A Modified Belief Propagation Algorithm for Code Word Quantization

Phillip A. Regalia, Fellow, IEEE

Abstract—Modern coding advances, including dirty paper coding and information hiding, require quantizing a given binary word to a code word. A 'good' solution would approach the rate-distortion bound in lossy source compression. Here we propose a simple variant on belief propagation which is observed to converge to a solution giving respectable rate-distortion performance. Comparisons with other recently proposed source quantization methods reveal that the proposed algorithm holds particular interest in short block-length applications, as encountered in packet-based communication systems.

Index Terms—Source compression; code word quantization; rate-distortion theory; information hiding; dirty paper coding.

I. INTRODUCTION

The general decoding problem for linear binary codes is to deduce a minimum-weight error pattern consistent with a given error syndrome. This problem arose in early studies of error correction codes since, under reasonable channel assumptions, the minimum weight correction to an error-prone received block of bits gives the maximum likelihood estimate of the true code word sent. Although conceptually straightforward, the general decoding problem actually belongs to the class of NP-complete problems [1], [2], [3], thus challenging the development of efficient algorithms for its resolution. The hard aspect of this problem also makes it use in cryptography (see, e.g., [4]) and underlies heuristic search procedures for low-weight error patterns to assess cryptographic strength [5], [6], [2], [7].

The general decoding problem has met with resurgent interest in modern coding applications, including dirty paper coding for multi-user communications [8], information hiding with robustness [9], [10], [11], and wet paper coding in steganography [12], [13], [14]. In these more recent applications, the error syndrome is replaced by constraints deriving from side information (typically messages to be hidden or users to be accommodated), thus exposing the duality with source coding and quantization [9], [10], [11], [15]. In this vein, recent algorithmic work has focused on approaching the rate-distortion bound for lossy source compression (e.g., [16]–[22]) using variants of belief propagation combined possibly with heuristic search procedures. Various reported results (such as [18], [19], [22]) indeed approach the theoretical rate-distortion curve, albeit at an appreciable level of algorithmic complexity.

The intent of this note is to propose a simple modification to the standard belief propagation algorithm which, when used in a low-density generator matrix configuration, is observed to yield a convergent algorithm for code word quantization, offering respectable rate-distortion performance. The resulting algorithm is significantly simpler than the survey propagation algorithm [16] and its variants [18]–[21], [23], or even pruning techniques applied to standard belief propagation [24], thus rendering it potentially suitable for real-time applications. At the same time, its performance displays consistent behavior over variations in block length and choice of generator matrix as compared to the message-passing algorithm of Murayama [25], although the latter is arguably still to be preferred in the long block-length case.

II. BACKGROUND

Let \( \mathbf{H} \) be an \((N-K) \times N\) parity check matrix comprised of ones and zeros, operating on the Galois field \( \mathbb{GF}(2) \) (modulo-2 arithmetic over integers). The set of vectors \( \mathbf{c} \in [\mathbb{GF}(2)]^N \) in its null space defines a linear code \( C \) of rate \( K/N \):

\[
\mathbf{H} \mathbf{c} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{c} \in C.
\]

The set of vectors \( \mathbf{y} \in [\mathbb{GF}(2)]^N \) which instead produce a prescribed syndrome \( s \in [\mathbb{GF}(2)]^{N-K} \) comprise the coset \( C(s) \) for that syndrome:

\[
C(s) = \{ \mathbf{y} \in [\mathbb{GF}(2)]^N : \mathbf{H} \mathbf{y} = s \}.
\]

The code \( C \) induced by \( \mathbf{H} \) is simply the coset for the zero syndrome: \( C = C(0) \). Note that with \( \mathbf{y} \in C(s) \) and \( \mathbf{c} \in C \), their modulo-2 sum remains in the coset: \( \mathbf{y} + \mathbf{c} \in C(s) \).

Suppose we are given an \( N \)-element binary vector \( \mathbf{w} \in [\mathbb{GF}(2)]^N \) and consider the problem of finding an \( N \)-element binary vector \( \mathbf{y} \in [\mathbb{GF}(2)]^N \) which minimizes the Hamming distance \( d(\mathbf{w}, \mathbf{y}) \) subject to \((N-K)\) parity constraints:

\[
\min_{\mathbf{y}} d(\mathbf{w}, \mathbf{y}), \quad \text{subject to} \quad \mathbf{s} = \mathbf{H} \mathbf{y} \tag{1}
\]

This problem arises in information hiding [9], [10], [11], dirty paper coding [8] (once transcribed to its \( \mathbb{GF}(2) \) setting [10]), and wet paper coding [12], [13].

If the bits in \( \mathbf{w} \) hail from a uniform source, then from rate-distortion theory [26], the minimum average distortion

\[
D_{\min} = \frac{1}{2^N} \sum_{\mathbf{w} \in [\mathbb{GF}(2)]^N} \min_{\mathbf{y} \in C(s)} \frac{d(\mathbf{w}, \mathbf{y})}{N}
\]

relates to the code rate \( K/N \) via the rate-distortion curve [26]

\[
H_2(D_{\min}) \geq 1 - (K/N) \tag{2}
\]

Manuscript received _______; revised _______; parts of this work were presented at the Asilomar Conference on Circuits, Systems and Computers, Nov. 2007.

P. A. Regalia is with the Department of Electrical Engineering and Computer Science, Catholic University of America, Washington, DC 20064; regalia@cua.edu. This work was supported by the National Science Foundation under grant CCF 0634757.
where \( H_2(p) = -p \log p - (1 - p) \log(1 - p) \) is the binary entropy function.

Finding a closest \( y \) is equivalent to finding a minimum weight vector consistent with the syndrome:

\[
\min_{\mathbf{e}} d(\mathbf{e}, \mathbf{0}) \quad \text{subject to} \quad \mathbf{s} = \mathbf{H}\mathbf{e}
\]

This vector \( \mathbf{e} \) is the coset leader of \( \mathcal{C}(\mathbf{s}) \), and the solution to (1) becomes \( \mathbf{y} = \mathbf{w} + \mathbf{e} \) (using the modulo-2 sum). This minimum weight problem, in turn, is known to belong to the class of NP-complete problems [1], [6], [2], [3], and thus the decoding problem (1) is hard.

Conventional belief propagation decoding applied to this problem, with \( \mathbf{w} \) a “received” vector, \( \mathbf{s} \) the “side information,” and \( \mathbf{y} \) the “hidden measurement” (as elucidated in [17]), does not in general converge to a meaningful solution, unless \( \mathbf{w} \) is fortuitously within a decoding vicinity of \( \mathcal{C}(\mathbf{s}) \) (e.g., [21]). As such, other approaches must be pursued.

A promising approach is offered by the low-density generator matrix formulation (e.g., [27], [28], [20], [21]). Let \( \mathbf{G} \) be an \( N \times K \) generator matrix for the code \( \mathcal{C}(\mathbf{0}) \), so that each code word \( \mathbf{c} \in \mathcal{C}(\mathbf{0}) \) may be expressed as

\[
\mathbf{c} = \mathbf{Gz}, \quad \text{for some} \ K\text{-bit vector} \ \mathbf{z}.
\]

If \( \mathbf{x} \) is a particular solution satisfying the parity constraint \( \mathbf{s} = \mathbf{Hx} \), then \( \mathbf{x} + \mathbf{c} \) also satisfies the parity constraint for any \( \mathbf{c} \in \mathcal{C}(\mathbf{0}) \), and indeed, all elements of the coset \( \mathcal{C}(\mathbf{s}) \) may be so expressed. As such, finding a minimum weight error vector \( \mathbf{e} \) is equivalent to minimizing the Hamming distance

\[
\min_{\mathbf{z}} d(\mathbf{x}, \mathbf{Gz}).
\]

An effective algorithm for this purpose has emerged via survey propagation [16] and its variants [18]–[21], [23], which share a message-passing nature with belief propagation, but require in general many runs with pruning stages between runs in order to converge. This results in a higher computational complexity. A simpler alternative would thus be of interest.

III. BELIEF PROPAGATION

We review the standard belief propagation algorithm here to facilitate the modified algorithm of section IV. Consider the factor graph of the generator equation \( \mathbf{x} = \mathbf{Gz} \), as in Figure 1. If \( \mathbf{x} \) is not in the column space of the generator matrix \( \mathbf{G} \), this gives an inconsistent system; belief propagation attempts to iteratively refine the information bits in \( \mathbf{z} \) so as to reconcile \( \mathbf{Gz} \) with \( \mathbf{x} \). The algorithm passes messages along edges in the graph between variable nodes (denoted as circles) and parity-check nodes (denoted as squares) [29]. These messages consist of two-element vectors containing pseudo-probabilities that sum to one.

The update equations at the variable nodes appear as

\[
m_{z_i \rightarrow f_j}(z_i) = \zeta \Pr(z_i) \prod_{k \in F(i), k \neq j} m_{k \rightarrow z_i}(z_i)
\]  

(3)

where \( m_{k \rightarrow z_i}(z_i) \) denotes an incoming messages at variable node \( i \), \( F(i) \) contains the indices \( k \) whose parity-check nodes \( f_k \) connect to variable node \( z_i \), and \( m_{z_i \rightarrow f_j}(z_i) \) denotes an outgoing message sent from variable node \( z_i \) to parity-check node \( f_j \). The scale factor \( \zeta \) (for a given node) is chosen to ensure that evaluations sum to one:

\[
m_{z_i \rightarrow f_j}(0) + m_{z_i \rightarrow f_j}(1) = 1.
\]

The variable node probability \( \Pr(z_i) \) reflects any \textit{a priori} information on the information bit \( z_i \), if available.

The update equations at the parity check nodes read as

\[
m_{f_j \rightarrow z_i}(z_i) = \frac{1}{2} \left\{ 1 + (-1)^{z_i} \left( 1 - 2 m_{z_i \rightarrow f_j}(1) \right) \prod_{k \in V(j), k \neq j} \left( 1 - 2 m_{z_k \rightarrow f_j}(1) \right) \right\}
\]  

(4)

where \( V(j) \) contains the indices \( k \) of the variable nodes \( z_k \) which enter into the \( j \)-th parity check. The “flooding schedule” runs (3) at each variable node, followed by (4) at each parity-check node, in successive iterations. If convergence occurs, the belief values for the bits \( \{z_i\} \) are given as

\[
b_i(z_i) = \zeta \Pr(z_i) \prod_{k \in F(i)} m_{k \rightarrow z_i}(z_i),
\]

where, again, \( \zeta \) ensures that evaluations \( b_1(0) \) and \( b_1(1) \) sum to one.

Let \( \tilde{x}_i \in \{0, 1\} \) denote the candidate \( i \)-th codeword bit from the generator matrix, as distinguished from the actual bit \( x_i \) the algorithm is attempting to match. By conventional belief propagation, the message \( m_{z_i \rightarrow f_j}(\tilde{x}_j) \) is that sent from the parity-check node \( q_j(\tilde{x}_j) \) (right-most squares in Figure 1) to the variable node \( x_j \). Setting \( q_i(1) \) to some “soft” probability skewed inward from the hard value \( x_j \) gives an algorithm which, in general, converges to seemingly meaningless probabilities. Setting \( q_i(1) = x_j \) (hard values) gives, in general, an inconsistent system, resulting in numerical singularities.

IV. TRUTHINESS PROPAGATION

To overcome the shortcomings of standard belief propagation, consider the choice

\[
q_j(1) = \alpha x_j + (1 - \alpha) m_{x_j \rightarrow q_j}(1)
\]  

(5)
Fig. 2. Messages at \( j \)-th parity-check node \( f_j(z) \), to illustrate modified algorithm. The constraint node \( q_j \) sends back a convex combination of its hard constraint \( x_j \) and its incoming message \( m_{x_j \rightarrow q_j} \), controlled by a “truthiness” factor \( \alpha \) in (5).

where \( 0 < \alpha < 1 \) and

\[
m_{x_j \rightarrow q_j}(\hat{x}_j) = m_{f_j \rightarrow x_j}(\hat{x}_j) = \frac{1}{2} \left\{ 1 + (-1)^{\hat{x}_j} \prod_{k \in V(j)} (1 - 2m_{z_k \rightarrow f_j}(1)) \right\} (6)
\]

is the message sent from parity-check node \( f_j \) to variable node \( x_j \); cf. Figure 2. The value \( m_{f_j \rightarrow x_j}(1) \) is the probability that the information bits \( \{z_k\} \) which impinge on the \( j \)-th parity check produce an odd parity [30], giving thus the “natural parity” (i.e., without the constraint from \( x_j \)) at that check node.

The rationale for choosing the convex combination in (5) is that if the natural parity agrees with the constraint \( x_j \), the constraining node (labeled \( q_j \) in fig. 2) maintains this “hard” constraint. If instead the natural parity is opposite to the constraint \( x_j \), the probability fed from \( q_j \) is softened, thus allowing further refinement in the message passing algorithm. The operation at the constraining node is akin to a system which feeds back not the actual “truth” (as strict belief propagation would do), but rather what it “wishes” to be true, via injection of the natural parity. Accordingly, we dub the algorithm truthiness propagation\(^1\), obtained by running (3), (6), (5) [giving \( m_{x_j \rightarrow f_j}(1) = q_j(1) \)], and then (4), and iterating; the resulting algorithm is summarized in Table I. Interestingly, this simple modification gives an algorithm that is observed to converge to a quantizing solution (the beliefs \( b_i(z_i) \) tend to 0 or 1), yet features negligible overhead compared to standard belief propagation. It is thus an order of magnitude simpler than the survey propagation methods developed in [16], [18]–[21], or the pruning approach applied to belief propagation [24].

V. SIMULATION RESULTS

Performance is compared with the Thouless-Anderson-Palmer approach of Murayama [31], [25], which is likewise a message passing algorithm derived from the factor graph of a low density generator matrix. Although the algorithm as published in [25] requires hyperbolic tangent and hyperbolic arctangent operations (presenting a significantly higher computational complexity per iteration), some straightforward algebraic manipulations can transform the algorithm into the computationally simpler sum-product form summarized in Table II. In this algorithm, the parity-check nodes integrate soft probabilities derived from \( \{x_i\} \), as controlled by a parameter \( \beta \) in the notations of [31], [25], while the variable nodes replace

\(^1\text{truthiness} \text{ (noun): “The quality of preferring concepts or facts one wishes to be true, rather than concepts or facts known to be true” (Merriam-Webster).}
the prior probabilities \( \Pr(z_i) \) by softened beliefs from the previous iteration, as controlled by a reinjection parameter \( \gamma \). As with the parameter \( \alpha \) of the proposed algorithm, analytic formulas for the optimal choices of \( \beta \) and \( \gamma \) are not available at present, but are instead adjusted manually. The survey propagation algorithm of [19], [18], finally, requires even higher computational complexity per iteration, additional parameters requiring manual adjustment (denoted \( w_{\text{non}} \) and \( w_{\text{info}} \) in [19]), and subsequent pruning steps between runs, as with [24]. These pruning-based approaches give a more cumbersome procedure outright, and although the reported quantization performance in [19] and [24] is close to the rate-distortion bound, the algorithms are not included in the comparisons to follow.

Regular low density generator matrices were initially used, as in [25], featuring two ones per row and \( C \) ones per column, giving an adjustable code rate of \( R = 2/C \). Using a block length of 300 bits, the obtained rate-distortion performance is plotted in figure 3. The average distortion at a given rate is

\[
D = \frac{E[d(x, Gz)]}{N}
\]

in which the expectation is replaced by the empirical mean of 20,000 quantization runs (40 different generator matrices, with 500 independent runs each), using identical data for the two algorithms at each run. (The conventional belief propagation algorithm was also run, but gave an average distortion of about 0.5 at each rate, which is no better than a coin toss.)

The seemingly poor performance of the T-A-P algorithm is due to the short block length. Figure 4 shows a histogram of the Hamming distortion rate over the 20,000 runs for the two algorithms at rate \( R = 1/2 \), illustrating a "double-hump" character for the T-A-P algorithm. (Similar histograms were observed for all code rates). The left-most hump shows excellent quantization performance; the right-most hump reflects the sizeable number of runs for which the beliefs failed to converge to a meaningful solution, even after 1000 iterations of the message passing algorithm. The truthiness propagation algorithm, by contrast, shows a single hump, exhibiting low variance about the mean value. In addition, whereas the convergent runs of the T-A-P algorithm required typically hundreds of iterations for the belief values to settle, the truthiness propagation algorithm was observed to converge within tens of iterations in most runs.

Simulations indicate that if the block length is increased into the tens of thousands, the left-most hump for the T-A-P scheme dominates the histogram; the performance then concurs with the respectable results reported in [25]. This indicates that the convergence difficulties noted here are likely due to a factor graph girth deficiency which becomes more prominent in the short block length case. Interestingly, the truthiness propagation algorithm does not appear to suffer these convergence difficulties, despite using the same generator matrices.

Further performance improvements were observed using irregular generator matrices, with the results plotted in figure 5; the degree distribution polynomials (in the formulation of [32]) for the generator matrices are listed in table III for code rates up to \( R = 0.6 \); codes of higher rate were obtained as the duals of lower rate codes from table III. The T-A-P scheme, unfortunately, failed to converge for any generator matrices featuring check node degrees exceeding two, indicating that the formulation is apparently functional only for the "pairwise interaction" model. The conventional belief propagation algorithm did converge for specific code rates using the irregular generator matrices, although the quantization performance, as plotted in figure 5, is less than spectacular.

**VI. CONCLUDING REMARKS**

We have proposed a simple modification to the standard belief propagation algorithm, which is observed to yield a convergent algorithm for the code word quantization problem. The algorithm is much simpler in implementation and complexity compared to survey propagation [16], [18]–[21], and

---

2 Simulations indicate that reducing \( \alpha \) or \( \beta \) generally gives lower quantization distortion, until a lower limit on either is reached, below which the respective algorithms fail to converge. The critical lower value for \( \alpha \) or \( \beta \) decreases with the code rate; empirically, the lower bound \( \alpha \geq 1 - 2D_{\text{min}}(K/N) \) with \( D_{\text{min}} \) the minimum Hamming distortion at rate \( K/N \) [cf. (2)], was found to work well. Some indications on choosing \( \beta \) are suggested from [31]. As in [25], \( \gamma = 0.1 \) was used.
avoiding the cumbersome pruning and decimation steps required of that procedure and others [24]. Its main advantage concerns its consistent performance in the short block-length case. For longer block lengths, the quantization results reported in [25], [19], [21] inch slightly closer to the rate-distortion curve. For such cases, the algorithm of [25] (once transcribed to its sum-product form of table II) is perhaps preferred, in view of its simple computational requirements and excellent performance, at least for pairwise interaction models.

REFERENCES


