Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 861093, 10 pages
http://dx.doi.org/10.1155/2015/861093

Research Article

An Effective Error Correction Scheme for Arithmetic Coding

Qiuzhen Lin,[1,2] Kwok-Wo Wong,[2] Ming Li,[3,4] and Jianyong Chen[1]

[1] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China*
[2] *Department of Electronic Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong 999077, Hong Kong*
[3] *Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, Shanghai 200241, China*
[4] *School of Information Science and Technology, East China Normal University, Shanghai 200241, China*

Correspondence should be addressed to Jianyong Chen; cjyok2000@hotmail.com

Received 9 September 2014; Accepted 28 November 2014

Academic Editor: Ezzat G. Bakhoum

We propose an effective error correction technique for arithmetic coding with forbidden symbol. By predicting the occurrence of the subsequent forbidden symbols, the forbidden region is actually expanded and theoretically, a better error correction performance can be achieved. Moreover, a generalized stack algorithm is exploited to detect the forbidden symbol beforehand. The proposed approach is combined with the *maximum a posteriori* (MAP) metric to keep the highly probable decoding paths in the stack. Simulation results justify that our scheme performs better than the existing MAP methods on the error correction performance, especially at a low coding rate.

1. Introduction

Traditionally, channel coding is performed after source coding to protect the compressed bit stream sent over a noisy channel. For example, an image file is first compressed by using Discrete Cosine Transformation or arithmetic coding [1–3] with high coding efficiency. Then, the compressed sequence is further protected by Turbo code [4] or Hamming code [5] against channel noise. This traditional separate scheme lacks the cooperation between source and channel coding processes and may not result in the optimal performance. Recent studies have revealed that the joint operation of them leads to some advantages when compared with the traditional separately operated approach [6–9]. As certain implicit redundancy still exists in the bit streams when the encoder cannot ideally decorrelate the source symbols, it can be utilized in the joint scheme to improve the overall error correcting performance. Thus, it is possible for the joint scheme to outperform the separate approach [10].

Early works on joint source-channel coding were devoted to the study of error resilience in variable length codes (VLC). In particular, most of which were focused on the resynchronization ability of Huffman code [7–9]. The corresponding hard and soft decoding schemes based on maximum likelihood (ML) or MAP metrics are well-studied for a binary symmetric channel (BSC) with additive white Gaussian noise (AWGN). As arithmetic coding (AC) represents a source symbol using a fractional number of bits, it leads to a better compression efficiency and achieves the optimal entropy coding. However, the high compression ratio makes the codeword more sensitive to channel noise and is difficult to be resynchronized. Therefore, there is a growing interest in improving the robustness of AC against channel noise.

In [11], a forbidden symbol introduced by a reduction in the coding interval is adopted to detect the transmission error continuously. These errors can be detected when the forbidden region is visited. This continuous nature in error detection is exploited to improve the overall performance of the communication system [12]. It provides a tradeoff between the extra redundancy and the delay in detecting an error since its occurrence. Instead of the forbidden symbol, the insertion of markers in some particular positions of the input sequence plays the role of synchronization between the encoder and the decoder [13]. The markers which do not appear in the expected positions indicate transmission errors. Three strategies for the selection of the markers were studied in [13]. A better compression ratio can be achieved using an adaptive [14] or an artificial marker scheme [15]. The adaptive marker scheme selects the most frequent source symbol as the marker symbol while the artificial marker scheme creates an artificial marker with an arbitrary probability. Making use of the error detection capacity of AC, error correction is performed by sequential decoding, which successively removes the erroneous decoding paths. In [16], depth-first and breadth-first decoding algorithms were proposed with binary branching based on a null zone. The decoding paths are discarded due to the error detection capacity of the forbidden symbol. All the decoding paths with the lowest Hamming distance from the received sequence are preserved in a list.

In [17], a MAP criterion based on the context-based AC was proposed with the insertion of synchronization markers, where the symbol clock and the bit clock models were analyzed. The iterative decoding of error resilient AC concatenated with a convolutional code is adopted and its error correcting capability is validated with the transmission of images over an AWGN channel. A novel MAP decoding approach based on the forbidden symbol was proposed in [10], with a high flexibility in adjusting the coding rate. Sequential decoding algorithms, such as stack algorithm and M-algorithm, are adopted and the proposed system outperforms the separate approach based on convolutional codes in terms of error correcting capability. It is serially concatenated with channel codes and iterative decoding is employed to further improve the overall performance [18, 19]. Chaos phenomenon, which generally exists in complex systems [20, 21], is also observed during the iterative decoding procedures. Thus, chaos control techniques can be adopted to further enhance the error correction performance [22–24]. A sequential MAP estimation for CABAC coder was proposed in [25], which employs an improved sequential decoding technique to determine the tradeoff between complexity and efficiency. In [26], a look-ahead technique for AC decoder was proposed to allow quick error detection. Considering the improvement in the implementation efficiency, AC can be modeled as a finite-state machine corresponding to a variable-length trellis code. The trellis code based on AC was proposed in [27, 28], where a list Viterbi decoding algorithm is applied on the corresponding trellis code and a cyclic redundancy check code is employed for detecting small Hamming-distance errors. The free distance of the corresponding AC-based VLC and its theoretical error correction performance were investigated in [29, 30]. Besides that, the practical implementations on this joint source-channel coding scheme were studied in [31, 32] for high coding speed.

The error detecting capability of AC was analyzed in our previous paper [15]. Here we extend our previous work to tackle the problem of error correction in AC. An effective error correction technique utilizing the forbidden symbol is proposed, which predicts the occurrence of the subsequent forbidden symbols. With our approach, the forbidden region is theoretically expanded and so a better error correction performance is achieved. Furthermore, a generalized stack

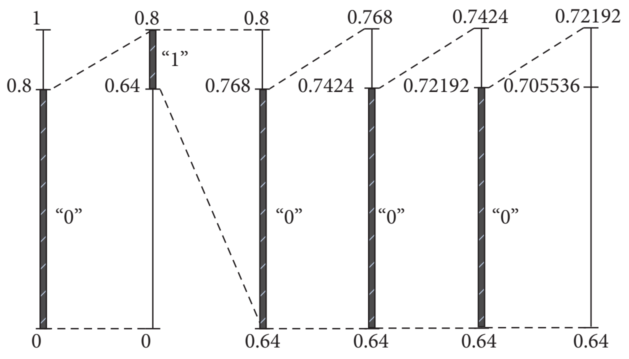


Figure 1: The arithmetic coding steps for encoding the sequence "01000."

algorithm (SA) extending 2^k branches from the best node is also studied for the detection of the forbidden symbol beforehand. The MAP metric [10] is integrated with our approach to preserve the most probable decoding paths in the stack. The idea of our approach was briefly presented in [33], which mainly focuses on the forecasting of the forbidden symbols. Here, the procedures of AC with forecasted forbidden symbols are described in detail. More analyses and simulation results are provided to justify that the proposed scheme outperforms the look-ahead scheme [26] and the original MAP scheme [10] on the error correction performance, especially at a low coding rate.

The rest of this paper is organized as follows. The background of AC is reviewed in Section 2. The proposed scheme is described in Section 3, where the estimation of the subsequent forbidden symbols and the generalized SA are introduced. Simulation results are presented in Section 4 to show the improvement of our scheme. Finally, conclusions are drawn in Section 5.

2. Background of Arithmetic Coding

Arithmetic coding is an iterative operation, which recursively assigns the coding interval to a sequence of source symbols. In general, a prior source model is required, which initializes the coding interval according to the occurrence probabilities of the source symbols. Considering the binary case that the occurrence probabilities of "0" and "1" are correspondingly 0.8 and 0.2, the coding units $[0, 0.8)$ and $[0.8, 1)$ are then assigned to the symbols "0" and "1," respectively. The arithmetic coding steps for encoding the source sequence "01000" are illustrated in Figure 1.

At last, the final coding unit $[0.64, 0.72192)$ is obtained, within which any real value can be selected and exported as the compressed bits. Theoretically, it is guaranteed to obtain a compressed sequence with the shortest length using $\lceil -\log(0.72192 - 0.64)\rceil = 4$ bits (1011) in this example. In the decoding process, the received codeword sequence (1011) is firstly put after the decimal point to make it 0.1011 which is within the range $[0, 1)$. Then the representation 0.1011 is converted to the decimal value 0.6875. As it falls into the intervals $[0, 0.8]$, $[0.64, 0.8]$, $[0.64, 0.768]$, $[0.64, 0.7424]$,

```
Function AC_Encoder
Input: $s_k$, $\mathbf{c}$, $l_k$, $u_k$
Output: $\mathbf{b}$
Set $l_{k+1} = l_k + (u_k - l_k + 1)c(s_k)$ and $u_{k+1} = l_k + (u_k - l_k + 1)c(s_k + 1) - 1$
While(True)
  If $u_{k+1} < Half$
    Set $l_{k+1} = 2 \times l_{k+1}$ and $u_{k+1} = 2 \times u_{k+1} + 1$
    Emit a bit 0 and $f_{k+1}$ bits 1 to $\mathbf{b}$
    Set $f_{k+1} = 0$
  Else If $l_{k+1} \geq Half$
    Set $l_{k+1} = 2 \times (l_{k+1} - Half)$ and $u_{k+1} = 2 \times (u_{k+1} - Half) + 1$
    Emit a bit 1 and $f_{k+1}$ bits 0 to $\mathbf{b}$
    Set $f_{k+1} = 0$
  Else If $l_{k+1} \geq First\_quarter$ and $u_{k+1} < Third\_quarter$
    Set $l_{k+1} = 2 \times (l_{k+1} - First\_quarter)$ and $u_{k+1} = 2 \times (u_{k+1} - First\_quarter) + 1$
    Set $f_{k+1} = f_k + 1$
  Else
    Break;
```

Algorithm 1: Pseudocode of the encoder.

```
Function AC_Decoder
Input: $b_k$, $c$, $l_k$, $u_k$, $v\_l_k$, $v\_u_k$
Output: $\mathbf{s}$
If $b_k == 0$
  Set $v\_l_{k+1} = v\_l_k$ and $v\_u_{k+1} = ((v\_u_k - v\_l_k + 1)/2) - 1$
Else
  Set $v\_l_{k+1} = (v\_u_k - v\_l_k + 1)/2$ and $v\_u_{k+1} = v\_u_k$
Set $V = l_k + (u_k - l_k + 1) \times c(1)$
While(True)
  If $V > v\_u_{k+1}$
    Emit source symbol "0" to $\mathbf{s}$
    Set $l_{k+1} = l_k$ and $u_{k+1} = V - 1$
    Scale the intervals $[v\_l_{k+1}, v\_u_{k+1}]$ and $[l_{k+1}, u_{k+1}]$ as done in AC_Enocder
  Else If $V \leq v\_l_{k+1}$
    Emit source symbol "1" to $\mathbf{s}$
    Set $l_{k+1} = V$ and $u_{k+1} = u_k$
    Scale the intervals $[v\_l_{k+1}, v\_u_{k+1}]$ and $[l_{k+1}, u_{k+1}]$ as done in AC_Enocder
  Else
    Break;
```

Algorithm 2: Pseudocode of the decoder.

and $[0.64, 0.72192]$, the decoder will sequentially export the symbols "0," "1," "0," "0," and "0." The decoded sequence is exactly the same as the source sequence since AC is a lossless source coding scheme.

A practical problem encountered in the implementation of AC is that the interval will continue to shrink in the iterative encoding steps. Thus, a high precision is needed to represent the very small real numbers encountered in the coding process. A solution to this problem is to use integer representation, where the coding interval can be rescaled when the most significant bits in the representation of the lower and upper bounds are the same. Suppose that the binary source sequence represented by $\mathbf{s} = \{s_1, s_2, \ldots, s_N\}$ with $p(s = 0) = p_0$ and $p(s = 1) = p_1$, where $(p_0 + p_1 = 1)$, is encoded as a variable-length codeword $\mathbf{b} = \{b_1, b_2, \ldots, b_L\}$. In Algorithm 1, the pseudocode of the encoder is given, where l_k and u_k are, respectively, the lower and upper bounds for encoding the source symbol s_k. The vector $\mathbf{c}$ represents the cumulative probabilities of the source model with $c(0) = 0$, $c(1) = p_0$ and $c(2) = 1$. The initial lower and upper bounds are set to 0 and $2^P - 1$, respectively, where P is the length of the register for storing the value of the bounds. The number of bits not emitted in the interval rescaling operations is recorded by f_k. The values of *First_quarter*, *Half* and *Third_quarter* are fixed and are set to 2^{P-2}, 2^{P-1}, and $3 \times 2^{P-2}$, respectively.

In Algorithm 2, the pseudocode of the corresponding sequential decoder is listed, which avoids the decoding delay. Once both the bounds, d_l_{k+1} and d_u_{k+1}, of the decoding

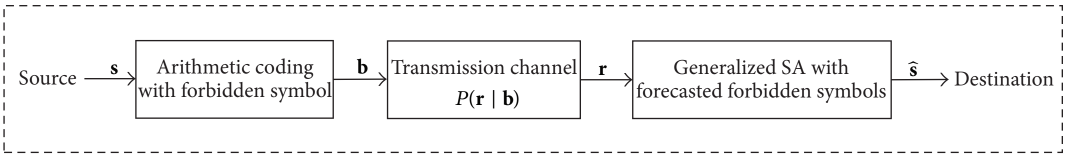

Source
s
Arithmetic coding with forbidden symbol
b
Transmission channel $P(\mathbf{r} \mid \mathbf{b})$
r
Generalized SA with forecasted forbidden symbols
ŝ
Destination

FIGURE 2: A block diagram of the transmission system.

interval for the compressed bit b_k are located in the encoding interval of a particular source symbol, the symbol can be decoded out. The encoding and decoding intervals are then rescaled as performed in the encoder. The lower and upper bounds of the decoding interval are also initialized to 0 and $2^P - 1$, respectively. The details of this kind of AC encoding and decoding can be found in [1, 34].

3. The Proposed Algorithm

Assume that the variable-length codeword $\mathbf{b} = \{b_1, b_2, \ldots, b_L\}$ is transmitted over a channel with transition probability $P(\mathbf{r} \mid \mathbf{b})$. The receiver obtains the demodulated sequence $\mathbf{r} = \{r_1, r_2, \ldots, r_L\}$, with which the recovered message $\hat{\mathbf{s}}$ is found using the generalized stack algorithm. A block diagram of this transmission system is depicted in Figure 2.

3.1. MAP Metric. In our scheme, the MAP metric [10] is employed for finding the most probable message $\hat{\mathbf{s}}$ from all possible sequences s, by maximizing the likelihood $P(\mathbf{s} \mid \mathbf{r})$, as expressed by

$$\hat{\mathbf{s}} = \arg\max_{s} P(\mathbf{s} = s \mid \mathbf{r}). \quad (1)$$

The Bayesian relationship states that

$$P(\mathbf{s} \mid \mathbf{r}) = \frac{P(\mathbf{r} \mid \mathbf{s}) P(\mathbf{s})}{P(\mathbf{r})} = \frac{P(\mathbf{r} \mid \mathbf{b}) P(\mathbf{s})}{P(\mathbf{r})}. \quad (2)$$

In the case of memoryless channels, it is straightforward to represent (2) in an additive form

$$m = \log P(\mathbf{s} \mid \mathbf{r}) = \log P(\mathbf{r} \mid \mathbf{b}) + \log P(\mathbf{s}) - \log P(\mathbf{r}). \quad (3)$$

For each bit of $\mathbf{r}$,

$$m_k = \log P(r_k \mid b_k) + \log P(\mathbf{s}_k) - \log P(r_k), \quad k \in [1, L], \quad (4)$$

where the vector $\mathbf{s}_k$ contains the decoded source symbols when the compressed bit b_k is shifted into the decoder. It should be noticed that $\mathbf{s}_k$ can be empty when no source symbol is outputted from the decoder. There are three terms at the right-hand-side of (4). The first term $\log P(r_k \mid b_k)$ is the channel transition probability while the second term $\log P(\mathbf{s}_k)$ represents the *a priori* probabilities of the source symbols. The first two terms can be evaluated based on the channel and source models, respectively. The last term $\log P(\mathbf{r})$ is complicated, which needs to sum up all the $\log P(\mathbf{r} \mid \mathbf{b}) P(\mathbf{b})$ terms, as follows:

$$\log P(\mathbf{r}) = \log \sum_{\mathbf{b}} P(\mathbf{r} \mid \mathbf{b}) P(\mathbf{b}). \quad (5)$$

As the full knowledge on the codeword $\mathbf{b}$ with length L is required, it is impractical to evaluate (5) exactly. However, assuming that the codeword $\mathbf{b}$ has equal probabilities of occurrence of "0" and "1," this term can be approximated by

$$\begin{aligned} P(r_k) &= P(r_k \mid b_k = 0) P(b_k = 0) + P(r_k \mid b_k = 1) P(b_k = 1) \\ &= \frac{P(r_k \mid b_k = 0) + P(r_k \mid b_k = 1)}{2}. \end{aligned} \quad (6)$$

When hard decoding is adopted in an AWGN channel using binary phase-shift keying (BPSK) modulation with a signal-to-noise ratio (SNR) E_b/N_0, the channel transition probability is

$$P(r_k \mid b_k) = \begin{cases} 1 - p, & \text{if } r_k = b_k \\ p, & \text{if } r_k \neq b_k, \end{cases} \quad (7)$$

where $p = (1/2)\operatorname{erfc}\sqrt{E_b/N_0}$. By (6), $P(r_k) = (p + 1 - p)/2 = 1/2$ in this case. Therefore,

$$m_k = \begin{cases} \log(1 - p) + \log P(\mathbf{s}_k) + \log 2, & \text{if } r_k = b_k \\ \log p + \log P(\mathbf{s}_k) + \log 2, & \text{if } r_k \neq b_k. \end{cases} \quad (8)$$

In the soft decoding process, each bit in $\mathbf{b}$ is mapped to $\mathbf{t}$ by $t_k = \sqrt{E_b}(2b_k - 1)$ before transmitted over the AWGN channel. The decoder receives the noisy signal $r_k = t_k + n_k$, where n_k is the additive white noise with standard deviation δ. Given the input sequence $\mathbf{b}$, the conditional probability of the received signal $\mathbf{r}$ is

$$\begin{aligned} P(r_k b_k) &= \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{(r_k - t_k)^2}{2\delta^2}\right) \\ &= \begin{cases} \dfrac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\dfrac{(r_k + \sqrt{E_b})^2}{2\delta^2}\right), & \text{if } b_k = 0 \\ \dfrac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\dfrac{(r_k - \sqrt{E_b})^2}{2\delta^2}\right), & \text{if } b_k = 1. \end{cases} \end{aligned} \quad (9)$$

Making use of (6), we have

$$\begin{aligned} P(r_k) = \Bigg(& \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{(r_k + \sqrt{E_b})^2}{2\delta^2}\right) \\ & + \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{(r_k - \sqrt{E_b})^2}{2\delta^2}\right)\Bigg) \cdot 2^{-1}. \end{aligned} \quad (10)$$

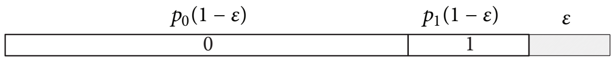


Figure 3: Source symbol model with forbidden symbol.

Thus,

$$\begin{aligned}\log P(r_k) &= \log\left(\frac{1}{\sqrt{2\pi\delta^2}}\exp\left(-\frac{(r_k+\sqrt{E_b})^2}{2\delta^2}\right)\right.\\ &\quad\left.\cdot\left(1+\frac{\exp\left(-(r_k-\sqrt{E_b})^2/2\delta^2\right)}{\exp\left(-(r_k+\sqrt{E_b})^2/2\delta^2\right)}\right)\right)\\ &\quad-\log 2\\ &= -\frac{(r_k+\sqrt{E_b})^2}{2\delta^2}-\log\sqrt{2\pi\delta^2}\\ &\quad+\log\left(1+\exp\left(\frac{2\sqrt{E_b}r_k}{\delta^2}\right)\right)-\log 2,\end{aligned} \tag{11}$$

$$m_k=\begin{cases}\log P(\mathbf{s}_k)+\log 2\\ \quad-\log\left[1+\exp\left(\dfrac{2\sqrt{E_b}r_k}{\sigma^2}\right)\right], & \text{if } b_k=0\\ \log P(\mathbf{s}_k)+\log 2+\dfrac{2\sqrt{E_b}r_k}{\sigma^2}\\ \quad-\log\left[1+\exp\left(\dfrac{2\sqrt{E_b}r_k}{\sigma^2}\right)\right], & \text{if } b_k=1.\end{cases} \tag{12}$$

3.2. Forecasted Forbidden Symbols. In order to embed error detecting capacity into AC, a forbidden symbol μ with probability of occurrence ε is inserted in the source model, as shown in Figure 3. The probabilities of “0” and “1” are changed to $p_0(1-\varepsilon)$ and $p_1(1-\varepsilon)$, respectively. The overhead of this approach is a lower coding rate as the available coding space for AC shrinks. This accounts for $R_\mu = -\log_2(1-\varepsilon)$ additional bits for each source symbol. The expected length of the compressed sequence is $L = N(H+R_\mu)$ when the forbidden symbol is adopted, where $H = -p_0\log p_0 - p_1\log p_1$ is the memoryless source entropy rate. As the forbidden symbol is never encoded, the decoder can assure that some estimated bits are erroneous once it observes the forbidden symbol in the decoding process. Thus, the erroneous decoding path can be pruned. Theoretically, the number of symbols decoded before an error is detected is greater than n at a probability of $(1-\varepsilon)^n$. Therefore, as more source symbols after the erroneous bits are decoded, the error can be detected with a higher probability. Moreover, a large value of ε enables short error detection delay at the expense of compression efficiency.

Thanks to the iterative nature of the AC encoding process, the forbidden symbols after the currently encoded source symbol can actually be estimated beforehand. This is useful in detecting the errors at an earlier stage, so as to prune the erroneous decoding tree quickly and to increase the chance for the correct decoding tree to remain in the stack. As shown in Figure 4(a), the second forbidden symbols in the original coding regions for “0” and “1” are forecasted, the lengths of which are $p_0(1-\varepsilon)\varepsilon$ and $p_1(1-\varepsilon)\varepsilon$, respectively. Similarly, the third forbidden symbols shown in Figure 4(b) are predicted with the corresponding lengths $p_0p_1(1-\varepsilon)^2\varepsilon$ and ${p_1}^2(1-\varepsilon)^2\varepsilon$. Theoretically, the lengths of all the successive forecasted forbidden symbols can be summed up as $p_0(1-\varepsilon)\varepsilon\sum_{i=0}^{n-1}{p_1}^i(1-\varepsilon)^i$ and $\varepsilon\sum_{i=0}^{n}{p_1}^i(1-\varepsilon)^i$ in the two coding units, where n is the number of the forecasted forbidden symbols. When n tends to ∞, we have the length of forecasted forbidden regions $fs(1)$ and $fs(2)$ as follows:

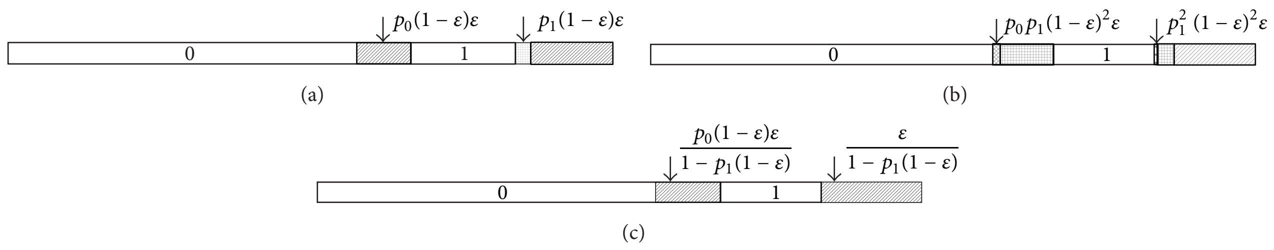
$$\begin{aligned}fs(1) &= p_0(1-\varepsilon)\varepsilon\sum_{i=0}^{\infty}{p_1}^i(1-\varepsilon)^i = \frac{p_0(1-\varepsilon)\varepsilon}{1-p_1(1-\varepsilon)},\\ fs(2) &= \varepsilon\sum_{i=0}^{\infty}{p_1}^i(1-\varepsilon)^i = \frac{\varepsilon}{1-p_1(1-\varepsilon)}.\end{aligned} \tag{13}$$

They are the theoretical limit for the length of successive forbidden symbols. As shown in Figure 4(c), the forecasted forbidden region is much larger than that in Figure 3, which obviously improves the error correcting capability.

On the other hand, the look-ahead technique [26] usually employed in AC decoders can detect forbidden symbol quickly by decoding the source symbol even when the decoding interval bounds d_l_{k+1} and d_u_{k+1} are located in the encoding intervals for a particular source symbol and the forbidden region by assuming that it is error-free. An example of which is given in Figure 5, where the source symbol “1” is decoded in Figure 5(a) and then the forbidden symbol can be detected in Figure 5(b). Compared with the look-ahead technique in AC decoders, our forecasted forbidden regions can effectively detect the possible errors that may be found by the look-ahead technique. For example, the forbidden symbol in Figure 5(b) can also be detected with our forecasted forbidden regions without the need to decode source symbol “1” in advance. Besides that, the forecasted forbidden regions in the middle of the source symbols “0” and “1” enable the adoption of generalized stack algorithm introduced in Section 3.3. The look-ahead technique can be adopted in our scheme to further enhance the overall correction performance. An example of this scenario is depicted in Figure 5(c) that the source symbol “1” is decoded in advance. The improvement of our scheme is further validated by the simulation results to be reported in Section 4.

With the look-ahead technique adopted in our scheme, it is possible that $d\_l_{k+1} < 0$ and $d\_u_{k+1} > 2^P - 1$ in the implementation of AC. The pseudocode of the modified decoder for error detection can be found in Algorithm 3.

3.3. Generalized Stack Algorithm. The generalized SA is a variation of SA, which is a metric first search algorithm. In the original SA, all the explored decoding paths with better metric are stored in an ordered stack with size M. The best decoding path, which has the maximum value given by (8) or



$p_0(1-\varepsilon)\varepsilon$
$p_1(1-\varepsilon)\varepsilon$
0
1
(a)
$p_0 p_1 (1-\varepsilon)^2\varepsilon$
$p_1^2(1-\varepsilon)^2\varepsilon$
0
1
(b)
$\frac{p_0(1-\varepsilon)\varepsilon}{1-p_1(1-\varepsilon)}$
$\frac{\varepsilon}{1-p_1(1-\varepsilon)}$
0
1
(c)

Figure 4: (a) The second forecasted forbidden symbol; (b) the third forecasted forbidden symbol; (c) theoretical length of all the successive forecasted forbidden symbols.

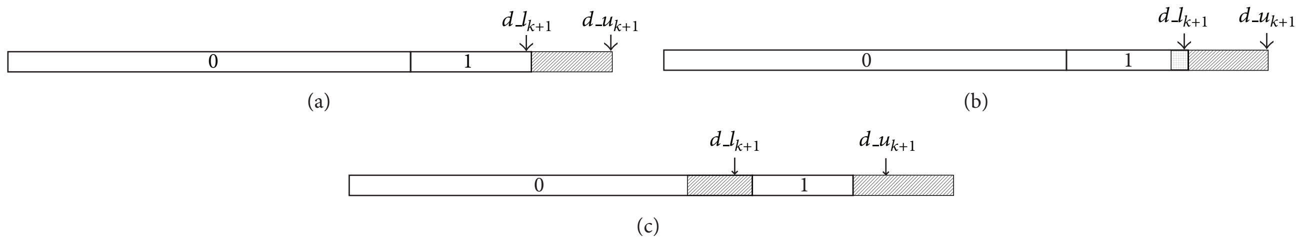


Figure 5: (a) Decode source symbol “1” using look-ahead technique; (b) detect forbidden symbol using look-ahead technique; (c) the use of look-ahead technique in our scheme.

(12), is usually stored at the top of the stack. It is extended to two branches after the current decoding node by estimating the subsequent decoding bits as “0” and “1,” respectively. Then the top node is removed and the two child nodes are inserted into the stack. Once the stack is full, the one with the worst metric will be discarded. In the generalized SA, 2^k branches instead of 2 branches are extended from the top node. As the coding region assigned to the forbidden symbol is small, it usually needs more bits to make sure that the decoder will visit the forbidden region or not. Thus, 2^k branches from the best node are able to result in a fast detection and the removal of erroneous decoding paths. This in turn means a higher probability to preserve the correct path in the stack. It is noted that the generalized SA is not applicable in the original MAP algorithm and the look-ahead technique as the underflow problem may happen when 2^k branches are extended from the very small decoding interval. However, as the forbidden regions in the middle of source symbols “0” and “1” are forecasted in our scheme, it guarantees that the decoding interval is not smaller than the length of the intermediate forbidden region. Therefore, the underflow problem can be avoided.

There are three conditions for discarding decoding paths in the generalized SA. The first condition is that the forbidden symbol is encountered. The second corresponds to the situation that the number of decoded symbols is equal to N but the number of decoded bits is smaller than L. The third case is that the number of decoded bits is equal to L but the number of decoded symbol is smaller than N. The generalized SA stops when the L decoded bits can exactly recover N source symbols or the stack is empty. A diagram illustrating the generalized SA is shown in Figure 6, with $k = 2$. Therefore, four child nodes are extended from the best nodes that are identified with gray color. The one marked with X is deleted as it visits the forbidden region.

4. Simulations

In this section, the proposed scheme is compared with the original MAP scheme [10] and the look-ahead scheme [26]. Binary source symbols with $p_0 = 0.8667$ are randomly generated, which correspond to the memoryless source entropy $H = 0.567$. This entropy is the same as that of the simulation data used in [10]. Each packet consists of 2304 binary symbols. It is then encoded by arithmetic coding with forbidden symbol to generate the variable-length compressed sequence **b**. The packet length and the priori bit probability p_0 are sent to the decoder as side information. They are protected by a high-redundant channel code to guarantee their correctness. Each packet is terminated with an EOS (End of Sequence) symbol having probability 10^{-5}, which protects the last few bits of **b**. The stack size M is chosen as 256 for all the algorithms. The value of k is set to 8 in the generalized SA. All the simulations are run for 10^5 times over an AWGN channel with BPSK modulation. The number of forecasted forbidden symbols is selected as 4. As the original MAP scheme [10] has already been shown to have a better error correction performance than the traditional separated source and channel coding scheme, a comparison with the latter scheme is not repeated here. As the placement of the forbidden symbol can affect the error correction performance [26, 29], two placements are considered in our simulations, which are identified with source models A and B in Figures 7(a) and 7(c). The corresponding forecasted forbidden symbols in source models A and B are illustrated in Figures 7(b) and 7(d). Note that the performance of

```
Function AC_FS_Decoder
Input: $b_k, c, l_k, u_k, d\_l_k, d\_u_k$
Output: s
If $b_k == 0$
    Set $d\_l_{k+1} = d\_l_k$ and $d\_u_{k+1} = ((d\_u_k - d\_l_k + 1)/2) - 1$
Else
    Set $d\_l_{k+1} = (d\_u_k - d\_l_k + 1)/2$ and $d\_u_{k+1} = d\_u_k$
While(True)
    Find two estimated forbidden regions in the encoding interval;
    If $d\_l_{k+1}$ and $d\_u_{k+1}$ are completely located in the forecasted forbidden regions
    or out of the encoding interval
    Then delete the decoding path and break;
    Set $V = l_k + (u_k - l_k + 1) \times c(1)$ and $V_{fs} = l_k + (u_k - l_k + 1) \times (c(1) - fs(1))$
    If $V > d\_u_{k+1}$
        Emit source symbol "0" to s
        Set $l_{k+1} = l_k$ and $u_{k+1} = V - 1$
        Rescale the intervals $[d\_l_{k+1}, d\_u_{k+1}]$ and $[l_{k+1}, u_{k+1}]$ as done in AC_Enocder
    Else If $V_{fs} \le d\_l_{k+1}$ and $d\_u_{k+1} \le u_k$
        Emit source symbol "1" to s
        Set $l_{k+1} = V$ and $u_{k+1} = u_k$
        Rescale the intervals $[d\_l_{k+1}, d\_u_{k+1}]$ and $[l_{k+1}, u_{k+1}]$ as done in AC_Enocder
    Else
        Break;
```

Algorithm 3: Pseudocode of the modified decoder for error detection using forecasted forbidden symbols.

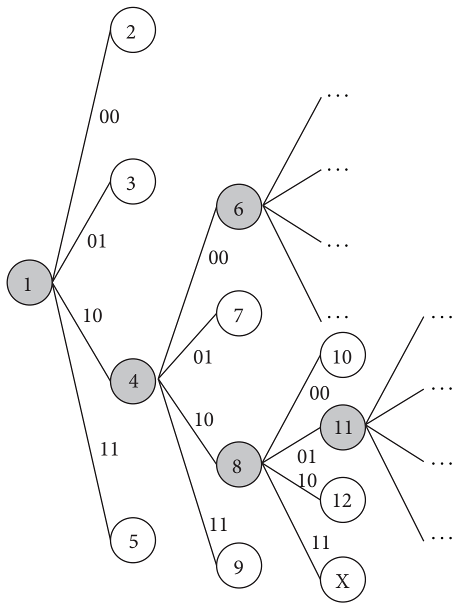
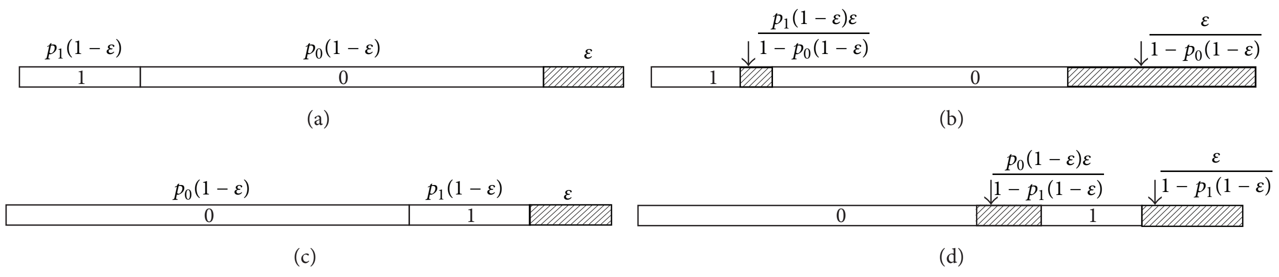


Figure 6: A diagram illustrating the generalized SA.

the compared schemes is evaluated by the packet error rate (PER).

The PERs of the proposed, the look-ahead, and the original MAP schemes with source models A and B at various channel SNRs are plotted in Figure 8. The value of ε is set to 0.185 in Figure 8, which corresponds to the coding rate of 2/3. Considering source model A, in which many forbidden symbols can be forecasted as indicated by Figure 7(b), it contributes to the major improvement of our scheme and the look-ahead scheme when compared with the original MAP scheme. The simulation results plotted in Figure 8 validate that the look-ahead scheme performs much better than the original MAP scheme while ours achieves the best results. However, the results obtained with source model B are better than that with source model A in all algorithms. These observations show that the error correcting capacity of source model B is better than that of source model A. Although many forbidden symbols can be forecasted in source model A, it will cause a lot of forbidden symbols assigned in the upper bound of the coding interval and leads to weak error detection in the errors occurring in the lower bound of the coding interval. In summary, our scheme performs much better than the look-ahead scheme and the original MAP scheme at all SNRs for the two source models. Of course, the best results in soft and hard decoding are found by using our scheme with source model B, which achieves a coding gain of around 0.5 dB for hard decoding and 0.25 dB for soft decoding, when compared with the original MAP scheme. Moreover, the values of ε at 0.097 and 0.05 are also selected for source model B, which correspond to the coding rate of 4/5 and 8/9, respectively. The PERs of our, the look-ahead, and the original MAP schemes are plotted in Figures 9-10. As indicated in these two figures, the coding gain decreases when the value of ε becomes small. The graphs reveal that, for large ε, our scheme has a much better performance than the look-ahead and the original MAP schemes. In other words, it is especially effective at a low coding rate.

Figure 11 shows the error correction performance with source model B at various ε using soft and hard decoding. In this figure, soft decoding is applied in our, the look-ahead,



$p_1(1-\varepsilon)$
$p_0(1-\varepsilon)$
ε
1
0
(a)
$\frac{p_1(1-\varepsilon)\varepsilon}{1-p_0(1-\varepsilon)}$
$\frac{\varepsilon}{1-p_0(1-\varepsilon)}$
(b)
$p_0(1-\varepsilon)$
$p_1(1-\varepsilon)$
(c)
$\frac{p_0(1-\varepsilon)\varepsilon}{1-p_1(1-\varepsilon)}$
$\frac{\varepsilon}{1-p_1(1-\varepsilon)}$
(d)

FIGURE 7: (a) Source model A; (b) source model A with forecasted forbidden symbols; (c) source symbol B; (d) source model B with forecasted forbidden symbols.

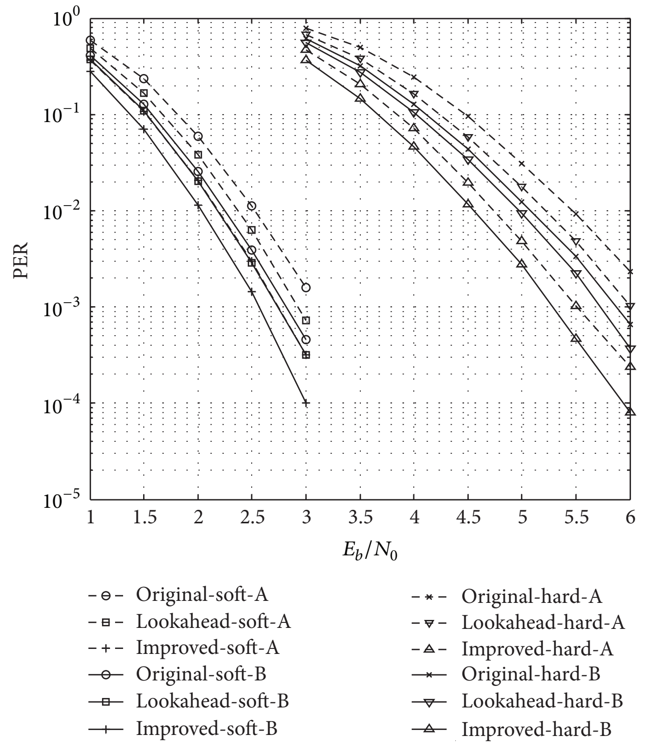


FIGURE 8: Error correction performance of our, the look-ahead, and the original MAP schemes at $\varepsilon = 0.185$.

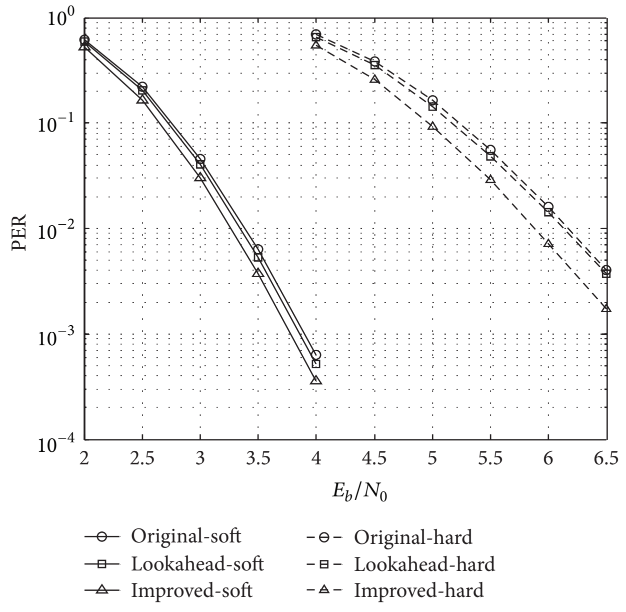


FIGURE 9: Error correction performance of our, the look-ahead, and the original MAP schemes at $\varepsilon = 0.097$.

and the original MAP schemes with the channel SNR fixed at 3.5 dB. The PERs are plotted against the ε value ranging from 0.04 to 0.16. With the increase of ε, the PERs of all schemes drop accordingly. This is reasonable as a large value of ε leads to more redundant bits for error detection, which are helpful in removing the erroneous decoding paths. When ε is large, the gain of our scheme over the look-ahead scheme and the original MAP scheme becomes apparent, which also indicates that our scheme performs much better at a low coding rate. Considering hard decoding in a channel with SNR 5.5 dB, the PERs of our, the look-ahead, and the original MAP schemes are also depicted in Figure 11 with various ε between 0.04 and 0.16. Results similar to those obtained using soft decoding are observed and they further confirm the superiority of our scheme at a low coding rate.

5. Conclusions

We have proposed an effective error detection technique based on the forecasting of forbidden symbols, which widens the forbidden region by estimating the occurrence of the subsequent forbidden symbols. A generalized SA is also adopted to detect the forbidden symbol beforehand and to remove the erroneous decoding paths earlier. As a result, the chance of preserving the correct decoding path increases and the error correction performance is improved. Simulation results validate the superiority of our approach over the look-ahead and the original MAP schemes, especially at a low coding rate.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

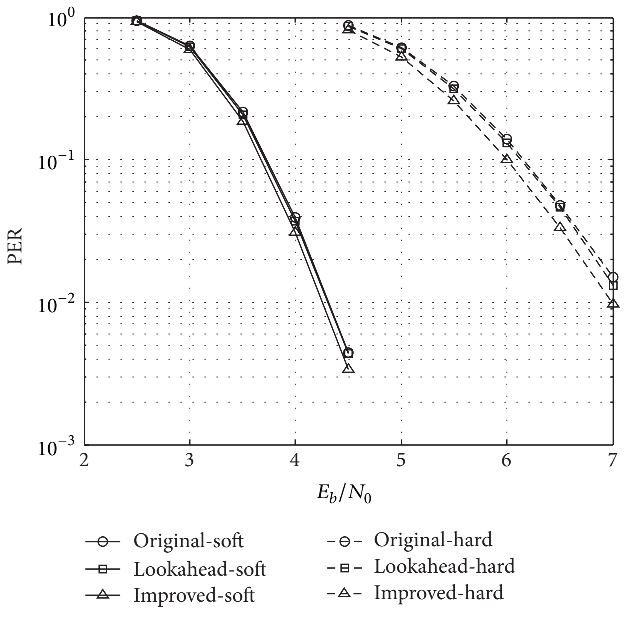


FIGURE 10: Error correction performance of our, the look-ahead, and the original MAP schemes at $\varepsilon = 0.05$.

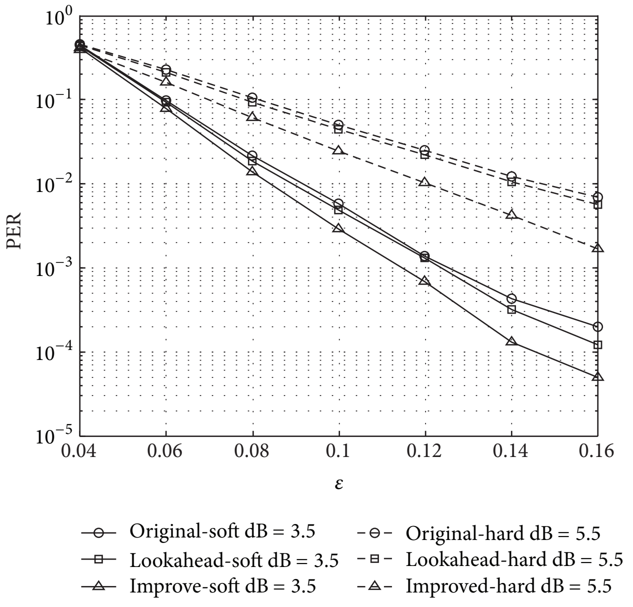


FIGURE 11: The PERs of our, the look-ahead, and the original MAP schemes at various ε, using soft decoding at channel SNR 3.5 dB and hard decoding at channel SNR 5.5 dB.

Acknowledgments

The work described in this paper was supported by a grant from CityU (Project no. 7004062), the National Natural Science Foundation of China under the Project Grant nos. 61402291, 61272402, 61070214, 60873264, and 61170283, National High-Technology Research and Development Program ("863" Program) of China under Grand 2013AA01A212, and Ministry of Education in the New Century Excellent Talents Support Program under Grand NCET-12-0649. Ming Li also thanks the support by the Science and Technology Commission of Shanghai Municipality under research Grant no. 14DZ2260800.

References

[1] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[2] H.-C. Hsin, T.-Y. Sung, Y.-S. Shieh, and C. Cattani, "Adaptive binary arithmetic coder-based image feature and segmentation in the compressed domain," *Mathematical Problems in Engineering*, vol. 2012, Article ID 490840, 14 pages, 2012.

[3] Y.-K. Chan, R.-C. Chen, P.-Y. Pai, and C.-C. Chang, "Lossless image compression based on multiple-tables arithmetic coding," *Mathematical Problems in Engineering*, vol. 2009, Article ID 128317, 13 pages, 2009.

[4] C. Koller, A. G. I. Amat, J. Kliewer, F. Vatta, K. S. Zigangirov, and D. J. Costello, "Analysis and design of tuned turbo codes," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4796–4813, 2012.

[5] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Enhanced detection of double and triple adjacent errors in hamming codes through selective bit placement," *IEEE Transactions on Device and Materials Reliability*, vol. 12, no. 2, pp. 357–362, 2012.

[6] O. Y. Bursalioglu, G. Caire, and D. Divsalar, "Joint source-channel coding for deep-space image transmission using rateless codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3448–3461, 2013.

[7] S. Chabbouh and C. Lamy, "A structure for fast synchronizing variable-length codes," *IEEE Communications Letters*, vol. 6, no. 11, pp. 500–502, 2002.

[8] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Transactions on Communications*, vol. 48, no. 1, pp. 1–6, 2000.

[9] K. Sayood, H. H. Otu, and N. Demir, "Joint source/channel coding for variable length codes," *IEEE Transactions on Communications*, vol. 48, no. 5, pp. 787–794, 2000.

[10] M. Grangetto, P. Cosman, and G. Olmo, "Joint source/channel coding and MAP decoding of arithmetic codes," *IEEE Transactions on Communications*, vol. 53, no. 6, pp. 1007–1016, 2005.

[11] C. Boyd, J. G. Cleary, S. A. Irvine, I. Rinsma-Melchert, and I. H. Witten, "Integrating error detection into arithmetic coding," *IEEE Transactions on Communications*, vol. 45, no. 1, pp. 1–3, 1997.

[12] R. Anand, K. Ramchandran, and I. V. Kozintsev, "Continuous error detection (CED) for reliable communication," *IEEE Transactions on Communications*, vol. 49, no. 9, pp. 1540–1549, 2001.

[13] G. F. Elmasry, "Joint lossless-source and channel coding using automatic repeat request," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 953–955, 1999.

[14] S. Chokchaitam and P. Teekaput, "Embedded error detection in arithmetic coding using markers," in *Proceedings of the International Conference on Information Technology: Coding Computing (ITCC '04)*, pp. 747–750, April 2004.

[15] K.-W. Wong, Q. Z. Lin, and J. Y. Chen, "Error detection in arithmetic coding with artificial markers," *Computers & Mathematics with Applications*, vol. 62, no. 1, pp. 359–366, 2011.

[16] B. D. Pettijohn, M. W. Hoffman, and K. Sayood, "Joint source/channel coding using arithmetic codes," *IEEE Transactions on Communications*, vol. 49, no. 5, pp. 826–835, 2001.

[17] T. Guionnet and C. Guillemot, "Soft decoding and synchronization of arithmetic codes: application to image transmission over noisy channels," *IEEE Transactions on Image Processing*, vol. 12, no. 12, pp. 1599–1609, 2003.

[18] M. Grangetto, B. Scanavino, G. Olmo, and S. Benedetto, "Iterative decoding of serially concatenated arithmetic and channel codes with JPEG 2000 applications," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1557–1567, 2007.

[19] S. Zaibi, A. Zribi, R. Pyndiah, and N. Aloui, "Joint source/channel iterative arithmetic decoding with JPEG 2000 image transmission application," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, article 114, 2012.

[20] C. Toma, "Wavelets-computational aspects of sterian realistic approach to uncertainty principle in high energy physics: a transient approach," *Advances in High Energy Physics*, vol. 2013, Article ID 735452, 6 pages, 2013.

[21] E. G. Bakhoum and C. Toma, "Modeling transitions in complex systems by multiplicative effect of temporal patterns extracted from signal flows," *Mathematical Problems in Engineering*, vol. 2012, Article ID 409856, 11 pages, 2012.

[22] L. Kocarev, F. Lehmann, G. M. Maggio, B. Scanavino, Z. Tasev, and A. Vardy, "Nonlinear dynamics of iterative decoding systems: analysis and applications," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1366–1384, 2006.

[23] L. X. Li, H. P. Peng, J. Kurths, Y. X. Yang, and H. J. Schellnhuber, "Chaos-order transition in foraging behavior of ants," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 23, pp. 8392–8397, 2014.

[24] L. Li, J. Xiao, H. Peng, Y. Yang, and Y. Chen, "Improving synchronous ability between complex networks," *Nonlinear Dynamics*, vol. 69, no. 3, pp. 1105–1110, 2012.

[25] S. B. Jamaa, M. Kieffer, and P. Duhamel, "Improved sequential MAP estimation of CABAC encoded data with objective adjustment of the complexity/efficiency tradeoff," *IEEE Transactions on Communications*, vol. 57, no. 7, pp. 2014–2023, 2009.

[26] T. Spiteri and V. Buttigieg, "Maximum a posteriori decoding of arithmetic codes in joint source-channel coding," *Communications in Computer and Information Science*, vol. 222, pp. 363–377, 2012.

[27] D. S. Bi, M. W. Hoffman, and K. Sayood, "State machine interpretation of arithmetic codes for joint source and channel coding," in *Proceedings of the Data Compression Conference (DCC '06)*, pp. 143–152, March 2006.

[28] H. Moradmand, A. Payandeh, and M. R. Aref, "Joint source-channel coding using finite state integer arithmetic codes," in *Proceedings of the IEEE International Conference on Electro/Information Technology (EIT '09)*, pp. 19–22, Windsor, Canada, June 2009.

[29] S. Ben-Jamaa, C. Weidmann, and M. Kieffer, "Analytical tools for optimizing the error correction performance of arithmetic codes," *IEEE Transactions on Communications*, vol. 56, no. 9, pp. 1458–1468, 2008.

[30] A. Diallo, C. Weidmann, and M. Kieffer, "Efficient computation and optimization of the free distance of variable-length finite-state joint source-channel codes," *IEEE Transactions on Communications*, vol. 59, no. 4, pp. 1043–1052, 2011.

[31] S. Zezza, S. Nooshabadi, and G. Masera, "A 2.63 Mbit/s VLSI implementation of SISO arithmetic decoders for high performance joint source channel codes," *IEEE Transactions on Circuits and Systems. I. Regular Papers*, vol. 60, no. 4, pp. 951–964, 2013.

[32] W. Zhang, F. Cen, and F. L. Zhu, "Research on practical implementation of binary arithmetic coding with forbidden symbol for error resilience," *Frontiers in Computer Education, Advances in Intelligent and Soft Computing*, vol. 133, pp. 791–798, 2012.

[33] Q. Z. Lin and K. W. Wong, "Improving the error correction capability of arithmetic coding by forecasting forbidden symbols," in *Proceedings of IEEE International Symposium of Circuits and Systems (ISCAS '13)*, pp. 1139–1142, IEEE, Beijing, China, May 2013.

[34] A. Said, "Hewlett-Packard Laboratories report," HPL 2004-76, HP Laboratories, Palo Alto, Calif, USA, 2004, http://www.hpl.hp.com/techreports/2004/HPL-2004-76.pdf.