Decoding LDPC Codes with Mutual Information-Maximizing Lookup Tables

Francisco Javier Cuadros Romero and Brian M. Kurkoski
School of Information Science
Japan Advanced Institute of Science and Technology
Ishikawa, Japan 923–1292
Email: fco.cuadros@jaist.ac.jp, kurkoski@jaist.ac.jp

***Abstract*—A recent result has shown connections between statistical learning theory and channel quantization. In this paper, we present a practical application of this result to the implementation of LDPC decoders. In particular, we describe a technique for designing the message-passing decoder mappings (or lookup tables) based on the ideas of channel quantization. This technique is not derived from sum–product algorithm or any other LDPC decoding algorithm. Instead, the proposed algorithm is based on an optimal quantizer in the sense of maximization of mutual information, which is inserted in the density evolution algorithm to generate the lookup tables. This algorithm has low complexity since it only employs 3–bit messages and lookup tables, which can be easily implemented in hardware. Two quantized versions of the min–sum decoding algorithm are used for comparison. Simulation results for a binary–input AWGN channel show 0.3 dB and 1.2 dB gains versus the two quantized min–sum algorithms. On the binary symmetric channel also a gain is seen.**

***Keywords*—LDPC codes, lookup tables, quantization**

I. Introduction

The sum-product algorithm (SPA) also called belief–propagation (BP) algorithm is an excellent decoding algorithm for LDPC codes [1], but its efficient implementation in hardware is a serious concern. As LDPC codes become more and more widely used, reducing the power consumption of the decoder remains a significant problem. A natural way to reduce power consumption is to reduce the number of bits used to represent the messages in a message-passing decoder. This topic has been intensively studied in the past several years, and some of the results are reviewed in the next section.

Recently, the problem of optimal quantization of communications channels has been shown to have a connection with the problem of classification from statistical learning theory [2]. A common classification problem deals with a Markov chain $X \to Y \overset{Q}{\to} Z$, where X is a variable of interest, Y is an observation, Q is a function called a *classifier* and Z is the classification, essentially an estimate of X [4]. A well-known metric for classification is to minimize conditional entropy [3]:

$$\min_{Q} H(X|Z), \tag{1}$$

which is equivalent to:

$$\max_{Q} I(X;Z) = H(X) - \min_{Q} H(X|Z), \tag{2}$$

and moreover, this quantization can be performed efficiently when X is binary [2]. In the case of channel quantization, X is the channel input, Y is the channel output, and Z is the quantized channel output.

The subject of this paper is the practical application of this quantization method. In particular, we present an LDPC–LUT decoding algorithm, with two distinctive characteristics: 1) it is not derived from sum–product algorithm or other LDPC decoding algorithms, and 2) it only uses lookup tables (or decoding mappings), which are based on the maximization of mutual information.

The idea of using lookup tables for LDPC decoding is not new, but in this paper the technique for designing the lookup tables is new. Mutual information is a reasonable design criterion because the channel capacity is precisely a maximization of mutual information.

Lookup tables are considered because they have a great simplicity in their implementation as an array indexing operation. It is possible to use one lookup table per check/variable node operation in the decoding process but, depending on both the degree of the node and the message precision in bits, the size of the lookup table can be large and consume too many resources. To solve this problem we apply two approaches. First, we perform a decomposition at the nodes, which generates a set of two–input lookup tables, and second, we use 3–bit per message, which reduces directly the size of the lookup tables. The lookup tables are generated in the context of density evolution [13]. Since the lookup tables are built by the optimal quantizer presented in [2], the lookup tables are locally optimal, in the sense of maximizing mutual information at each decoding iteration; we cannot say anything about global optimality over all iterations.

The remainder of the paper is organized as follows: in Section II-A some previous work on LDPC message representation is reviewed. In Section II-B, the quantizer which maximizes mutual information is reviewed. In Section II-C, notation on LDPC codes is given. In Section III, a density evolution algorithm with quantization is described. In Section IV, advantages of a decomposition of a lookup table are described. In Section V, the simulation results are shown. Finally, conclusions are stated in Section VI.

II. Background

A. Previous Work

The efficient representation of messages in LDPC decoding algorithm has been a topic of intensive study, due to its practical importance. Naturally, LDPC message–passing decoding

algorithms with high complexity have better performance than low complexity algorithms. But low complexity algorithms with a reasonable loss are desirable for reliable hardware implementations because they have low power consumption, reduced size and lower cost. This section describes some of the previous work on this topic.

In [5], the normalized BP decoding algorithms with quantized messages has performance close to BP, with a suitable choice of normalization factor. In [6] a parity likelihood ratio technique using 6 bits is presented for overcoming the BP algorithm's sensitivity to quantization effects.

Quantized LDPC belief-propagation decoders can be designed by considering mutual information that produce a nonuniform message quantization, which using four bits per message is quite close to unquantized performance [7]. This is significant since conventional uniform quantization requires about six bits per message to achieve similar error performance. But that technique required hand optimization.

On the other hand, the min–sum (MS) algorithm [8] has an error performance just a few tenths of a decibel inferior to SPA, and this is simpler for hardware implementation. Since min–sum is less complex, in [9] the effects of clipping and quantization on the performance of MS are studied.

In [10] a 3–bit Finite Alphabet Iterative Decoder (FAID) was presented. FAIDs are designed using the knowledge of potentially harmful subgraphs that could be present in a given code. Results were presented for the binary symmetric channel (BSC). On loopy graphs, performance better than BP was obtained.

On the other hand, the bit–flipping algorithm [1], works with hard information (1-bit messages) instead of soft information (floating point messages), as the SPA and MS do. Work on bit–flipping algorithm has also been described in [11] and [12], although these algorithms' error performance is still far from that achieved by the MS algorithm.

B. Quantizer that maximizes mutual information

In this section, we briefly review the optimal quantization algorithm that maximizes mutual information [2], since this is applied to build the set of lookup tables and also to perform the channel quantization.

The framework of this quantizer and its considerations are as follows. Consider a conditional probability distribution on Y given X (in some cases, this conditional probability distribution represents a discrete memoryless channel (DMC)). A quantizer Q maps B values of Y to K values of Z, which is a new random variable (in the case of a DMC, this is the quantized output). The alphabet size of X, Y and Z are 2 (binary), B and K respectively. Then, $p_j = Pr(X = j)$ and $P_{b|j} = Pr(Y = b|X = j)$ for $j = 1, 2$ and $b = 1, \ldots, B$. The case of interest and considered in this paper is when $K < B$.

Assuming $\mathcal{Q}$ as the set of all possible quantizers, the optimal quantizer Q^* that maximizes the mutual information $I(X;Z)$ between X and Z is:

$$Q^* = \underset{Q \in \mathcal{Q}}{\operatorname{argmax}} I(X;Z).$$

Therefore, Q^* is a matrix of size $K \times B$, where for each distribution output b there is exactly one value k', for which $Q_{k'|b} = 1$, and for all other values of k, $Q_{k|b} = 0$ [2, Lemma 2]. For a given quantizer value k let $\mathcal{A}_k$ be the set of values b for which $Q_{k|b} = 1$. The quantizer is a mapping from B values of Y to K values of Z. Under this mapping $\mathcal{A}_k$ is the preimage of k. The sets $\mathcal{A}_m$ and $\mathcal{A}_n$ are disjoint for $m \neq n$ and the union of all the sets is $\{1, 2, \ldots, B\}$. For convenience the algorithm is denoted as $Q^* = \text{Quant}(P_{b|j}, K)$.

C. LDPC Codes

An LDPC code is defined by a $K_c \times N$ parity check matrix $\mathbf{H}$ that has low density of 1's (in the LDPC literature K is used insted of K_c, but we previously used K to denote the quantization levels in Section II-B, which is the same notation used in [2]). When the $\mathbf{H}$ matrix that represents an LDPC code has the same number of 1's in each column and row, the code is called a *regular* LDPC code, otherwise it is called an *irregular* LDPC code. Throughout this work we shall consider only binary regular LDPC codes with block length N, number of parity bits no more than K_c, number of information bits at least $M = N - K_c$, number of 1's per column as d_v (degree of the variable node) and number of 1's in each row as d_c (degree of the check node). Therefore, the rate of a *regular* LDPC code is $R \geq 1 - \frac{d_v}{d_c}$, with equality when $\mathbf{H}$ is of full rank.

III. Density evolution with quantization

The lookup tables are generated in the context of density evolution [13]. Classical density evolution is restricted to channels with certain symmetry properties. But here, arbitrary and asymmetrical channels are allowed, and the optimized decoding lookup tables, and thus the distributions, may be asymmetrical even if the channel was symmetrical. Wang et al. generalized density evolution to asymmetric channels [14]. They showed that while error rates are codeword–dependent, it is sufficient to consider the evolution of densities only for the two code bits, that is densities conditioned on $X = 0$ and $X = 1$. The same method will be used here.

An arbitrary, binary-input DMC with input X and output W is used for transmission. The channel transition probabilities are denoted by $r^{(0)}$:

$$r^{(0)}(x_0, y_0) = \Pr(W = y_0 | X = x_0). \tag{3}$$

At iteration ℓ, the check node with degree d_c finds the check–to–variable node messages L from alphabet $\mathcal{L}$ using $d_c - 1$ incoming messages V from alphabet $\mathcal{V}$, for each outgoing message L, using a mapping function $\Psi_c^{(\ell)}$:

$$\Psi_c^{(\ell)} : \mathcal{V}^{d_c - 1} \to \mathcal{L}. \tag{4}$$

This step is shown diagrammatically in Fig. 1–(a).

Similarly, at iteration ℓ, the variable node with degree d_v finds the variable–to–check messages V using the channel value W and incoming messages L for each outgoing message V using a mapping function $\Phi_v^{(\ell)}$:

$$\Phi_v^{(\ell)} : \mathcal{W} \times \mathcal{L}^{d_v - 1} \to \mathcal{V}. \tag{5}$$

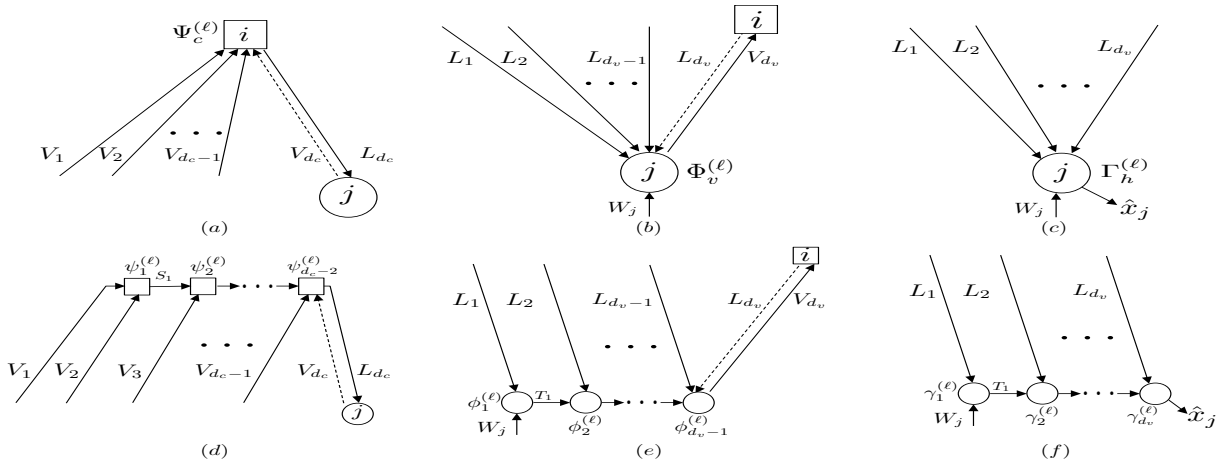

Fig. 1. Decomposition of the nodes. (a) Check node update operation. (b) Variable node update operation. (c) Hard decision operation. (d) Decomposition of the check node update operation $\Psi_c^{(\ell)}$ into the set of lookup tables $\psi_1^{(\ell)}, \ldots, \psi_{d_c-2}^{(\ell)}$. (e) Decomposition of the variable node update operation $\Phi_v^{(\ell)}$ into the set of lookup tables $\phi_1^{(\ell)}, \ldots, \phi_{d_v-1}^{(\ell)}$. (f) Decomposition of the hard decision operation $\Gamma_c^{(\ell)}$ into the set of lookup tables $\gamma_1^{(\ell)}, \ldots, \gamma_{d_v}^{(\ell)}$.

This step is shown diagrammatically in Fig. 1–(b).

At iteration ℓ, the variable node with degree d_v calculates the estimate $\hat{x} \in \{0, 1\}$ using the channel value W and the d_v incoming messages L:

$$\Gamma_h^{(\ell)} : \mathcal{W} \times \mathcal{L}^{d_v} \to \{0, 1\}. \tag{6}$$

This step is shown diagrammatically in Fig. 1–(c).

On iteration ℓ, the probability distribution for V is $r^{(\ell)}(x, y) = \Pr(V = y | X = x)$ with $y \in \mathcal{V}$, and the probability distribution for L is: $l^{(\ell)}(x, y) = \Pr(L = y | X = x)$ with $y \in \mathcal{L}$.

The following method finds the message-passing decoding lookup tables Ψ_c (check node update) and Φ_v (variable node update), as well as the probability distributions r and l using quantization. In particular, for each iteration and each node type, there are four steps: **(a)** given the node input distribution, a cross product distribution is found, **(b)** the quantization algorithm produces a quantizer to K levels, **(c)** the reduced distribution is found, which is used in the next step of the density evolution and **(d)** the decoding lookup tables are found for each quantizer.

Two functions f_c and f_v are of interest when decoding LDPC codes. At the check node:

$$f_c(x_1, \ldots, x_{d_c-1}) = x_1 + \cdots + x_{d_c-1} \pmod 2 \tag{7}$$

and at the variable node:

$$f_v(x_0, \ldots, x_{d_v-1}) = \begin{cases} 0 & \text{if } x_0 = x_1 = \cdots = 0 \\ 1 & \text{if } x_0 = x_1 = \cdots = 1 \\ & \text{otherwise undefined} \end{cases}$$

where x_i are binary values. It is useful to use a single symbol that is a concatenation of the component messages in the cross-product distribution. In the context of the check node, let $y\prime$ denote the concatenation.

$$y' = (y_1, y_2, \ldots, y_{d_c-1}) \tag{8}$$

where $y' \in \mathcal{V}^{d_c-1}$. And in the context of the variable node, let $y\prime$ denote the concatenation:

$$y' = (y_0, y_1, y_2, \ldots, y_{d_v-1}) \tag{9}$$

where $y' \in \mathcal{W} \times \mathcal{L}^{d_v-1}$.

Step **(a)** is to find the cross-product distributions $\tilde{l}^{(\ell)}(x, y')$ and $\tilde{r}^{(\ell)}(x, y')$, given by:

$$\tilde{l}^{(\ell)}(x, y') = \left(\frac{1}{2}\right)^{d_c-2} \sum_{\mathbf{x}: f_c(\mathbf{x})=x} \prod_{i=1}^{d_c-1} r^{(\ell-1)}(x_i, y_i) \tag{10}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_{d_c-1})$, and

$$\tilde{r}^{(\ell)}(x, y') = \sum_{\mathbf{x}: f_v(\mathbf{x})=x} r^{(0)}(x_0, y_0) \prod_{i=1}^{d_v-1} l^{(\ell-1)}(x_i, y_i) \tag{11}$$

where $\mathbf{x} = (x_0, x_1, \ldots, x_{d_v-1})$.

Step **(b)**. The matrix-form quantizers $Q_c^{(\ell)}$ and $Q_v^{(\ell)}$ are produced at each iteration ℓ, given by:

$$Q_c^{(\ell)} = \text{Quant}(\tilde{l}^{(\ell)}, K) \text{ and} \tag{12}$$

$$Q_v^{(\ell)} = \text{Quant}(\tilde{r}^{(\ell)}, K), \tag{13}$$

where $\text{Quant}(\cdot, \cdot)$ is the quantization algorithm described in Section II-B. Step **(c)** is to find the reduced distributions as:

$$l^{(\ell)} = Q_c^{(\ell)} \tilde{l}^{(\ell)} \text{ and} \tag{14}$$

$$r^{(\ell)} = Q_v^{(\ell)} \tilde{r}^{(\ell)}. \tag{15}$$

Step **(d)** is to find the decoding maps, which are given by:

$$\Psi_c^{(\ell)}(y') = y \text{ if } \; Q_c^{(\ell)}(y, y') = 1 \text{ and} \tag{16}$$

$$\Phi_v^{(\ell)}(y') = y \text{ if } \; Q_v^{(\ell)}(y, y') = 1. \tag{17}$$

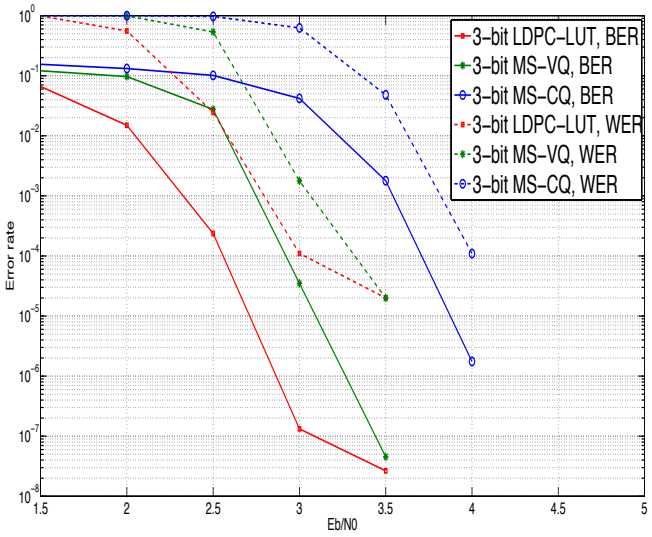

Fig. 2. Bit–error rate (—-) and word–error rate (– –) achieved by the proposed 3–bit LDPC–LUT ($\square$), 3–bit MS–VQ ($*$) and 3–bit MS–CQ ($\circ$).

Note that the quantization algorithm may also be used to design the lookup table $\Gamma_h^{(\ell)}$, which perform the hard decision on x. For each iteration, repeat the variable node steps (a)–(c), using all d_v inputs and quantize to $K = 2$ levels to make hard decision.

The decoding process is the same as the conventional although Ψ_c, Γ_h and Φ_v are decomposed in a set of smaller lookup tables (see decomposition showed in Fig. 1, item d) to f))for reducing complexity, according to [15].

IV. Lookup tables for low complexity decoding

Lookup tables are desirable for hardware implementation since they can be implemented in a small amount of memory as an array indexing operation. Furthermore, the time processing required to read a memory can be significantly less than to compute conventional operations such as multiplication, addition. Since the main goal of this work is to propose a message–passing decoding algorithm with good characteristics and features for implementation, it is assumed that $\Psi_c^{(\ell)}$, $\Phi_v^{(\ell)}$ and $\Gamma_h^{(\ell)}$ are implemented as lookup tables.

Consider one lookup table for each function $\Psi_c^{(\ell)}$, $\Phi_v^{(\ell)}$ and $\Gamma_h^{(\ell)}$, which receives d incoming messages. Note that the size of the lookup table is subjected to the degree of the node. Consider $d_v \geq 3$ and $d_c \geq 4$ for different LDPC codes of interest [16]. The number of memory addresses for the lookup tables $\Psi_c^{(\ell)}$, $\Phi_v^{(\ell)}$ and $\Gamma_h^{(\ell)}$ is $|\mathcal{V}|^{d_c-1}$, $|\mathcal{L}|^{d_v}$ and $|\mathcal{L}|^{d_v+1}$ respectively. Note that the number of memory addresses is exponential in the degree of the node. Since the decoding process is iterative, the memory space needed to save the lookup tables $\Psi_c^{(\ell)}$, $\Phi_v^{(\ell)}$ and $\Gamma_h^{(\ell)}$ can become unpractical, even for a serial LDPC decoding implementation.

For this reason we perform a decomposition of each large lookup table $\Psi_c^{(\ell)}$, $\Phi_v^{(\ell)}$ and $\Gamma_h^{(\ell)}$ into a set of smaller two–input lookup tables, this allows a reduction in the memory requirements for the implementation (the number of memory addresses using this decomposition is linear in the degree of the node). The decomposition also facilitates the identification of patterns in the tables as iterations progress, which can be useful for resource management in hardware.

V. Simulation results

For all simulation results, the maximum number of iterations was fixed at 25. The alphabets used for the lookup tables in the 3–bit LDPC–LUT decoding algorithm are $\mathcal{V} = \mathcal{L} = \{1, 2, \ldots, 8\}$ either using the binary–input AWGN channel or binary symmetric channel, this means that, any decoding message is represented by 3–bit.

A. Simulation results for binary–input AWGN channel

Because we are proposing a low complexity LDPC decoding algorithm, the min–sum decoding algorithm was considered for comparison.

First, two types of quantization are described.

- In *constant quantization* (CQ), the boundaries $a_1^*, \ldots, a_{K-1}^*$ are optimized using σ^*, the noise threshold value found for a given rate given by the density evolution algorithm in Section III. These boundaries are then used, independently of the signal–to–noise ratio, in the simulation.
- In *variable quantization* (VQ), the boundaries $a_1^*, \ldots, a_{K-1}^*$ are optimized for each signal–to–noise ratio in the simulation.

Employing CQ and VQ we compared with two 3–bit min–sum decoding algorithms:

1) 3–bit MS–CQ, this algorithm uses CQ for both the channel quantization and the decoding message quantization.
2) 3–bit MS–VQ, this algorithm uses VQ for both the channel quantization and the decoding message quantization.

For the AWGN channel simulation results we considered the code $(N, K_c) = (2640, 1320)$ with rate $1/2$ [16], and channel quantization levels $K = 4$. The threshold used for CQ is $\sigma^* = 0.83$.

Fig. 2 shows the bit–error rate and word–error rate results for the proposed 3–bit LDPC–LUT decoding algorithm and the two quantized versions of the min–sum decoding algorithm. In this plot, we can see that at 10^{-5} the bit–error rate achieved by the 3–bit LDPC–LUT ($\square$) outperforms to that achieved by the 3–bit MS–CQ ($\circ$) by around 1.2 dB. For the word–error rates achieved at 10^-4, we can observe a gain of 1 dB for the 3–bit LDPC–LUT decoding algorithm ($\square$) over the 3–bit MS–CQ decoding algorithm ($\circ$).

The second comparison in Fig. 2, is between the 3–bit LDPC–LUT ($\square$) and the 3–bit MS–VQ ($*$). We can see the benefit that the VQ represents in the decoding process by observing the gain that exists between the 3–bit MS–VQ ($*$) and the 3–bit MS–CQ ($\circ$).

Even though 3–bit MS–VQ shows the better error rate performance of the two quantized versions of the min–sum

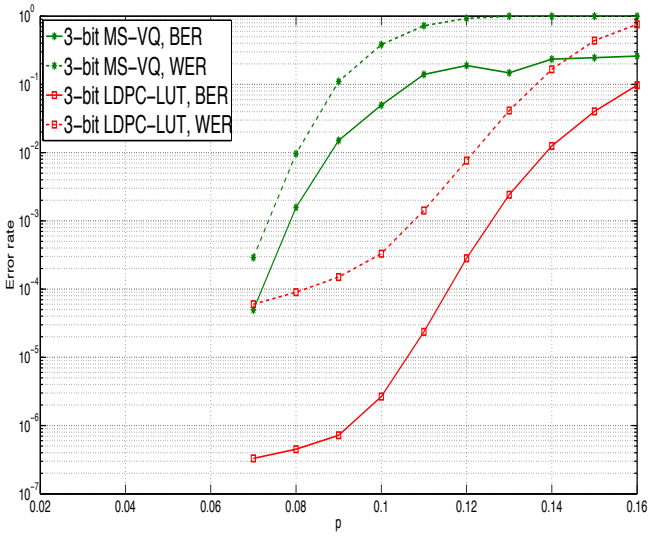

Fig. 3. Bit–error rate (—-) and word–error rate (– –) achieved by the proposed 3–bit LDPC–LUT ($\square$) and the 3–bit min–sum ($*$) over the BSC.

decoding algorithm, this does not outperform the error rate performance achieved by the 3–bit LDPC–LUT decoding algorithm ($\square$).

In Fig. 2 at 10^{-7} the proposed algorithm ($\square$) has a bit–error gain of around 0.3 dB versus the bit–error rate achieved by the 3–bit MS–VQ ($*$). In the same graph for the word–error rate at 10^{-4} also a gain of around 0.3 dB achieved by the 3–bit LDPC–LUT decoding algorithm ($\square$) versus that achieved by the 3–bit MS–VQ ($*$) decoding algorithm is seen.

Although the error rate performance of the 3–bit LDPC–LUT is a few tenths better than that achieved by the 3–bit MS–VQ, it is important to emphasize that the 3–bit MS–VQ is more complex, since it uses distinct quantization boundaries for each SNR while the lookup tables used by the 3–bit LDPC–LUT are the same for any SNR.

B. Simulation results for BSC

For the simulation results on the binary symmetric channel (BSC), we use the code $(N, K_c) = (1000, 250)$ with rate $1/4$ [16]. Since we consider a BSC with error probability p, the channel alphabet is $\mathcal{W} = \{0, 1\}$. In this case the noise threshold is $p = 0.156$ with which the lookup tables were built. In this case, we only consider the min–sum decoding algorithm with variable quantization (3–bit MS–VQ), since this quantization in Section V-A was shown to be the most effective for bit–error rate and word–error rate comparison.

In Fig. 3 the bit–error rate and word–error rate performance achieved by both the 3–bit LDPC–LUT and the 3–bit MS–VQ is shown. In this figure we can note a favorable gap for the proposed 3–bit LDPC–LUT decoding algorithm ($\square$) with respect to the quantized 3–bit MS–VQ decoding algorithm ($*$).

VI. Conclusions

We proposed a 3–bit LDPC–LUT decoding algorithm, which is based on both 3–bit messages and lookup tables that maximize mutual information. The 3–bit LDPC–LUT is locally optimal in the sense of maximizing mutual information for each iteration, although we cannot say anything about global optimality.

The proposed algorithm has low complexity, since the lookup tables are desirable for VLSI implementation due to the fact that these can be implemented as a simple array operation and can also be easily implemented in parallel.

Through simulations, we showed that the 3–bit LDPC LUT has better decoding performance than two quantized versions of min–sum decoding algorithm over either the binary–input AWGN channel or BSC.

References

[1] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. IT–8, pp. 21–28, January 1962.

[2] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 60, pp. 4544–4552, August 2014.

[3] P. A. Chou, "Optimal partitioning for classification and regression trees," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, pp. 340–354, April 1991.

[4] D. Burshtein, V. D. Pietra, D. Kanevsky, and A. Nadas, "Minimum impurity partitions," *The Annals of Statistics*, vol. 20, no. 3, pp. 1637–1646, 1992.

[5] J. Chen and M. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," *in Proceedings of IEEE Globecom*, pp. 1378–1382, November 2002.

[6] L. Ping and W. K. Leung "Decoding low density parity check codes with finite quantization bits," *IEEE Communications Letters*, vol. 4, no. 2, pp. 62–64, February 2000.

[7] J. Kwok–San Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," *in Proceedings of IEEE International Symposium on Information Theory*, Adelaide, Australia, pp. 459463, September 2005.

[8] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern,* Cambridge University Press, USA, pp. 226–233, 2009.

[9] J. Zhao, F. Zarkeshvari and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549–554, April 2005.

[10] S. Kumar Planjery, D. Declercq, L. Danjean and B. Vasic, "Finite alphabet iterative decoders-Part I: decoding beyond belief propagation on the binary symmetric channel," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4033–4045, October 2013.

[11] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, "An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes," *IEEE Communications Letters*, vol. 9, no. 9, pp. 814–816, September 2005.

[12] Telex Magloire N. N., M. Bossert, A. Fahrner and F. Takawira, "Two bit–flipping decoding algorithms for low–density parity–check codes," *IEEE Transactions on Communications*, vol. 57, no. 3, pp. 591–596, March 2009.

[13] T. J. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, February 2001.

[14] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Density evolution for asymmetric memoryless channels," *IEEE Transactions on Information Theory*, vol. 51, pp. 4216–4236, December 2005.

[15] B. M. Kurkoski and H. Yagi, "Noise thresholds for discrete LDPC decoding mappings," *in Proceedings IEEE Global Telecommunications Conference*, New Orleans, USA, pp. 1–5, November 2008.

[16] D.J.C. MacKay, *Encyclopedia of sparse graph codes,* http://www.inference.phy.cam.ac.uk/mackay/codes/data.html