Lossless Compression of Grayscale and Colour Images Using Multidimensional CSE

Danny Dubé
Université Laval
Quebec City, Canada
Danny.Dube@ift.ulaval.ca

***Abstract*—Originally, compression by substring enumeration (CSE) is a lossless compression technique that is intended for strings of bits. As such, the original version is one-dimensional. An extension of CSE for strings drawn from a larger alphabet has later been introduced. Also, CSE has recently been extended to two-dimensional (2D) data. As such, 2D CSE can be used directly to compress images. Unfortunately, CSE generally does not perform on data drawn from large alphabets as well as on binary data. This means that, although we can expect 2D CSE to perform well on bilevel images, we must expect a loss of performance on grayscale and colour images, where the alphabet sizes may be 2^8 and 2^{24}, respectively, as in common image formats. As a workaround for this difficulty, we propose to handle grayscale and colour images by remaining in the realm of binary data but by extending CSE to higher dimensions. Grayscale images may have the levels of gray of their pixels decomposed into bit planes and, then, get compressed using a 3D CSE. Colour images may have their three colour channels treated as yet another dimension and, then, get compressed using a 4D CSE. Actual empirical measurements are deferred to another paper as we do not have a working implementation of multidimensional CSE yet.**

***Index Terms*—data compression, image compression, lossless data compression, bit planes, multidimensional data**

I. Introduction

Many lossless image compressors proceed by predicting and encoding the colour of pixels. In certain compression techniques, it is instead the prediction error that is encoded. The process is applied on one pixel after the other, in some predefined order; e.g. row by row. The prediction of the colour of a pixel can be improved by taking into account the colours of surrounding, already described pixels. The collection of these surrounding pixels is usually referred to as the *context* of the prediction. This frequently adopted and general strategy can be seen as the two-dimensional (2D) equivalent of the *prediction by partial matching* (PPM) lossless text compression technique [1].

Not all text compression techniques lend themselves to a natural adaptation to image compression. Some techniques have no 2D equivalent, either because the presumption of linearity of the data is too inherent or simply because a 2D equivalent hasn't been figured out yet. It was the case of a technique called *compression by substring enumeration* (CSE) for which it is only recently that a 2D variant has been introduced. In this paper, we point out that certain kinds of images can automatically be handled by 2D CSE and, moreover, we propose to use a straightforward generalization of CSE to higher dimensions in order to handle all the kinds of images.

The paper is structured as follows. Section II presents background information on CSE, with a focus on the research developments that are relevant to the technique proposed here. Section III introduces the generalization of CSE for data represented under an arbitrary number of dimensions. Section IV proposes to handle grayscale images using 3D CSE. Similarly, Section V proposes to handle colour images using 4D CSE.

II. Background on CSE

A. Original CSE

CSE has been introduced by Dubé and Beaudoin in 2010 [2]. It is intended to compress a string of bits, which we refer to as the input text. The basic idea in CSE is exactly what its name says: it indicates how many of each possible substring there exists in the input text. Figure 1 shows an example of enumeration. The table indicates how many times a particular substring occurs. Implicitly, when a substring is not mentioned in the table, it means that it does not occur in the input text. There are two things we need to mention about this example. First, CSE considers the input text to be circular; i.e. to the right of the last bit of the text, there lies the first bit of the text; analogously in the other direction. Second, the table merely illustrates the kind of information that is manipulated by CSE. It is not intended to be an exact description of how CSE proceeds to the enumeration. We refer the reader to previous work to obtain full details about the procedure followed by CSE.

To make mathematical reasoning easier, we denote the number of occurrences of substring w by C_w. For example, the indication "2×010" in Figure 1 translates into the fact that C_{010} is 2. Note that C_w exists even when w does not occur in the input text; then, we simply have $C_w = 0$. Technically, a "substring" may even be longer than the input text. In that case, the substring wraps around the boundary one or multiple times. Note that the table in Figure 1 intentionally includes a row for substrings of length 8 to illustrate that point. The numbers of occurrences obey the following equations, where N is the length of the input text and ϵ denotes the empty string.

Length	Substrings						
0	$7 \times \epsilon$						
1	3×0			4×1			
2	3×01			3×10			1×11
3	2×010		1×011	3×101			1×110
4	2×0101		1×0110	2×1010		1×1011	1×1101
5	1×01010	1×01011	1×01101	2×10101		1×10110	1×11010
6	1×010101	1×010110	1×011010	1×101010	1×101011	1×101101	1×110101
7	1×0101011	1×0101101	1×0110101	1×1010101	1×1010110	1×1011010	1×1101010
8	1×01010110	1×01011010	1×01101010	1×10101011	1×10101101	1×10110101	1×11010101

Fig. 1. Enumeration table for the substrings of 1011010.

$$C_\epsilon = N \quad (1)$$

$$C_{0w} + C_{1w} = C_w = C_{w0} + C_{w1}, \quad \forall w \in \{0,1\}^* \quad (2)$$

$$\sum_{w \in \{0,1\}^n} C_w = N, \quad \forall n \in \mathbb{N} \quad (3)$$

Although we do not present the full procedure for CSE, we must mention that the enumeration strictly proceeds row by row, i.e. by completely enumerating all the substrings of length ℓ (and less) before enumerating those of length $\ell + 1$. This order allows CSE to exploit the information it has about the shorter substrings to improve its predictions about the longer substrings. In particular, the following bounds constrain the set of values that are considered in the prediction of C_{0w0}.

$$\max(0, C_{0w} - C_{w1}) \leq C_{0w0} \leq \min(C_{0w}, C_{w0}) \quad (4)$$

Here, the prediction on the number of occurrences of the substring $0w0$ takes advantage of the preliminary knowledge about substrings like $0w$, $w0$, and $w1$. We call the central part w the *core* of $0w0$. This notion of core is crucial in CSE. Below, we see that it is too, in 2D CSE.

Some of the developments on CSE include the theoretical analysis of its performance [3]–[6], improvements on the bounds used in the prediction of C_{0w0} [7]–[9], its relation to antidictionaries [10], [11], and approaches to achieve a practical implementation [12]–[14].

B. CSE with a Larger Alphabet

Multiple researchers have proposed an extension of CSE to larger alphabets [15]–[21]. When considering larger alphabets, we lose the simplicity of Eq. 2. The equivalent for a larger alphabet Σ is the following.

$$\sum_{a \in \Sigma} C_{aw} = C_w = \sum_{b \in \Sigma} C_{wb}, \quad \forall w \in \Sigma^* \quad (5)$$

In particular, when CSE tries to predict C_{awb}, there are many counters that "rival" with C_{awb} to obtain their share of occurrence units from C_{aw} and C_{wb}. A nice illustration of that effect is made using a *contingency table*, which has been introduced by Ota et al. [19]. When predicting C_{awb}, the counters for all the extensions of core w have to be considered. Figure 2(b) shows this table. The first and last symbols in a general alphabet are symbolically denoted by ¢ and \$, respectively. Bounds can still be obtained but they tend to be quite loose. Ota et al. explain how to best use the information carried by the counters that have already been encoded; these appear in gray in Figure 2(b). On the other hand, Figure 2(a) shows the contingency table that one faces when predicting C_{0w0} (denoted $C_{¢w¢}$ here). Only the top-left counter needs to be predicted and encoded, since the other three can be deduced afterwards. The looser bounds in the case of larger alphabets has the tendency to worsen the performance of CSE, unfortunately.

$C_{¢w¢}$	$C_{¢w\$}$	$C_{¢w}$
$C_{\$w¢}$	$C_{\$w\$}$	$C_{\$w}$
$C_{w¢}$	$C_{w\$}$	C_w

(a) Binary alphabet

$C_{¢w¢}$	$\cdots$	$C_{¢wb}$	$\cdots$ $C_{¢w\$}$	$C_{¢w}$
$\vdots$		$\vdots$	$\vdots$	$\vdots$
$C_{aw¢}$	$\cdots$	C_{awb}	$\cdots$ $C_{aw\$}$	C_{aw}
$\vdots$		$\vdots$	$\vdots$	$\vdots$
$C_{\$w¢}$	$\cdots$	$C_{\$wb}$	$\cdots$ $C_{\$w\$}$	$C_{\$w}$
$C_{w¢}$	$\cdots$	C_{wb}	$\cdots$ $C_{w\$}$	C_w

(b) Larger alphabet

Fig. 2. Contingency tables.

C. Phase-Aware CSE

Béliveau and Dubé have rather addressed the issue of larger alphabets in a different way [22]. The input text, drawn from large alphabet Σ, is first converted into a binary text in a straightforward way: each original symbol is transformed into a string of $\lceil \log_2 |\Sigma| \rceil$ bits; all of the latter get concatenated together. Then, CSE compresses the converted text. However, in order to preserve CSE from confusing bits with different significance, the notion of *phase* has been integrated directly into the procedure of CSE. For instance, a byte of value 75 would be converted into the string of bits $0_7 1_6 0_5 0_4 1_3 0_2 1_1 1_0$. The indices on the bits are used to mark the phase of each bit, relative to the separations between bytes. This phase marking causes CSE to work with 8 distinct binary alphabets. Thanks to the markings, the prediction made for C_{011010} in phase, say, 6 is kept separate from the predictions for C_{011010} in the 7 other phases. Indeed, it is normal to expect different statistical behaviours from bits located at different phases. Earlier work has indirectly given a sense of phase to CSE by using synchronization codes [23]–[25].

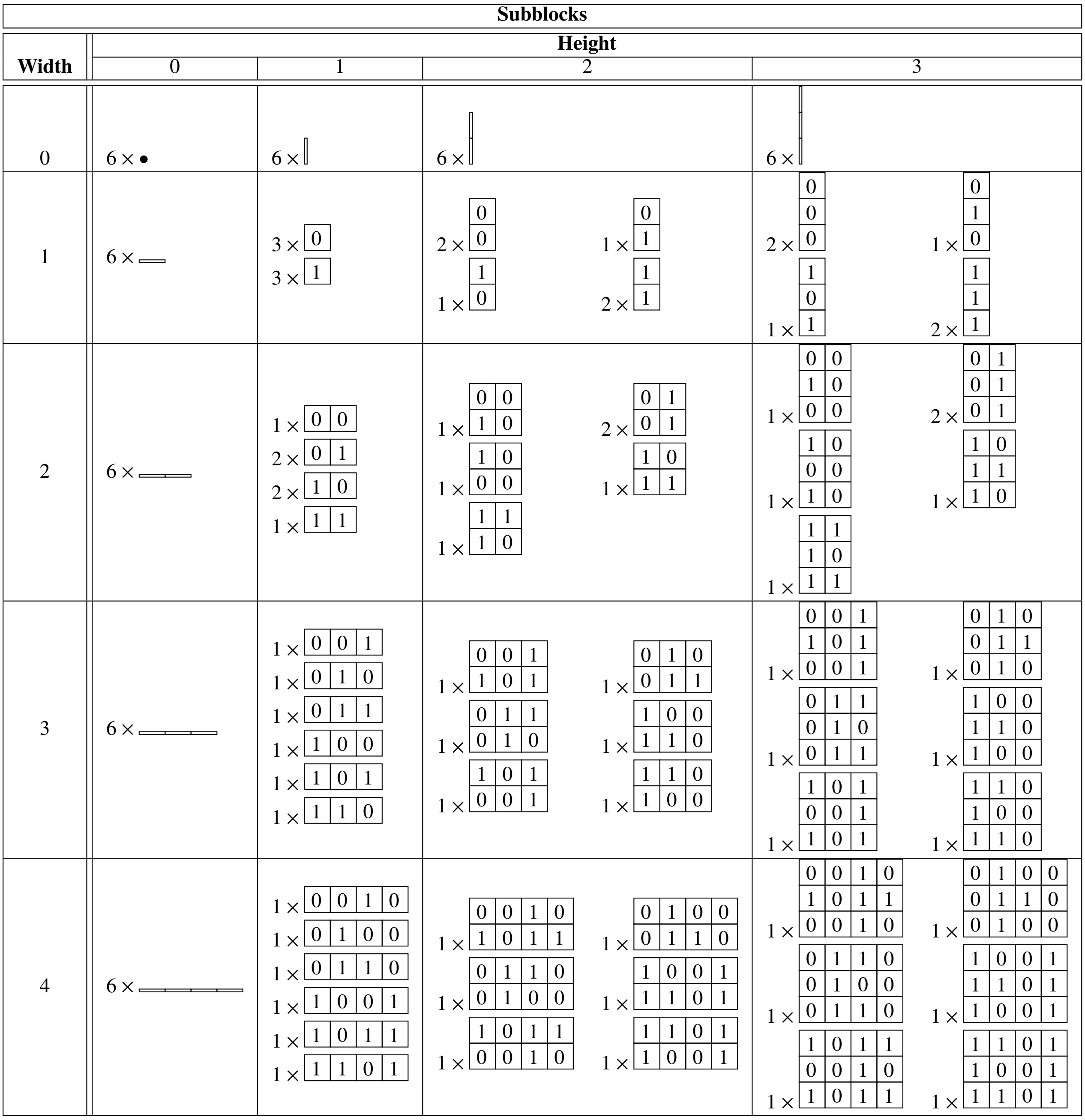

Fig. 3. Enumeration table for the subblocks of $\begin{smallmatrix}0&1&0\\0&1&1\end{smallmatrix}$.

D. Two-Dimensional CSE

In 2017, Ota et al. have introduced a 2D variant of CSE [26]. Instead of compressing a string drawn from Σ, 2D CSE compresses a rectangular array of symbols drawn from Σ. Like 1D CSE, 2D CSE considers the input block to be circular. In fact, the block is circular in each of the two dimensions: to the right of the last column, there lies the first column; below the last row, there lies the first row; and so on. Although 2D CSE could readily handle input blocks drawn from a large alphabet, we restrict our presentation to the case of the binary alphabet. Figure 3 shows the enumeration of the subblocks for an example input block. Once again, a subblock may be wider or taller than the input block. This is illustrated in the table by considering subblocks wider by one column and subblocks higher by one row. Notice how bits start to repeat in subblocks that are wider or higher than the input block.

An extra complication of 2D CSE is that predicting and encoding the number of occurrences of a particular subblock can be made by viewing it as either a horizontal or a vertical extension of smaller subblocks. This fact is depicted in Figure 4.

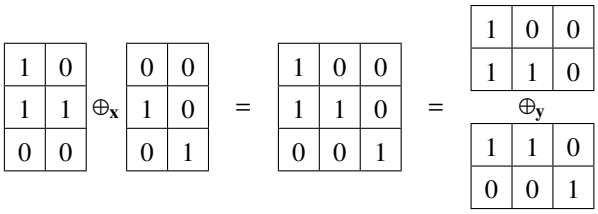

Fig. 4. Extended blocks as horizontal or vertical extensions.

Let us introduce some notation. Let $\Sigma_{w,h}$ be the set of rectangular arrays of width w and height h of symbols drawn from Σ. Let '$\cdot_{\mathbf{x}}$' be the operator that horizontally concatenates two blocks of sizes w' by h and w'' by h, respectively. The resulting block has size w by h, where w is $w'+w''$. Likewise, let '$\cdot_{\mathbf{y}}$' be the operator that vertically concatenates two blocks of sizes w by h' and w by h'', respectively. Extended block B is the *horizontal extension* of B' and B'', denoted $B' \oplus_{\mathbf{x}} B''$, if there exist $E_L, E_R \in \Sigma_{1,h}$ and $D \in \Sigma_{w,h}$, such that $B' = E_L \cdot_{\mathbf{x}} D$, $B'' = D \cdot_{\mathbf{x}} E_R$, and $B = E_L \cdot_{\mathbf{x}} D \cdot_{\mathbf{x}} E_R$. Similarly, B is the *vertical extension* of B' and B'', denoted $B' \oplus_{\mathbf{y}} B''$, if there exist $E_t, E_b \in \Sigma_{w,1}$ and $D \in \Sigma_{w,h}$, such that $B' = E_t \cdot_{\mathbf{y}} D$, $B'' = D \cdot_{\mathbf{y}} E_b$, and $B = E_t \cdot_{\mathbf{y}} D \cdot_{\mathbf{y}} E_b$. Note that D is the core, in either direction.

Given that an extended block B can be seen as either a horizontal or a vertical extension whenever its size is non trivial, this leads to the existence of two distinct pairs of bounds, one per direction. Each pair of bounds is similar to those obtained in 1D CSE in Eq. 4. Ota et al. propose to calculate both pairs of bounds and then to choose the direction that leads to the pair of bounds that constrain C_B the most. Theoretically, contingency tables can be defined for 2D CSE. However, their sizes are huge. For instance, if we consider B to be a horizontal extension, i.e. $B = E_L \cdot_{\mathbf{x}} D \cdot_{\mathbf{x}} E_R$, the number of rows of the table would be the number of different columns E_L that it would be possible to concatenate to the left of D and the number of columns of the table would be the number of different columns E_R that it would be possible to concatenate to the right of D. It is not clear, due to the novelty of 2D CSE and the actual sizes of the contingency tables, how to make the technique fast. The paper on 2D CSE does not include experimental results [26].

III. Generalization of CSE to Higher Dimensions

Following the work by Ota et al., it is natural to extend 2D CSE to more dimensions. In k dimensions, we may simply define k concatenation operators '$\cdot_1$', ..., '$\cdot_k$', one per dimension, and then derive the corresponding k extension operators '$\oplus_1$', ..., '$\oplus_k$'. We merely state that it is simple to *mathematically* define the operators and generalize the equations introduced by Ota et al. Note that we do not describe any means to build a simple, fast, and efficient implementation of 3D CSE, 4D CSE, ...

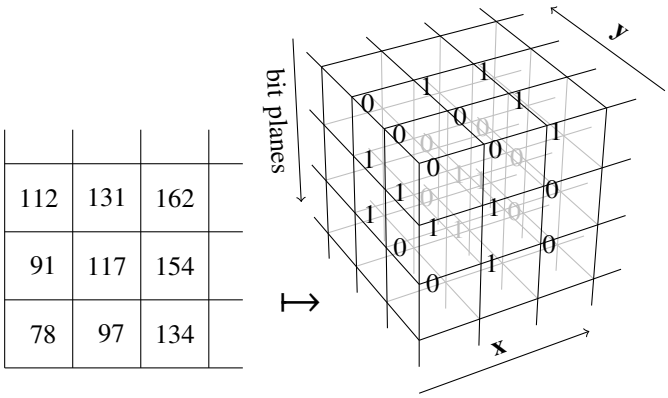

Fig. 5. Decomposition of a grayscale image into a 3D block of bits.

IV. Handling of Grayscale Images

As presented by Ota et al., 2D CSE could in principle handle all kinds of images: bilevel, grayscale, and colour. Clearly, bilevel images could readily be handled by 2D CSE. But in the case of grayscale images, the alphabet would typically be quite large. And in the case of colour images, the alphabet could be ridiculously large; e.g., when the colour of each pixel is encoded using three bytes. Recall that large alphabets lead to large (to the square) contingency tables. Moreover, due to the inefficiency observed in the case of non-binary alphabets, we believe that non-binary alphabets should be avoided.

Let us propose a different way to handle grayscale images, to start with. A grayscale image may be converted into a 3D block of bits, as illustrated in Figure 5. The **x** and **y** dimensions of the image are preserved in the 3D block. A third dimension is introduced to accommodate for the decomposition of the levels of gray into bit planes. As in the approach by Béliveau and Dubé [22], each gray level drawn from the alphabet Σ is converted into a bit string of length $\lceil \log_2 |\Sigma| \rceil$. Due to the various statistical behaviours to be expected from each of the $\lceil \log_2 |\Sigma| \rceil$ bits of the converted gray levels, we argue that we should make CSE aware of the phase of the bits in this third dimension. This conversion would turn a "regular" 3D CSE compressor into a compressor specialized from grayscale images. Indeed, 3D CSE would not need to be aware in any way that the 3D block of bits originates from an image.

V. Handling of Colour Images

In a similar way, we propose to handle the compression of colour images by first converting the images into 4D blocks of bits. In this case, a fourth dimension is used to separate the three colour channels. Figure 6 illustrates the conversion. Of course, illustrating 4D objects can only be suggestive, at best. By using the picture of three bits that connect to the next, we suggest that the fourth dimension is used to list the bits for the successive bits of a particular significance for a particular pixel of the original image. We propose that the dimension of the colour channels should also require phase

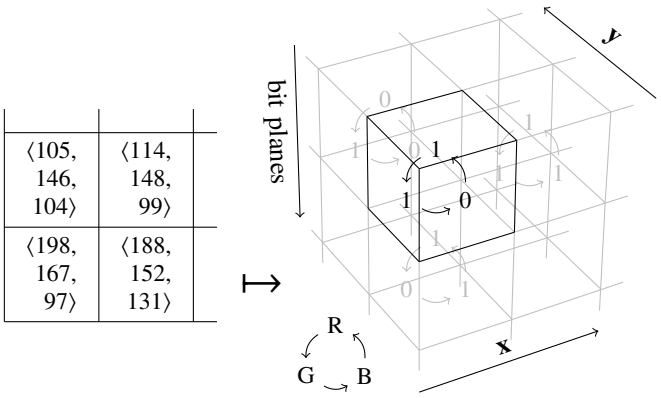

Fig. 6. Decomposition of a colour image into a 4D block of bits.

awareness from the 4D CSE compressor. To summarize the dimensions and their characteristics, the 4D block would go along the **x**, **y**, significance, and colour directions, where the first two dimensions should be left phase unaware and the last two, phase aware.

VI. Conclusion

We have presented a theoretical proposition to use CSE as an image compressor. The original CSE is intended to compress strings of bits. Recently, a 2D variant of CSE has been proposed by Ota et al. Also, variants of CSE that are able to handle strings drawn for a non-binary alphabet had been introduced earlier. Unfortunately, CSE seems to suffer from lower efficiency on non-binary data. Here, we have proposed to remain in the realm of binary data for the compression of images by considering CSE variants of higher dimensions and turning gray levels into bit planes and coloured pixels into vectors in yet another dimension. We describe our proposition as "theoretical" since we have not run experiments using a multidimensional CSE. In fact, we do not have an implementation of it, yet.

There remains much work to carry out from this point. We need to implement a multidimensional variant of CSE. A sub-task of this consists in dealing with the huge size of the contingency tables, once 2D (or higher) CSE is considered. Once a multidimensional CSE is implemented, a relatively simple extra task consists in implementing the conversions from grayscale or coloured images to multidimensional blocks of bits.

Acknowledgement

The author wishes to thank those involved in the review process for the valuable comments.

References

[1] J. G. Cleary and W. J. Teahan, "Unbounded length contexts for PPM," *The Computer Journal*, vol. 40, no. 2/3, pp. 67–75, 1997.

[2] D. Dubé and V. Beaudoin, "Lossless data compression via substring enumeration," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, March 2010, pp. 229–238.

[3] D. Dubé and H. Yokoo, "The universality and linearity of compression by substring enumeration," in *Proceedings of the International Symposium on Information Theory*, Saint-Petersburg, Russia, July 2011, pp. 1519–1523.

[4] K. Iwata, M. Arimura, and Y. Shima, "On the maximum redundancy of CSE for i.i.d. sources," in *Proceedings of the International Symposium on Information Theory and Applications*, Honolulu, Hawaii, USA, October 2012, pp. 489–492.

[5] ——, "Maximum redundancy of lossless data compression via substring enumeration for Markov sources," IEICE Technical Report, Tech. Rep. IT2012-76, March 2013, (in Japanese).

[6] ——, "Evaluation of maximum redundancy of data compression via substring enumeration for *k*-th order Markov sources," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 8, pp. 1754–1760, 2014.

[7] ——, "An improvement in lossless data compression via substring enumeration," in *Proceedings of the IEEE/ACIS International Conference on Computer and Information Science*, Sanya, Hainan Island, China, May 2011, pp. 219–223.

[8] H. Yokoo, "Asymptotic optimal lossless compression via the CSE technique," in *Proceedings of the International Conference on Data Compression, Communications and Processing*, Palinuro, Italy, June 2011, pp. 11–18.

[9] ——, "An information-theoretic interpretation of the CSE lossless compression scheme," in *Proceedings of IT*, vol. IEICE-111, no. 390, University of Tsukuba, Japan, January 2012, pp. 37–42, (in Japanese).

[10] T. Ota and H. Morita, "Relationship between antidictionary automata and compacted substring automata," in *Proceedings of the Information Theory and Applications Workshop*, San Diego, CA, USA, January 2013, pp. 1–4.

[11] ——, "On antidictionary coding based on compacted substring automaton," in *Proceedings of the International Symposium on Information Theory*, Istanbul, Turkey, July 2013, pp. 1754–1758.

[12] S. Kanai and H. Yokoo, "Implementation of compression by substring enumeration via BWT matrix," *Information Processing Society of Japan SIG Notes*, vol. 2014-AL-148, no. 7, pp. 1–8, June 2014, (in Japanese).

[13] S. Kanai, H. Yokoo, K. Yamazaki, and H. Kaneyasu, "Efficient implementation and empirical evaluation of compression by substring enumeration," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99-A, no. 2, pp. 601–611, February 2016.

[14] K. Yamazaki, H. Kaneyasu, and H. Yokoo, "Efficient implementation of compression by substring enumeration," IEICE Technical Report, Tech. Rep. IT2013-51, January 2014, (in Japanese).

[15] K. Iwata and M. Arimura, "Maximum redundancy of lossless data compression via substring enumeration with a finite alphabet," IEICE Technical Report, Tech. Rep. IT2013-55, 2014.

[16] ——, "Lossless data compression via substring enumeration for k-th order Markov sources with a finite alphabet," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, April 2015, p. 452.

[17] ——, "Lossless data compression via substring enumeration for *k*-th order Markov sources with a finite alphabet," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99-A, no. 12, pp. 2130–2135, 2016.

[18] T. Ota and H. Morita, "On a universal antidictionary coding for stationary ergodic sources with finite alphabet," in *Proceedings of the International Symposium on Information Theory and Applications*, Melbourne, Australia, October 2014, pp. 294–298.

[19] T. Ota, H. Morita, and A. Manada, "Compression by substring enumeration with a finite alphabet using sorting," in *Proceedings of the International Symposium on Information Theory and Applications*, Singapore, October 2018, pp. 587–591.

[20] S. Sakuma, K. Narisawa, and A. Shinohara, "Generalization of efficient implementation of compression by substring enumeration—finite alphabet and explicit phase awareness," in *Proceedings of COMP*, vol. IEICE-115, no. 205, Shinshu University, Japan, September 2015, pp. 13–20, (in Japanese).

[21] ——, "Generalization of efficient implementation of compression by substring enumeration," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, March 2016, p. 630.

[22] M. Béliveau and D. Dubé, "Improving compression via substring enumeration by explicit phase awareness," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, March 2014, p. 399.

[23] D. Dubé, "Using synchronization bits to boost compression by substring enumeration," in *Proceedings of the International Symposium on Information Theory and its Applications*, Taichung, Taiwan, October 2010, pp. 82–87.

[24] ——, "On the use of stronger synchronization to boost compression by substring enumeration," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, March 2011, p. 454.

[25] D. Vohl, C.-G. Quimper, and D. Dubé, "Finding synchronization codes to boost compression by substring enumeration," in *Proceedings of the International Workshop on Constraint Modelling and Reformulation*, Quebec City, Quebec, Canada, October 2012.

[26] T. Ota and H. Morita, "Two-dimensional source coding by means of subblock enumeration," in *Proceedings of the International Symposium on Information Theory*, Aachen, Germany, July 2017, pp. 311–315.