A Two-Stage Algorithm to Reduce Encoding Delay of Turbo Source Coding

Javad Haghighat and David V. Plant
Department of Electrical and Computer Engineering, McGill University
McConnell Engineering Building, Room 644, 3480 University Street, Montreal, Quebec, H3A 2A7, Canada
Email: {javad.haghighat, david.plant}@mcgill.ca

***Abstract*—Lossless turbo source coding employs an iterative encoding algorithm to search for the smallest codeword length that guarantees zero distortion. Although such encoder achieves promising compression rates, running the iterative algorithm for each individual message block imposes a large delay on the system. To reduce this delay, we propose a two-stage encoding algorithm for turbo source coding. We show that converging to zero distortion after a definite number of iterations, can be predicted from the earlier behavior of the distortion function. This will enable us to produce a quick, and yet sufficiently accurate, estimate of the codeword length in the first encoding stage. In the second stage, we iteratively increase this estimated codeword length until reaching zero distortion. Also, we show that employing an auxiliary distortion measure at the first stage of encoding may allow for better estimates and decrease the delay furthermore. Numerical results show that the proposed algorithm will decrease the encoding delay up to 19%. Although there are previous works in the literature on delay reduction of turbo source coding, those works achieve lower delays by reducing the message block length. However, the proposed algorithm achieves lower delays for the same block length and therefore the actual "per bit" encoding delay is decreased.**

I. Introduction

In recent years application of turbo codes for source coding has received a great deal of attention. Researchers mainly focus on the case when the message is a biased binary i.i.d. sequence and consider two different types of turbo source coding schemes: (a) *near-lossless* turbo source coding where small levels of average distortion (bit error rate) can be tolerated. This definition for near-lossless coding is borrowed from [5]. It is worth mentioning that in image compression near-lossless coding is defined by setting the maximum distortion of a pixel below a threshold (see [10] for example). (b) lossless turbo source coding, where the distortion is forced to be zero.

Near-lossless turbo source coding is considered in many works including [1]-[4]. The message is delivered to a turbo encoder to generate parity bits. The parity bits are then punctured to a fixed compression rate; and the non-punctured parities are transmitted to the decoder. The decoder regenerates the message which is now distorted. This distortion is a function of the selected compression rate; i.e. a desired distortion level is achievable by defining the proper compression rate. Although applicable in most cases, *near-lossless* coding is not acceptable in applications such as compression of medical images, where image details are of extreme importance. This is because allowing a small average distortion may result in high levels of distortion on some parts of the image and damage some important details.

Lossless turbo source coding is first presented in [5] and then revisited in works including [6], [7]. The main idea of lossless turbo source coding is to place a decoder at the encoder side and check if the (tentatively) generated codeword can be successfully decoded (a similar idea is presented for low density parity check codes and fountain codes in [8] and [9], respectively). If decoding errors are observed, the encoder appends more bits to the codeword and repeats tentative decoding until the first successful decoding is reached. We refer to this procedure as *iterative encoding*. Such encoder generates variable length codewords with minimum length required to reproduce the original message with no distortion (although the message block length is fixed). Therefore, it achieves the minimum compression rate required to achieve zero distortion. Large block length codes can achieve compression rates close to the source entropy. For example, the work in [6] studies repeat-accumulate (RA) codes as special cases of serially concatenated turbo codes, and designs RA codes with message block length 2^{16} bits that losslessly compress a binary i.i.d. source with entropy 0.47 bits/sample to a compression rate of 0.50 bits/sample. One important drawback of lossless turbo source coding is that several rounds of tentative decoding require several turbo decoding iterations that impose a large encoding delay on the system.

Our contribution is to propose a two-stage iterative encoding algorithm to achieve same compression rates with smaller encoding delay. In the first stage, we utilize an auxiliary distortion measure that quickly searches for a sufficiently accurate estimate of the codeword length. The desired accuracy is defined by an auxiliary threshold. In the second stage, the estimated length is examined by the actual distortion measure and tentative decoding continues until reaching zero distortion. The problem formulation, and presentation of the new algorithm contains all types of discrete i.i.d. sources. However, for simulation results we focus on binary i.i.d. sources. The paper is organized as follows. Section II formulates the problems of source coding under two different conditions (i) When the expected value of distortion is required to be below a threshold. *Near-lossless* turbo source coding can be viewed as a special case of this problem (ii) When the maximum distortion is required to be below a defined threshold. In this case we need to run the iterative encoding algorithm. Lossless turbo source coding can be viewed as a special case of this problem. In Section III we propose our two-stage encoding algorithm. In Section IV we present simulation results for lossless turbo source coding of a binary biased i.i.d. source. We present numerical results that show the modified algorithm can decrease the encoding delay up to 19%, while maintaining the same compression rate. Concluding remarks are presented in Section V.

978-1-4244-1722-3/08/$25.00 ©2008 IEEE.

It is worth mentioning that delay reduction of turbo source coding is also considered in [7]. However, [7] reduces the delay by decreasing the message block length and designing powerful codes for that shorter block length. Thus, the "per-block" encoding delay (latency [7]) is reduced. However, our proposed algorithm reduces the number of encoding iterations for a message with a given block length. Therefore, the actual "per-bit" encoding delay is reduced.

II. Source Coding Under Expected and Maximum Distortion Constraints

Let $\mathbf{x}$ be a message vector consisting of n samples of a source X, each sample taken from a finite alphabet χ. Consider a source encoder with an encoding function $f(.,.)$ that encodes $\mathbf{x}$ to a codeword $\mathbf{y} = f(\mathbf{x}, l)$, where l is called the codeword length (in bits). This encoder has a compression rate of $R = \frac{l}{n}$ bits/sample. The codeword $\mathbf{y}$ is delivered to a source decoder with a decoding function $g(.,.)$ to generate $\widehat{\mathbf{x}} = g(\mathbf{y}, T^*)$ as an estimate of $\mathbf{x}$. T^* is the amount of time required to terminate decoding. The accuracy of this estimation, $\widehat{\mathbf{x}}$, is evaluated by a distortion measure $d_1(\mathbf{x}, \widehat{\mathbf{x}})$. By absorbing functions $f(.,.)$ and $g(.,.)$, the distortion could be alternatively represented by a function $d(\mathbf{x}, l, T^*)$ which represents the distortion caused by encoding $\mathbf{x}$ to a codeword with length l bits, and allowing a decoding time of T^*. For example, the normalized Hamming distortion between a binary vector $\mathbf{x}$ and its binary reconstruction $\hat{\mathbf{x}}$ is defined as

$$d(\mathbf{x}, l, T^*) = \frac{1}{n} \sum_{i=1}^{n} (x_i \oplus \hat{x}_i), \tag{1}$$

where $\mathbf{x} = x_1 x_2 ... x_n$, $\hat{\mathbf{x}} = g(f(\mathbf{x}, l), T^*)$, and $\oplus$ represents binary addition. Generally, we assume that $d(\mathbf{x}, l, T^*)$ is a non-increasing function of l and T^*.

Now we consider two different source coding problems, source coding under expected distortion constraint and source coding under maximum distortion constraint.

A. Source Coding Under Expected Distortion Constraint

In this problem the goal is to find the minimum compression rate to have $E_{\mathbf{x}} d(\mathbf{x}, l, T^*) \leq D$ where $E_{\mathbf{x}}$ is the expected value with respect to $\mathbf{x}$, and $D \geq 0$ is a defined threshold. In this case we search for a codeword length

$$l^* = \underset{E_{\mathbf{x}} d(\mathbf{x}, l, T^*) \leq D}{Min} l, \tag{2}$$

and fix this codeword length for every message vector. This scheme has the advantages of having a fixed length output and no encoding delay (except for the amount of time required to execute function $f(.,.)$). For *near-lossless* turbo source coding of biased binary sources, we normally consider the normalized Hamming distortion measure. The expected value of this measure will give the bit error rate. The threshold D is usually set to some value about 10^{-5} (i.e. bit error rate below 10^{-5} is considered *near-lossless*).

B. Source Coding Under Maximum Distortion Constraint

In this problem the goal is to find the minimum *average* compression rate to have $d(\mathbf{x}, l, T^*) \leq D$, $\forall \mathbf{x}$ (this definition is different from the standard maximum distortion constraint defined by $\underset{i}{Max}\, d_1(x_i, \hat{x}_i) \leq D$. For lossless source coding these conditions agree as $d(\mathbf{x}, l, T^*) = \underset{i}{Max}\, d_1(x_i, \hat{x}_i) = 0$). In this case the codeword length varies for different message vectors and is calculated as

$$l^*_{\mathbf{x}} = \underset{d(\mathbf{x}, l, T) \leq D}{Min} l. \tag{3}$$

For lossless turbo source coding we normally consider the normalized Hamming distortion measure and allow no distortion by setting $D = 0$. To calculated $l^*_{\mathbf{x}}$ for each $\mathbf{x}$, the source encoder has a built-in decoder and runs the following iterative algorithm to find $l^*_{\mathbf{x}}$ (This is similar to the algorithm presented in [5]).

Algorithm I

$l^*_{\mathbf{x}} = l_o$, $i^*_{\mathbf{x}} = 1$
$if\, d(\mathbf{x}, l_o, T^*) \leq D$
  $\{while\, d(\mathbf{x}, l^*_{\mathbf{x}}, T^*) \leq D$
    $\{l^*_{\mathbf{x}} = l^*_{\mathbf{x}} - m$
    $i^*_{\mathbf{x}} = i^*_{\mathbf{x}} + 1\};$
  $l^*_{\mathbf{x}} = l^*_{\mathbf{x}} + m\};$
$if\, d(\mathbf{x}, l_o, T^*) > D$
  $\{while\, d(\mathbf{x}, l^*_{\mathbf{x}}, T^*) > D$
    $\{l^*_{\mathbf{x}} = l^*_{\mathbf{x}} + m$
    $i^*_{\mathbf{x}} = i^*_{\mathbf{x}} + 1\};\};$

$i^*_{\mathbf{x}}$ is the number of iterations required to find $l^*_{\mathbf{x}}$, and l_o is an optimum initial length, calculated to minimize $E(i^*_{\mathbf{x}})$ (the calculation method is explained in Appendix A). Note that calculation of $i^*_{\mathbf{x}}$ is not necessary for encoding of the message block $\mathbf{x}$. The counter $i^*_{\mathbf{x}}$ is employed to assess the encoding delay. The reason we set $i^*_{\mathbf{x}} = 1$ in Line 1, is that the algorithm begins by calculating $d(\mathbf{x}, l_o, T^*)$ (Lines 2 *or* 7) that already requires one encoding iteration. The reason $l^*_{\mathbf{x}}$ is incremented in Line 6 is that the "while" loop terminates when $d(\mathbf{x}, l^*_{\mathbf{x}}, T^*)$ has exceeded the threshold D. Therefore, to keep the distortion below D, we have to increase the codeword length. The integer $m \geq 1$ is a parameter that trades off the number of iterations for the codeword length. By increasing m, a larger codeword length (less compression) is found by running the encoding algorithm for a smaller number of iterations (smaller $i^*_{\mathbf{x}}$). The encoding delay associated with each message vector $\mathbf{x}$ is represented by $\Delta^*_{\mathbf{x}} = T^* \times i^*_{\mathbf{x}}$.

Generally speaking, source coding schemes under maximum distortion constraint provide a smaller expected distortion (for the same threshold D) and a smaller average compression rate, $\overline{R} = \frac{E(l^*_{\mathbf{x}})}{n}$. The disadvantages are having a variable-length scheme and an average encoding delay time of $\overline{\Delta} = E(\Delta^*_{\mathbf{x}}) = T^* \times E(i^*_{\mathbf{x}})$.

As stated before, by changing the step size m in the iterative encoding algorithm, one can trade off compression rate for delay. Also, increasing parameter T^* allows for a more accurate decoding at the expense of increasing the delay. A more accurate decoding may keep the distortion below the threshold for a smaller codeword length. This will decrease the compression rate. Therefore, the decoding time, T^*, is another parameter to trade off delay for rate.

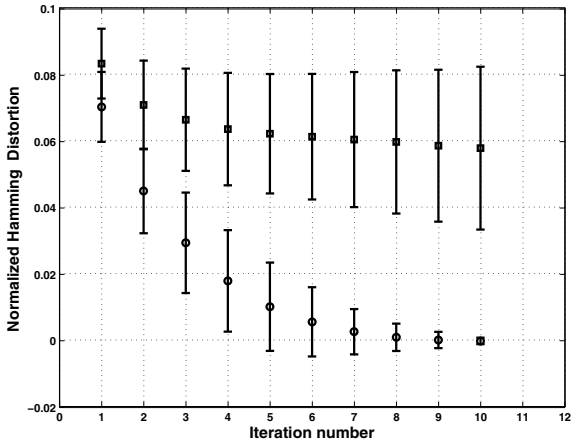

Fig. 1. Mean and standard deviation of $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ (circles) and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ (squares), for $n = 1024$, $m = 64$, and $T^* = 10$. The source is binary i.i.d. with $Pr(1) = p = 0.10$.

III. The Proposed Encoding Algorithm

To explain the idea of our proposed algorithm, let us consider two random processes $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$. For a fixed decoding time t, $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ are functions of the random message vector $\mathbf{x}$, and are regarded as random variables. From (3), it is known that at time $t = T^*$, the probability density functions (or probability mass functions) of $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ are zero for any value greater than D, and less than or equal to D, respectively. Thus, the threshold D can be used to detect the codeword length $l^*_{\mathbf{x}}$. For times $t < T^*$ distributions of $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ are not separated and setting any threshold $\widehat{D}$ will result in detection errors. However, if this auxiliary threshold is set properly, it may result in a small error probability and the errors can be corrected in a second stage of encoding. On the other hand, allowing a decoding time $t < T^*$ will reduce the encoding delay.

Figure 1 shows the mean and standard deviation of $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ (circles) and $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ (squares) for different values of t. The message block length is $n = 1024$, the step size is $m = 64$ bits, and maximum number of iterations is $T^* = 10$. The source is binary i.i.d. with $Pr(1) = p = 0.10$. It is observed that by increasing t, mean and standard deviation of $d(\mathbf{x}, l^*_{\mathbf{x}}, t)$ converge to zero. However the mean of $d(\mathbf{x}, l^*_{\mathbf{x}} - m, t)$ does not converge to zero and the standard deviation slightly increases. The important observation here is that for $t = 1$, the two random variables are distributed very close to each other. However, for all values $t > 1$ (especially for $t > 2$) they are fairly separated and a threshold can be set to detect $l^*_{\mathbf{x}}$.

Based on the above observations, we propose a two-stage encoding algorithm as follows:

Initialization: choose $t < T^*$, $T \geq T^*$, and an auxiliary threshold $\hat{D}$.

Stage 1: Estimate $l^*_{\mathbf{x}}$ as $\widehat{l}_{\mathbf{x}}$, where $\widehat{l}_{\mathbf{x}}$ is the smallest codeword length which provides $d(\mathbf{x}, \widehat{l}_{\mathbf{x}}, t) \leq \widehat{D}$.

Stage 2: Increase $\widehat{l}_{\mathbf{x}}$ to reach the first codeword length $l_{\mathbf{x}}$ with $d(\mathbf{x}, l_{\mathbf{x}}, T) \leq D$.

Notice that in Stage 2, we only increase $\widehat{l}_{\mathbf{x}}$ (if necessary). Therefore, if $T = T^*$, the achieved codeword length is always grater than or equal to $l^*_{\mathbf{x}}$ (since $l^*_{\mathbf{x}}$ is the smallest length to achieve $d(\mathbf{x}, l_{\mathbf{x}}, T^*) \leq D$). In fact, in this case $l_{\mathbf{x}} = Max(\widehat{l}_{\mathbf{x}}, l^*_{\mathbf{x}})$. To decrease $l_{\mathbf{x}}$ and reduce the average compression rate to the rate that is achieved by Algorithm *I*, we let a more accurate decoding in Stage 2 by increasing the number of decoding iterations (choosing $T \geq T^*$). The final codeword length $l_{\mathbf{x}}$ is also an estimate that can be greater or less (because of $T \geq T^*$) than $l^*_{\mathbf{x}}$. However, by choosing proper values for t, $\hat{D}$, and T, the proposed algorithm can achieve $E(l_{\mathbf{x}}) = E(l^*_{\mathbf{x}})$ (i.e. the same average compression rate).

In some cases it is observed that replacing the actual distortion measure, $d(.,.,t)$, by an auxiliary distortion measure, $\hat{d}(.,.,t)$, in Stage 1 will allow for better separation of distribution functions of $d(\mathbf{x}, l_{\mathbf{x}}, t)$ and $d(\mathbf{x}, l_{\mathbf{x}} - m, t)$, and will give a better estimate of the codeword length. Therefore, we will use this auxiliary distortion measure in formal presentation of our proposed algorithm. Also, simulation results contain an example that gives an idea how this auxiliary function may help reduce the delay (see Fig. 4). However, in this paper we do not aim to explore methods to search for good auxiliary measures. Also note that in Stage 2 we still use the actual distortion measure, $d(.,.,T)$.

The proposed algorithm is formally presented below:

Algorithm II

Stage 1:

-Initialization: choose $t < T^*$, $T \geq T^*$, and an auxiliary threshold $\hat{D}$.

-Run Algorithm I to find $\widehat{l}_{\mathbf{x}} = \underset{\hat{d}(\mathbf{x},l,t) \leq \widehat{D}}{Min} l$ after $\hat{i}_{\mathbf{x}}$ iterations.

Stage 2:

-Initialization: Let $i_{\mathbf{x}} = 1$, $l_{\mathbf{x}} = \widehat{l}_{\mathbf{x}}$.

-While $d(\mathbf{x}, l_{\mathbf{x}}, T) > D$

$\{l_{\mathbf{x}} = l_{\mathbf{x}} + m$

$i_{\mathbf{x}} = i_{\mathbf{x}} + 1\}$;

In Stage 1 the iterative encoding begins from an optimum initial length $\hat{l}_o$ calculated based on the probability mass function of $\hat{l}_{\mathbf{x}}$ and according to the approach presented in Appendix A. This initial length will minimize $E(\hat{i}_{\mathbf{x}})$. Then in Stage 2 we execute only the lower half of Algorithm *I* to reach the first codeword length $l_{\mathbf{x}} \geq \hat{l}_{\mathbf{x}}$ with $d(\mathbf{x}, l_{\mathbf{x}}, T) \leq D$. By running this algorithm, the encoding delay for each message block $\mathbf{x}$ is

$$\Delta_{\mathbf{x}} = t \times \widehat{i}_{\mathbf{x}} + T \times i_{\mathbf{x}}. \tag{4}$$

Therefore, we achieve a rate-distortion pair of $\overline{R} = \frac{E(l_{\mathbf{x}})}{n}$, $\overline{\Delta} = t \times E(\widehat{i}_{\mathbf{x}}) + T \times E(i_{\mathbf{x}})$. By changing the function $\hat{d}(.,.,.)$ and the parameters T, t, and $\widehat{D}$, different pairs $(\overline{R}, \overline{\Delta})$ are generated and a *rate-delay region* is found. The lower bound of this region will represent the smallest delay achievable by the proposed algorithm, given a fixed compression rate (or vice versa).

Notice that now the source decoder must perform $T \geq T^*$iterations to successfully decode $\mathbf{x}$ to a distortion below D. However, this increase in decoding time can be tolerated; because the system is constrained by the encoding delay that is larger than the decoding delay.

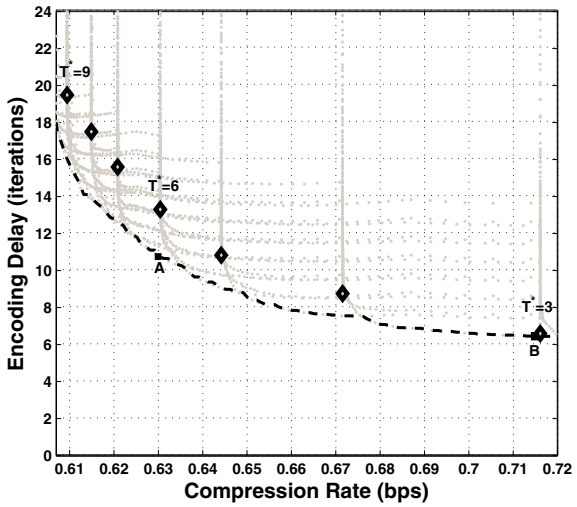

Fig. 2. The rate-delay region achieved by Algorithm *II* for message block length $n = 1024$, and step size $m = 64$. Lower envelope of the region is marked by dashes. Diamonds show rate-delay pairs found by running Algorithm *I* for different values of T^*.

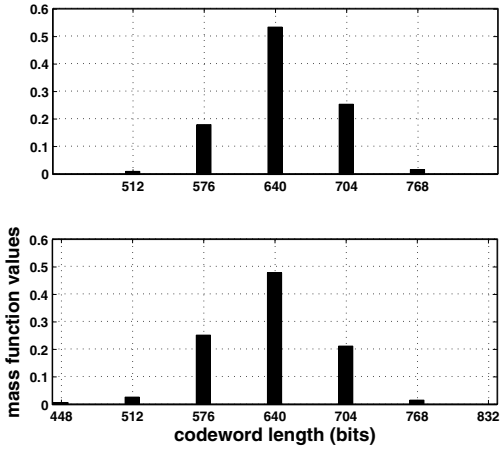

Fig. 3. Distribution of $l^*_{\mathbf{x}}$ for $T^* = 6$ (top) and $\widehat{l}_{\mathbf{x}}$ for $(t, \widehat{D}) = (2, .052)$ (bottom). Message block length is $n = 1024$, and step size is $m = 64$.

IV. Simulation Results

In this section we focus on the case where the source is biased binary i.i.d., i.e. $\mathbf{x}$ consists of n independent bits with $Pr(1) = p < 0.5$. We consider the turbo source coding scheme with the following encoding-decoding functions: (i) The encoding function $f(\mathbf{x}, l)$ is executed by passing $\mathbf{x}$ through two parallel concatenated convolutional codes, interleaving the parities, and then puncturing the interleaved parities to achieve the proper codeword length $l_{\mathbf{x}}$. The second convolutional code is preceded by an interleaver that could be different from the one applied to interleave parities. In practice, all the steps before puncturing are executed only once; then the output is stored and each time is punctured to generated the codewords with the proper lengths. See [5], [7] for details on encoder's structure. (ii) The decoding function, $g(\mathbf{y}, T)$ is executed by performing the usual turbo decoding on codeword $\mathbf{y}$ for T iterations. The only difference is that the source *a priori* knowledge ($Pr(x_i = 1) = p$) is present at the decoder in the form of an L-value, $log\frac{1-p}{p}$. See [5] for details on decoding.

Stage 2 of Algorithm *II* begins by calculating $d(\mathbf{x}, \hat{l}_{\mathbf{x}}, T)$ that requires running turbo decoding on codeword $\mathbf{y} = f(\mathbf{x}, \hat{l}_{\mathbf{x}})$ for T iterations. Since in Stage 1 we already calculated the *extrinsic* L-values for this codeword after t iterations, these L-values may be given to the decoder in Stage 2 to decrease the encoding delay by t iterations. This decrease is considered in presenting our simulation results. For the simulations in this section we use convolutional codes with generator polynomials $\frac{1+D^2}{1+D+D^2}$. The message block length and the step size are set to $n = 1024$, and $m = 64$, respectively. Before delivering to the second convolutional code, the message is interleaved using a random interleaver (generated once and fixed after that). This interleaver is fixed for all simulations. Also, a 4×256, block interleaver is used to interleave parities generated by the convolutional codes.

Figure 2 shows the inner rate-delay region (gray dots) achieved by Algorithm *II* for a binary i.i.d. source with $p = 0.10$, message block length $n = 1024$, and step size $m = 64$. The same normalized Hamming function is used as the auxiliary distortion measure in Stage 1. Other parameters are changed in the following ranges: $2 \leq t \leq 10$ (turbo decoding) *iterations* , $t \leq T \leq 10$ *iterations*, and $0 \leq \hat{D} \leq p$. Turbo decoding *iteration* is used as the unit to measure all delays. The lower envelope of this region is marked by dashes. This envelope is not convex; but it can be made convex by using two encoders and allowing time sharing. The diamonds in Fig. 2 mark rate-delay pairs achieved by running Algorithm *I* for different decoding times, T^*. As observed, most of these points fall above the lower envelope, which means by running Algorithm *II* we are able to achieve smaller encoding delays for the same compression rates. For example, setting $T^* = 6$ in Algorithm *I*, gives a compression rate $\overline{R} = 0.630$ bps. The distribution of codeword length $l^*_{\mathbf{x}}$ is depicted in Fig. 3. On the average, this encoding takes $E(\Delta^*_{\mathbf{x}}) = 13.33$ turbo decoding iterations. However, by setting $(T, t, \hat{D}) = (7, 2, .052)$, Algorithm *II* achieves the same compression rate by spending an average encoding delay of $E(\Delta_{\mathbf{x}}) = 10.78$ iterations (point **A** in Fig. 2). For this, Algorithm *II* first executes Stage 1 by setting $(t, \hat{D}) = (2, .052)$ and achieves an estimated codeword length $\hat{l}_{\mathbf{x}}$. The distribution of this estimated length is shown in Fig. 3. Then the algorithm sets the codeword length to $l_{\mathbf{x}} = \hat{l}_{\mathbf{x}}$ and the decoding time to $T = 7$ (Stage 2) and continues by incrementing $l_{\mathbf{x}}$ until achieving zero distortion. Notice that this time $T = 7$ iterations are required to reconstruct the message at the decoder side (see Remark III). By calculation, $\frac{E(\Delta^*_{\mathbf{x}} - \Delta_{\mathbf{x}})}{E(\Delta^*_{\mathbf{x}})} = 0.191$, i.e. Algorithm *II* decreases the encoding delay by 19.1%. As the compression rate increases, the modified algorithm becomes less effective. For example, the rate-delay point achieved by $T^* = 3$, is very close to its corresponding point on the lower envelope (marked by point **B** in Fig. 2). Table I lists different rate-delay points found by Algorithm *I* and compares them with the point on the lower envelope with the same rate.

To illustrate the effect of auxiliary distortion measure, we define an auxiliary measure that is a function of the vector of *a posteriori* L-values. If the vector of *a posteriori* values is

TABLE I
AVERAGE ENCODING DELAY USING ALGORITHM *I* WITH PARAMETER T^*, AND ALGORITHM *II* WITH PARAMETERS $(T, t, \widehat{D})$. THE MESSAGE BLOCK LENGTH IS $n = 1024$, AND THE STEP SIZE IS $m = 64$.

T^*	$(T, t, \widehat{D})$	$E(\Delta^*_{\mathbf{x}})$	$E(\Delta_{\mathbf{x}})$	$\overline{R}$
9	$(10, 3, .053)$	19.50	15.66	0.610
8	$(9, 3, .048)$	17.52	13.88	0.615
7	$(8, 3, .045)$	15.62	12.79	0.620
6	$(7, 2, .052)$	13.33	10.78	0.630
5	$(6, 2, .044)$	10.86	9.01	0.645
4	$(5, 2, .034)$	8.78	7.62	0.670
3	$(4, 2, .018)$	6.62	6.48	0.715

represented by $\mathbf{v} = v_1 v_2 ... v_n$, then we know [5]:

$$Pr(\hat{x}_i = x_i) = \begin{cases} \frac{e^{v_i}}{1+e^{v_i}}, & x_i = 0 \\ \frac{1}{1+e^{v_i}}, & x_i = 1 \end{cases}, \quad (5)$$

where $\mathbf{x} = x_1 x_2 ... x_n$, $\widehat{\mathbf{x}} = \widehat{x}_1 \widehat{x}_2 ... \widehat{x}_n$. After running t turbo decoding iterations, we calculate the values $Pr(\hat{x}_i = x_i)$ from the vector $\mathbf{v}$; then we define the auxiliary distortion measure as

$$\hat{d}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} Pr(\hat{x}_i \neq x_i), \quad (6)$$

where $Pr(\hat{x}_i \neq x_i) = 1 - Pr(\hat{x}_i = x_i)$. This distortion measure takes all real values from 0 to 1, in contrast to the normalized Hamming distortion measure that only takes $n + 1$ values $(0, \frac{1}{n}, ..., 1)$. Figure 4 shows the lower envelope of the rate-delay region, without and with applying the auxiliary distortion measure in Stage 1 of Algorithm *II*. Three types of binary i.i.d. sources with $p = 0.08, 0.10, 0.12$ are considered. It is observed that in all cases, applying the auxiliary measure slightly improves the performance of the encoding algorithm (up to 1.0 iterations). By exploring better auxiliary distortion measures, one might further decrease the delay of turbo source encoding.

V. CONCLUSION

We proposed a two-stage algorithm to reduce encoding delay of lossless turbo source coding. By defining an auxiliary threshold, and allowing a small decoding time, the algorithm gives a quick estimate of the codeword length. Then this estimated length is iteratively increased to reach zero distortion. In contrast to previous works that achieve lower delays by reducing the message block length, the proposed algorithm achieves lower delays for the same block length; therefore, the actual "per bit" encoding delay is decreased. According to our numerical results, the proposed algorithm reduces the encoding delay up to 19%. We also showed that employing an auxiliary distortion measure in the first stage of encoding, could give a better estimate of the codeword length; that results in a further reduction of the encoding delay.

APPENDIX A
CALCULATING THE OPTIMUM INITIAL CODEWORD LENGTH

The codeword length $l^*_{\mathbf{x}}$ is a discrete random variable with a probability mass function (see Fig. 3 for example). Let $q_l = Pr(l^*_{\mathbf{x}} = l)$ be the mass function value for length l. Now let Algorithm *I* begin by setting $l^*_{\mathbf{x}} = l_o$, for all $\mathbf{x}$ and for some fixed integer l_o. From Algorithm *I*, it is observed that for all

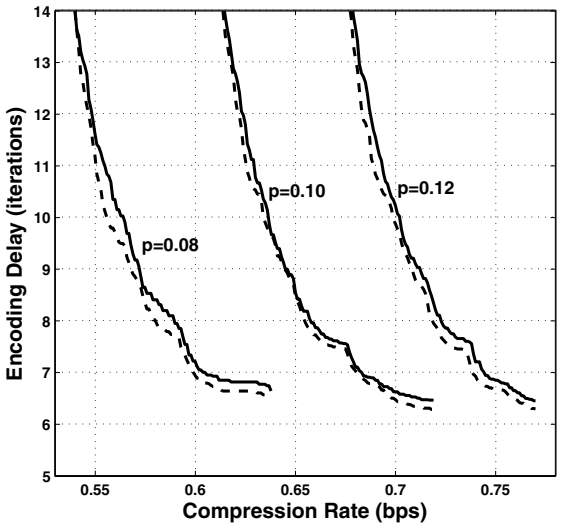

Fig. 4. The lower envelope of the rate-delay region without applying auxiliary distortion measures (Solid) and with applying auxiliary distortion measure (Dashes). Message block length is $n = 1024$, and step size is $m = 64$ bits.

message blocks $\mathbf{x}$ with $l^*_{\mathbf{x}} \leq l_o$ (i.e. $d(\mathbf{x}, l_o, T^*) \leq D$), the required number of encoding iterations is $i^*_{\mathbf{x}} = i^*_{l^*_{\mathbf{x}}} = l_o - l^*_{\mathbf{x}} + 2$. Also, for every $\mathbf{x}$ with $l^*_{\mathbf{x}} > l_o$ (i.e. $d(\mathbf{x}, l_o, T^*) > D$), the required number of iterations is $i^*_{\mathbf{x}} = i^*_{l^*_{\mathbf{x}}} = l_o - l^*_{\mathbf{x}} + 1$. We choose the initial length l_o that minimizes $E(i^*_{\mathbf{x}} | \, l_o) = \sum_{l^*_{\mathbf{x}}} q_{l^*_{\mathbf{x}}} i_{l^*_{\mathbf{x}}}$.

ACKNOWLEDGMENT

The authors would like to thank the Natural Science and Engineering Council of Canada for its support.

REFERENCES

[1] J. Garcia-Frias, and Ying Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Communications letters*, vol. 5, No. 10, pp. 417-419, Oct. 2001.

[2] J. Garcia-Frias, and Y. Zhao, "Compression of binary memoryless sources using punctured turbo codes," *IEEE Commun. Lett.*, Vol. 6, No. 9, pp. 394-396, Sept. 2002.

[3] J. Bajcsy, and P. Mitran, "Coding for the Slepian-Wolf problem with turbo codes," *In Proc. IEEE Globecom 2001*, pp. 1400-1404, Nov. 2001.

[4] A. Aaron, and B. Girod, "Compression with side information using turbo codes," *In Proc. Data Compression Conf., DCC 2002*, pp. 252-261, April 2002.

[5] J. Hagenauer, J. Barros, and A. Schaeffer, "Lossless turbo source coding with decremental redundancy," *in Proc. 5th int. ITG conf. on Source and Channel Coding, SCC 04*, Erlangen, Germany, Jan. 2004.

[6] N. Dütsch, "Code optimisation for lossless compression of binary memoryless sources based on FEC codes," *European Transactions on Telecommunications,* Vol. 17, Issue 2, March-April 2006, pp. 219-229.

[7] J. Haghighat, W. Hamouda, and M. R. Soleymani, "Design of Lossless Turbo Source Encoders," *IEEE Signal Processing Letters*, vol. 13, Issue 8, Aug. 2006, pp. 453-456.

[8] G. Caire, S. Shamai, and S. Verdu, "Noiseless data compression with low-density parity-check codes," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 66, pp. 263-284, 2004, American Mathematical Soc.

[9] G. Caire, S. Shamai, A. Shokrollahi, and S. Verdu, "Fountain codes for lossless data compression," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 68, pp. 1-20, 2006, American Mathematical Soc.

[10] R. Ansari, N. Memon, and E. Ceran, "Near-lossless Image Compression Techniques," Journal of Electronic Imaging, pp. 486-494, July 1998.