*

We show the NP-completeness of OptLE in the following and present an algorithm for exact computation in the next section.

► **Theorem 3.** *OptLE is NP-complete.*

**Proof.** We give a reduction from the vertex cover problem to OptLE. Let  $G = (V, E)$  be a graph with a set of vertices  $V = \{v_1, \dots, v_n\}$  and a set of edges  $E = \{e_1, \dots, e_m\}$ . Suppose that an input graph  $G$  of the vertex cover problem is connected and  $|\Gamma(v)| \geq 2$  for any

$\mathcal{P}$	$v_1 v_1 v_1 \# v_1 v_1 \$ \# v_1 \$ \$ \# v_1 \$ e_1 \# v_1 \$ e_3 \# e_1 e_1 v_1 \# e_3 e_3 v_1 \# $ $v_2 v_2 v_2 \# v_2 v_2 \$ \# v_2 \$ \$ \# v_2 \$ e_1 \# v_2 \$ e_2 \# e_1 e_1 v_2 \# e_2 e_2 v_2 \# $ $v_3 v_3 v_3 \# v_3 v_3 \$ \# v_3 \$ \$ \# v_3 \$ e_2 \# v_3 \$ e_3 \# e_2 e_2 v_3 \# e_3 e_3 v_3 \# $	
$\mathcal{Q}$	$e_1 e_1 e_1 \# e_2 e_2 e_2 \# e_3 e_3 e_3 \# $	
$\mathcal{R}$	$v_1 v_1 v_1 v_1 \$ e_1 e_1 e_1 v_1 v_1 \$ e_3 e_3 e_3 v_1 v_1 \$ \$ \# $ $v_2 v_2 v_2 v_2 \$ e_1 e_1 e_1 v_2 v_2 \$ e_2 e_2 e_2 v_2 v_2 \$ \$ \# $ $v_3 v_3 v_3 v_3 \$ e_2 e_2 e_2 v_3 v_3 \$ e_3 e_3 e_3 v_3 v_3 \$ \$ \# $	greedy
$\mathcal{S}$	$\$ e_1 e_1 e_1 \# \$ e_2 e_2 e_2 \# \$ e_3 e_3 e_3 \# $	
$\mathcal{R}$	$v_1 v_1 v_1 v_1 \$ e_1 e_1 e_1 v_1 v_1 \$ e_3 e_3 e_3 v_1 v_1 \$ \$ \# $ $v_2 v_2 v_2 v_2 \$ e_1 e_1 e_1 v_2 v_2 \$ e_2 e_2 e_2 v_2 v_2 \$ \$ \# $ $v_3 v_3 v_3 v_3 \$ e_2 e_2 e_2 v_3 v_3 \$ e_3 e_3 e_3 v_3 v_3 \$ \$ \# $	optimal
$\mathcal{S}$	$\$ e_1 e_1 e_1 \# \$ e_2 e_2 e_2 \# \$ e_3 e_3 e_3 \# $	

■ **Figure 2** Let  $G = (V, E)$  be the complete graph of three vertices  $v_1, v_2, v_3$  and  $e_1 = \{v_1, v_2\}$ ,  $e_2 = \{v_2, v_3\}$ ,  $e_3 = \{v_1, v_3\}$ .  $\mathcal{W}_G$  and the greedy parsing and an optimal parsing are illustrated in the figure. The first two parts ( $\mathcal{P}$  and  $\mathcal{Q}$ ) share the same parsing. The last two parts ( $\mathcal{R}$  and  $\mathcal{S}$ ) are different. The upper part in the figure shows the greedy parsing and the lower part shows an optimal parsing. For instance, in the optimal parsing, we can choose  $\$e_1^3$  and  $\$e_3^3$  as phrases by using non-greedy parsing in  $\mathcal{R}_1$ . In other words, we can reduce two phrases in  $\mathcal{S}$ -part by adding one phrase in  $\mathcal{R}_1$ . In this example, the optimal parsing represents a vertex cover  $\{v_1, v_3\} \subset V$  of  $G$  (since  $\mathcal{R}_2$  selects the greedy parsing and the others are not).

$v \in V$ . We identify each vertex  $v_i$  as a symbol  $v_i$  and each edge  $e_i$  as a symbol  $e_i$ . We also introduce the symbol  $\$$ , and a set of symbols that occur uniquely in the string. The latter is represented, for simplicity, by the special symbol  $\#$ , i.e.,  $\#$  represents a different symbol each time it occurs in our description. We consider the string  $\mathcal{W}_G$  defined by graph  $G$  as follows.

- $\mathcal{W}_G = \prod_{i=1}^n \mathcal{P}_i \cdot \prod_{j=1}^m \mathcal{Q}_j \cdot \prod_{i=1}^n \mathcal{R}_i \cdot \prod_{j=1}^m \mathcal{S}_j$
- $\mathcal{P}_i = v_i^3 \# v_i^2 \$ \# v_i \$^2 \# \cdot \mathcal{X}_i \cdot \mathcal{Y}_i$
- $\mathcal{Q}_j = e_j^3 \#$
- $\mathcal{R}_i = v_i^4 \$ \prod_{e_j \in \Gamma(v_i)} (e_j^3 v_i^2 \$) \$ \#$
- $\mathcal{S}_j = \$ e_j^3 \#$
- $\mathcal{X}_i = \prod_{e_j \in \Gamma(v_i)} (v_i \$ e_j \#)$
- $\mathcal{Y}_i = \prod_{e_j \in \Gamma(v_i)} (e_j^2 v_i \#)$

An example of this string is illustrated in Figure 2. Note that we use  $i$  for representing indices of vertices and  $j$  for indices of edges.

Before we show the detail of the reduction, we start with an intuitive description of our reduction. The string  $\mathcal{W}_G$  consists of four parts (which are represented by  $\mathcal{P}$ ,  $\mathcal{Q}$ ,  $\mathcal{R}$ , and  $\mathcal{S}$ ). (1) The first two parts are non-functional parts. They can only be parsed in a single sensible way. These parts play a role as sources of the third part ( $\mathcal{R}$ -part). (2) In the third part,  $\mathcal{R}_i$  corresponds to the vertex  $v_i$ . Roughly speaking,  $\mathcal{R}_i$  can be parsed in two sensible ways such that the parsing represents whether the vertex  $v_i$  is in the vertex cover or not. If a vertex  $v_i$  is in the vertex cover, then the parsing of  $\mathcal{R}_i$  needs one more phrase. (3) In the last part,  $\mathcal{S}_j$  corresponds to the edge  $e_j$ .  $\mathcal{S}_j$  will be parsed into two phrases iff one of the incident vertex of the edge  $e_j$  is in the vertex cover. Otherwise,  $\mathcal{S}_j$  has three phrases. Overall, minimizing the number of vertices for the vertex cover of a graph  $G$  corresponds to minimizing the total penalties in the  $\mathcal{R}$ -part such that every  $\mathcal{S}_j$  will be parsed into two phrases.

### 3:6 Optimal LZ-End Parsing Is Hard

We show that the number of phrases of the optimal parsing of  $\mathcal{W}_G$  is less than  $13n+22m+k$  if and only if the vertex cover number  $\tau_G$  is less than  $k$ .

First, we observe an optimal LZ-End parsing of  $\mathcal{W}_G$ . Let us consider a parsing of  $\prod_{i=1}^n \mathcal{P}_i$ . In this part, the greedy parsing gives  $10n + 13m$  phrases. In the greedy parsing of  $\prod_{i=1}^n \mathcal{P}_i$ , phrases  $v_i^2$  in  $v_i^2\$\#, v_i\$\$$  in  $v_i\$\$^2\#, v_i\$\$e_j\#$ , and the second occurrence of  $e_j^2$  have length 2, and the other phrases have length 1. It is easy to see that this parsing is a smallest possible parsing of  $\prod_{i=1}^n \mathcal{P}_i$ . Moreover, other parsings of the same size do not affect the parsing of the rest of the string; candidates for a source cannot be increased by selecting any other parsings since the phrases of length 2 are preceded by unique symbols  $\#$ . Hence, we can choose this greedy parsing as a part of an optimal parsing.

In the second part  $\prod_{j=1}^m \mathcal{Q}_j$ , the greedy parsing also gives an optimal parsing which has  $3m$  phrases (i.e., each  $\mathcal{Q}_j$  is parsed into three phrases since  $e_j^2$  occurs in  $\mathcal{P}_i$  for some  $i$  and  $e_j^3$  is unique in  $\prod_{i=1}^n \mathcal{P}_i \cdot \prod_{j=1}^m \mathcal{Q}_j$ ). This parsing is also a smallest possible parsing and does not affect any parsings of the rest of the string.

The remaining suffix  $\prod_{i=1}^n \mathcal{R}_i \cdot \prod_{j=1}^m \mathcal{S}_j$  is a key of the reduction. The key idea is that  $\mathcal{S}_j$  represents whether the edge  $e_j$  is an incident edge of some vertex in a subset of vertices or not.  $\$e_j^3$  in  $\mathcal{S}_j$  has exactly two previous occurrences in the  $\mathcal{R}$ -part (since each edge is incident to exactly two vertices). Hence  $\mathcal{S}_j$  can be parsed into two phrases (i.e.,  $\$e_j^3, \#$ ) if and only if  $\$e_j^3$  has an occurrence which ends with an LZ-End phrase in the  $\mathcal{R}$ -part. Now we consider the greedy parsing of the  $\mathcal{R}_i$ -part (let  $\Gamma(v_i) = \{e_{(i,1)}, \dots, e_{(i,|\Gamma(v_i)|)}\}$ ), which is as follows:

$$v_i^3, v_i\$\$e_{(i,1)}, e_{(i,1)}^2 v_i, \dots, v_i\$\$e_{(i,|\Gamma(v_i)|)}, e_{(i,|\Gamma(v_i)|)}^2 v_i, v_i\$\$^2, \#.$$

The parsing has  $2|\Gamma(v_i)| + 3$  phrases. We claim that this parsing is the smallest possible parsing: If the length of every phrase is at most 3, then  $2|\Gamma(v_i)| + 3$  is the minimum size since the length of  $\mathcal{R}_i$  is  $6|\Gamma(v_i)| + 7$ . On the other hand, we can see that substrings of length at least 4 which contain a symbol  $v_i$  are unique in the whole string  $\mathcal{W}_G$  by the definition. Namely,  $\$e_j^3$  is the only substring of length at least 4 which is not unique. Let us consider a parsing of  $\mathcal{R}_i$  such that the parsing has  $\alpha$  length-4 phrases. In other words, we choose  $\alpha$  incident edges out of  $|\Gamma(v_i)|$  edges. Let  $(i_1, \dots, i_\alpha)$  be the sequence of indexes of selected edges. We observe that the length of substrings that are covered by length at most 3 phrases. The length of the prefix of  $\mathcal{R}_i$  that is succeeded by the first length-4 phrase  $\$e_{(i,i_1)}^3$  is  $6(i_1 - 1) + 4$ . This implies that there are at least  $2(i_1 - 1) + 2$  phrases. The length of substring between  $\$e_{(i,i_{d-1})}^3$  and  $\$e_{(i,i_d)}^3$  is  $6(i_d - i_{d-1} - 1) + 2$ . Thus there are at least  $2(i_d - i_{d-1} - 1) + 1$  phrases in each middle part. The length of the suffix that is preceded by the last length-4 phrase  $\$e_{(i,i_\alpha)}^3$  is  $6(|\Gamma(v_i)| - i_\alpha) + 5$ . Since the last symbol is a unique symbol  $\#$ , there are at least  $2(|\Gamma(v_i)| - i_\alpha) + 3$  phrases in the last part. Hence, there are at least

$$\alpha + 2(i_1 - 1) + 2 + \sum_{d=2}^{\alpha} (2(i_d - i_{d-1} - 1) + 1) + 2(|\Gamma(v_i)| - i_\alpha) + 3 = 2|\Gamma(v_i)| + 4$$

phrases. Thus the minimum number of phrases of  $\mathcal{R}_i$  is  $2|\Gamma(v_i)| + 3$  and the above greedy parsing is the only candidate which is the minimum size. Notice that phrases of this parsing do not end with  $\$e_j^3$ . Let us consider the other possible parsing of  $\mathcal{R}_i$ -part as follows:

$$v_i^2, v_i^2\$, e_{(i,1)}^3, \dots, v_i^2\$, e_{(i,|\Gamma(v_i)|)}^3, v_i^2\$, \$, \#.$$

This parsing has  $2|\Gamma(v_i)| + 4$  phrases. Notice that this parsing has phrases which end with  $\$e_j^3$ . Thus  $\mathcal{S}_j$  can be parsed into two phrases if we choose a non-greedy parsing such that there exists a phrase that ends at one of these positions. In other words, if we choose such a parsing

in the  $\mathcal{R}_i$ -part, we can reduce at most  $|\Gamma(v_i)|$  phrases in the  $\mathcal{S}$ -part. These observations imply that  $\mathcal{R}_i$  is parsed into  $2|\Gamma(v_i)| + 3$  or  $2|\Gamma(v_i)| + 4$  phrases in any optimal parsing of  $\mathcal{W}_G$ .

Let us consider an optimal LZ-End parsing. Let  $r$  be the number of substrings  $\mathcal{R}_i$  which contain  $2|\Gamma(v_i)| + 4$  phrases, and  $s$  be the number of substrings  $\mathcal{S}_j$  which contain exactly two phrases. Then the size of the parsing is

$$(10n + 13m) + (3m) + (2 \sum_{i=1}^n |\Gamma(v_i)| + 3n + r) + (3m - s) = 13n + 23m + r - s.$$

We consider a subset  $V'$  of vertices such that  $v_i \in V'$  if and only if  $\mathcal{R}_i$  is parsed into  $2|\Gamma(v_i)| + 4$  phrases (i.e.,  $|V'| = r$ ), and a subset  $E'$  of edges such that  $e_j \in E'$  if and only if  $\mathcal{S}_j$  is parsed into three phrases (i.e.,  $|E'| = m - s$ ). If  $E' = \emptyset$  (i.e.,  $s = m$ ),  $V'$  is a vertex cover of  $G$ . Otherwise,  $V'$  is not a vertex cover of  $G$ . However we can obtain the vertex cover number by using the parsing. Since the parsing is an optimal parsing, we can observe that there is no vertex  $v_i$  in  $V \setminus V'$  which has two or more incident edges in  $E'$  (we can reduce two or more phrases in  $\mathcal{S}$ -part by adding one phrase in  $\mathcal{R}_i$ , a contradiction). This implies that we can obtain a vertex cover by choosing one vertex in  $V \setminus V'$  for each edge in  $E \setminus E'$ . Then there exists an optimal LZ-End parsing of the same size which can directly represent a vertex cover. In other words, the vertex cover number is  $r + m - s$  if there exists an optimal LZ-End parsing of  $13n + 22m + (r + m - s)$  phrases. It is clear from the above constructions that there exists an optimal LZ-End parsing of  $13n + 22m + k$  phrases iff the vertex cover number is  $k$ .

Since we can check a parsing is an LZ-End parsing in linear time, OptLE is clearly in class NP. ◀

## 4 MAX-SAT Formulation

An approach for exact computation of various NP-hard repetitiveness measures was shown in [1], where they formulated them as MAX-SAT instances so that very efficient solvers could be taken advantage of. Here, we show that this approach can be adapted to computing the optimal LZ-End parsing as well.

Let the input string be  $T[1..n]$ , and for any  $i \in [2, n]$ , let  $M_i = \{j \mid 1 \leq j < i, T[j] = T[i]\}$ . Below, we use 1 to denote true, and 0 to denote false. We introduce the following Boolean variables:

- $p_i$  for all  $i \in [1, n]$ :  $p_i = 1$  if and only if position  $i$  is a starting position of an LZ-End phrase. Note that  $p_1 = 1$ .
- $c_i$  for all  $i \in [1, n]$ :  $c_i = 1$  if and only if position  $i$  is the left-most occurrence of symbol  $T[i]$ .
- $r_{i \rightarrow j}$  for all  $i \in [2, n]$  and  $j \in M_i$ :  $r_{i \rightarrow j} = 1$  if and only if position  $i$  references position  $j$  via an LZ-End factor.

Notice that the truth values of  $c_i$  are all fixed for a given string and are easy to determine. Furthermore, the left-most occurrence must be beginning of a phrase, so, some values of  $p_i$  can also be fixed. For all  $i \in [1, n]$ :

$$c_i = p_i = 1 \text{ if } i \text{ is left-most occurrence of } T[i], \tag{1}$$

$$c_i = 0 \text{ otherwise.} \tag{2}$$

### 3:8 Optimal LZ-End Parsing Is Hard

The truth values of  $p_i$  define the factors, so in order to minimize the number of factors, we define the soft clauses as  $\neg p_i$  for all  $i \in [1, n]$ . Below, we give other constraints between the variables that must be satisfied, i.e., hard clauses.

The symbol at any position must either be a left-most occurrence, or it must reference some position to its left. That is, for any  $i \in [1, n]$ :

$$c_i + \sum_{j \in M_i} r_{i \rightarrow j} = 1. \quad (3)$$

In order to ensure that references in the same LZ-End phrase are consistent, we have the following two constraints. The first ensures that if  $i$  references  $j$  and the symbols at positions  $i - 1$  and  $j - 1$  are different or do not exist (i.e.,  $j = 1$ ), positions  $i$  and  $i - 1$  cannot be in the same LZ-End phrase. For all  $i \in [2, n]$  and  $j \in M_i$  s.t.  $j = 1$  or  $T[j - 1] \neq T[i - 1]$ :

$$r_{i \rightarrow j} \implies p_i, \quad (4)$$

The second ensures that if position  $i$  references position  $j$  and  $i$  is not a start of an LZ-End phrase, then, position  $i - 1$  must reference position  $j - 1$ . For all  $i \in [2, n]$  and  $j \in M_i \setminus \{1\}$  s.t.  $T[j - 1] = T[i - 1]$ :

$$r_{i \rightarrow j} \wedge \neg p_i \implies r_{i-1 \rightarrow j-1}. \quad (5)$$

Finally, the following constraints ensure that the reference of each LZ-End phrase must end at an end of a previous LZ-End phrase. For all  $i \in [1, n]$  and  $j \in M_i$ :

$$\begin{cases} r_{i \rightarrow j} \wedge p_{i+1} \implies p_{j+1} & \text{if } i \in [1, n) \\ r_{i \rightarrow j} \implies p_{j+1} & \text{if } i = n. \end{cases} \quad (6)$$

It is easy to see that the truth assignments that are derived from any LZ-End parsing will satisfy the above constraints.

We now show that any truth assignment that satisfies the above constraints will represent a valid LZ-End parsing. The truth values for  $p_i$  implies a parsing where each phrase starts at a position  $i$  if and only if  $p_i = 1$ . Constraint (1),(2),(3) ensure that each position is either a left-most occurrence or references a unique previous position. Thus, it remains to show that the referencing of each position of a given factor is consistent (adjacent positions reference adjacent positions) and ends at a previous phrase end.

For any position  $i$  such that  $c_i = 0$ , let  $j \in M_i$  be the unique value such that  $r_{i \rightarrow j} = 1$ . We can see that any such position  $i$  that is not at the beginning of a phrase (i.e.,  $p_i = 0$ ) will reference a position consistent with the reference of position  $i - 1$ : If  $j = 1$  or  $T[j - 1] \neq T[i - 1]$ , then Constraint (4) would imply  $r_{i \rightarrow j} = 0$ . Thus, we have  $j > 1$  and  $T[j - 1] = T[i - 1]$ , and from Constraint (5), we have that  $r_{i-1 \rightarrow j-1}$ , and the referencing inside a factor is consistent. Finally, from Constraint (6), the last reference in a phrase always points to an end of a previous LZ-End phrase.

The MAX-SAT instance contains  $O(n^2)$  variables, and the total size of the CNF is  $O(n^2)$ :  $O(n)$  clauses of  $O(n)$  size (Constraint (3) using linear size encodings of cardinality constraints, e.g. [14]), and  $O(n^2)$  clauses of size  $O(1)$  (the soft clauses, and Constraints (4), (5), (6)).

We note that it is not difficult to obtain a MAX-SAT formulation for the original definition of LZ-End by minor modifications.



## 5 Approximation ratio of greedy parsing to optimal parsing

In this section, we consider an approximation ratio of the size  $z_e$  of the greedy LZ-End parsing to the size  $z_{end}$  of the optimal LZ-End parsing. Here, we give a lower bound of the ratio.

► **Theorem 4.** *There exists a family of binary strings such that the ratio  $z_e/z_{end}$  asymptotically approaches 2.*

**Proof.** Let  $K = \sum_{i=1}^k 2^i (= 2^{k+1} - 2)$  for any positive integer  $k \geq 1$ . The following binary string  $w_k$  over an alphabet  $\{a, b\}$  gives the lower bound:

$$w_k = aa \cdot \prod_{i=1}^k (a^{2^i}) \cdot b^4 \cdot \prod_{i=1}^K (a^i b^3).$$

It is easy to see that  $K$  is the length of the substring  $\prod_{i=1}^k (a^{2^i})$ . First, we show the greedy parsing of  $w_k$ . Let  $W_0 = aa \cdot \prod_{i=1}^k (a^{2^i}) \cdot b^4$  (i.e., a prefix of  $w_k$ ) and  $W_j = W_{j-1} \cdot a^j b^3$  for any  $1 \leq j \leq K$ . Notice that  $W_K = w_k$ . We show that

$$LZEnd(W_j) = LZEnd(W_{j-1}), a^j b^2, b \quad (7)$$

by induction on  $j$ . Initially, we consider the greedy parsing of  $W_0$ . The greedy parsing of the first run (i.e., maximal substring with a unique symbol) is  $a, a, a^2, \dots, a^{2^k}$  of size  $k+2$ . The second run is parsed into three phrases  $b, b, b^2$ . Thus

$$LZEnd(W_0) = a, a, a^2, \dots, a^{2^k}, b, b, b^2.$$

Moreover, we can see that the greedy parsing of  $W_1$  is

$$LZEnd(W_1) = a, a, a^2, \dots, a^{2^k}, b, b, b^2, ab^2, b.$$

Hence Equation 7 holds for  $j = 1$ . Suppose that Equation 7 holds for any  $j \leq p$  for some integer  $p \geq 1$ . We show that Equation 7 holds for  $j = p+1$ . Assume that there exists a phrase  $x$  of  $LZEnd(W_{p+1})$  which begins in  $W_p$  and ends in a new suffix  $a^{p+1}b^3$  of  $W_{p+1}$ . By the induction hypothesis, phrases of  $LZEnd(W_p)$  which end with  $a$  are only in the first  $a$ 's run. This implies that  $x$  cannot end with  $a$  and  $x$  can be written as  $x = x'ba^{p+1}b^\ell$  for some prefix  $x'$  of  $x$  and some positive integer  $\ell$ . However,  $a^{p+1}$  only occurs in the first  $a$ 's run. Thus  $LZEnd(W_{p+1})$  cannot have such a phrase  $x$ , namely  $LZEnd(W_{p+1}) = LZEnd(W_p), S$  for some factorization  $S$  of the remaining suffix  $a^{p+1}b^3$ . It is easy to see that the remaining suffix  $a^{p+1}b^3$  of  $W_{p+1}$  is parsed into  $a^{p+1}b^2, b$ . Hence Equation 7 holds for  $j = p+1$ , and it also holds for any  $j$ . Notice that  $|LZEnd(w_k)| = 2K + k + 5$  holds.

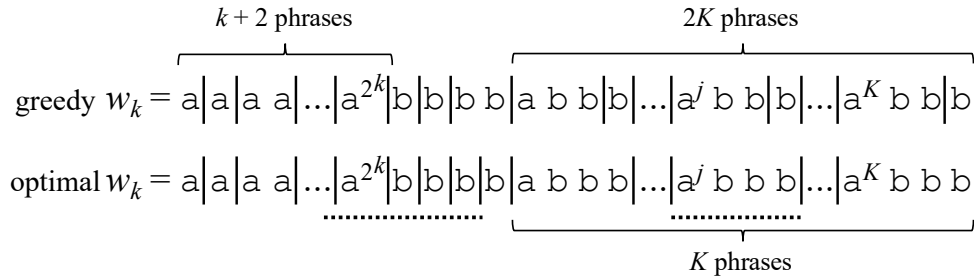
Finally, we give a smaller parsing of  $w_k$ . We consider the same parsing for the first run and a different parsing for the second run as  $b, b, b, b$ . In the greedy parsing,  $a^j b^3$  cannot be a phrase since the only previous occurrence does not have an LZ-End phrase. We can use a substring  $a^j b^3$  as a new phrase of  $W_j$  (see also Figure 3). Thus there exists an LZ-End parsing

$$a, a, a^2, \dots, a^{2^k}, b, b, b, b, a^1 b^3, \dots, a^K b^3.$$

The size of the parsing is  $K + k + 6$ .

Therefore the ratio  $z_e/z_{end}$  asymptotically approaches 2 for this family of strings. ◀

Note that this family of strings also gives a lower bound of the ratio  $z_e/z_{no}$  since  $(z_e/z_{end}) \leq (z_e/z_{no})$  holds.



■ **Figure 3** Illustration for two variants of LZ-End parsings of a string  $w_k$  (Theorem 4). In the optimal parsing, we can choose  $a^j b^3$  (dotted lines) as a phrase for each  $j$  ( $1 \leq j \leq K$ ) by adding a single letter phrase  $b$ .

## 6 Conclusions

In this paper, we first studied the optimal version of the LZ-End variant. We showed the NP-completeness of the decision version of computing the optimal LZ-End parsing and presented an approach for exact computation of the optimal LZ-End by formulating as MAX-SAT instances. We also gave a lower bound of the possible gap (as the ratio) between the greedy LZ-End and the optimal LZ-End. Finally, we note possible future work in the following.

- Our reduction from the vertex cover problem uses a polynomially large alphabet. How can we construct a reduction with a small alphabet?
- The most interesting remaining problem is an upper bound of the ratio discussed in Section 5. We conjecture that there exists a constant upper bound (i.e.,  $z_e/z_{end} \leq c$  for any strings where  $c$  is a constant). This implies that the greedy parsing gives a constant-approximation of the optimal parsing. On the other hand, if there exists a family of strings which gives  $c > 2$  or non-constant ratio, then the conjecture  $z_e \leq 2z_{no}$  does not stand.

---

## References

- 1 Hideo Bannai, Keisuke Goto, Masakazu Ishihata, Shunsuke Kanda, Dominik Köppl, and Takaaki Nishimoto. Computing np-hard repetitiveness measures via MAX-SAT. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPICs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ESA.2022.12.
- 2 Sergio De Agostino and James A. Storer. On-line versus off-line computation in dynamic text compression. *Information Processing Letters*, 59(3):169–174, 1996. doi:10.1016/0020-0190(96)00068-3.
- 3 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.
- 4 Takumi Ideue, Takuya Mieno, Mitsuru Funakoshi, Yuto Nakashima, Shunsuke Inenaga, and Masayuki Takeda. On the approximation ratio of LZ-End to LZ77. In Thierry Lecroq and Hélène Touzet, editors, *String Processing and Information Retrieval - 28th International Symposium, SPIRE 2021, Lille, France, October 4-6, 2021, Proceedings*, volume 12944 of *Lecture Notes in Computer Science*, pages 114–126. Springer, 2021. doi:10.1007/978-3-030-86692-1\_10.

- 5 Dominik Kempa and Dmitry Kosolobov. LZ-End parsing in compressed space. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer, editors, *2017 Data Compression Conference, DCC 2017, Snowbird, UT, USA, April 4-7, 2017*, pages 350–359. IEEE, 2017. doi:10.1109/DCC.2017.73.
- 6 Dominik Kempa and Dmitry Kosolobov. LZ-End parsing in linear time. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPICs*, pages 53:1–53:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ESA.2017.53.
- 7 Dominik Kempa and Nicola Prezza. At the roots of dictionary compression: string attractors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 827–840. ACM, 2018. doi:10.1145/3188745.3188814.
- 8 Dominik Kempa and Barna Saha. An upper bound and linear-space queries on the LZ-End parsing. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2847–2866. SIAM, 2022. doi:10.1137/1.9781611977073.111.
- 9 Sebastian Kreft and Gonzalo Navarro. LZ77-like compression with fast random access. In James A. Storer and Michael W. Marcellin, editors, *2010 Data Compression Conference (DCC 2010), 24-26 March 2010, Snowbird, UT, USA*, pages 239–248. IEEE Computer Society, 2010. doi:10.1109/DCC.2010.29.
- 10 Sebastian Kreft and Gonzalo Navarro. On compressing and indexing repetitive sequences. *Theor. Comput. Sci.*, 483:115–133, 2013. doi:10.1016/j.tcs.2012.02.006.
- 11 Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976. doi:10.1109/TIT.1976.1055501.
- 12 Gonzalo Navarro. Indexing highly repetitive string collections. *CoRR*, abs/2004.02781, 2020. arXiv:2004.02781.
- 13 Gonzalo Navarro. Indexing highly repetitive string collections, part I: repetitiveness measures. *ACM Comput. Surv.*, 54(2):29:1–29:31, 2021. doi:10.1145/3434399.
- 14 Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005. doi:10.1007/11564751\_73.
- 15 Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. doi:10.1109/TIT.1977.1055714.
- 16 Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978. doi:10.1109/TIT.1978.1055934.