Asymptotic Optimal Lossless Compression via the CSE Technique

Hidetoshi Yokoo
Department of Computer Science
Gunma University
Kiryu, 376-8515 Japan
yokoo@cs.gunma-u.ac.jp

***Abstract*—A novel lossless compression algorithm known as compression by substring enumeration (CSE) is analyzed and modified. The CSE compression algorithm is a block-based, off-line method, as is the case with enumerative codes and the block-sorting compression scheme. First, we propose an encoding model that achieves asymptotic optimality for stationary ergodic sources. The codeword length attained by the proposed model converges almost surely to the entropy rate of a source when the length of a string generated by the source tends to infinity. Then, we propose a novel decoding algorithm that requires fewer codewords than the original CSE.**

***Keywords*-Asymptotic optimality; BWT; CSE; data compression; lossless; universal codes;**

I. Introduction

Compression by substring enumeration (CSE), proposed by Dubé and Beaudoin [4], is a relatively new universal lossless compression algorithm. Although it has not been investigated thoroughly, initial experiments [4], [3] show that it is a potential alternative to existing high-performance compression methods. Some similarities between CSE and other existing methods have been reported; however, they have not been analyzed in detail. CSE has significant potential from the viewpoint of both research and practice. In this paper, we propose an appropriate coding model for CSE to achieve asymptotic optimality for stationary ergodic sources. In addition, we introduce a novel decoding algorithm that requires fewer codewords than the original CSE for decoding the source data.

The CSE compression algorithm operates only on binary data, and it performs compression by encoding the number of occurrences of every distinct substring of data, enumerating the substrings from the shortest to the longest. It is similar to *enumerative coding* [2] in that both enumerate the number of symbols or substrings of data. However, CSE is quite different in that it repeats the enumeration process until it uniquely identifies the input data.

CSE assumes the data to be circular, and it uses the technique described above to encode an equivalence class of strings under rotation. Then, it encodes the correct rotation of data as the rank or the lexicographic order in the set of all rotations. In this respect, CSE is quite similar to the block-sorting compression scheme based on the Burrows–Wheeler Transform (BWT) [1]. However, the BWT outputs the right-most column of the BWT-transformed matrix whereas CSE outputs the information on the matrix itself from the first column. For circular data, Shields [9] showed the existence of universal codes by introducing the *circular k-type*, which is the relative frequency of a substring of length k, counted on a circular string. We actually use the circular k-type to bound the length of the codeword produced by CSE. However, the code proposed by Shields is essentially an enumerative code, and it is different from CSE.

In this paper, we first review the basics of CSE in Section II. In section III, we describe the proposed model, which predicts and encodes the number of occurrences of a substring in the CSE framework; in addition we prove its asymptotic optimality for stationary ergodic sources. In spite of its asymptotic optimality, the original CSE is inefficient, as will be shown in Section IV. In the section, we first present an example to show that the original CSE tends to enumerate more substrings than are necessary. To overcome this drawback, we focus on the similarity between CSE and the BWT. We apply a finite-order variant [6], [7], [11] of the BWT to the development of a novel decoder that makes fuller use of the circularity of an input string. We show that the proposed decoder is more efficient than the original decoder in most cases. In Section V, we briefly discuss future work related to the current study.

Throughout the paper, all logarithms are taken to the base 2.

II. Compression by Substring Enumeration (CSE)

We consider the lossless compression of a binary string of length N. We represent the data string by $\mathbf{D} \in \{0,1\}^+$ and its length by $N = |\mathbf{D}|$. In this paper, $|\cdot|$ denotes the length of a binary string or the size of a set, depending on the context. When we refer to a particular element in $\mathbf{D}$, we regard $\mathbf{D}$ as an N-dimensional array, i.e., $\mathbf{D}[0..N-1] = \mathbf{D}[0]\cdots\mathbf{D}[N-1]$. Other vectors and matrices are represented as arrays with indices beginning from 1.

The CSE encoder converts $\mathbf{D}$ into two components: its equivalence class of strings under rotation and its rank in the class. In the literature, such an equivalence class of strings is known as a *necklace* [8]. We identify each necklace by the lexicographically smallest string in its equivalence class.

978-0-7695-4528-8/11 $26.00 © 2011 IEEE
DOI 10.1109/CCP.2011.32

Let C_w denote the number of occurrences of a substring $w \in \{0,1\}^*$ in $\mathbf{D}$; we define $C_\epsilon = N$ for the empty string ϵ. Since we assume that $\mathbf{D}$ is circular, we have the following *compatibility conditions*:

$$\begin{aligned} C_w &= C_{w0} + C_{w1} \\ &= C_{0w} + C_{1w} \quad \text{for any } w \in \{0,1\}^*. \end{aligned} \tag{1}$$

We also have

$$\sum_{w \in \{0,1\}^k} C_w = N \text{ for any } k = 0, 1, \ldots. \tag{2}$$

From Eq. (1), we can derive

$$C_{0w1} = C_{0w} - C_{0w0}, \tag{3}$$
$$C_{1w0} = C_{w0} - C_{0w0}, \tag{4}$$
$$C_{1w1} = C_{w1} - C_{0w1}, \tag{5}$$

which are used to compute each quantity on the left-hand sides. We combine these equations with $C_{0w0} \geq 0$, $C_{0w1} \geq 0$, $C_{1w0} \geq 0$, and $C_{1w1} \geq 0$ to obtain

$$\max(0, C_{0w} - C_{w1}) \leq C_{0w0} \leq \min(C_{0w}, C_{w0}). \tag{6}$$

When we already have C_{0w}, C_{w0}, and C_{w1}, we can efficiently encode C_{0w0} by using the bounds (6)[1]. The range of possible values of C_{0w0} is given by

$$\begin{aligned} &\min(C_{0w}, C_{w0}) - \max(0, C_{0w} - C_{w1}) + 1 \\ &\leq \min(C_{0w}, C_{1w}, C_{w0}, C_{w1}) + 1. \end{aligned} \tag{7}$$

We can summarize these observations into the following CSE compression algorithm.

1) /* Encode the string length */
   Encode N;
2) /* Encode the number of zeros */
   $i := 1$; **Encode** C_0;
3) /* Main loop for encoding the necklace */
   For $l := 2$ **to** N **do**
   For every $w \in \mathbf{D}$ such that $|w| = l - 2$ **and** $\min(C_{0w}, C_{1w}, C_{w0}, C_{w1}) \geq 1$ **do**
   $i := i + 1$; **Encode** C_{0w0};
4) /* Encode the rank of the string */
   Encode rank($\mathbf{D}$);

The counter variable i is unnecessary for encoding, and it is simply introduced for explanation. In the main loop, possible substrings for w are selected from $\{0,1\}^*$ from the shortest ($= \epsilon$) to the longest (see Lemma 2, below), in lexicographic order. Therefore, the length of the ith substring (0 for $i = 1$, and $0w0$ for $i \geq 2$) whose number of occurrences is actually encoded is greater than or equal to $\lceil \log i \rceil + 1$. Thus, we have

$$\log i + 1 \leq \lceil \log i \rceil + 1 \leq |0w0| \tag{8}$$

in the main loop.

In order to upperbound the maximal i or the maximum number of outputted numbers, we introduce the following two sets of strings.

$$\begin{aligned} U(\mathbf{D}) &= \{w \mid \text{Both } w0 \text{ and } w1 \text{ occur in } \mathbf{D}\}, \\ V(\mathbf{D}) &= \{w \mid \text{Both } 0w \text{ and } 1w \text{ occur in } \mathbf{D}\}. \end{aligned}$$

For example, we have, for $\mathbf{D} = 00000101$,

$$\begin{aligned} U(\mathbf{D}) &= \{\epsilon, 0, 00, 10, 000, 010, 0000\}, \\ V(\mathbf{D}) &= \{\epsilon, 0, 00, 01, 000, 010, 0000\}. \end{aligned} \tag{9}$$

The main loop of the CSE algorithm encodes C_{0w0} only for $w \in U(\mathbf{D}) \cap V(\mathbf{D})$. We can easily show the following bounds [5].

Lemma 1: For a string $\mathbf{D}$ of length N, we have $|U(\mathbf{D})| \leq N - 1$ and $|V(\mathbf{D})| \leq N - 1$. The equalities hold for a non-repetitive (aperiodic) $\mathbf{D}$.

From this lemma, it follows that the numbers encoded using CSE do not exceed the string length. Furthermore, we can prove the following.

Lemma 2: For a string $\mathbf{D}$, the longest substrings in $U(\mathbf{D})$ and $V(\mathbf{D})$ are the same, and therefore, they are included in $U(\mathbf{D}) \cap V(\mathbf{D})$.

Proof: Let w_{M} be the longest substring in $V(\mathbf{D})$. By the definition of $V(\mathbf{D})$, both $0w_{\mathrm{M}}$ and $1w_{\mathrm{M}}$ occur in $\mathbf{D}$. If their following bits are the same, w_{M} is no longer the longest substring in $V(\mathbf{D})$. Therefore, w_{M} is followed by both 0 and 1 in $\mathbf{D}$. This means that $w_{\mathrm{M}} \in U(\mathbf{D})$. Similarly, we can show that the longest substring in $U(\mathbf{D})$ is also in $V(\mathbf{D})$. Hence, the lemma is proved. ■

III. Coding Model and its Asymptotic Optimality

A. Preliminaries

Let $\mathcal{N}_{N,k}(\{C_w\}) = \mathcal{N}_{N,k}(\{C_w\}_{w \in \{0,1\}^k})$ be the set of necklaces determined by the numbers of occurrences of all substrings of length k. The argument $\{C_w\}$ is a list of 2^k numbers from C_{0^k} to C_{1^k}, arranged in the lexicographic order of the corresponding substrings. Thus, $\mathcal{N}_{8,2}(4,2,2,0)$ is the set of necklaces of length $N = 8$ with $C_{00} = 4, C_{01} = 2, C_{10} = 2$, and $C_{11} = 0$. From Eq. (2), the sum of the numbers in $\mathcal{N}_{N,k}(\ldots)$ is equal to the string length N. Note that for enumerating necklaces in $\mathcal{N}_{N,k}(\{C_w\})$, we consider only the set of fixed-length ($= k$) substrings. The numbers of shorter substrings are automatically determined by Eq. (1). Table I shows examples of necklaces. The sets of necklaces are monotonic in the sense

$$\mathcal{N}_{N,k}(\{C_w\}_{w \in \{0,1\}^k}) \supseteq \mathcal{N}_{N,k+1}(\{C_{w'}\}_{w' \in \{0,1\}^{k+1}}),$$

where $C_w = \sum C_{w'}$ for $w' = w0$ and $w' = w1$.

[1] In some cases, the upper bound is not tight. For example, if $C_0 > 0$ and $C_1 > 0$, then naturally $C_{01} > 0$ and $C_{10} > 0$. In this case, $C_{00} \leq C_0 - 1$. If $C_0 > C_1$, then we have at least one substring 001, so $C_{001} \geq 1$. In this case, $C_{000} \leq C_{00} - 1$. Shima et al. [10] have recently succeeded in incorporating these facts into a general upper bound.

Table I
NECKLACE EXAMPLES.

$\mathcal{N}_{8,1}(6,2) = \{00000011, 00000101, 00001001, 00010001\}$
$\mathcal{N}_{8,2}(4,2,2,0) = \{00000101, 00001001, 00010001\}$
$\mathcal{N}_{8,3}(3,1,2,0,1,1,0,0) = \{00000101\}$

First, we prove an upperbound on the size of the set of necklaces.

Lemma 3: For $N \geq 1$ and $k \geq 0$,

$$|\mathcal{N}_{N,k}(\{C_w\}_{w\in\{0,1\}^k})|^k \leq \frac{N!}{\prod_{w\in\{0,1\}^k} C_w!}. \tag{10}$$

Proof: We prove the lemma by induction on k for fixed N.

For $k = 0$, Eq. (10) obviously holds with equality.

Now, we assume that Eq. (10) holds for $k = l \geq 0$:

$$|\mathcal{N}_{N,l}(\{C_w\})|^l \leq \frac{N!}{\prod_{w\in\{0,1\}^l} C_w!}, \tag{11}$$

and we prove this for $k = l + 1$.

Suppose that we are given a set of numbers of occurrences of all substrings of length $l+1$. For a string $\mathbf{D} = \mathbf{D}[0..N-1]$, if we observe C_w occurrences of a substring w of length l in the concatenation $\mathbf{D}\cdot\mathbf{D}[0..l]$, there are at most $\binom{C_w}{C_{w0}}$ possible combinations of their following bits. The number of necklaces never exceeds the product of these possible combinations over all substrings of length l. That is, for $|w'| = l + 1$,

$$|\mathcal{N}_{N,l+1}(\{C_{w'}\}_{w'\in\{0,1\}^{l+1}})| \leq \prod_{w\in\{0,1\}^l} \binom{C_w}{C_{w0}}. \tag{12}$$

From the monotonicity on the size of the necklaces, we have

$$|\mathcal{N}_{N,l+1}(\{C_{w'}\})|^l \leq |\mathcal{N}_{N,l}(\{C_w\})|^l \quad \text{for } |w'| = l+1 \text{ and } |w| = l.$$

Combining this and (12) with the induction hypothesis (11), we have

$$\begin{aligned} &|\mathcal{N}_{N,l+1}(\{C_{w'}\})|^{l+1} \\ &\leq |\mathcal{N}_{N,l}(\{C_w\})|^l \prod_{w\in\{0,1\}^l} \binom{C_w}{C_{w0}} \\ &\leq \frac{N!}{\prod_{w\in\{0,1\}^l} C_w!} \frac{\prod_{w\in\{0,1\}^l} C_w!}{\prod_{w\in\{0,1\}^l} C_{w0}!C_{w1}!} \\ &= \frac{N!}{\prod_{w'\in\{0,1\}^{l+1}} C_{w'}!}. \end{aligned} \tag{13}$$

Thus, we confirm that Eq. (10) is the case with $k = l + 1$; hence the lemma is proved. ∎

Next, we introduce another expression to represent the same quantity as $\mathcal{N}_{N,k}(\{C_w\})$. Let $\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_{0w0})$ denote the set of necklaces determined by the numbers $N, C_0, C_{00}, \ldots, C_{0w0}$, which are actually encoded by the CSE compressor. Obviously, we have

$$\begin{aligned} \mathcal{N}_{\mathrm{CSE}}(N) \supseteq \mathcal{N}_{\mathrm{CSE}}(N; C_0) \supseteq \mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}) \\ \supseteq \cdots \supseteq \mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_{0w0}). \end{aligned}$$

We can regard the main loop of CSE as a process in which we gradually reduce the set of necklaces to a singleton. When we encode $0w0$ in the ith step and v in the $i+1$st step, if $|v| > |0w0| = k$, we have

$$\mathcal{N}_{\mathrm{CSE}}(N; C_0, \ldots, C_{0w0}) = \mathcal{N}_{N,k}(\{C_{w'}\}_{w'\in\{0,1\}^k}). \tag{14}$$

B. Coding Model

When we encode C_{0w0} in the main loop, it is reasonable to assign the probability

$$\frac{|\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_u, C_{0w0})|}{|\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_u)|} \tag{15}$$

to the number C_{0w0}, where u is a substring such that C_u is encoded immediately before C_{0w0}. We have no closed form expressions for $|\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_{0w0})|$, except for some initial sets [8]; however, we continue our analysis under the assumption that these numbers are available.

When we use (15) to encode C_{0w0}, we can encode it with

$$-\log \frac{|\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_u, C_{0w0})|}{|\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_u)|} \tag{16}$$

bits provided that we can perform the encoding ideally (i.e., by using an *ideal* arithmetic code). Hereafter, we refer to the ideal codeword length assumed by the model as the *model entropy*.

We now present a coding model for encoding the value of C_{0w0} in the main loop. The proposed model uses the following two probability estimators, depending on the length of the substring.

Uniform over (6) when $|0w0| \leq \lfloor \log\log N \rfloor$,

Prediction by (15) when $|0w0| \geq \lfloor \log\log N \rfloor + 1$.

It follows from (8) that the proposed model encodes the ith C_{0w0} using a uniform distribution over the range (6) when $\log i + 1 \leq \lfloor \log\log N \rfloor$, which implies that $i < \log N$. Since any number in (6) can be encoded with at most $\log N$ bits, the total number of bits to be outputted by the uniform model never exceeds $(\log N)^2$. After that, the model begins to encode C_{0w0} by prediction (15).

When the model uses the prediction in (15), Eq. (16) is our model entropy for C_{0w0}. Since the set of necklaces monotonically decreases and eventually converges to a singleton, the sum of (16) over all $0w0$s that satisfy $|0w0| \geq \lfloor \log\log N \rfloor + 1$ is equal to

$$\begin{aligned} &\log |\mathcal{N}_{\mathrm{CSE}}(N; C_0, C_{00}, \ldots, C_u)| \\ &= \log |\mathcal{N}_{N,|u|}(\{C_w\}_{w\in\{0,1\}^{|u|}})|, \end{aligned} \tag{17}$$

in which u is a substring such that C_u is encoded immediately before the use of model (15), and its length is $\lfloor \log\log N \rfloor$ or shorter. For example, for $N=8$, $\lfloor \log\log 8 \rfloor = 1$ bit. In the example provided in Table I, C_{00} and C_{000} are encoded by prediction (15), and the corresponding model entropy is

$$\begin{aligned} &-\log \frac{|\mathcal{N}_{\rm CSE}(N; C_0, C_{00})|}{|\mathcal{N}_{\rm CSE}(N; C_0)|} \\ &\quad -\log \frac{|\mathcal{N}_{\rm CSE}(N; C_0, C_{00}, C_{000})|}{|\mathcal{N}_{\rm CSE}(N; C_0, C_{00})|} \\ &= -\log\frac{3}{4} - \log\frac{1}{3} = \log 4 \\ &= \log|\mathcal{N}_{\rm CSE}(N; C_0)| = \log|\mathcal{N}_{N,1}(C_0, C_1)|. \end{aligned}$$

We now bound the model entropy in (17) using the following lemma.

Lemma 4: For any natural numbers $n, a_1, a_2, \ldots, a_d$ such that $n = a_1 + a_2 + \cdots + a_d$,

$$\frac{n!}{\prod_{i=1}^{d} a_i!} \le \frac{n^n}{\prod_{i=1}^{d} a_i^{a_i}}. \tag{18}$$

Proof: Omitted, but standard. ∎

Noting Eq. (2), we can show the following.

Theorem 1: For any natural numbers $1 \le k \le N$,

$$\log|\mathcal{N}_{N,k}(\{C_w\})| \le -\frac{N}{k}\sum_{w\in\{0,1\}^k}\frac{C_w}{N}\log\frac{C_w}{N}. \tag{19}$$

Proof: Follows from Lemmas 3 and 4. ∎

Let $L_{\rm CSE}(\mathbf{D})$ denote the model entropy or the ideal codeword length attained by the CSE compressor with the model described above for $\mathbf{D}$ of length N. We can now establish that it is upperbounded by

$$\begin{aligned} L_{\rm CSE}(\mathbf{D}) \le -\frac{N}{k(N)}\sum_{w\in\{0,1\}^{k(N)}}\frac{C_w}{N}\log\frac{C_w}{N} \\ +(\log N)^2 + O(\log N), \end{aligned} \tag{20}$$

where

$$k(N) = \lfloor \log\log N \rfloor. \tag{21}$$

On the right-hand side of (20), the second term represents the cost incurred by the uniform model, and the last term corresponds to the costs for representing the value of N and rank of $\mathbf{D}$.

C. Asymptotic Optimality

Consider a stationary ergodic source $\boldsymbol{X} = \{X_i\}_{i=1}^{\infty}$, $X_i \in \{0,1\}$ with entropy rate $H(\boldsymbol{X})$. Let X_1^n denote an n-tuple of random variables $X_1, X_2, \ldots, X_n$ drawn from the source. Define

$$\begin{aligned} H^{(n)}(\boldsymbol{X}) &= \frac{1}{n}H(X_1^n) \\ &= -\frac{1}{n}\sum_{w\in\{0,1\}^n}\mu_n(w)\log\mu_n(w), \end{aligned} \tag{22}$$

where μ_n is a probability measure that defines the source $\boldsymbol{X}$, i.e.,

$$\mu_n(w) = \Pr\{X_1^n = w\}, \quad w \in \{0,1\}^n. \tag{23}$$

Then,

$$H(\boldsymbol{X}) = \lim_{n\to\infty} H^{(n)}(\boldsymbol{X}). \tag{24}$$

Here, we show our main result.

Theorem 2: For any stationary ergodic source $\boldsymbol{X} = \{X_i\}_{i=1}^{\infty}$, $X_i \in \{0,1\}$ with entropy rate $H(\boldsymbol{X})$, the model entropy $L_{\rm CSE}(X_1^N)/N$ per bit satisfies the following:

$$\lim_{N\to\infty}\frac{L_{\rm CSE}(X_1^N)}{N} = H(\boldsymbol{X}), \quad \text{a.s.} \tag{25}$$

Proof: For an integer $k \le N$, let $\tilde{X}_1^{N+k-1}$ be the concatenation of X_1^N and X_1^{k-1}. Define

$$p^{(k)}(w \mid X_1^N) = \frac{|\{i : \tilde{X}_i^{i+k-1} = w,\ 1 \le i \le N\}|}{N}$$

for $w \in \{0,1\}^k$, and

$$\tilde{H}^{(k)}(X_1^N) = -\frac{1}{k}\sum_{w\in\{0,1\}^k} p^{(k)}(w \mid X_1^N)\log p^{(k)}(w \mid X_1^N).$$

From (20), we have

$$\frac{L_{\rm CSE}(X_1^N)}{N} \le \tilde{H}^{(k(N))}(X_1^N) + \frac{(\log N)^2 + O(\log N)}{N}.$$

Thus, it is sufficient to show

$$\lim_{N\to\infty}\tilde{H}^{(k(N))}(X_1^N) = H(\boldsymbol{X}), \quad \text{a.s.} \tag{26}$$

to prove the theorem. Note that $H^{(n)}(\boldsymbol{X})$ is monotonic in the sense that

$$H^{(1)}(\boldsymbol{X}) \ge H^{(2)}(\boldsymbol{X}) \ge \cdots \ge H(\boldsymbol{X}).$$

Equation (24) means that for an arbitrary $\varepsilon > 0$ we can choose an integer K such that

$$H^{(K)}(\boldsymbol{X}) \le H(\boldsymbol{X}) + \varepsilon. \tag{27}$$

Since $p^{(k)}(w \mid X_1^N)$ satisfies the compatibility condition:

$$p^{(k)}(w \mid X_1^N) = p^{(k+1)}(w0 \mid X_1^N) + p^{(k+1)}(w1 \mid X_1^N),$$

$\tilde{H}^{(k)}(X_1^N)$ is also monotonic, i.e.,

$$\tilde{H}^{(1)}(X_1^N) \ge \cdots \ge \tilde{H}^{(k)}(X_1^N) \ge \tilde{H}^{(k+1)}(X_1^N) \ge \cdots. \tag{28}$$

For the integer K chosen above, since

$$\lim_{N\to\infty} p^{(K)}(w \mid X_1^N) = \mu_K(w), \quad \text{a.s.}$$

from the ergodicity of the source, we have

$$\lim_{N\to\infty}\tilde{H}^{(K)}(X_1^N) = H^{(K)}(\boldsymbol{X}), \quad \text{a.s.}$$

This implies that for almost every infinite binary string x there exists an integer $N_o(x,\varepsilon)$ such that

$$\tilde{H}^{(K)}(x_1^N) \le H^{(K)}(\boldsymbol{X}) + \varepsilon \text{ for } \forall N \ge N_o(x,\varepsilon). \tag{29}$$

From (27) and (29), for almost every x we have

$$\tilde{H}^{(K)}(x_1^N) \le H(\boldsymbol{X}) + 2\varepsilon \quad \text{for } \forall N \ge N_o(x,\varepsilon).$$

Since $k(N)$ is given in (21), we can make $k(N) \ge K$ for sufficiently large N. For such N, we have

$$\tilde{H}^{(k(N))}(x_1^N) \le \tilde{H}^{(K)}(x_1^N)$$

from (28). Thus, for almost every x we have

$$\tilde{H}^{(k(N))}(x_1^N) \le H(\boldsymbol{X}) + 2\varepsilon \quad \text{for } \forall N \ge N_o(x,\varepsilon).$$

Since ε is arbitrary, this shows that (26) is the case, and therefore the theorem is proved. ■

Theorem 2 shows the asymptotic optimality of our model in the sense that the per-bit model entropy approaches almost surely the entropy rate of the source that produces a source string when its length tends to infinity.

IV. New Decoding Algorithm

A. Redundancy in Coding Process

In the previous section, we have shown that there exists an asymptotically optimal coding model in the CSE framework. However, it is essential to have complete knowledge on the size of the set of necklaces when we wish to realize the model. We may overcome this difficulty by introducing an estimator of the size itself or of the predictor in (15). For example, we may use the kth root of the upperbound in (10) instead of the size of a necklace set for the computation of the ratio in (15). In fact, we have another combinatorial predictor [5], which can be relatively easily computed and shown to be universal in a similar sense to our present model. Even if we use such a model that requires no knowledge on the size of necklaces, we need to be able to detect if the necklace is unique so that we can efficiently terminate the main loop of the CSE compressor. We here say that the necklace $\mathbf{D}$ is *unique* for a given length k if $\mathbf{D}$ is the sole element in the necklace set $\mathcal{N}_{N,k}(\{C_w\})$ that are determined by the numbers of occurrences of all substrings of length k in $\mathbf{D}$.

As an example, consider $\mathbf{D} = 00000101$ ($N = 8$). Since we know $U(\mathbf{D}) \cap V(\mathbf{D}) = \{\epsilon, 0, 00, 000, 010, 0000\}$ from (9), the CSE compressor may encode N and the numbers in the following table.

w	$-$	ϵ	0	00	000	010	0000
C_{0w0}	C_0	C_{00}	C_{000}	C_{0000}	C_{00000}	C_{00100}	C_{000000}
	6	4	3	2	1	0	0

On the decoding end, however, upon receiving $N = 8$, $C_0 = 6$, $C_{00} = 4$, and $C_{000} = 3$, we can immediately know that only the necklace 00000101 can fulfill these numbers and any further numbers from C_{0000} are no longer needed. The numbers C_{0000}, C_{00000} and so on are redundant to encode $\mathbf{D} = 00000101$. Here, we propose a new implementation of CSE that can remove this kind of redundancy.

B. New Decoder

The proposed implementation can be characterized by the decoding procedure, in which the numbers $(N; C_0, \ldots, C_w)$ are used as an input for recovering the necklace corresponding to an original string. Let k be the length of the longest substring in the input. We defer to Section IV-C our discussion on the length k to be required for decoding.

The decoding procedure for a necklace can be divided into three steps. The first step is devoted to recovering the left k columns of the BWT-transformed matrix. In the following two steps, we apply a finite-order variant [6], [7], [11] of the BWT.

Step 1. (Ordering of substrings)

We assume that the numbers $\{C_w\}_{w\in\{0,1\}^k}$ are already computed for all substrings of length k. We sort all the substrings of length k with $C_w > 0$ lexicographically, and put them into an $N \times k$ matrix $M'[1..N][1..k]$ as its N rows. The number of the same row vectors as w must be equal to C_w. Then, perform a stable sort on the row vectors of $M'[1..N][1..k]$ in disregard of the first column, and let $M[1..N][1..k]$ be the obtained $N \times k$ matrix.

Step 2. (Computation of auxiliary vectors)

Correspond the first and second columns of $M[1..N][1..k]$ in a stable manner (i.e., respecting the orders), and represent the correspondence by $Q[1..N]$. That is, if the jth bit $M[j][1]$ in the first column corresponds to the ith bit $M[i][2]$ in the second column, then we set

$$Q[i] = j \quad (1 \le i, j \le N).$$

The vector $Q[1..N]$ can be calculated in the following way:

1) Set $p_0 := 1$ and $p_1 := C_0 + 1$;
2) **For** $j := 1$ **to** N **do**
   a) **If** $M[j][1] = 0$ **then** set $Q[p_0] := j$ and p_0++;
   b) **If** $M[j][1] = 1$ **then** set $Q[p_1] := j$ and p_1++;

We introduce other two vectors $C[1..N]$ and $T[1..N]$, which are computed using the following procedure. Nonzero elements in $C[1..N]$ represent the numbers of occurrences of substrings of length $k-1$.

1) Initialize $C[1..N]$ as a zero-vector;
2) **For** $i := 1$ **to** N **do**
   a) **If** $i = 1$ or $M[i][2..k] \neq M[i-1][2..k]$ **then** set $j := i$;
   b) Set $T[Q[i]] := j$;
   c) Set $C[j]$++;

Step 3. (Reconstruction of a necklace)

The necklace corresponding to an input string $\mathbf{D}$ is obtained in $x[1..N]$ in the following way:

1) Set $i := 1$;
2) **For** $j := N$ **downto** 1 **do**
   a) Set $C[i]{-}{-}$;
   b) Set $i := i + C[i]$;
   c) Set $x[j] := M[i][1]$;

i	$M[1..8][1..3]$			$Q[i]$	$C[i]$	$T[i]$
1	0_1	0_1	0	1	4	1
2	0_2	0_2	0	2	0	1
3	0_3	0_3	0	3	0	1
4	1_4	0_5	0	5	0	7
5	0_5	0_7	1	7	2	1
6	1_6	0_8	1	8	0	7
7	0_7	1_4	0	4	2	5
8	0_8	1_6	0	6	0	5

(a) Step 2

	a)	b)	c)	d)
j	$C[i]$	i	$x[j]$	i
8	$C[1]=3$	4	$x[8]=M[4][1]=1$	$T[4]=7$
7	$C[7]=1$	8	$x[7]=M[8][1]=0$	$T[8]=5$
6	$C[5]=1$	6	$x[6]=M[6][1]=1$	$T[6]=7$
5	$C[7]=0$	7	$x[5]=M[7][1]=0$	$T[7]=5$
4	$C[5]=0$	5	$x[4]=M[5][1]=0$	$T[5]=1$
3	$C[1]=2$	3	$x[3]=M[3][1]=0$	$T[3]=1$
2	$C[1]=1$	2	$x[2]=M[2][1]=0$	$T[2]=1$
1	$C[1]=0$	1	$x[1]=M[1][1]=0$	$T[1]=1$

(b) Step 3

Figure 1. Decoding example from $(N; C_0, C_{00}, C_{000}) = (8; 6, 4, 3)$.

d) Set $i := T[i]$;

Before analyzing the decodability, we give an example that shows how the above procedure works.

Suppose that we have $(N; C_0, C_{00}, C_{000}) = (8; 6, 4, 3)$ with $k = 3$. Then, we derive C_1 from (1), and C_{01}, C_{10}, C_{11}, $C_{001} = 1$, $C_{010} = 2$, $C_{100} = 1$, $C_{101} = 1$ (no occurrences of other three-bit substrings) from (3), (4), and (5). Thus, we have the following two matrices in Step 1.

$$M'[1..8][1..3] = \begin{pmatrix} 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&1\\ 0&1&0\\ 0&1&0\\ 1&0&0\\ 1&0&1 \end{pmatrix}, \qquad M[1..8][1..3] = \begin{pmatrix} 0&0&0\\ 0&0&0\\ 0&0&0\\ 1&0&0\\ 0&0&1\\ 1&0&1\\ 0&1&0\\ 0&1&0 \end{pmatrix}.$$

Keys for the sort in the last half of Step 1 are shown in the box. The first column of matrix M will be treated as if it were the last column of the BWT matrix.

We then proceed to Step 2, which produces the three vectors shown in Fig. 1 (a). In the figure, we add subscripts to each of the bits in order to explicitly show their correspondences in the first and second columns of matrix M. All N substrings of length k in circular $\mathbf{D}$ are arranged lexicographically in $M'[1..N][1..k]$, in which in turn the row vectors are rearranged in lexicographic order of $M'[\cdot][2..k]$. Therefore, in the obtained matrix $M[1..N][1..k]$, we have

$$M[i][2..k] = M[Q[i]][1..k-1], \text{ for } i = 1, 2, \ldots, N.$$

Note also that

$$M[T[i]][2..k] = M[i][1..k-1], \text{ for } i = 1, 2, \ldots, N. \quad (30)$$

Finally, the loop in Step 3 works as shown in Fig. 1 (b). Actually, $x[1..8] = 00000101$ equals $\mathbf{D}$ in our running example.

C. Decodability

In order to demonstrate the decodability of the input string by the above decoder, we need some notions. Assume that $\mathbf{D} = \mathbf{D}[0..N-1]$ is a necklace. That is, $\mathbf{D}$ is circular, and is the lexicographically smallest string of all the cyclic shifts. We refer to its substring:

$$s_i^{(\ell)} = \mathbf{D}[i..i+\ell-1] \text{ for } i = 0, 1, \ldots, N-1$$

as the *ith (forward) context of length* ℓ, where we take indices mod N. Next, for an ℓ-bit binary substring $w \in \{0,1\}^\ell$, we define the *index set* for symbols occurred in the context w of length $\ell \geq 0$ by

$$I_{\mathbf{D}}(w) = \{i \mid s_{i+1}^{(\ell)} = w,\ 0 \leq i \leq N-1\}.$$

For $\mathbf{D} = 00000101$, we have, for example,

$$\begin{aligned} I_{\mathbf{D}}(\epsilon) &= \{0,1,2,3,4,5,6,7\},\\ I_{\mathbf{D}}(0) &= \{0,1,2,3,5,7\},\\ I_{\mathbf{D}}(10) &= \{4,6\}. \end{aligned} \quad (31)$$

Definition 1: (Consistency of an index set) The index set $I_{\mathbf{D}}(w)$ is *consistent* either if $\mathbf{D}[i] \leq \mathbf{D}[j]$ for any $i \leq j \in I_{\mathbf{D}}(w)$, or if $I_{\mathbf{D}}(w)$ is empty.

Generally, the longer the contexts, the more likely they will be consistent. Hence, we introduce the following length:

$$\ell_{\min}(\mathbf{D}) = \min\{\ell \mid \text{all index sets } I_{\mathbf{D}}(w) \text{ for } \forall w \in \{0,1\}^\ell \text{ are consistent}\}. \quad (32)$$

For a necklace $\mathbf{D}$ and a given $\ell > 0$, consider a pair

$$\langle C_{0w}, C_{1w} \rangle$$

for every $w \in \{0,\ 1\}^\ell$. Then, we have the following lemma.

Lemma 5: For two different necklaces $\mathbf{D}$ and $\mathbf{D}'$, the two sets $\{\langle C_{0w}, C_{1w} \rangle\}$ of pairs corresponding to $\mathbf{D}$ and $\mathbf{D}'$ are distinct for any $|w| \geq \max\{\ell_{\min}(\mathbf{D}), \ell_{\min}(\mathbf{D}')\}$.

Proof: The case where the lengths of $\mathbf{D}$ and $\mathbf{D}'$ are different is trivial. Thus, we assume that they have the same

length N. We also assume without loss of generality that $|w| = \ell \geq \ell_{\min}(\mathbf{D}') \geq \ell_{\min}(\mathbf{D})$. Let

$$b_0 b_1 \cdots b_{\ell-1},\ b_i \in \{0,\, 1\}$$

be the lexicographically smallest substring of length ℓ among such w's that $C_w = C_{0w} + C_{1w} > 0$ in $\mathbf{D}$. This substring corresponds to the ℓ-bit prefix of $\mathbf{D}$. Suppose on the contrary that a necklace $\mathbf{D}'$ ($\neq \mathbf{D}$) with $\ell_{\min}(\mathbf{D}') \geq \ell_{\min}(\mathbf{D})$ produces the same set of pairs as $\{\langle C_{0w}, C_{1w}\rangle\}$ of $\mathbf{D}$. $\mathbf{D}'$ must also begin with the same prefix as $b_0 b_1 \cdots b_{\ell-1}$. Consider the two strings:

$$\mathbf{D}[0..N-1]\, b_0 b_1 \cdots b_{\ell-1} = x_0 x_1 \cdots x_{N+\ell-1},$$
$$\mathbf{D}'[0..N-1] b_0 b_1 \cdots b_{\ell-1} = x'_0 x'_1 \cdots x'_{N+\ell-1},$$

and the rightmost position j for which $x_j \neq x'_j$. Then, we have

$$\ell \leq j \leq N-1,$$
$$x_{j+1} x_{j+2} \cdots x_{j+\ell} = x'_{j+1} x'_{j+2} \cdots x'_{j+\ell}.$$

Since $\ell \geq \ell_{\min}(\mathbf{D}') \geq \ell_{\min}(\mathbf{D})$, the position j is included in a consistent index set both in $\mathbf{D}$ and $\mathbf{D}'$. For $w = x_{j+1} x_{j+2} \cdots x_{j+\ell} = x'_{j+1} x'_{j+2} \cdots x'_{j+\ell}$, since $\mathbf{D}$ and $\mathbf{D}'$ share the same $\langle C_{0w}, C_{1w}\rangle$, the numbers of occurrences of w are the same in $\mathbf{D}$ and $\mathbf{D}'$. The assumption $x_j \neq x'_j$ contradicts the fact that $\mathbf{D}$ and $\mathbf{D}'$ have the same C_{0w}. Therefore, $\mathbf{D}$ and $\mathbf{D}'$ must be identical. Thus, if $\mathbf{D}$ and $\mathbf{D}'$ are different, then $\{\langle C_{0w}, C_{1w}\rangle\}$ are also different between $\mathbf{D}$ and $\mathbf{D}'$. ■

Consider again the table in Fig. 1 (a). We now insert a dotted line just before a row that has a non-zero $C[i]$, as shown in Fig. 2, where we add the values of $T[Q[i]]$ as well. The dotted lines divide the rows of matrix M into regions of the same $M[\cdot][2..k]$. In each region, the values of $T[Q[i]]$ are all the same, and are equal to the first row number of the region. The first column of matrix M is *consistent* in each region in the sense that $M[i][1] \leq M[j][1]$ for $i < j$. If we consider $M[\cdot][2..k]$ as w, as shown in Fig. 2, the matrix M is equivalent to the set of pairs $\{\langle C_{0w}, C_{1w}\rangle\}$.

For any necklace $\mathbf{D}$, in the first region: $1 \leq i \leq C[1]$, $M[i][2..k]$ corresponds to the $k-1$-bit prefix of $\mathbf{D}$, and

$$M[C[1]][1..k] = \mathbf{D}[N-1] \cdot \mathbf{D}[0..k-2]. \tag{33}$$

In the proposed decoding algorithm, Step 3 begins with

1) Set $i := 1$;
   a) Set $C[i]{-}{-}$;
   b) Set $i := i + C[i]$;
   c) Set $x[N] := M[i][1]$,

which is equivalent to $x[N] := M[C[1]][1]$. From (33), we know that $M[C[1]][1] = \mathbf{D}[N-1]$. After that, Step 3 produces the string that has $M[\cdot][1..k]$ as the associated set of pairs. Combining this fact with (30) and Lemma 5, we can now establish the following.

i		w		$Q[i]$	$C[i]$	$T[i]$	$T[Q[i]]$
1	0	0	0	1	4	1	1
2	0	0	0	2	0	1	1
3	0	0	0	3	0	1	1
4	1	0	0	5	0	7	1
5	0	0	1	7	2	1	5
6	1	0	1	8	0	7	5
7	0	1	0	4	2	5	7
8	0	1	0	6	0	5	7

Figure 2. Regions corresponding to pairs $\{\langle C_{0w}, C_{1w}\rangle\}$.

Theorem 3: For any necklace $\mathbf{D}$, the proposed decoding algorithm with $k \geq \ell_{\min}(\mathbf{D}) + 1$ can recover $\mathbf{D}$ losslessly.

Remark: For $\mathbf{D} = 00000101$, we have $I_{\mathbf{D}}(0)$ in (31) and $I_{\mathbf{D}}(1) = \{4, 6\}$. Since $0 = \mathbf{D}[0] = \mathbf{D}[1] = \mathbf{D}[2] = \mathbf{D}[3] < \mathbf{D}[5] = \mathbf{D}[7] = 1$ and $\mathbf{D}[4] = \mathbf{D}[6] = 0$, the index sets $I_{\mathbf{D}}(0)$ and $I_{\mathbf{D}}(1)$ of length 1 are both consistent, so $\ell_{\min}(\mathbf{D}) = 1$. Therefore, it is sufficient to encode $N = 8$, $C_0 = 6$, and $C_{00} = 4$ for recovering $\mathbf{D}$. Using these numbers, Step 2 of the decoding procedure constructs the table in Fig. 3 (a). Note that $C[1] = C_0$ and $C[7] = C_1$. Step 3 uses the table to recover $\mathbf{D}$ in $x[1..8]$ as shown in Fig. 3 (b).

Intuitively, it looks strange that $\mathbf{D} = 00000101$ can be recovered only from $N = 8$, $C_0 = 6$, and $C_{00} = 4$, which are also the case with "00001001" and "00010001." In fact, in our decoding algorithm, "00000101" is recognized as the necklace that has $N = 8$, $C_0 = 6$, and $C_{00} = 4$; "00001001" as the necklace that has $N = 8$, $C_0 = 6$, $C_{00} = 4$, and $C_{000} = 2$; "00010001" as the necklace that has $N = 8$, $C_0 = 6$, $C_{00} = 4$, $C_{000} = 2$, and $C_{0000} = 0$. Hence, in a strict sense, the code that our decoder assumes is *not a prefix code* when we regard an entire data block as a single codeword. However, in practice, this is not a substantial drawback because such a data block is usually identified with a special delimiter or by specifying the block length.

D. Comparison of the Number of Substrings to be Encoded

In the last one ($\mathbf{D} = 00010001$) of the above examples, since $U(\mathbf{D}) = V(\mathbf{D}) = \{\epsilon, 0, 00\}$, the numbers $\{C_{0w0}\}$ for $w \in \{\epsilon, 0, 00\}$ are encoded in the main loop of the original CSE. These numbers are exactly the same as those that we need to recover $\mathbf{D}$ using our decoder. In this sense, the original CSE and our proposed one have the same performance for this particular example. In general, however, our algorithm is never worse than the original CSE in terms of the number of substrings to be encoded. We now show this.

Theorem 4: For the longest substring $w_{\mathrm{M}} \in U(\mathbf{D}) \cap V(\mathbf{D})$ and its length $|w_{\mathrm{M}}|$,

$$\ell_{\min}(\mathbf{D}) \leq |w_{\mathrm{M}}| + 1. \tag{34}$$

i	$M[\cdot][1,2]$		$Q[i]$	$C[i]$	$T[i]$
1	0_1	0_1	1	6	1
2	0_2	0_2	2	0	1
3	0_3	0_3	3	0	1
4	0_4	0_4	4	0	1
5	1_5	0_7	7	0	7
6	1_6	0_8	8	0	7
7	0_7	1_5	5	2	1
8	0_8	1_6	6	0	1

(a) Step 2

j	$C[i]$	i	$x[j]$	i
8	$C[1]=5$	6	$x[8]=M[6][1]=1$	$T[6]=7$
7	$C[7]=1$	8	$x[7]=M[8][1]=0$	$T[8]=1$
6	$C[1]=4$	5	$x[6]=M[5][1]=1$	$T[5]=7$
5	$C[7]=0$	7	$x[5]=M[7][1]=0$	$T[7]=1$
4	$C[1]=3$	4	$x[4]=M[4][1]=0$	$T[4]=1$
3	$C[1]=2$	3	$x[3]=M[3][1]=0$	$T[3]=1$
2	$C[1]=1$	2	$x[2]=M[2][1]=0$	$T[2]=1$
1	$C[1]=0$	1	$x[1]=M[1][1]=0$	$T[1]=1$

(b) Step 3

Figure 3. Decoding example from $(N; C_0, C_{00}) = (8; 6, 4)$.

Proof: From Lemma 2, $w_{\rm M}$ is the longest substring in $V(\mathbf{D})$. Therefore, both $0w_{\rm M}$ and $1w_{\rm M}$ occur in $\mathbf{D}$, but only either of $00w_{\rm M}$ or $10w_{\rm M}$ occurs in $\mathbf{D}$. Similarly, only either of $01w_{\rm M}$ or $11w_{\rm M}$ occurs in $\mathbf{D}$. Thus, both $I(0w_{\rm M})$ and $I(1w_{\rm M})$ are consistent. Therefore, $\ell_{\min}(\mathbf{D})$ is at most the length of $w_{\rm M}$ plus one. ■

In the original CSE, the numbers $\{C_{0w0}\}$ are encoded for substrings from $w = \epsilon$ to $w = w_{\rm M}$, and the length of the longest one is $|0w_{\rm M}0| = |w_{\rm M}| + 2$. Theorems 3 and 4 show that when we incorporate our decoder into the CSE framework, the required length of substrings is at most $\ell_{\min}(\mathbf{D}) + 1$, which is shorter than or equal to $|w_{\rm M}| + 2$. However, this does not imply that we can relate $\ell_{\min}(\mathbf{D})+1$ to the minimum length of substrings that is required to uniquely identify the necklace. In some cases, $\ell_{\min}(\mathbf{D}) + 1$ is even shorter than the minimum length, as shown in the above example, and in some cases vice versa.

V. Conclusion

We have given an asymptotic optimal encoding model to CSE whose codeword length per input bit converges almost surely to the entropy rate of a stationary ergodic source when the length of a source string tends to infinity. We have also proposed a relatively efficient decoder that is based on a finite-order variant of the BWT.

We have already developed another asymptotic optimal model for CSE [5]. Future work includes the characterization of such optimal models or the development of a class of these models. In particular, finding a simple and practically efficient model is important. The development of a fast method for computing the value of $\ell_{\min}(\mathbf{D})$ is also future work for the proposed decoder and its corresponding encoder to be realized. In addition, we have to develop a method for detecting the uniqueness of a necklace. The CSE assumes the data to be binary. An extension to the case with an arbitrary alphabet is also an interesting issue.

Acknowledgment

The author would like to thank Danny Dubé for his comments and discussions.

References

[1] M. Burrows and D. J. Wheeler, A block-sorting lossless data compression algorithm, *SRC Research Report*, 124, 1994.

[2] T. M. Cover, Enumerative source encoding, *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, 1973.

[3] D. Dubé, Using synchronization bits to boost compression by substring enumeration, *2010 Int. Symp. Information Theory and Appl., ISITA2010*, Taichung, Taiwan, Oct. 2010, pp. 82–87.

[4] D. Dubé and V. Beaudoin, Lossless data compression via substring enumeration, *Proc. of the Data Compression Conf.*, pp. 229–238, Snowbird, Utah, USA, Mar. 2010.

[5] D. Dubé and H. Yokoo, The universality and linearity of compression by substring enumeration, to appear in *2011 IEEE Int. Symp. Information Theory, ISIT 2011*, Saint-Petersburg, Russia, Aug. 2011.

[6] G. Nong and S. Zhang, Efficient algorithms for the inverse sort transform, *IEEE Trans. Computers*, vol. 56, no. 11, pp. 1564–1574, 2007.

[7] G. Nong, S. Zhang, and W. H. Chan, Computing inverse ST in linear complexity, *Combinatorial Pattern Matching: 19th Annual Symposium, CPM 2008*, Pisa, Italy, Jun. 2008, LNCS, vol. 5029, pp. 178–190.

[8] F. Ruskey and J. Sawada, An efficient algorithm for generating necklaces with fixed density, *SIAM J. Comput.*, vol. 29, no. 2, pp. 671–684, 1999.

[9] P. C. Shields, *The Ergodic Theory of Discrete Sample Paths*, American Mathematical Society, 1996.

[10] Y. Shima, K. Iwata, and M. Arimura, An improvement of lossless data compression via substring enumeration (in Japanese), *IEICE Technical Report*, IT2011-1, pp. 1–6, 2011.

[11] H. Yokoo, Extension and faster implementation of the GRP transform for lossless compression, *Combinatorial Pattern Matching: 21st Annual Symposium, CPM 2010*, Brooklyn, NY, Jun. 2010, LNCS, vol. 6129, pp. 338–347.