Vol.2014-AL-148 No.7
2014/6/13

Compression by Substring Enumeration符号化法のBWT行列による実現

金井 翔[1,a)]　横尾 英俊[1,b)]

概要：無ひずみデータ圧縮法に分類される比較的最近の手法として，Compression by Substring Enumeration 法がある．本論文では，そのための新しい符号化アルゴリズムを提案する．同符号化法と BWT 行列の関係を利用することで，木構造を利用した従来法よりも単純で高速な符号化法を実現した．本論文では，アルゴリズムの詳細と実験による評価結果を報告する．

キーワード：データ圧縮，符号化，ユニバーサル符号，BWT, CSE 法

Implementation of Compression by Substring Enumeration via BWT Matrix

Sho Kanai[1,a)]　Hidetoshi Yokoo[1,b)]

Abstract: A new encoder implementation is proposed for the lossless universal code of *Compression by Substring Enumeration*. Compared to the existing methods utilizing tree structures, the proposed encoder is much simpler and faster in that it relies only on array structures that appear in relation to the Burrows–Wheeler transform.

Keywords: BWT, coding, CSE, data compression, universal codes

1. まえがき

本論文では, Compression by Substring Enumeration (部分列数え上げデータ圧縮法，以下 “CSE 法” という）と呼ばれる無ひずみデータ圧縮法 [9] の新しい符号化アルゴリズムを提案する．

次のような問題を考える．まず，長さ n ビットの 2 元系列のすべての巡回シフトを各行に辞書式順に配置した $n \times n$ 行列を BWT 行列とよぶ．BWT 行列では，第 1 行目と第 n 列目とは，どちらも行列全体を再現することができるという意味で等価である [4]．それでは，第 1 列目から何列目までをどのように記述すれば，やはり，行列全体を再現することができるであろうか．

Lempel–Ziv 符号，PPM データ圧縮法，文法圧縮法などの無ひずみデータ圧縮法は，BWT 行列の各行を効率よく記述するためツールと考えることができる．同様に，第 n 列目は Move-to-Front, Inversion Frequency, Distance Coding 等 [2], [15], [25] を仲介として効率的な記述が可能である．これらに対し，本論文で考察の対象とする CSE 法は前段落の問題に対する解答例と考えることができる．

CSE 法は 2010 年に Dubé–Beaudoin [9] によって提案された，2 元系列を対象とする無ひずみデータ圧縮法である．最初の試行的な実験により，比較的良好な圧縮性能を有することが報告されると，理論的な解析も精力的に行われるようになってきている．

まず，適切な符号化モデルを組み合わせることによって，漸近最良なユニバーサル性を有することが示されている [10],[26]．特に，マルコフ情報源に対する漸近的最良性

[1] 群馬大学大学院 理工学府
Graduate School of Science and Technology, Gunma University
[a)] sho.kanai@daisy.cs.gunma-u.ac.jp
[b)] yokoo@cs.gunma-u.ac.jp

については，任意の次数のエントロピーを圧縮率の漸近限界として設定することが可能であり，極めて特徴的な性質である．また，この場合の最悪冗長度が最適な冗長度に十分近いことが岩田ら [11],[12],[13] によって明らかにされている．さらに，CSE 法は他のユニバーサルデータ圧縮法との関連も多様であり，本論文で議論するような BWT 変換 [4] との強い関連のほか，数え上げ符号 [5],[17] や反辞書符号化法 [7] との関連が指摘されている．たとえば，反辞書符号化法の漸近的最良性が，CSE 法実装用のデータ構造として導入された Compacted Substring Tree (CST) [9] の作るオートマトンと反辞書オートマトンとの同型性を利用して証明されている [18].

一方，現実の圧縮力の実験的評価，もしくは，そのための実装法の検討については，CSE 法の発明者自身によるものがわずかに存在する程度で [8], [9]，十分な評価がなされているとは言えない．それは，CSE 法のアイデアの単純さに比較して実装が相対的に面倒なこと，および，最終的なエントロピー符号化のために必要となる確率モデルが自明ではないこと等に起因すると考えられる．本論文では，CSE 法のより簡素で高速な実装法を目指して，これまで提案されてきた木構造に基づく方法 [9], [24] とは異なる符号化アルゴリズムを提案する．提案法では，CSE 法と BWT 変換との関係をより積極的に利用している．

以下本論文では，次の 2 節で CSE 法の概要を説明した後，従来提案されてきた実現法を簡単に紹介する．続く 3 節で，本論文で新たに提案する符号化アルゴリズムについて述べる．提案法の計算量の理論的な見積りに加え，4 節では実験による符号化時間の測定結果も示す．なお，データ圧縮法は，一般に，狭い意味での符号化法と復号法とから成る．本論文で提案するのは，CSE 法の符号化アルゴリズムであり，復号法については考慮していない．符号化法と復号法の対応については，4.2 節で簡単に考察する．

2. CSE 無ひずみデータ圧縮法

2.1 CSE 法の基礎

長さ n ビットの 2 元系列 $\boldsymbol{x} \in \{0,1\}^n$ の無ひずみ圧縮を考える[*1]．本論文では，$\boldsymbol{x}$ を第 I 行に持つ次のような 2 元 $n \times n$ 行列 $\mathcal{M}$ を BWT 行列とよぶ．まず，第 1 行目を $\boldsymbol{D} = \boldsymbol{D}[0..n-1] \in \{0,1\}^n$ で表すと，$\mathcal{M}$ の各行は $\boldsymbol{D}$ の巡回シフトを辞書式順序に並べたものである．すなわち，$\mathcal{M}$ の各行は巡回シフトについて同値類をなし[*2]，その第 I 行が入力系列 $\boldsymbol{x}$ に一致する．なお，本論文を通じて，非反復的な $\boldsymbol{x}$, $\boldsymbol{D}$ だけを考える．このとき，辞書式順序の最も先行する $\boldsymbol{D}$ を Lyndon word という [6]．また，$\mathcal{M}$ の各行に対し，$\mathcal{M}$ の右端の列を BWT 変換 [4] という．$\boldsymbol{D}$ から $\mathcal{M}$ が一意に定まり，BWT 変換も一意に定まる．逆に，$\mathcal{M}$ から $\boldsymbol{D}$ も一意に定まる．これ以降，入力 2 元系列の巡回シフトによる同値類を $\boldsymbol{D}$ で代表させ，$\boldsymbol{D}$ 自身も巡回列と見なす．入力系列を $\boldsymbol{D}$ と区別する必要がある場合には入力系列を $\boldsymbol{x}$ で表し，そうでない場合には $\boldsymbol{D}$ という記法を優先する．$\boldsymbol{D}$ および対応する $\mathcal{M}$ の例を**図 1** に示す．

CSE 法は，2 元系列 $\boldsymbol{D}$ 中に生起する部分列 $w \in \{0,1\}^*$ の出現回数を求め，その値を符号化することで $\boldsymbol{D}$ 自身の符号化を行うものである[*3]．部分列の出現回数の数え上げにおいては，$\boldsymbol{D}$ を巡回列と考えた上で，重複を許して数え上げる．

系列 $\boldsymbol{D}$ 中の部分列 w の生起回数を C_w で表し，空系列 ε に対し，$C_\varepsilon = n$ と定義する．$\boldsymbol{D}$ を巡回列と考えていることにより，次の整合性条件が成り立つ．

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

整合性条件に基づき，C_{0w0} について以下の下限と上限を導出することができる [9].

$$\max\{0, C_{0w} - C_{w1}\} \le C_{0w0} \le \min\{C_{0w}, C_{w0}\}. \tag{2}$$

式 (2) の上下限が一致する場合には C_{0w0} の値が一意に決まってしまい，符号化の必要がない．すなわち，C_{0w0} の値の取り得る可能性は

$$\begin{aligned} &\min\{C_{0w}, C_{w0}\} - \max\{0, C_{0w} - C_{w1}\} + 1 \\ &= \min\{C_{0w}, C_{1w}, C_{w0}, C_{w1}\} + 1 \end{aligned} \tag{3}$$

i	w	LCP	S	E
0		-1		
1	0	1	0	3
2	010	3	1	2
3	01	2	1	3
4	ε	0	0	7
5	10	2	4	6
6	1010	4	5	6
7	1	1	4	7

$\mathcal{M}$

								R
0	0	1	1	0	1	0	1	0
0	1	0	0	1	1	0	1	0
0	1	0	1	0	0	1	1	0
0	1	1	0	1	0	1	0	1
1	0	0	1	1	0	1	0	2
1	0	1	0	0	1	1	0	3
1	0	1	0	1	0	0	1	3
1	1	0	1	0	1	0	0	4

図 1 $\boldsymbol{D} = 00110101$ に対応する BWT 行列 $\mathcal{M}$ (中央)．行列の右端の列が $\boldsymbol{D}$ の BWT 変換に対応する．その他の配列については，本文の記述参照のこと．なお，$\mathsf{LCP}[n] = -1$, $R[-1] = 0$.

Fig. 1 BWT matrix $\mathcal{M}$ for $\boldsymbol{D} = 00110101$, and its related arrays.

[*1] 2 元に限定するのは，オリジナルの CSE 法が 2 元データを対象としているからであるが，最近，CSE 法の一般のアルファベットへの拡張が提案されている [13], [19].

[*2] このような同値類は**共役** (conjugates [6])，あるいは**ネックレス** (necklace [21]) とよばれる．Conjugates が複数形，necklace が単数形であることに注意．BWT 行列の各行それぞれを共役とよんでいるのに対し，それら一つひとつを区別せずに 1 個の巡回列としてネックレスとよぶ．

[*3] $\boldsymbol{x}$ の符号化では，これに加えて I の値の符号化が必要であるが，本論文では考慮から除外する．

通りである．したがって，

$$U(\boldsymbol{D}) = \{w \in \{0,1\}^* \mid C_{w0} \geq 1 \text{ and } C_{w1} \geq 1\},$$
$$V(\boldsymbol{D}) = \{w \in \{0,1\}^* \mid C_{0w} \geq 1 \text{ and } C_{1w} \geq 1\},$$
$$I(\boldsymbol{D}) = U(\boldsymbol{D}) \cap V(\boldsymbol{D})$$

を定義すると，CSE 法の符号化法は次のとおりである．

CSE 符号化法 ($\boldsymbol{D} \in \{0,1\}^n$ の符号化)

1. **Encode** n; **Encode** C_0;
2. **for** $l := 0$ **to** $n-2$ **do**
   for every $w \in \{0,1\}^l \cap I(\boldsymbol{D})$ **do Encode** C_{0w0};

(例) $\boldsymbol{D} = 00110101$ の符号化：
$I(\boldsymbol{D}) = \{\varepsilon, 0, 1, 01, 10, 1010\}$

w	$-$	$-$	ε	0	1	01	10	1010
$n, C_0,$	n	C_0	C_{00}	C_{000}	C_{010}	C_{0010}	C_{0100}	C_{010100}
C_{0w0}	8	4	1	0	2	0	1	1

2.2 CSE 法の実現

CSE 法の符号化を実装するために，原論文 [9] では，次のような手順を提案している．

(1) $\boldsymbol{x}^{\langle 2\rangle} = \boldsymbol{x} \cdot \boldsymbol{x}$ に対応する接尾辞木の構成
(2) 深さ n までの頂点での**重み** C_w の算出
(3) Compacted Substring Tree (CST) への変換
(4) n，C_0，および必要な $\{C_{0w0}\}$ の符号化

ステップ (1) では $\boldsymbol{x}^{\langle 2\rangle}$ を対象とすることで，巡回列としての $\boldsymbol{x}$ を模倣している．それに対応する接尾辞木 (suffix tree) を構成するのは，計算量を系列長の線形オーダに保つためである．次に現れる CST は CSE 法のために導入され [9]，その後 [10]，その線形性等が明らかにされたデータ構造である．その名称にもかかわらず，実際には「木」ではなく根つきの有向グラフである．根から各頂点までの有向道に含まれる辺のラベルの連結を w とすると，その頂点は C_w の値を保持している．それらの値を CSE 法の基準に従って符号化しようとするのが原論文 [9] の考え方である．

これに対し山崎ら [24] は，CSE 法の無駄に着目し，符号化で利用する木構造の必要な高さを前もって算出する方法を提案した．たとえば，上記の例では，$C_{0010} = 0$ までで $\boldsymbol{D}$ が一意的に決まってしまい，残りの C_{0100}, C_{010100} の値は不要である．山崎らは，この場合の「必要な $\{C_{0w0}\}$」のうちの '$0w0$' の最大長を $h(\boldsymbol{D})$ もしくは $h(\boldsymbol{x})$ として，それを具体的に計算するアルゴリズムを与えた．その計算法を**図 2** に示す．さらに，符号化においては CST 経由の必要がないことも明らかにし，次のような符号化法を提案した．

(1) $h(\boldsymbol{x})$ の計算
(2) 2 元系列 $\boldsymbol{x} \cdot \boldsymbol{x}[0..h(\boldsymbol{x})-2]$ を対象として，高さを $h(\boldsymbol{x})$ に抑えた接尾辞トライ (suffix trie) を構成する．ただし，深さ $h(\boldsymbol{x})$ の頂点 (葉) に到達するたびに，その重みを表すカウンタの値をインクリメント
(3) 深さ 0 から $h(\boldsymbol{x})-1$ までの頂点での重み C_w の算出 (子頂点の重みの和をその頂点の重みとして頂点に格納)
(4) n，C_0，および必要な $\{C_{0w0}\}$ の符号化 (上述の CSE 符号化法の 2 行目の **for** ループの上限を $\ell = h(\boldsymbol{x}) - 2$ とする.)

上記 2 符号化法とも，入力系列をまずは木構造に変換することで，効率的な処理を可能にしようとしている．しかし，入力系列長が長大になるに従って必要な領域が増大し，単純な配列等にくらべて自明でないプログラミング技法も要求される．そこで本論文では，木構造によらない，より単純な実現法を提案する．

3. BWT 行列による CSE 法の実現

3.1 部分列の出現回数と BWT 行列

提案法の要点は，CSE 符号化法で必要となる諸概念・諸数量を BWT 行列上に対応させることにある．

まず，巡回列としての $\boldsymbol{D}$ において，長さ n ビットの部分列が 2 個あるとき，それらの先頭からの一致部分を両者の lcp (longest common prefix) という．BWT 行列 $\mathcal{M} = \mathcal{M}[0..n-1][0..n-1]$ の隣り合う行要素の lcp の長さを順に配列に格納したものを LCP 配列という．本論文では，これを形式的に次のように定義する．

$$\begin{aligned} &\mathsf{LCP}[0] = -1, \\ &\mathsf{LCP}[i] = \max\{\ell \mid \mathcal{M}[i-1][0..\ell-1] = \mathcal{M}[i][0..\ell-1]\} \\ &\qquad\qquad \text{for } i = 1, 2, \ldots, n-1, \qquad (4) \\ &\mathsf{LCP}[n] = -1. \end{aligned}$$

$\boldsymbol{D}$ が非反復的であるという仮定から，すべての i に対し

符号化に利用する '$0w0$' の最大長 $h(\boldsymbol{x})$ の算出 [24]

1. **Compute** LCP from $\boldsymbol{x}$;
   LCP_MAX := 0; No_of_sus[$n+1$] := 0;
2. **for** $i := 0$ **to** n **do**
   No_of_lcp[i] := 0; No_of_sus[i] := 0;
3. **for** $i := 0$ **to** $n-1$ **do**
   if LCP[i] ≥ 0 **then** No_of_lcp[LCP[i]] $++$;
   No_of_sus[$1 + \max\{$LCP[i], LCP[$i+1$]$\}$] $++$;
   LCP_MAX := $\max\{$LCP_MAX, LCP[i]$\}$;
4. **for** $\ell := 0$ **to** LCP_MAX $+ 1$ **do**
   if (No_of_lcp[ℓ] $= 1$ AND No_of_sus[$\ell+1$] ≥ 1)
   OR No_of_lcp[ℓ] $= 0$
   then return ($\ell + 1$);

図 2 符号化に利用する接尾辞トライの高さ ('$0w0$' の最大長) を算出するアルゴリズム

Fig. 2 Algorithm for computing the height $h(\boldsymbol{x})$ of suffix trie used in CSE.

$\mathsf{LCP}[i] \le n-1$ である $(i = 0, 1, \ldots, n)$．そして，

$$w[i] = \mathcal{M}[i][0..\mathsf{LCP}[i]-1] \text{ for } i = 1, 2, \ldots, n-1$$

を定義すると，

$$\{w[i] \mid i = 1, 2, \ldots, n-1\} = U(\boldsymbol{D}) \tag{5}$$

である．$|U(\boldsymbol{D})| = n-1$ であり，$U(\boldsymbol{D})$ の要素と $w[i]$ が 1 対 1 に対応する．さらに，それぞれ大きさ $n-1$ の配列 S と E を用意し，$i = 1, 2, \ldots, n-1$ に対し，

$$S[i] = \min\{s \mid \mathcal{M}[s][0..\mathsf{LCP}[i]-1] = w[i]\}, \tag{6}$$

$$E[i] = \max\{e \mid \mathcal{M}[e][0..\mathsf{LCP}[i]-1] = w[i]\} \tag{7}$$

と定義する．$\mathcal{M}[S[i]]$ から $\mathcal{M}[E[i]]$ までの連続した領域の各行は $w[i]$ を prefix として有し，また，これらの領域以外で $w[i]$ から始まる行は存在しない．すなわち，

$$C_{w[i]} = E[i] - S[i] + 1 \tag{8}$$

である．また，LCP 配列の定義 (4) から，$i = 1, 2, \ldots, n-1$ に対し，$\mathcal{M}[i-1][\mathsf{LCP}[i]] = 0$，$\mathcal{M}[i][\mathsf{LCP}[i]] = 1$ である．したがって，

$$C_{w[i]0} = (i-1) - S[i] + 1 = i - S[i]. \tag{9}$$

次に，$\boldsymbol{D}$ の BWT 変換，すなわち BWT 行列の右端の列を $B = B[0..n-1] = \mathcal{M}[0..n-1][n-1]$ で表し，$B[0..i]$ に含まれるビット '0' の個数を $R[i]$ で表す，そして，$R[-1] = 0$ と定義すると，$i = 1, 2, \ldots, n-1$ に対し，

$$C_{0w[i]} = R[E[i]] - R[S[i]-1], \tag{10}$$

$$C_{0w[i]0} = R[i-1] - R[S[i]-1] \tag{11}$$

が成り立つ．すなわち，C_{0w0} の値の符号化に際して参照が必要となる諸量の計算は，$w = w[i]$ に対し次のようにまとめられる $(i = 1, 2, \ldots, n-1)$.

まず，C_{0w0} 自身の値は式 (11) によって求められる．C_w, C_{w0}, C_{0w} は，それぞれ，式 (8), (9), (10)，および

$$C_{w1} = C_w - C_{w0}, \tag{12}$$

$$C_{1w} = C_w - C_{0w} \tag{13}$$

である．

3.2 符号化法の枠組

提案する符号化法の全体的な流れをまとめると次のようになる．

(1) 入力系列 $\boldsymbol{x}$ の BWT 変換，および LCP 配列の計算
(2) 配列 $S[1..n-1]$, $E[1..n-1]$, $R[-1..n-1]$ の算出
(3) $h(\boldsymbol{x})$ の計算
(4) $0 \le \mathsf{LCP}[i] \le h(\boldsymbol{x}) - 2$ を満たす $i \in \{1, 2, \ldots, n-1\}$ を LCP 値の昇順に安定整列．その結果を

$$0 = \mathsf{LCP}[i_1] < \mathsf{LCP}[i_2] \le \cdots \le \mathsf{LCP}[i_M] \le h(\boldsymbol{x}) - 2$$

とする．
(5) $j = 1, 2, \ldots, M$ について，$w = w[i_j]$ に対し式 (11) で求まる C_{0w0} の値が式 (2) の範囲にある場合に符号化

3.3 符号化法の詳細と計算量について

まず，符号化法の各ステップの詳細とその時間計算量について述べ，最後にまとめて領域計算量についての考察を行う．

上記のステップ (1) における BWT 変換は，SAIS アルゴリズム [16] を適用することで $O(n)$ 時間で実行することができる．LCP 配列も $O(n)$ で生成できるアルゴリズムが既に知られている ([3],[14] など)．

ステップ (2) の配列 S, E は，[1] の考えを利用することで，いずれも $O(n)$ 時間で求めることができる．もっとも単純な実現例を**図 3** に示した．配列 R も $O(n)$ 時間で算出可能である．ステップ (3) の $h(\boldsymbol{x})$ が図 2 に示したアルゴリズムによって求まることの詳細は省くが，LCP 配列の計算がステップ (1) で済んでいるので，図 2 の計算時間も $O(n)$ である．

ステップ (4) は，4 節で述べる実装においては，$j = 0, 1, \ldots, h(\boldsymbol{x}) - 2$ に対応するバケットを用意したバケットソートを利用した．バケット j には，$\{i \mid \mathsf{LCP}[i] = j\}$ であるような添字 i が昇順に格納される．このようなバケットソートは，$O(n)$ 時間で実行することができる．

以上が符号化の前処理であるのに対し，続くステップ (5) が符号化の主ルーチンである．式 (5) に示したように，$w = w[i_j]$ に対し $w \in U(\boldsymbol{D})$ であるため，$w \in V(\boldsymbol{D})$ であ

配列 $S[1..n-1]$ の計算

1. $\boldsymbol{Q} := \emptyset$; //スタック
2. **for** $i := 1$ **to** $n-1$ **do**
   while $\boldsymbol{Q} \neq \emptyset$ and $\mathsf{LCP}[i] \le \mathsf{LCP}[\mathrm{top}(\boldsymbol{Q})]$ **do**
   $\mathrm{pop}(\boldsymbol{Q})$;
   if $\boldsymbol{Q} = \emptyset$ **then** $S[i] := 0$ **else** $S[i] := \mathrm{top}(\boldsymbol{Q})$;
   push i onto $\boldsymbol{Q}$;

配列 $E[1..n-1]$ の計算

1. $\boldsymbol{Q} := \emptyset$; //スタック
2. **for** $i := n-1$ **to** 1 **do**
   while $\boldsymbol{Q} \neq \emptyset$ and $\mathsf{LCP}[i] \le \mathsf{LCP}[\mathrm{top}(\boldsymbol{Q})]$ **do**
   $\mathrm{pop}(\boldsymbol{Q})$;
   if $\boldsymbol{Q} = \emptyset$ **then** $E[i] := n-1$ **else** $E[i] := \mathrm{top}(\boldsymbol{Q}) - 1$;
   push i onto $\boldsymbol{Q}$;

図 3 配列 S と E の算出．push の回数は $n-1$ 回である．pop の回数は push の回数を上回ることはないので，全体としての計算時間は $O(n)$ である．

Fig. 3 An example of the computation of arrays S and E, which can be performed in $O(n)$ time.

るかの確認を行って，その場合にのみ C_{0w0} の値を符号化する．$|U(\boldsymbol{D})| < n$ であり，ステップ (5) の時間計算量も $O(n)$ である．

以上のように，提案法においては，全ステップを通して $O(n)$ の時間で符号化することができる．ここで必要な領域について簡単に考察する．

まず，提案法で明示的に利用する配列として，LCP, S, E, R がある．また，BWT 行列自身は不要であるが，BWT 変換と LCP の計算に際し，接尾辞配列 (SA) および，その逆配列 (SA^{-1}) を利用している．これらはいずれも，$[0, n)$ に値を取る大きさ n の配列である．この点において，提案法の領域計算量は $O(n \log n)$ である．このほかに，配列 S, E の計算に際して必要となるスタックと LCP 値整列時のバケット管理用の領域も必要となるが，それらもオーダとしては $O(n \log n)$ を超えることはない．

以上の領域計算量を改善する一つの可能性として，配列それぞれの領域を縮小することが考えられる．たとえば，$B[0..i]$ に含まれる '0' の個数を表す $R[i]$ は，簡潔データ構造である完備辞書 [20] と rank 計算の組合せによって $n + o(n)$ の領域を使った定数時間の計算で求めることができる．同じく，LCP 配列に対しても簡潔データ構造が知られている [22] が，主ルーチンで参照されるのは配列 R に加え，配列 S, E である．これらが $O(n)$ まで削減できるかどうかは今後の課題である．

4. 評価と考察

4.1 実験による評価

前節で提案した符号化法をプログラムとして実装し，符号化時間や圧縮率の測定を行った．具体的には，C++で実装した符号化法を gcc 4.5.3 でコンパイルし，Windows 7, Intel(R), Core(TM) i5-2450M CPU@2.50GHz，メモリ 4GB のマシンで計測した．

まず，Calgary Corpus [27] の “bib.txt” の先頭から数キロバイトを 2 元データとみた入力系列を対象として，符号化に要する実時間を測定した．測定した符号化時間は，最初のバイトデータを読み込んで 2 元系列に変換した直後から最後の符号語の出力までの全時間である．ただし，最終的なエントロピー符号化は組み込んでいない．測定結果を**図 4**, **図 5** に示す．図 4 には，比較の対象として 2.2 節で述べた山崎らの符号化法の符号化時間を含めた．

2.2 節で述べたように，山崎らの手法では接尾辞トライを利用するため，入力 2 元系列が長くなるにしたがって頂点数が急激に増大する．そのため，上述の実験環境では 9KB 程度から符号化の処理そのものができなくなっている．一方，提案法は入力系列長にほぼ比例する時間で，ファイル全体の長さまで（109KB, $n = 890,088$）安定した時間で符号化が実現できている．ただし，山崎らの手法は接尾辞トライのかわりに接尾辞木を用いた実装が可能であり [24]，その場合には，符号化可能な系列長も符号化の速度も大幅に改善すると予想される．

続いて圧縮性能の評価結果を**表 1**，**表 2** に示す．圧縮対象として，Canterbury Corpus (Calgary Corpus および Large Corpus を含む) [27] の各ファイルを利用した．Calgary Corpus を対象とした同様の測定は Dubé–Beaudoin [9] においても行われており，比較のために彼らの評価結果 ($\overline{\text{Btf}}$, Btf, BTF) を表 1 にそのまま引用してある．$\overline{\text{Btf}}$ は，入力記号列を 32KB ごとのブロックに分割し，分割したブロックごとに符号化を行った結果，Btf は同じブロックに対して特殊なモデル化を利用した結果，BTF はブロック長 1MB に対し Btf と同様のモデルを適用した結果である．提案法では，このようなブロック分割は行っていない．

上で述べたように，実際のエントロピー符号化は導入していないため，圧縮性能の評価は，入力系列 1 byte (= 8 bit) あたりの理想的な符号長 (モデルのエントロピー) で行った．そのために想定した符号化モデルは，「一様分布モデル」である．これは，C_{0w0} の値が式 (2) の範囲に一様

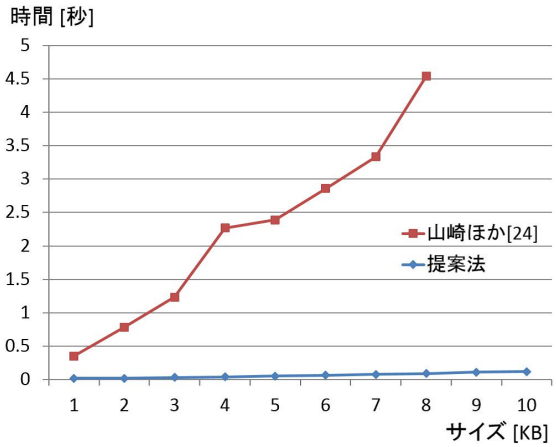

図 4 符号化に要する実時間 (1)
Fig. 4 Comparison of encoding time.

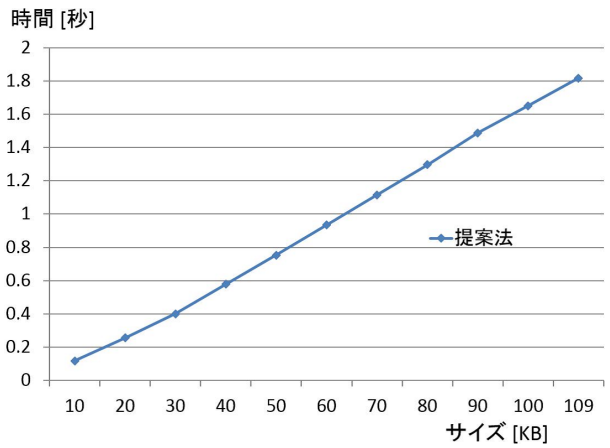

図 5 符号化に要する実時間 (2)
Fig. 5 Encoding time for File “bib.txt.”

表 1 1 byte あたりの平均符号長 [bit]．提案法以外の数値は Calgary Corpus [27] と文献 [9] からの引用．

Table 1 Compression comparisons in bits per byte.

File [KB]	n	gzip-b	bzip2-9	ppmD5	$\overline{\text{Btf}}$	Btf	BTF	提案法
bib [109]	890,088	2.51	1.97	1.89	2.54	2.56	1.98	1.97
book1 [751]	6,150,168	3.25	2.42	2.34	3.14	3.06	2.27	2.41
book2 [597]	4,886,848	2.70	2.06	1.98	2.74	2.72	1.98	2.03
geo [100]	819,200	5.34	4.45	4.96	6.03	5.52	5.35	5.80
news [368]	3,016,872	3.06	2.52	2.42	3.33	3.32	2.52	2.54
obj1 [21]	172,032	3.84	4.01	3.70	5.10	4.46	4.46	4.91
obj2 [241]	1,974,512	2.63	2.48	2.35	3.03	3.02	2.71	2.71
paper1 [52]	425,288	2.79	2.49	2.36	2.79	2.80	2.54	2.54
paper2 [80]	657,592	2.89	2.44	2.34	2.77	2.77	2.41	2.53
pic [501]	4,105,728	0.82	0.78	0.95	2.05	0.79	0.77	1.78
progc [39]	316,888	2.68	2.53	2.40	2.76	2.77	2.60	2.59
progl [70]	573,168	1.80	1.74	1.69	1.90	1.89	1.71	1.71
progp [48]	395,032	1.81	1.74	1.72	1.99	1.96	1.78	1.79
trans [91]	749,560	1.61	1.53	1.50	2.16	2.07	1.60	1.67

表 2 1 byte あたりの平均符号長 [bit]．提案法以外の数値は Canterbury Corpus と Large Corpus [27] からの引用．

Table 2 Compression comparisons in bits per byte.

File [KB]	n	gzip-b	bzip2-9	ppmD5	提案法
alice29 [149]	1,216,712	2.85	2.27	2.20	2.24
asyoulik [122]	1,001,432	3.12	2.53	2.49	2.56
cp.html [24]	196,824	2.59	2.48	2.32	2.53

File [KB]	n	gzip-b	bzip2-9	ppmD5	提案法
E.coli [4530]	37,109,520	2.24	2.16	1.99	2.31
bible [3953]	32,379,136	2.33	1.67	1.58	1.52
world [2415]	19,787,200	2.33	1.58	1.52	1.33

に分布すると仮定したモデルである．値の可能性が式 (3) に示すとおりだけあるので，この場合のモデルのエントロピーは

$$\frac{8}{n}\sum_{\substack{w\in I(\boldsymbol{D}),\\ |w|<h(\boldsymbol{D})-1}} \log_2(\min\{C_{0w}, C_{1w}, C_{w0}, C_{w1}\}+1) \qquad (14)$$

で与えられる．ただし，w が 0 だけの連接，もしくは 1 だけの連接の場合には，式 (2) の上限はタイトではないことが知られているので [23]，その結果も反映させてある．

Dubé–Beaudoin [9] の評価結果 $\overline{\text{Btf}}$ と提案法の評価結果を比較すると，いずれのファイルにおいても提案法が一様に優れていることが観察できる．これは，$\overline{\text{Btf}}$ ではファイルを 32KB ごとに分割し，ブロックごとに符号化を行っているのに対し，提案法ではそのようなブロック分割を行っていないこと，及び，不要な符号語の出力を抑制した結果であると考えられる．

「一様分布モデル」とは異なる符号化モデルも既に提案されている．既存の評価結果の Btf, BTF もそのようなモデルの一種と思われるが，詳細は不明なので，比較検討はしていない．また，[10] で提案された超幾何分布を仮定したモデルや一様分布との切り替えモデルについても，現実の圧縮力を有意に改善する利用法は確立できていないので，圧縮結果の詳細は割愛する．

4.2 符号化法と復号法について

以上述べてきたように，本論文での提案は符号化に関わる部分だけであり，復号法には全く言及していない．ここでは，CSE 法における符号化法と復号法の関連について簡単に考察する．

これまで CSE 法の復号法に具体的に言及している文献は余りない．BWT 変換に基礎を置くという点で本論文の提案法と関連の深い復号法として [26] がある．しかし，この方法は，CSE 法の符号化結果から BWT 行列の左何列かが再現できている状態を前提としているので，完全な復号法としては不十分である．ある程度完全な形で復号法が述べられているのは [24] である．

図 6 に示すように，山崎ら [24] の復号法では，符号化法で利用しなかった CST を利用している．現在のところ，本論文の提案法のように配列のみを利用する復号法が得られていないので，本論文の提案法を実装した符号化プログ

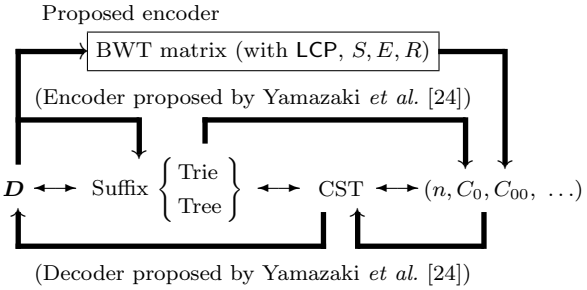

図 6 符号化法，復号法とデータ構造の関係

Fig. 6 Relation among encoder, decoder, and data structures.

ラムの出力結果が正しく復号できることの確認は，山崎らの復号プログラムで行っている．

既に述べたように，CST は有向グラフであり，長さ n ビットの入力記号列に対して，その頂点数が高々 $2n-1$ である [10]．一方，符号化法で木構造を利用する場合，頂点数をこれと同程度に抑えるには接尾辞木を利用する必要がある．しかし，CSE 法に接尾辞木を導入する場合，接尾辞トライを導入する場合に比較してプログラミングが複雑となる．実際，2.2 節の最初に述べた符号化法を提案した原論文 [9] でも，その提案法ではなく，$O(n^2)$ の時間計算量を必要とする別の実装法が試みられている．一方，CST は巡回列に対する接尾辞トライと見ることができる．接尾辞トライに相当するという点において，接尾辞木構築よりもプログラミングが容易であり，頂点数が接尾辞木と同程度という点において，符号化法に使った場合の接尾辞トライよりメモリ消費量が少なくて済む．そのため，CST を利用した復号法は十分実用的であると考えることができる．

以上が，本研究において，復号法ではなく符号化法にのみ着目している理由である．もちろん復号法についても更なる効率化の検討が必要であり，今後の課題である．

5. むすび

無ひずみデータ圧縮法 CSE の符号化法として，BWT 行列を利用したアルゴリズムを提案した．木構造を利用した従来の符号化法と比較して概念的にも実装上も簡潔であり，実際的にも高速であることをプログラムによる実装を通じて明らかにした．これによって，CSE 法に組み込む符号化モデルの現実的な比較が可能になったので，今後はそのようなモデルの比較と新規開発に取り組む計画である．ただし，これまで試験的に調べた限りでは，そのようなモデルによって CSE 法の現実的な圧縮力が有意に改善できることは確認できていない．その理由の一つとして，圧縮対象としたファイルの多くがバイトデータであるのに対し，CSE 法および既存の符号化モデルの対象が 2 元データであることが考えられる．既に，CSE 法の多元アルファベット版が提案されている [13], [19] ので，本論文の提案法がそのような場合に拡張可能かどうか，また，提案法の対象が符号化法のみであるため復号法にも応用可能かどうか，こうした拡張可能性を検討することも今後の課題である．

謝辞 本研究および実験の一部において，本学大学院修了生の山崎世界と金安英明両名の開発した手法 [24] および復号プログラムを活用させていただいた．両名に感謝する．

参考文献

[1] Asano, T., Bereg, S. and Kirkpatrick, D.: Finding nearest larger neighbors, *Efficient Algorithms, Lecture Notes in Computer Science*, vol. 5760, pp. 249–260, 2009.

[2] Adjeroh, D., Bell, T. and Mukherjee, A.: *The Burrows–Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*, Springer, 2008.

[3] Beller, T., Gog, S., Ohlebusch, E. and Schnattinger, T.: Computing the longest common prefix array based on the Burrows–Wheeler transform, *J. of Discrete Algorithms*, vol. 18, pp. 22–31, 2013.

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

[5] Cover, T.M.: Enumerative source coding, *IEEE Trans. Inform. Theory*, vol. IT-19, no. 3, pp. 395–399, 1972.

[6] Crochemore, M. and Rytter, W.: *Text Algorithms*, Oxford University Press, 1994.

[7] Crochemore, M. *et al.*: Data compression using antidictionaries, *Proc. IEEE*, vol. 88, no. 11, pp. 1756–1768, 2000.

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

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

[10] Dubé, D. and Yokoo, H.: The universality and linearity of compression by substring enumeration, *2011 IEEE Int. Symp. Inform. Theory, ISIT 2011*, pp. 1619–1623, Saint-Petersburg, Russia, Aug. 2011.

[11] Iwata, K., Arimura, M. and Shima, Y.: On the maximum redundancy of CSE for i.i.d. sources, *2012 Int. Symp. Inform. Theory and Appl., ISITA2012*, pp. 489–492, Honolulu, Hawaii, Oct. 2012.

[12] 岩田賢一，有村光晴，嶋　優希：Lossless data compression via substring enumeration のマルコフ情報源に対する最悪冗長度，電子情報通信学会技術研究報告，vol. 112, no. 460, IT2012-76, pp. 95–100, March 2013.

[13] 岩田賢一，有村光晴：多値アルファベット版部分列数え上げデータ圧縮法に対する最悪冗長度，電子情報通信学会技術研究報告，vol. 113, no. 483, IT2013-55, pp. 7–12, March 2014.

[14] Kasai, T., Lee, G., Arimura, H., Arikawa, S. and Park, K.: Linear-time longest-common-prefix computation in suffix arrays and its applications, *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001*, pp. 181–192, *Lecture Notes in Computer Science*, vol. 2089, 2001.

[15] Gagie, T. and Manzini, G.: Move-to-front, distance coding, and inversion frequencies revisited, *Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007*, pp. 71–82, *Lecture Notes in Computer Science*, vol. 4580, 2007.

[16] Nong, G., Zhang, S. and Chan, W.H.: Linear suffix array construction by almost pure induced-sorting, *Proc. 2009 Data Compression Conf., DCC 2009*, pp. 193–202, 2009.

[17] Öktem, L. and Astola, J.: Hierarchical enumerative coding of first-order Markovian binary sources, *Electron. Lett.*, vol. 35, pp. 2003–2005, 1999.

[18] Ota, T. and Morita, H.: On antidictionary coding based on compacted substring automaton, *2013 IEEE Int. Symp. Inform. Theory, ISIT 2013*, pp. 1754–1758, Istanbul, Turkey, July 2013.

[19] 太田隆博，森田啓義：多値アルファベットに対するユニバーサルな 2 パス反辞書符号化法，第 36 回情報理論と

その応用シンポジウム，SITA2013, pp. 26–29, Shizuoka, Japan, Nov. 2013.
[20] 岡野原大輔：高速文字列解析の世界—データ圧縮・全文検索・テキストマイニング，岩波書店，2012.
[21] Ruskey, F. and Sawada, J.: An efficient algorithm for generating necklaces with fixed density, *SIAM J. Comput.*, vol. 29, no. 2, pp. 671–684, 1999.
[22] Sadakane, K.: Succinct representations of lcp information and improvements in the compressed suffix arrays, *Proc. 13th ACM-SIAM Symp. Discrete Algorithms, SODA '02*, pp. 225–232, 2002.
[23] 嶋　優希, 岩田賢一, 有村光晴 : Lossless data compression via substring enumeration におけるある改良，電子情報通信学会技術研究報告，vol. 111, no. 51, IT2011-1, pp. 1–6, May 2011.
[24] 山崎世界, 金安英明, 横尾英俊 : Compression by substring enumeration データ圧縮法の効率的実現，電子情報通信学会技術研究報告，vol. 113, no. 411, IT2013-51, pp. 35–40, Jan. 2014.
[25] Yokoo, H.: A comparison of methods for redundancy reduction in recurrence time coding, *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3793–3799, 2006.
[26] Yokoo, H.: Asymptotic optimal lossless compression via the CSE technique, *2011 1st Int. Conf. on Data Compression, Communication and Processing, CCP 2011*, pp. 1–8, Palinuro, Italy, June 2011.
[27] *The Canterbury Corpus*，入手先 `<http://corpus.canterbury.ac.nz/index.html>` 2014 年 5 月 8 日アクセス.