Compression by Substring Enumeration Using Sorted Contingency Tables

著者（英）	Takahiro OTA, Hiroyoshi MORITA, Akiko MANADA
journal or publication title	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences
volume	E103.A
number	6
page range	829-835
year	2020-06-01
URL	http://id.nii.ac.jp/1438/00009595/

doi: 10.1587/transfun.2019EAP1063

PAPER

Compression by Substring Enumeration Using Sorted Contingency Tables*

Takahiro OTA[†a)], ***Member***, **Hiroyoshi MORITA**[††b)], ***Senior Member***, ***and*** **Akiko MANADA**[†††c)], ***Member***

SUMMARY This paper proposes two variants of improved Compression by Substring Enumeration (CSE) with a finite alphabet. In previous studies on CSE, an encoder utilizes inequalities which evaluate the number of occurrences of a substring or a minimal forbidden word (MFW) to be encoded. The inequalities are derived from a contingency table including the number of occurrences of a substring or an MFW. Moreover, codeword length of a substring and an MFW grows with the difference between the upper and lower bounds deduced from the inequalities, however the lower bound is not tight. Therefore, we derive a new tight lower bound based on the contingency table and consequently propose a new CSE algorithm using the new inequality. We also propose a new encoding order of substrings and MFWs based on a sorted contingency table such that both its row and column marginal total are sorted in descending order instead of a lexicographical order used in previous studies. We then propose a new CSE algorithm which is the first proposed CSE algorithm using the new encoding order. Experimental results show that compression ratios of all files of the Calgary corpus in the proposed algorithms are better than those of a previous study on CSE with a finite alphabet. Moreover, compression ratios under the second proposed CSE get better than or equal to that under a well-known compressor for 11 files amongst 14 files in the corpus.

key words: *CSE, sorting, contingency table, lossless data compression*

1. Introduction

Dubé and Beaudoin proposed *Compression by Substring Enumeration* (CSE) [1], a two-stage lossless data compression algorithm with a binary source. CSE is a kind of enumerative code and encodes the number of occurrences of all substrings and *Minimal Forbidden Words* (MFWs) in the circular string of an input string. A set of MFWs, called *antidictionary*, is used in antidictionary coding [2], [3].

There have been previous studies on CSE. For example, its compression performance has been evaluated by computer simulations [1], [4], and these simulations show that the performance of CSE in [4] is better than that of a well-known data compression application. Indeed, the CSE in [4] gives the best compression performance amongst all variants of CSE. Moreover, Yokoo proposed a modified CSE which utilizes combined probabilistic models and proved the asymptotic optimality of the modified CSE [5]. Furthermore, Kanai et al. proposed a fast and memory-efficient array-based CSE [6].

The CSE algorithms shown above work only for binary alphabet, that is CSE over binary alphabet called binary CSE. As for finite CSE (CSE over q-ary alphabet with $q > 2$), it was first produced from antidictionary coding [7], [8]. In [7], an encoder of binary CSE is extended to that of finite CSE, and it is proven that an encoder of antidictionary coding and that of finite CSE are isomorphic. Moreover, both of the asymptotic optimality of antidictionary coding and the finite CSE are proven by extending Yokoo's results for $q = 2$ to those of $q > 2$. Iwata and Arimura modified in [9] the finite CSE and derived the maximum redundancy rate for the k-th order Markov sources. Furthermore, Sakuma et al. extended the array-based CSE and the binary CSE in [4] to those of finite CSE [10]. On the other hand, as for compression ratios, no experimental result of finite CSE is better than that of the original CSE in [1] (and clearly that of CSE in [4]) to the best of our knowledge.

This paper proposes two variants of improved finite CSE with respect to compression ratios. Previous studies on finite CSE utilize inequalities derived in [9] in encoding for providing a range of the number of occurrences of a substring and an MFW. Clearly, the difference between the upper and lower bounds of the inequalities has to be tight for better encoding. However, the lower bound is not tight.

In this paper, we derive a new inequality which derives a tighter lower bound, and present a new CSE algorithm using the new inequality. Moreover, for improving compression ratios, we propose a new encoding order of substrings and MFWs which are sorted by row and column marginal totals of the proper substrings and MFWs, while previous studies on finite CSE uses an encoding order of them sorted in lexicographical order. The second proposed CSE uses the new equality and the new order in encoding. We further examine the proposed CSE algorithms by computer simulations.

This paper is organized as follows. Section 2 gives the basic notations and definitions. Then, in Sect. 3, we review conventional CSE algorithms. In Sect. 4, we propose two new variants of CSE. Section 5 gives experimental results of the proposed algorithms for files of Calgary corpus. Section 6 summarizes our results.

Manuscript received May 9, 2019.
Manuscript revised January 9, 2020.
[†]The author is with School of Network and Information, Senshu University, Kawasaki-shi, 214-8580 Japan.
[††]The author is with Dept. of Computer and Network Engineering, Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu-shi, 182-8585 Japan.
[†††]The author is with Dept. of Information Science, Shonan Institute of Technology, Fujisawa-shi, 251-8511 Japan.
*This paper was presented in part at International Symposium on Information Theory and its Applications, Singapore, Oct. 2018 [16].
a) E-mail: ota@isc.senshu-u.ac.jp (Corresponding author)
b) E-mail: morita@uec.ac.jp
c) E-mail: amanada@info.shonan-it.ac.jp
DOI: 10.1587/transfun.2019EAP1063

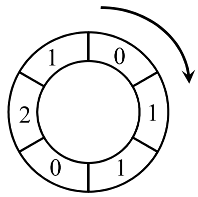


Fig. 1 The circular string of $\boldsymbol{x} = 011021$.

2. Basic Notations and Definitions

Let Σ be a finite source alphabet $\Sigma = \{0, 1, \ldots, J-1\}$ such that $0 < 1 < \cdots < J-1$. For a string $\boldsymbol{x}$ over Σ, let $|\boldsymbol{x}|$ be the length of the string; that is, $|\boldsymbol{x}| = n$ for $\boldsymbol{x} = x_1 \ldots x_n$. We denote by Σ^n the set of all strings of length n over Σ, and define the set of all finite strings over Σ to be $\Sigma^* = \cup_{n \geq 0} \Sigma^n$, including the *empty string* ϵ of length 0.

For $\boldsymbol{x} \in \Sigma^n$, $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{uvw}$ in terms of a concatenation of strings $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \Sigma^*$. In this case, $\boldsymbol{v}$ is called a *substring* of $\boldsymbol{x}$.

For $\boldsymbol{u}, \boldsymbol{v} \in \Sigma^k$ $(k \geq 1)$, $\boldsymbol{u}$ is said to be smaller than $\boldsymbol{v}$ in *lexicographical order* if and only if *i)* $u_1 < v_1$ or *ii)* $u_1 = v_1$ and $\boldsymbol{u}'$ is smaller than $\boldsymbol{v}'$ in lexicographical order, where $\boldsymbol{u} = u_1\boldsymbol{u}'$, $\boldsymbol{v} = v_1\boldsymbol{v}'$, and $u_1, v_1 \in \Sigma$.

For a string $\boldsymbol{w}$ of length $|\boldsymbol{w}| \geq 1$, $\boldsymbol{w}$ can be written as $a\boldsymbol{w}'$ and $\boldsymbol{w}''b$ for $a, b \in \Sigma$ and $\boldsymbol{w}', \boldsymbol{w}'' \in \Sigma^*$. Note that when $|\boldsymbol{w}| = 1$ holds, $a = b$ and $\boldsymbol{w}' = \boldsymbol{w}'' = \epsilon$. If $\boldsymbol{w}$ satisfies the following three conditions,

(a) $\boldsymbol{w}$ is not a substring of $\boldsymbol{x}$,

(b) $\boldsymbol{w}'$ is a substring of $\boldsymbol{x}$,

(c) $\boldsymbol{w}''$ is a substring of $\boldsymbol{x}$,

then $\boldsymbol{w}$ is called a *minimal forbidden word* (MFW) of $\boldsymbol{x} \in \Sigma^*$ [2], [8].

For a given $\boldsymbol{x} = x_1x_2 \ldots x_n \in \Sigma^n$, the string obtained by circularly concatenating the last symbol x_n and the first symbol x_1 is called the *circular string* of $\boldsymbol{x}$. Figure 1 shows the circular string of $\boldsymbol{x} = 011021$.

Let $C_{\boldsymbol{w}}(\boldsymbol{x})$ be the number of occurrences of a string $\boldsymbol{w} \in \Sigma^*$ in the circular string of $\boldsymbol{x}$, where $C_\epsilon(\boldsymbol{x})$ is defined to be $|\boldsymbol{x}|$ by convention. For convenience, we adopt the notation $C_{\boldsymbol{w}}$ instead of $C_{\boldsymbol{w}}(\boldsymbol{x})$. For example, for the circular string shown in Fig. 1, $C_\epsilon = 6, C_0 = 2, C_1 = 3, C_2 = 1$, $C_{10} = 2$, and $C_{\boldsymbol{w}} \leq 1$ otherwise. For a non-negative integer k and $\boldsymbol{v} \in \Sigma^*$, observe that

$$\sum_{\boldsymbol{w} \in \Sigma^k} C_{\boldsymbol{w}} = n, \tag{1}$$

$$C_{\boldsymbol{v}} = \sum_{a \in \Sigma} C_{a\boldsymbol{v}} = \sum_{b \in \Sigma} C_{\boldsymbol{v}b}. \tag{2}$$

3. Review of CSE

3.1 The Upper and Lower Bounds on C_{awb}

Given an upper bound and a lower bound on $C_{\boldsymbol{awb}}$, the difference between them is used to encode $C_{\boldsymbol{awb}}$. Consequently, the smaller the difference is, the fewer output bits of codeword of $C_{\boldsymbol{awb}}$ is. We use a table representation of Eq. (2) to explain the bounds because the formulas of the bounds shown in [9] are complicated. Fig. 2 depicts a table representation, called the $J \times J$ *contingency table* of $C_{\boldsymbol{cwd}}$ $(c, d \in \Sigma)$ for a given $\boldsymbol{w}$ and a fixed $\boldsymbol{awb}$. Any $\boldsymbol{v}$ such that $C_{\boldsymbol{v}}$ is within the thick lines implies that $\boldsymbol{v}$ is smaller than $\boldsymbol{awb}$ in lexicographical order.

							Row marginal total
	C_{0w0}	$\cdots$	C_{0wb}	$\cdots$	$C_{0w(J\text{-}1)}$		C_{0w}
	$\vdots$		$\vdots$		$\vdots$		$\vdots$
	C_{aw0}	$\cdots$	C_{awb}	$\cdots$	$C_{aw(J\text{-}1)}$		C_{aw}
	$\vdots$		$\vdots$		$\vdots$		$\vdots$
	$C_{(J\text{-}1)w0}$	$\cdots$	$C_{(J\text{-}1)wb}$	$\cdots$	$C_{(J\text{-}1)w(J\text{-}1)}$		$C_{(J\text{-}1)w}$
Column marginal total	C_{w0}	$\cdots$	C_{wb}	$\cdots$	$C_{w(J\text{-}1)}$		C_{w}

Fig. 2 A $J \times J$ contingency table of $C_{\boldsymbol{cwd}}$ $(c, d \in \Sigma)$ for a given $\boldsymbol{w}$ and a fixed $\boldsymbol{awb}$.

In the table in Fig. 2, the J elements of each row (resp. column) from the left (resp. top) are sorted in lexicographical ascending order. Moreover, the rightmost side (resp. bottom) element in the c-th row (resp. the d-th column) is the row (resp. column) marginal total ($C_{(c-1)\boldsymbol{w}} = \sum_{h=0}^{J-1} C_{(c-1)\boldsymbol{w}h}$) (resp. $C_{\boldsymbol{w}(d-1)} = \sum_{g=0}^{J-1} C_{g\boldsymbol{w}(d-1)}$) which corresponds to Eq. (2). Note that C_w is the grand total in Fig. 2.

The table in Fig. 2 contains 16 subtables separated by solid and thick lines. Note that the number of elements in a subtable may not be equal to that in the other subtable. Furthermore, we convert the $J \times J$ contingency table to the 3×3 simplified contingency table using a total of elements in a subtable. Fig. 3 shows the 3×3 simplified contingency table for the given $J \times J$ contingency table shown in Fig. 2, where elements in Fig. 3 are defined by Definition 1.

Definition 1 (Relationship between elements in contingency tables in Fig. 2 and Fig. 3)**.**

$$S_{11} = \sum_{\substack{c(<a)\in\Sigma,\\ d(<b)\in\Sigma}} C_{\boldsymbol{cwd}},\ S_{12} = \sum_{c(<a)\in\Sigma} C_{\boldsymbol{cwb}},\ S_{13} = \sum_{\substack{c(<a)\in\Sigma,\\ e(>b)\in\Sigma}} C_{\boldsymbol{cwe}},$$

$$S_{21} = \sum_{d(<b)\in\Sigma} C_{\boldsymbol{awd}},\ S_{23} = \sum_{e(>b)\in\Sigma} C_{\boldsymbol{awe}},$$

$$S_{31} = \sum_{\substack{f(>a)\in\Sigma,\\ d(<b)\in\Sigma}} C_{\boldsymbol{fwd}},\ S_{32} = \sum_{f(>a)\in\Sigma} C_{\boldsymbol{fwb}},\ S_{33} = \sum_{\substack{f(>a)\in\Sigma,\\ e(>b)\in\Sigma}} C_{\boldsymbol{fwe}},$$

$$S_{1.} = S_{11} + S_{12} + S_{13},\ S_{3.} = S_{31} + S_{32} + S_{33},$$

$$C_{\boldsymbol{aw}} = S_{21} + C_{\boldsymbol{awb}} + S_{23},\ C_{\boldsymbol{wb}} = S_{12} + C_{\boldsymbol{awb}} + S_{32},$$

$$S_{.1} = S_{11} + S_{21} + S_{31},\ S_{.3} = S_{13} + S_{23} + S_{33}.$$

				Row marginal total
	S_{11}	S_{12}	S_{13}	$S_{1.}$
	S_{21}	C_{awb}	S_{23}	C_{aw}
	S_{31}	S_{32}	S_{33}	$S_{3.}$
Column marginal total	$S_{.1}$	C_{wb}	$S_{.3}$	C_{w}

Fig. 3 The 3×3 simplified contingency table for a given $\boldsymbol{w}$ and a fixed $\boldsymbol{awb}$.

The inequalities used for encoding $C_{\boldsymbol{awb}}$, which are derived in [9], are given by

$$\begin{aligned} &\max(0, C_{\boldsymbol{aw}} - S_{21} - S_{.3}, C_{\boldsymbol{wb}} - S_{12} - S_{3.}) \\ &\leq C_{\boldsymbol{awb}} \leq \min(C_{\boldsymbol{aw}} - S_{21}, C_{\boldsymbol{wb}} - S_{12}). \end{aligned} \tag{3}$$

Note that the upper bound is given by $\min(C_{\boldsymbol{aw}} - S_{21}, C_{\boldsymbol{wb}} - S_{12}) - 1$ when $\boldsymbol{awb}$ is a repetition of a single letter; that is, $\boldsymbol{awb} = c \ldots c$ for $c \in \Sigma$ [9]. Let $N_{\boldsymbol{awb}}$ be the difference between the upper and lower bounds on $C_{\boldsymbol{awb}}$, that is,

$$\begin{aligned} N_{\boldsymbol{awb}} = \min\{&C_{\boldsymbol{aw}} - S_{21}, S_{.3}, C_{\boldsymbol{w}} - S_{1.} - C_{\boldsymbol{wb}} + S_{12} - S_{21}, \\ &C_{\boldsymbol{wb}} - S_{12}, S_{3.}, C_{\boldsymbol{w}} - S_{.1} - C_{\boldsymbol{aw}} + S_{21} - S_{12}\}. \end{aligned} \tag{4}$$

When $N_{\boldsymbol{awb}} = 0$, $C_{\boldsymbol{awb}}$ is calculable because the upper bound, the lower bound, and $C_{\boldsymbol{awb}}$ turn out to be equal.

3.2 Encoding and Decoding Algorithm

The encoding algorithm for CSE runs as follows.

Algorithm CSE Encoding
input : an input string $\boldsymbol{x} \in \Sigma^n$
output : the codeword $(E(n), \varepsilon(\mathrm{rank}(\boldsymbol{x})), e(\boldsymbol{x}))$
1 **begin**
/ Step 1:Encode the length of* $|\boldsymbol{x}| (= n)$ **/*
2 Output n encoded by an integer coding such as [11];
/ Step 2:Encode the rank of* $\boldsymbol{x}$ **/*
3 Output the rank of $\boldsymbol{x}$ encoded using $\lceil \log_2 n \rceil$ bits;
/ Step 3:Encode* C_a *for* $a \in \Sigma$ **/*
4 **for** $a := 0$ **to** $J - 2$ **do**
5   Output C_a encoded using $\lceil \log_2 n \rceil$ bits;
/ Step 4:Encode* $C_{\boldsymbol{awb}}$ *(Main Loop) */*
6 **for** $|\boldsymbol{w}| := 0$ **to** $n - 2$ **do** (s.t. $C_{\boldsymbol{w}} > 0$)
*/** $\boldsymbol{w}$ *is selected in lexicographical order */*
7   **for** $a := 0$ **to** $J - 1$ **do**
8   **for** $b := 0$ **to** $J - 1$ **do**
9   **if** $N_{\boldsymbol{awb}} > 0$
10   Output the encoding of $C_{\boldsymbol{awb}}$;
11 **end**.

We assume that an input string $\boldsymbol{x}$ consists of at least two kinds of symbols. In the algorithm, $E(n)$ represents the encoding of n by an integer coding such as [11]. The rank of $\boldsymbol{x}$ denoted by $\mathrm{rank}(\boldsymbol{x})$ in Step 3 (line 5 in the algorithm) represents the number of strings of length $|\boldsymbol{x}|$ which appear within the circular string of $\boldsymbol{x}$ and are smaller than $\boldsymbol{x}$ in lexicographical order. The rank is used to retrieve $\boldsymbol{x}$ from the substrings in decoding, and $\varepsilon(\mathrm{rank}(\boldsymbol{x}))$ represents the encoding of $\mathrm{rank}(\boldsymbol{x})$. Moreover, $e(\boldsymbol{x})$ represents a sequence of encoded C_a and $C_{\boldsymbol{awb}}$ in encoding order.

There are some variations for encoding $C_{\boldsymbol{awb}}$. For example, methods [1], [4], [6] assign a probability to $C_{\boldsymbol{awb}}$, and the probability is encoded by an entropy coding. The method in [9] assigns a probability to the sequence of all $C_{c\boldsymbol{w}d}$ for $c, d \in \Sigma$ and fixed $\boldsymbol{w}$, that is, the $J \times J$ contingency table except the row and column marginal totals. Moreover, the uniform distribution [1], [6] and a combination of the uniform distribution and the hypergeometric distribution [9] are used as a probabilistic model.

In any conventional CSE with a finite alphabet, inequalities (3) play a key role for encoding $C_{\boldsymbol{awb}}$ because the codeword length of $C_{\boldsymbol{awb}}$ increases as the difference between the upper and lower bounds of (3) grows. Therefore, by tightening the upper or lower bounds, performance on compression ratios of any CSE with a finite alphabet can be improved. We will tighten the lower bound and improve the difference in Sect. 4.

Next, we show the decoding algorithm for CSE. The decoding algorithm is a simplified algorithm shown in [8]. In the algorithm, $\mathcal{W}$ is a set of all substrings of $\boldsymbol{x}$.

Algorithm CSE Decoding
input : a codeword $(E(n), \varepsilon(\mathrm{rank}(\boldsymbol{x})), e(\boldsymbol{x}))$
output : the decoded input source $\boldsymbol{x}$
1 **begin**
2 */* Step 0: Initialize */*
3 $\mathcal{W} \leftarrow \{\epsilon\}$;
4 */* Step 1: Decode* n **/*
5 Decode n from $E(n)$;
6 */* Step 2: Decode* $\mathrm{rank}(\boldsymbol{x})$ **/*
7 Decode $\mathrm{rank}(\boldsymbol{x})$ from $\varepsilon(\mathrm{rank}(\boldsymbol{x}))$;
8 */* Step 3: Decode* C_a *for* $a \in \Sigma$ **/*
9 **for** $a := 0$ **to** $J - 2$ **do**
10   Decode C_a from $e(\boldsymbol{x})$;
11 $C_{(J-1)} \leftarrow n - \sum_{j=0}^{J-2} C_j$;
12 $\mathcal{W} \leftarrow \mathcal{W} \cup \{a \in \Sigma : C_a > 0\}$;
13 */* Step 4: Decode* $C_{\boldsymbol{awb}}$ *for* $\boldsymbol{w} \in \Sigma^*$ **/*
14 **for** $|\boldsymbol{w}| := 0$ **to** $n - 2$ **do** (s.t. $\boldsymbol{w} \in \mathcal{W}$)
*/** $\boldsymbol{w}$ *is selected in lexicographical order */*
15   **for** $a := 0$ **to** $J - 1$ **do**
16   **for** $b := 0$ **to** $J - 1$ **do**
17   **if** $N_{\boldsymbol{awb}} > 0$
18   Decode $C_{\boldsymbol{awb}}$ from $e(\boldsymbol{x})$;
19   **else**
20         $C_{\boldsymbol{awb}} \leftarrow \min(C_{\boldsymbol{aw}} - S_{21}, C_{\boldsymbol{wb}} - S_{12})$;
21   **if** $C_{\boldsymbol{awb}} > 0$

```
        𝒲 ← 𝒲 ∪ {awb};
  /* Step 5: Decode x */
  x is selected in {w ∈ 𝒲 : |w| = n} by rank(x);
  return x;
end.
```

4. Proposed CSE Algorithms

4.1 Tightening the Lower Bound

When $C_{a\boldsymbol{w}b}$ is encoded under CSE Encoding, all the elements in the thick lines in the table in Fig. 2 have been already encoded because encoding $C_{a\boldsymbol{w}b}$ is implemented in string length ascending order of $\boldsymbol{w}$ and lexicographical order for a given $\boldsymbol{w}$. In other words, any element in the thick lines can be used to determine the lower and upper bounds on $C_{a\boldsymbol{w}b}$.

However, even though there are 11 elements in the thick lines in the table in Fig. 3, the bounds in (3) are determined by only six elements amongst them; that is, $C_{a\boldsymbol{w}}, S_{21}, S_{.3}, C_{\boldsymbol{w}b}, S_{12}$, and $S_{3.}$. Therefore, in this subsection, we propose a new inequality for $C_{a\boldsymbol{w}b}$, as denoted in Proposition 1 below, to tighten the lower bound of (3) by efficiently utilizing the remaining four elements $S_{11}, S_{13}, S_{1.}$, and $S_{.1}$ except for $C_{\boldsymbol{w}}$. We remark that S_{31} is not an element in the thick lines but is calculable by elements $S_{.1}, S_{11}$, and S_{21} because $S_{31} = S_{.1} - S_{11} - S_{21}$.

Proposition 1. *For a given $a\boldsymbol{w}b$,*

$$\max(0, C_{a\boldsymbol{w}} - S_{21} - S_{.3} + S_{13}, C_{\boldsymbol{w}b} - S_{12} - S_{3.} + S_{31}) \leq C_{a\boldsymbol{w}b} \leq \min(C_{a\boldsymbol{w}} - S_{21}, C_{\boldsymbol{w}b} - S_{12}).$$

Proof. The upper bound is the same as that of (3), so we focus on showing

$$\max(0, C_{a\boldsymbol{w}} - S_{21} - S_{.3} + S_{13}, C_{\boldsymbol{w}b} - S_{12} - S_{3.} + S_{31}) \leq C_{a\boldsymbol{w}b}.$$

From the table in Fig. 3, $C_{a\boldsymbol{w}b}$ satisfies

$$C_{a\boldsymbol{w}} - S_{21} - S_{23} = C_{a\boldsymbol{w}b}, \tag{5}$$

$$C_{\boldsymbol{w}b} - S_{12} - S_{32} = C_{a\boldsymbol{w}b}. \tag{6}$$

Since all elements are non-negative, S_{23} and $S_{.3} - S_{13}$ satisfy the inequality (7), and S_{32} and $S_{3.} - S_{31}$ satisfy the inequality (8)

$$S_{23} \leq S_{23} + S_{33} = S_{.3} - S_{13}, \tag{7}$$

$$S_{32} \leq S_{32} + S_{33} = S_{3.} - S_{31}. \tag{8}$$

Replacing S_{23} in (5) with $S_{.3} - S_{13}$, and S_{32} in (6) with $S_{3.} - S_{31}$, we have

$$C_{a\boldsymbol{w}} - S_{21} - S_{.3} + S_{13} \leq C_{a\boldsymbol{w}b}, \tag{9}$$

$$C_{\boldsymbol{w}b} - S_{12} - S_{3.} + S_{31} \leq C_{a\boldsymbol{w}b}, \tag{10}$$

and therefore, we obtain

Local row marginal total

C_{awb}	S_{23}	C_{aw} - S_{21}
S_{32}	S_{33}	$S_{3.}$ - S_{31}
C_{wb} - S_{12}	$S_{.3}$ - S_{13}	$C_{awb} + S_{23} + S_{32} + S_{33}$

Local column marginal total

Fig. 4 A local contingency table for $C_{a\boldsymbol{w}b}$, S_{23}, S_{32}, and S_{33}.

$$\max(0, C_{a\boldsymbol{w}} - S_{21} - S_{.3} + S_{13}, C_{\boldsymbol{w}b} - S_{12} - S_{3.} + S_{31}) \leq C_{a\boldsymbol{w}b}$$

as required. □

Observe that the lower bound in Proposition 1 gives a better bound of $C_{a\boldsymbol{w}b}$ than that in (3). Furthermore, when $S_{13} > 0$ and $S_{31} > 0$, the lower bound in (3) is strictly lower than $C_{a\boldsymbol{w}b}$ since (9) and (10) can be written by

$$C_{a\boldsymbol{w}} - S_{21} - S_{.3} < C_{a\boldsymbol{w}} - S_{21} - S_{.3} + S_{13} \leq C_{a\boldsymbol{w}b},$$

$$C_{\boldsymbol{w}b} - S_{12} - S_{3.} < C_{\boldsymbol{w}b} - S_{12} - S_{3.} + S_{31} \leq C_{a\boldsymbol{w}b}.$$

The difference $\tilde{N}_{a\boldsymbol{w}b}$ between the upper and lower bounds of $C_{a\boldsymbol{w}b}$ in Proposition 1 is given by

$$\begin{aligned}\tilde{N}_{a\boldsymbol{w}b} = \min\{ & C_{a\boldsymbol{w}} - S_{21}, S_{.3} - S_{13}, C_{\boldsymbol{w}} - S_{1.} - C_{\boldsymbol{w}b} + S_{12} - S_{21} - S_{31}, \\ & C_{\boldsymbol{w}b} - S_{12}, S_{3.} - S_{31}, C_{\boldsymbol{w}} - S_{.1} - C_{a\boldsymbol{w}} + S_{21} - S_{12} - S_{13}\} \\ = \min\{ & C_{a\boldsymbol{w}} - S_{21}, S_{.3} - S_{13}, C_{\boldsymbol{w}b} - S_{12}, S_{3.} - S_{31}\}.\end{aligned} \tag{11}$$

We explain Eq. (11) in detail by using Fig. 4 which shows a local contingency table for $C_{a\boldsymbol{w}b}$, S_{23}, S_{32}, and S_{33}. Four values shown in local column and row marginal totals except $C_{a\boldsymbol{w}b} + S_{23} + S_{32} + S_{33}$ are the same with the four values in the last formula in (11). Moreover, if one of four values $C_{a\boldsymbol{w}b}$, S_{23}, S_{32}, and S_{33} is known, then the other three values are calculable because local column and row marginal totals are known in Fig. 4. Therefore, the difference between the upper and the lower bounds of $C_{a\boldsymbol{w}b}$ is equal to that of S_{23}, S_{32}, and S_{33}. Hence, we can obtain the difference between the upper and lower bounds of $C_{a\boldsymbol{w}b}$ by $\tilde{N}_{a\boldsymbol{w}b}$.

For a binary source alphabet $\Sigma = \{0, 1\}$ and $a = b = 0$, Eq. (11) is given by $\min\{C_{0\boldsymbol{w}}, C_{1\boldsymbol{w}}, C_{\boldsymbol{w}0}, C_{\boldsymbol{w}1}\}$ which is equal to the difference between an upper and a lower bound shown in Eq. (7) in [5] where $C_{a\boldsymbol{w}b} = C_{0\boldsymbol{w}0}$, $S_{23} = C_{0\boldsymbol{w}1}$, $S_{32} = C_{1\boldsymbol{w}0}$, $S_{33} = C_{1\boldsymbol{w}1}$, $S_{.3} = C_{\boldsymbol{w}1}$, $S_{3.} = C_{1\boldsymbol{w}}$, $S_{21} = S_{13} = S_{12} = S_{31} = 0$, and $C_{\boldsymbol{w}} = C_{a\boldsymbol{w}b} + S_{23} + S_{32} + C_{33}$ in Fig. 4. Therefore, the proposed techniques shown in Sects. 4.1 and 4.2 cannot improve the difference for a binary alphabet. On the other hand, the techniques are effective for a q-ary alphabet with $q > 2$.

							Row marginal total
	$C_{0'w0''}$	$\cdots$	$C_{0'wb''}$	$\cdots$	$C_{0'w(J\text{-}1)''}$		$C_{0'w}$
	$\vdots$		$\vdots$		$\vdots$		$\vdots$
	$C_{a'w0''}$	$\cdots$	$C_{a'wb''}$	$\cdots$	$C_{a'w(J\text{-}1)''}$		$C_{a'w}$
	$\vdots$		$\vdots$		$\vdots$		$\vdots$
	$C_{(J\text{-}1)'w0''}$	$\cdots$	$C_{(J\text{-}1)'wb''}$	$\cdots$	$C_{(J\text{-}1)'w(J\text{-}1)''}$		$C_{(J\text{-}1)'w}$
Column marginal total	$C_{w0''}$	$\cdots$	$C_{wb''}$	$\cdots$	$C_{w(J\text{-}1)''}$		C_w

Fig. 5 A $J \times J$ sorted contingency table of $C_{c'\boldsymbol{w}d''}$ $(c', d'' \in \Sigma)$ for a given $\boldsymbol{w}$ and a fixed $a'\boldsymbol{w}b''$ such that $C_{0'\boldsymbol{w}} \geq \cdots \geq C_{(J-1)'\boldsymbol{w}}$ and $C_{\boldsymbol{w}0''} \geq \cdots \geq C_{\boldsymbol{w}(J-1)''}$.

4.2 Improving the Difference $\tilde{N}_{a\boldsymbol{w}b}$ Using a Contingency Table with Sorted Marginal Total

We give a new encoding order of strings $a\boldsymbol{w}b$ for improving the compression ratio while the conventional CSE uses the lexicographical order such as $0\boldsymbol{w}0, \ldots, (J-1)\boldsymbol{w}(J-1)$ during encoding. The difference $\tilde{N}_{a\boldsymbol{w}b}$ depends on the order of strings in encoding because values of $S_{21}, S_{.3}, S_{13}, S_{12}$, and S_{31} also depend on the order. We focus on $S_{3.}$ and $S_{.3}$ for reducing $\tilde{N}_{a\boldsymbol{w}b}$ because $S_{3.}$ (resp. $S_{.3}$) is sum of wide range of the row (resp. col.) total. Hence, an order that makes $S_{3.}$ and $S_{.3}$ small derives a small value for $\tilde{N}_{a\boldsymbol{w}b}$. Note that the value may not be the minimum amongst all possible values for the differences. For reducing $S_{3.}$ (resp. $S_{.3}$), we sort elements of the row (resp. col.) marginal totals such that

$$C_{0'\boldsymbol{w}} \geq C_{1'\boldsymbol{w}} \geq \cdots \geq C_{a'\boldsymbol{w}} \geq \cdots \geq C_{(J-1)'\boldsymbol{w}}, \tag{12}$$

$$C_{\boldsymbol{w}0''} \geq C_{\boldsymbol{w}1''} \geq \cdots \geq C_{\boldsymbol{w}b''} \geq \cdots \geq C_{\boldsymbol{w}(J-1)''}, \tag{13}$$

where $c' < d'$ when $C_{c'\boldsymbol{w}} = C_{d'\boldsymbol{w}}$ and $e'' < f''$ when $C_{\boldsymbol{w}e''} = C_{\boldsymbol{w}f''}$ for $c', d', e'', f'' \in \Sigma$.

Figure 5 depicts the $J \times J$ contingency table such that elements of row (resp. col.) marginal total are sorted in descending order from the top (resp. the left) for a given $\boldsymbol{w}$ and a fixed $a'\boldsymbol{w}b''$. Roughly speaking, elements having large values tend to be gathered around the top-left in the table in Fig. 5 while elements having small values such as zero tend to be gathered around the bottom-right in the table in Fig. 5.

By using the table in Fig. 5, we obtain the second proposed algorithm by using $(0', \ldots, (J-1)')$ instead of $(0, \ldots, J-1)$ in lines 3 and 7 of CSE Encoding, and $(0'', \ldots, (J-1)'')$ is used instead of $(0, \ldots, J-1)$ in line 8 of the encoding. Note that we build the table shown in Table 5 before $C_{0'\boldsymbol{w}0''}$ is encoded. Elements of row (resp. column) marginal total in the converted table are sorted in descending order, so that $S_{3.}$ (resp. $S_{.3}$) in the table is smaller than or equal to that in an unsorted contingency table.

Table 1 Compression ratios of the conventional CSE and the proposed algorithms with a finite alphabet for the Calgary corpus.

File	Conventional CSE $J = 256$	Proposed CSE-P $J = 256$	Proposed CSE-PS $J = 256$
bib	0.27	0.26	0.24
book1	0.35	0.33	0.30
book2	0.29	0.28	0.25
geo	0.69	0.67	0.58
news	0.37	0.35	0.31
obj1	0.58	0.56	0.51
obj2	0.35	0.33	0.31
paper1	0.35	0.33	0.31
paper2	0.35	0.33	0.30
pic	0.13	0.12	0.12
progc	0.36	0.33	0.31
progl	0.23	0.22	0.21
progp	0.24	0.22	0.21
trans	0.21	0.19	0.19

5. Experimental Results

Table 1 shows compression ratios of the Calgary corpus [12] under the conventional CSE using (3) (conventional CSE), a proposed CSE using Proposition 1 (Proposed CSE-P), and a proposed CSE using Proposition 1 and a sorting contingency table (Proposed CSE-PS). Note that the three algorithms execute over a one-byte alphabet ($J = 256$). The compression ratio is given as the ratio of the compressed file size and its original file size. All algorithms encode the probability assigned to $C_{a\boldsymbol{w}b}$ by an adaptive entropy coding such as adaptive arithmetic coding of order-0 [13], sequentially. The algorithms first use the uniform distribution and update a non-negative frequency based upon the difference between $C_{a\boldsymbol{w}b}$ and the lower bound of Proposition 1.

As shown in Table 1, compression ratios get improved for all files using the proposed algorithms. In particular, the compression ratio of CSE-PS for the file (geo) is 11% better than that of the conventional CSE. Moreover, CSE-PS derives better compression ratios than CSE-P for 12 files amongst 14 files.

Table 2 shows compression ratios for the proposed CSE-PS, two conventional CSE algorithms [1], [4], and an well-known data compression application (bzip2) [14] using the Burrows-Wheeler transformation [15]. The proposed CSE-PS and bzip2 execute compression over a one-byte alphabet ($J = 256$) while the two conventional CSE algorithms do over a binary alphabet ($J = 2$). The conventional CSE [1], called BTF, encodes in a way similar to our proposed encoding. More precisely, the CSE (BTF) encodes the probability of $C_{a\boldsymbol{w}b}$, sequentially, by an adaptive entropy coding. However, the details are not described in [1]. The conventional CSE [4], called EPA, is the best CSE with respect to compression ratio.

As shown in Table 2, compression ratios under the proposed CSE-PS get better than those under the CSE (BTF) for 12 files and those under bzip2 for 11 files. These results show good performance on compression ratios for the

Table 2 Compression Ratios of the proposed CSE-PS, conventional CSE with a binary alphabet [1], [4], and a well-known data compression application (bzip2) [14].

file	Proposed CSE-PS $J = 256$	CSE [1] (BTF) $J = 2$	CSE [4] (EPA) $J = 2$	bzip2 [14] $J = 256$
bib	0.24	0.25	0.23	0.25
book1	0.30	0.28	0.28	0.30
book2	0.25	0.25	0.24	0.26
geo	0.58	0.69	0.57	0.56
news	0.31	0.32	0.30	0.31
obj1	0.51	0.56	0.49	0.50
obj2	0.31	0.34	0.31	0.31
paper1	0.31	0.32	0.30	0.31
paper2	0.30	0.30	0.29	0.30
pic	0.12	0.10	0.10	0.10
progc	0.31	0.33	0.30	0.32
progl	0.21	0.21	0.20	0.22
progp	0.21	0.22	0.21	0.22
trans	0.19	0.20	0.18	0.19

proposed CSE-PS.

However, compression ratios for 12 files under the proposed CSE-PS do not overcome those under the CSE (EPA); the maximum difference is 2% (geo and pic). To improve compression ratios, a technique using divided blocks instead of a whole file [1] can be applied to the proposed CSE-PS.

6. Conclusion

In this paper, we proposed a new inequality which derives a tighter lower bound on the number of occurrences of a substring and an MFW to be encoded. We then proposed a new CSE algorithm using the new inequality. Moreover, for improving compression ratios, we proposed a new encoding order of substrings and MFWs which are sorted by row and column marginal totals of the proper substrings and MFWs, instead of lexicographical order used in previous studies. We also proposed a new CSE algorithm which combines the first proposed CSE algorithm and the new sorted encoding order.

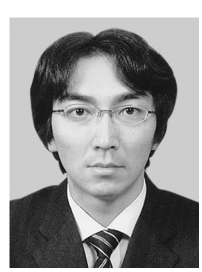
Experimental results showed that for all files on Calgary corpus, the proposed CSE algorithms exhibited better compression ratios than those of a previous study on CSE with a finite alphabet. Moreover, the proposed CSE using the new inequality and sorted encoding order gave better compression ratios for 11 files amongst 14 files in the corpus, compared with a well-known data compression application (bzip2).

Acknowledgments

The authors truly thank the anonymous reviewers for their valuable comments. This work was supported by JSPS KAKENHI Grant Numbers JP17K00147, JP17K00400, and JP19K11833.

References

[1] D. Dubé and V. Beaudoin, "Lossless data compression via substring enumeration," Proc. Data Compression Conference 2010, pp.229–238, March 2010.
[2] M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi, "Data compression using antidictionaries," Proc. IEEE, vol.88, no.11, pp.1756–1768, Nov. 2000.
[3] T. Ota and H. Morita, "On the adaptive antidictionary code using minimal forbidden words with constant lengths," Proc. International Symposium on Information Theory and its Applications, pp.72–77, Oct. 2010.
[4] M. Béliveau and D. Dubé, "Improving compression via substring enumeration by explicit phase awareness," Proc. Data Compression Conference 2014, pp.26–28, March 2014.
[5] H. Yokoo, "Asymptotic optimal lossless compression via the CSE technique," Proc. Data Compression, Communications and Processing, pp.11–18, June 2011.
[6] S. Kanai, H. Yokoo, K. Yamazaki, and H. Kaneyasu, "Efficient implementation and empirical evaluation of compression by substring enumeration," IEICE Trans. Fundamentals, vol.E99-A, no.2, pp.601–611, Feb. 2016.
[7] T. Ota and H. Morita, "On a universal antidictionary coding for stationary ergodic sources with finite alphabet," Proc. International Symposium on Information Theory and its Applications, pp.294–298, Oct. 2014.
[8] T. Ota and H. Morita, "A compact tree representation of an antidictionary," IEICE Trans. Fundamentals, vol.E100-A, no.9, pp.1973–1984, Sept. 2017.
[9] K. Iwata and M. Arimura, "Lossless data compression via substring enumeration for k-th order Markov sources with a finite alphabet," IEICE Trans. Fundamentals, vol.E99-A, no.12, pp.2130–2135, Dec. 2016.
[10] S. Sakuma, K. Narisawa, and A. Shinohara, "Generalization of efficient implementation of compression by substring enumeration," Proc. Data Compression Conference, p.630, March 2016.
[11] P. Elias, "Universal codeword sets and representations of the integers," IEEE Trans. Inf. Theory, vol.IT-21, no.2, pp.194–203, 1975.
[12] The Calgary text compression corpus. ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/
[13] A. Moffat and A. Turpin, Compression and Coding Algorithms, Kluwer Academic Publishers, 2002.
[14] bzip2, http://www.bzip2.org
[15] M. Burrows and D. Wheeler, "A block-sorting lossless data compression algorithm," SRC Research Report, pp.73–93, May 1994.
[16] T. Ota, H. Morita, and A. Manada, "Compression by substring enumeration with a finite alphabet using sorting," Proc. International Symposium on Information Theory and its Applications, pp.587–591, Oct. 2018.



Takahiro Ota received the B.E. and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 1993 and 2007, respectively. In 1997, he joined Nagano Prefectural Institute of Technology, Nagano, Japan, first as a Lecturer at the Department of Electronic Engineering, where from 2009, he was an Associate Professor. Since 2012, he was an Associate Professor with the Department of Computer and Systems Engineering, where from 2019, he was a Professor. In 2020, he joined at School of Network and Information, Senshu University, as an Associate Professor. His current research interests are in information theory, source coding, and bio-informatics.

Hiroyoshi Morita received the B.E., M.E., and D.E. degrees from Osaka University, Osaka, Japan, in 1978, 1980 and 1983, respectively. In 1983, he joined Toyohashi University of Technology, Aichi, Japan as a Research Associate in the School of Production System Engineering. In 1990, he joined the University of Electro-Communications, Tokyo, Japan, first an Assistant Professor at the Department of Computer Science and Information Mathematics, where from 1992, he was an Associate Professor. From 1995, he was with the Graduate School of Information Systems, where from 2005, he was a Professor. From 2017, he has been with the Graduate School of Informatics and Engineering at the University of Electro-Communications. He was a Visiting Fellow at the Institute of Experimental Mathematics, University of Essen, Essen, Germany during 1993–1994. His research interests are in combinatorial theory, information theory, and coding theory, with applications to the digital communication systems.

Akiko Manada received the M.S. degree from Tsuda College, Tokyo, Japan, in 2004, and the Ph.D. degree in Mathematics from Queen's University, Kingston, Canada, in 2009. She then worked at Claude Shannon Institute at University College Dublin, Ireland, as a postdoctoral fellow from 2009 through 2011. She was an assistant professor at the University of Electro-Communications, Tokyo, Japan from 2012 to 2018. Since April 2018, she has been a lecturer at Dept. of Information Science, Shonan Institute of Technology, Kanagawa, Japan. Her research interests are discrete mathematics (especially in graph theory) and its applications towards coding theory.