


Compressed Communication Complexity of Hamming Distance

Shiori Mitsuya¹, Yuto Nakashima¹ , Shunsuke Inenaga^{1,2,*} , Hideo Bannai³  and Masayuki Takeda¹ 

¹ Department of Informatics, Kyushu University, 744, Motoooka, Nishiku, Fukuoka 819-0395, Japan; mitsuya.shiori@inf.kyushu-u.ac.jp (S.M.); yuto.nakashima@inf.kyushu-u.ac.jp (Y.N.); inenaga@inf.kyushu-u.ac.jp (S.I.); takeda@inf.kyushu-u.ac.jp (M.T.)

² PRESTO, Japan Science and Technology Agency, 4-1-8, Honcho, Kawaguchi, Saitama 332-0012, Japan

³ M&D Data Science Center, Tokyo Medical and Dental University, Tokyo 101-0062, Japan; hdbn.dsc@tmd.ac.jp

* Correspondence: inenaga@inf.kyushu-u.ac.jp

Abstract: We consider the communication complexity of the *Hamming distance* of two strings. Bille et al. [SPIRE 2018] considered the communication complexity of the longest common prefix (LCP) problem in the setting where the two parties have their strings in a compressed form, i.e., represented by the Lempel-Ziv 77 factorization (LZ77) with/without self-references. We present a randomized public-coin protocol for a joint computation of the *Hamming distance* of two strings represented by LZ77 without self-references. Although our scheme is heavily based on Bille et al.'s LCP protocol, our complexity analysis is original which uses Crochemore's C-factorization and Rytter's AVL-grammar. As a byproduct, we also show that LZ77 with/without self-references are not monotonic in the sense that their sizes can increase by a factor of $4/3$ when a prefix of the string is removed.

Keywords: communication complexity; *Hamming distance*; Lempel-Ziv compression



Citation: Mitsuya, S.; Nakashima, Y.; Inenaga, S.; Bannai, H.; Takeda, M. Compressed Communication Complexity of *Hamming Distance*. *Algorithms* **2021**, *14*, 116. <https://doi.org/10.3390/a14040116>

Academic Editor: Frank Werner

Received: 3 March 2021

Accepted: 31 March 2021

Published: 3 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Communication complexity, first introduced by Yao [1], is a well-studied sub-field of complexity theory which aims at quantifying the total amount of communication (bits) between the multiple parties who separately hold partial inputs of a function f . The goal of the k (≥ 2) parties is to jointly compute the value of $f(X_1, \dots, X_k)$, where X_i denotes the partial input that the i th party holds. Communication complexity studies lower bounds and upper bounds for the communication cost of a joint computation of a function f . Due to the rapidly increasing amount of distributed computing tasks, communication complexity has gained its importance in the recent highly digitalized society. This paper deals with the most basic and common setting where the two parties, Alice and Bob, separately hold partial inputs A and B and they perform a joint computation of $f(A, B)$ for a function f following a specified protocol.

We pay our attention to communication complexity of string problems where the inputs A and B are strings over an alphabet Σ . Communication complexity of string problems has played a critical role in the space lower bound analysis of several streaming processing problems including Hamming/edit/swap distances [2], pattern matching with k -mismatches [3], parameterized pattern matching [4], dictionary matching [5], and quasi-periodicity [6].

Bille et al. [7] were the first to consider the communication complexity of the longest common prefix (LCP) problem in the setting where the two parties have their strings in a compressed form, i.e., represented by the *Lempel-Ziv 77 factorization* (LZ77) [8] with/without self-references. Bille et al. [7] proposed a randomized public-coin protocol for the LCP problem with $O(\log z_\ell)$ communication rounds and $O(\log \ell)$ total bits of communication, where ℓ denotes the length of the LCP of the two strings A and B and z_ℓ denotes the size of the *non-self-referencing* LZ77 factorization of the LCP $A[1..\ell]$. In addition, Bille et al. [7] showed a randomized public-coin protocol for the LCP problem with

- (i) $O(\log z'_\ell + \log \log \ell)$ communication rounds and $O(\log \ell)$ total bits of communication, or
- (ii) $O(\log z'_\ell)$ communication rounds and $O(\log \ell + \log \log \log n)$ total bits of communication,

where z'_ℓ denotes the size of the *self-referencing* LZ77 factorization of the LCP $A[1..\ell]$ and $n = |A|$.

In this paper, we consider the communication complexity of the *Hamming distance* of two strings of equal length, which are represented in a compressed form. We present a randomized public-coin protocol for a joint computation of the *Hamming distance* of two strings represented by *non-self-referencing* LZ77, with $O(d \log z)$ communication rounds and $O(d \log \ell_{\max})$ total bits of communication, where d is the *Hamming distance* between A and B , z is the size of the LZ77 factorization of string A , and ℓ_{\max} is the largest gap between two adjacent mismatching positions between A and B (if the first/last characters of A and B are equal, then we can add terminal symbols as $\#A\#$ and $\#B\#$ and subtract 2 from the computed distance). Although our scheme is heavily based on Bille et al.'s LCP protocol, our complexity analysis is original which uses Crochemore's C-factorization [9] and Rytter's AVL-grammar [10].

Furthermore, as a byproduct of our result for the *Hamming distance* problem, we also show that LZ77 with/without self-references are *non-monotonic*. For a compression algorithm A let $A(S)$ denote the size of the compressed representation of string S by A . We say that compression algorithm A is *monotonic* if $A(S[1..j]) \leq A(S)$ for any $1 \leq j < |S|$ and $A(S[i..|S|]) \leq A(S)$ for any $1 < i \leq |S|$, and we say it is *non-monotonic* otherwise. It is clear that LZ77 with/without self-references satisfy the first property, however, to our knowledge the second property has not been studied for the LZ77 factorizations. We prove that LZ77 with/without self-references is *non-monotonic* by giving a family of strings such that removing each prefix of length from 1 to \sqrt{n} increases the number of factors in the LZ77 factorization by a factor of $4/3$, where n denotes the string length. We also show that in the worst-case the number of factors in the non-self-referencing LZ77 factorization of any suffix of any string S of length n can be larger than that of S by at most a factor of $O(\log n)$.

Monotonicity of compression algorithms and string repetitive measures has gained recent attention. Lagarde and Perifel [11] showed that *Lempel-Ziv 78 compression* [12] is non-monotonic by showing that removing the first character of a string can increase the size of the compression by a factor of $\Omega(\log n)$. The recently proposed repetitive measure called the *substring complexity* δ is known to be monotonic [13]. Kociumaka et al. [13] posed an open question whether the smallest bidirectional macro scheme size b [14] or the smallest string attractor size γ [15] is monotonic. It was then answered by Mantaci et al. [16] that γ is non-monotonic.

2. Preliminaries

2.1. Strings

Let Σ be an *alphabet* of size σ . An element of Σ^* is called a *string*. The length of a string S is denoted by $|S|$. The empty string ε is the string of length 0, namely $|\varepsilon| = 0$. The i -th character of a string S is denoted by $S[i]$ for $1 \leq i \leq |S|$, and the *substring* of a string S that begins at position i and ends at position j is denoted by $S[i..j]$ for $1 \leq i \leq j \leq |S|$. For convenience, let $S[i..j] = \varepsilon$ if $j < i$. Substrings $S[1..j]$ and $S[i..|S|]$ are respectively called a *prefix* and a *suffix* of S . For simplicity, let $S[.j]$ denote the prefix of S ending at position j and $S[i..]$ the suffix $S[i..|S|]$ of S beginning at position i . A suffix $S[j..]$ with $j > 1$ is called a *proper suffix* of S .

For string X and Y , let $\text{lcp}(X, Y)$ denote the length of the *longest common prefix* (LCP) of strings X, Y , namely $\text{lcp}(X, Y) = \max(\{\ell \mid X[..\ell] = Y[..\ell], 1 \leq \ell \leq \min\{|X|, |Y|\}\} \cup \{0\})$. The *Hamming distance* $d_H(X, Y)$ of two strings X, Y of equal length is the number of positions where the underlying characters differ between X and Y , namely $d_H(X, Y) = |\{i \mid X[i] \neq Y[i], 1 \leq i \leq |X|\}|$.

2.2. Lempel-Ziv 77 Factorizations

Of many versions of Lempel-Ziv 77 factorization [8] which divide a given string in a greedy left-to-right manner, the main tool we use is the non-self-referencing LZ77, which is formally defined as follows:

Definition 1 (Non-self-referencing LZ77 factorization). *The non-self-referencing LZ77 factorization of string S , denoted $LZN(S)$, is a factorization $S = f_1 \cdots f_{zn}$ that satisfies the following: Let u_i denote the beginning position of each factor f_i in the factorization $f_1 \cdots f_{zn}$, that is, $u_i = |f_1 \cdots f_{i-1}| + 1$. (1) If $i > 1$ and $\max_{1 \leq j < u_i} \{\text{lcp}(S[u_i..], S[j..u_i - 1])\} \geq 1$, then for any position $s_i \in \arg \max_{1 \leq j < u_i} \text{lcp}(S[u_i..], S[j..u_i - 1])$ in S , let $p_i = \text{lcp}(S[u_i..], S[s_i..u_i - 1])$. (2) Otherwise, let $p_i = 0$. Then, $f_i = S[s_i..u_i + p_i]$ for each $1 \leq i \leq zn$.*

Intuitively, each factor f_i in $LZN(S)$ is either a fresh letter, or the shortest prefix of $f_1 \cdots f_{zn}$ that does not have a previous occurrence in $f_1 \cdots f_{i-1}$. This means that self-referencing is *not* allowed in $LZN(S)$, namely no previous occurrences $S[s_i..s_i + p_i]$ of each factor f_i can overlap with itself.

The size $zn(S)$ of $LZN(S)$ is the number zn of factors in $LZN(S)$.

We encode each factor f_i by a triple $(s_i, p_i, \alpha_i) \in ([1..n] \times [1..n] \times \Sigma)$, where s_i is the left-most previous occurrence of f_i , p_i is the length of f_i , and α_i is the last character of f_i .

Example 1. For $S = \text{abaababaabaabaabaabb}$, $LZS(S) = a \mid b \mid aa \mid bab \mid aabaa \mid baabaab \mid aabb \mid$ and it can be represented as $(0, 0, a), (0, 0, b), (1, 2, a), (2, 3, b), (3, 5, a), (7, 7, b), (3, 4, b)$. The size of $LZS(S)$ is 7.

The self-referencing counterpart is defined as follows:

Definition 2 (Self-referencing LZ77 factorization). *The self-referencing LZ77 factorization of string S , denoted $LZS(S)$, is a factorization $S = g_1 \cdots g_{zs}$ that satisfies the following: Let v_i denote the beginning position of each factor g_i in the factorization $g_1 \cdots g_{zs}$, that is, $v_i = |g_1 \cdots g_{i-1}| + 1$. (1) If $i > 1$ and $\max_{1 \leq j < v_i} \{\text{lcp}(S[v_i..], S[j..])\} \geq 1$, then for any position $t_i \in \arg \max_{1 \leq j < v_i} \text{lcp}(S[v_i..], S[j..])$ in S , let $q_i = \text{lcp}(S[v_i..], S[t_i..])$. (2) Otherwise, let $q_i = 0$. Then, $g_i = S[v_i..v_i + q_i]$ for each $1 \leq i \leq zs$.*

Intuitively, each factor g_i of $LZS(S)$ is either a fresh letter, or the shortest prefix of $g_1 \cdots g_{zs}$ that does not have a previous occurrence beginning in $g_1 \cdots g_{i-1}$. This means that self-referencing is allowed in $LZS(S)$, namely the left-most previous occurrence with smallest t_i of each factor g_i may overlap with itself.

The size $zs(S)$ of $LZS(S)$ is the number zs of factors in $LZS(S)$.

Likewise, we encode each factor g_i by a triple $(t_i, q_i, \beta_i) \in ([1..n] \times [1..n] \times \Sigma)$, where t_i is the left-most previous occurrence of g_i , q_i is the length of g_i , and β_i is the last character of g_i .

Example 2. For $S = \text{abaababaabaabaabaabb}$, $LZN(S) = a \mid b \mid aa \mid bab \mid aabaa \mid baabaabaabb \mid$ and it can be represented as $(0, 0, a), (0, 0, b), (1, 2, a), (2, 3, b), (3, 5, a), (7, 11, b)$. The size of $LZS(S)$ is 6.

2.3. Communication Complexity Model

Our approach is based on the standard communication complexity model of Yao [1] between two parties:

- The parties are Alice and Bob;
- The problem is a function $f : X \times Y \rightarrow Z$ for arbitrary sets X, Y, Z ;
- Alice has instance $x \in X$ and Bob has instance $y \in Y$;

- The goal of the two parties is to output $f(x, y)$ for a pair (x, y) of instances by a joint computation;
- The joint computation (i.e., the communication between Alice and Bob) follows a specified protocol \mathcal{P} .

The communication complexity [1] usually refers merely to the total amount of bits that need to be transferred between Alice and Bob to compute $f(x, y)$. In this paper, we follow Bille et al.'s model [7] where the communication complexity is evaluated by a pair $\langle r, b \rangle$ of the number of communication rounds r and the total amount of bits b exchanged in the communication.

In a (Monte-Carlo) randomized public-coin protocol, each party (Alice/Bob) can access a shared infinitely long sequence of independent random coin tosses. The requirement is that the output must be correct for every pair of inputs with probability at least $1 - \epsilon$ for some $0 < \epsilon < 1/2$, which is based on the shared random sequence of coin tosses. We remark that one can amplify the error rate to an arbitrarily small constant by paying a constant factor penalty in the communication complexity. Please note that the public-coin model differs from a randomized private-coin model, where in the latter the parties do not share a common random sequence and they can only use their own random sequence. In a deterministic protocol, every computation is performed without random sequences.

2.4. Joint Computation of Compressed String Problems

In this paper, we also consider the communication complexity of the *Hamming distance* problem between two compressed strings of equal length, which are compressed by LZ77 without self-references.

Problem 1 (*Hamming distance with non-self-referencing LZ77*).

Alice's input: $\text{LZN}(A)$ for string A of length n .

Bob's input: $\text{LZN}(B)$ for string B of length n .

Goal: Both Alice and Bob obtain the value of $d_H(A, B)$.

The following LCP problem for two strings compressed by non-self-referencing LZ77 has been considered by Bille et al. [7].

Problem 2 (*LCP with non-self-referencing LZ77*).

Alice's input: $\text{LZN}(A)$ for string A .

Bob's input: $\text{LZN}(B)$ for string B .

Goal: Both Alice and Bob obtain the value of $\text{lcp}(A, B)$.

Bille et al. proposed the following protocol for a joint computation of the LCP of two strings compressed by non-self-referencing LZ77:

Theorem 1 ([7]). *Suppose that the alphabet Σ and the length n of string A are known to both Alice and Bob. Then, there exists a randomized public-coin protocol which solves Problem 2 with communication complexity $\langle O(\log z_\ell), O(\log \ell) \rangle$, where $\ell = \text{lcp}(A, B)$ and $z_\ell = \text{zn}(A[1..\ell])$.*

The basic idea of Bille et al.'s protocol [7] is as follows: In their protocol, the sequences of factors in the non-self-referencing LZ77 factorizations $\text{LZN}(A)$ and $\text{LZN}(B)$ are regarded as strings of respective lengths $\text{zn}(A)$ and $\text{zn}(B)$ over an alphabet $[1..n] \times [1..n] \times \Sigma$. Then, Alice and Bob jointly compute the LCP of $\text{LZN}(A)$ and $\text{LZN}(B)$, which gives them the first mismatching factors between $\text{LZN}(A)$ and $\text{LZN}(B)$. This LCP of $\text{LZN}(A)$ and $\text{LZN}(B)$ is computed by a randomized protocol for doubling-then-binary searches with $O(\log z_\ell)$ communication rounds. Finally, Alice sends the information about her first mismatching factor to Bob, and he internally computes the LCP of A and B . The total number of bits exchanged is bounded by $O(\log \ell)$.

In Section 3, we present our protocol for Problem 1 of jointly computing the *Hamming distance* of two strings compressed by *non-self-referencing LZ77*. The scheme itself is a simple application of the LCP protocol of Theorem 1 for *non-self-referencing LZ77*, but our communication complexity analysis is based on non-trivial combinatorial properties of LZ77 factorization which, to our knowledge, were not previously known.

3. Compressed Communication Complexity of Hamming Distance

In this section, we show a Monte-Carlo randomized protocol for Problem 1 that asks for the *Hamming distance* $d_H(A, B)$ of strings A and B that are compressed by *non-self-referencing LZ77*. Our protocol achieves $\langle O(d \log z), O(d \log \ell_{\max}) \rangle$ communication complexity, where $d = d_H(A, B)$, $z = zn(A)$, and ℓ_{\max} is the largest value returned by the sub-protocol of the LCP problem for two strings compressed by *non-self-referencing LZ77*.

The basic idea is to apply the so-called *Kangaroo jumping* method, namely if d is the number of mismatching positions between A and B , then one can compute $d = d_H(A, B)$ with at most $d + 1$ LCP queries. More specifically, let $1 \leq i_1 < \dots < i_d \leq n$ be the sequence of mismatching positions between A and B . By using the protocol of Theorem 1 as a black box, and using the fact that $zn(S) \geq zn(S[1..j])$ for any prefix $S[1..j]$ of any string S , we immediately obtain the following:

Lemma 1. *Suppose that the alphabet Σ and the length n of strings A and B are known to both Alice and Bob. Then, there exists a randomized public-coin protocol which solves Problem 1 with communication complexity $\langle O(\sum_{k=1}^d \log zn(A[i_k + 1..])), O(d \log \ell_{\max}) \rangle$, where $\ell_{\max} = \max_{1 < k \leq d} \{i_k - i_{k-1} + 1\}$.*

3.1. On the Sizes of Non-Self-Referencing LZ77 Factorization of Suffixes

Our next question is how large the $zn(A[i_k + 1..])$ term in Lemma 1 can be in comparison to $zn(A)$. To answer this question, we consider the following general measure: For any string of length n , let

$$\zeta(n) = \max\{zn(S[i..])/zn(S) \mid S \in \Sigma^n, 1 < i \leq n\}.$$

3.1.1. Lower Bound for $\zeta(n)$

In this subsection, we present a family of strings S such that $zn(S[i..]) > zn(S)$ for some suffix $S[i..]$, namely $\zeta(n) > 1$. More specifically, we show the following:

Lemma 2. *$\zeta(n)$ is asymptotically lower bounded by $4/3$.*

Proof. For simplicity, we consider an integer alphabet $\{0, 1, \dots, \sigma\}$ of size $\sigma + 1$. Consider the string

$$S = (012 \dots \sigma - 1 \ \sigma)(0124)(012346)(01234568) \dots (012 \dots \sigma - 2 \ \sigma)$$

and its proper suffix

$$S[2..] = (12 \dots \sigma - 1 \ \sigma)(0124)(012346)(01234568) \dots (012 \dots \sigma - 2 \ \sigma).$$

The non-self-referencing LZ77 factorization of S and $S[2..]$ are:

$$\begin{aligned} \text{LZN}(S) &= 0|1|2|\dots|\sigma-1|\sigma|0124|012346|01234568|\dots|012\dots\sigma-2|\sigma| \\ \text{LZN}(S[2..]) &= 1|2|\dots|\sigma-1|\sigma|01|24|0123|46|012345|68|\dots|012\dots|\sigma-2|\sigma| \end{aligned}$$

Observe that after the first occurrence of character σ , each factor of $\text{LZN}(S)$ is divided into two smaller factors in $\text{LZN}(S[2..])$. Since $zn(S) = |\text{LZN}(S)| = (\sigma + 1) + (\frac{\sigma}{2} - 1) = \frac{3\sigma}{2}$ and $zn(S[2..]) = |\text{LZN}(S[2..])| = (\sigma) + (\sigma - 2) = 2\sigma - 2$, $zn(S[2..])/zn(S) = \frac{2\sigma-2}{(3\sigma/2)} =$

$\frac{4}{3} - \frac{2}{3\sigma}$, which tends to $4/3$ as σ goes to infinity. We finally remark that $|S| = n = \Theta(\sigma^2)$ which in turn means that $\sigma = \Theta(\sqrt{n})$. \square

Remark 1. One can generalize the string S of Lemma 2 by replacing 0 with 0^h for arbitrarily fixed $1 < h \leq a \cdot \sigma$ for any constant a . The upper limit $a \cdot \sigma$ comes from the fact that the number of 0's in the original string S is exactly $\frac{\sigma}{2}$. Since $|S| = n = \Theta(\sigma^2)$, replacing 0 by 0^h with $h < a \cdot \sigma$ keeps the string length within $O(n)$. This implies that one can obtain the asymptotic lower bound $4/3$ for any suffix $S[h..]$ of length roughly up to $n - \sqrt{n}$.

Note also that the factorizations shown in Lemma 2 coincide with the self-referencing counterparts $LZS(S)$ and $LZS(S[2..])$, respectively. The next corollary immediately follows from Lemma 2 and Remark 1.

Corollary 1. *The Lempel-Ziv 77 factorization with/without self-references is non-monotonic.*

3.1.2. Upper Bound for $\zeta(n)$

Next, we consider an upper bound for $\zeta(n)$. The tools we use here are the C-factorization [9] without self-references, and a grammar compression called AVL-grammar [10].

Definition 3 (Non-self-referencing C-factorization). *The non-self-referencing C-factorization of string S , denoted $CN(S)$, is a factorization $S = c_1 \cdots c_{cn}$ that satisfies the following: Let w_i denote the beginning position of each factor c_i in the factorization $c_1 \cdots c_{cn}$, that is, $w_i = |c_1 \cdots c_{i-1}| + 1$. (1) If $i > 1$ and $\max_{1 \leq j < w_i} \{\text{lcp}(S[w_i..], S[j..w_i - 1])\} \geq 1$, then for any position $r_i \in \arg \max_{1 \leq j < w_i} \text{lcp}(S[w_i..], S[j..w_i - 1])$ in S , let $y_i = \text{lcp}(S[w_i..], S[r_i..w_i - 1]) - 1$. (2) Otherwise, let $y_i = 0$. Then, $c_i = S[w_i..w_i + y_i]$ for each $1 \leq i \leq cn$.*

The size $cn(S)$ of $CN(S)$ is the number cn of factors in $CN(S)$.

Example 3. For $S = \text{abaababaabaabaabaabb}$, $CN(S) = \text{a} \mid \text{b} \mid \text{a} \mid \text{ab} \mid \text{abaab} \mid \text{aaba} \mid \text{abaabaab} \mid \text{b}$ and its size is 8.

The difference between $LZN(S)$ and $CN(S)$ is that while each factor f_i in $LZN(S)$ is the shortest prefix of $S[u_i..]$ that does not occur in $S[1..u_i - 1]$, each factor c_i in $CN(S)$ is the longest prefix of $S[w_i..]$ that occurs in $S[1..w_i - 1]$. This immediately leads to the next lemma.

Lemma 3. *For any string S , $cn(S) \geq zn(S)$.*

We also use the next lemma in our upper bound analysis for $\zeta(n)$.

Lemma 4. *For any string S , $cn(S) \leq 2zn(S)$.*

Proof. Suppose that there are two consecutive factors c_i, c_{i+1} of $CN(S)$ and a factor f_j of $LZN(S)$ such that c_i, c_{i+1} are completely contained in f_j and the ending position of c_{i+1} is less than the ending position of f_j . Since $c_i c_{i+1}$ is a substring of $f_j[1..|f_j| - 1]$ and $f_j[1..|f_j| - 1]$ has a previous occurrence in $f_1 \cdots f_{j-1}$, this contradicts that c_i terminated inside $f_j[1..|f_j| - 1]$.

Thus, the only possible case is that $c_i c_{i+1}$ occurs as a suffix of f_j . Please note that in this case c_{i-1} cannot occur inside f_j by the same reasoning as above. Therefore, at most two consecutive factors of $CN(S)$ can occur completely inside of each factor of $LZN(S)$. This leads to $cn(S) \leq 2zn(S)$. \square

An AVL-grammar of a string S is a kind of a *straight-line program (SLP)*, which is a context-free grammar in the Chomsky-normal form which generates only S . The parse-tree of the AVL-grammar is an AVL-tree [17] and therefore, its height is $O(\log n)$ if n is the length of S . Let $avl(S)$ denote the size (i.e., the number of productions) in the AVL-grammar

for S . Basically, the AVL-grammar for S is constructed from the C-factorization of S , by introducing at most $O(\log n)$ new productions for each factor in the C-factorization. Thus, the next lemma holds.

Lemma 5 ([10]). *For any string S of length n , $\text{avl}(S) = O(\text{cn}(S) \log n)$.*

Now we show our upper bound for $\zeta(n)$.

Lemma 6. $\zeta(n) = O(\log n)$.

Proof. Suppose we have two AVL-grammars for strings X and Y of respective sizes $\text{avl}(X)$ and $\text{avl}(Y)$. Rytter [10] showed how to build an AVL-grammar for the concatenated string XY of size $\text{avl}(X) + \text{avl}(Y) + O(h)$, where h is the height of the taller parse-tree of the two AVL-grammars before the concatenation. This procedure is based on a folklore algorithm (cf [18]) that concatenates two given AVL-trees of height h with $O(h)$ node rotations. In the concatenation procedure of AVL-grammars, $O(1)$ new productions are produced per node rotation. Therefore, $O(h)$ new productions are produced in the concatenation operation.

Suppose we have the AVL-grammar of a string S of length n . It contains $\text{avl}(S)$ productions and the height of its parse-tree is $h = O(\log n)$ since an AVL-tree is a balanced binary tree. For any proper suffix $S' = S[i..n]$ of S with $1 < i \leq n$, we split the AVL-grammar into two AVL-grammars, one for the prefix $S[1..i]$ and the other for the suffix $S[i..n]$. We ignore the former and concentrate on the latter for our analysis. Since split operations on a given AVL-grammar can be performed in a similar manner to the aforementioned concatenation operations, we have that $\text{avl}(S') \leq \text{avl}(S) + a \log n$ for some constant $a > 0$. Now it follows from Lemma 3, Lemma 4, Lemma 5, and that the size cn of the C-factorization of any string is no more than the number of productions in any SLP generating the same string [10], we have

$$\text{zn}(S') \leq \text{cn}(S') \leq \text{avl}(S') \leq \text{avl}(S) + a \log n \leq a' \text{cn}(S) \log n \leq 2a' \text{zn}(S) \log n$$

where $a' > 0$ is a constant. This gives us $\text{zn}(S')/\text{zn}(S) = O(\log n)$ for any string S of length n and any of its proper suffix S' . \square

Since the size $\text{zn}(S)$ of the non-self-referencing LZ77 factorization of any string S of length n is at least $\log n$, the next corollary is immediate from Lemma 6:

Corollary 2. *For any string S and its proper suffix S' , $\text{zn}(S')/\text{zn}(S) = O(\text{zn}(S))$.*

3.2. Compressed Communication Complexity of Hamming Distance

Now we have the main result of this section.

Theorem 2. *Suppose that the alphabet Σ and the length n of strings A and B are known to both Alice and Bob. Then, there exists a randomized public-coin protocol which solves Problem 1 with communication complexity $\langle O(d \log \text{zn}), O(d \log \ell_{\max}) \rangle$, where $\text{zn} = \text{zn}(A)$ and $\ell_{\max} = \max_{1 < k \leq d} \{i_k - i_{k-1} + 1\}$.*

Proof. The protocol of Lemma 1 has $O(\sum_{k=1}^d \log \text{zn}(A[i_k + 1..n]))$ rounds. By Corollary 2, we have that $\text{zn}(A[i_k + 1..n]) = O(\text{zn}(A)^2)$. Therefore, $\sum_{k=1}^d \log \text{zn}(A[i_k + 1..n]) = O(d \log \text{zn}(A))$, which proves the theorem. \square

4. Conclusions and Open Questions

This paper showed a randomized public-coin protocol for a joint computation of the Hamming distance of two compressed strings. Our Hamming distance protocol relies on Bille et al.'s LCP protocol for two strings that are compressed by non-self-referencing LZ77,

while our communication complexity analysis is based on new combinatorial properties of non-self-referencing LZ77 factorization.

As a further research, it would be interesting to consider the communication complexity of the *Hamming distance* problem using self-referencing LZ77. The main question to this regard is whether $zs(S[i..]) = O(\text{poly}(zs(S)))$ holds for any suffix $S[i..]$ of any string S . In the case of non-self-referencing LZ77, $zn(S[i..]) = O(zn(S)^2)$ holds due to Lemma 2.

Author Contributions: Conceptualization, Y.N, S.I., H.B. and M.T.; methodology, S.M., Y.N., and S.I.; writing—original draft preparation, S.I.; writing—review and editing, Y.N. and H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JSPS KAKENHI Grant Numbers JP18K18002 (YN), JP20H04141 (HB), JP18H04098 (MT), and JST PRESTO Grant Number JPMJPR1922 (SI).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yao, A.C. *Some Complexity Questions Related to Distributive Computing (Preliminary Report)*; In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Theory of Computing, Atlanta, Georgia, 30 April – 2 May 1979, Publisher: Association for Computing Machinery, New York, US; pp. 209–213.
2. Clifford, R.; Jalsenius, M.; Porat, E.; Sach, B. Space lower bounds for online pattern matching. *Theor. Comput. Sci.* **2013**, *483*, 68–74. [[CrossRef](#)]
3. Radoszewski, J.; Starikovskaya, T. Streaming k -mismatch with error correcting and applications. *Inf. Comput.* **2020**, *271*, 104513. [[CrossRef](#)]
4. Jalsenius, M.; Porat, B.; Sach, B. Parameterized Matching in the Streaming Model. *STACS 2013*, 400–411. 2013.400. [[CrossRef](#)]
5. Gawrychowski, P.; Starikovskaya, T. Streaming Dictionary Matching with Mismatches. In Proceedings of the 30th Annual Symposium on Combinatorial Pattern Matching (CPM 2019), Pisa, Italy, 18–20 June 2019; pp. 21:1–21:15.
6. Gawrychowski, P.; Radoszewski, J.; Starikovskaya, T. Quasi-Periodicity in Streams. In Proceedings of the 30th Annual Symposium on Combinatorial Pattern Matching (CPM 2019), Pisa, Italy, 18–20 June 2019; pp. 22:1–22:14.
7. Bille, P.; Ettiienne, M.B.; Grossi, R.; Gørtz, I.L.; Rotenberg, E. Compressed Communication Complexity of Longest Common Prefixes. In *International Symposium on String Processing and Information Retrieval, Proceedings of the 25th International Symposium, SPIRE 2018, Lima, Peru, 9–11 October 2018*; Springer: Cham, Switzerland, 2018; pp. 74–87.
8. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343. [[CrossRef](#)]
9. Crochemore, M. Linear searching for a square in a word. *Bull. Eur. Assoc. Theor. Comput. Sci.* **1984**, *24*, 66–72.
10. Rytter, W. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.* **2003**, *302*, 211–222. [[CrossRef](#)]
11. Lagarde, G.; Perifel, S. Lempel-Ziv: A “one-bit catastrophe” but not a tragedy. In Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–10 January 2018; pp. 1478–1495.
12. Ziv, J.; Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory* **1978**, *24*, 530–536. [[CrossRef](#)]
13. Kociumaka, T.; Navarro, G.; Prezza, N. Towards a Definitive Measure of Repetitiveness. In *Latin American Symposium on Theoretical Informatics, Proceedings of the 14th Latin American Symposium, São Paulo, Brazil, 5–8 January 2020*; Springer: Cham, Switzerland, 2020; pp. 207–219.
14. Storer, J.A.; Szymanski, T.G. Data compression via textual substitution. *J. ACM* **1982**, *29*, 928–951. [[CrossRef](#)]
15. Kempa, D.; Prezza, N. At the roots of dictionary compression: string attractors. In Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, Los Angeles, CA, USA, 25–29 June 2018; pp. 827–840.
16. Mantaci, S.; Restivo, A.; Romana, G.; Rosone, G.; Sciortino, M. A combinatorial view on string attractors. *Theor. Comput. Sci.* **2021**, *850*, 236–248. [[CrossRef](#)]
17. Adelson-Velskii, G.; Landis, E. An algorithm for the organization of information. *Sov. Math. Dokl.* **1962**, *3*, 1259–1263.
18. Knuth, D.E. *The Art of Computer Programming*, 2nd ed.; Addison-Wesley: Boston, MA, USA, 1998; Volume III.