Density Evolution for GF(q) LDPC Codes Via Simplified Message-passing Sets

Brian M. Kurkoski, Kazuhiko Yamaguchi, Kingo Kobayashi
Dept. of Information and Communications Engineering
University of Electro-Communications
Tokyo, 182-8585, Japan
Email: {kurkoski,yama,kingo}@ice.uec.ac.jp

***Abstract*— A message-passing decoder for $GF(q)$ low-density parity-check codes is defined, which uses discrete messages from a subset of all possible binary vectors of length q. The proposed algorithm is a generalization to $GF(q)$ of Richardson and Urbanke's decoding "Algorithm E" for binary codes. Density evolution requires a mapping between the probability distribution spaces for the channel, variable and check messages, and under the proposed algorithm, exact density evolution is possible. Symmetries in the message densities permit reduction in the size of the probability distribution space. Noise thresholds are obtained for LDPC codes on discrete memoryless channels, and as with Algorithm E, are remarkably close to noise thresholds under more complex belief propagation decoding.**

I. Introduction

Although binary LDPC codes can approach the capacity of a variety of channels for asymptotically long block lengths, for moderate block lengths, $GF(q)$ LDPC codes decoded using belief propagation have better performance than their binary counterparts on AWGN and BSC channels [2]. Furthermore, codes constructed over $GF(q)$ are appealing for use on bursty channels such as partial-response channels. However, belief propagation decoding of such codes is complex, although efforts have been made to reduce the decoding complexity [3] [5] [4].

Density evolution has been successfully used for the design and analysis of binary LDPC codes and belief-propagation decoders because scalar log-ratios are easily quantized [1]. However, for belief propagation decoding of $GF(q)$ LDPC codes, the message is a vector with dimension $q-1$, and the complexity of density evolution is quite high, even for small field sizes $q \geq 3$. Davey and MacKay used a Monte Carlo approach to estimate the message densities, and showed good codes with average variable node degree of between 2 and 3 [6]. Monte Carlo approaches, however, are computationally intensive, and are not effective for designing more powerful codes.

To address this quantization problem, Bennatan and Burshtein approximated log-ratio messages with a multi-dimensional Gaussian distribution, and used EXIT chart techniques to design irregular $GF(q)$ LDPC codes. However, as with binary LDPC codes, the check-to-variable message is poorly modeled by a Gaussian distribution [7].

In this paper, a framework for message-passing decoding of $GF(q)$ LDPC codes is given, and then a specific decoder is analyzed. Here, belief-propagation decoders, where the messages are probabilities and decoding rules are probabilistically derived, are viewed as a case of message-passing decoders. In Section II, some notation and preliminaries are given. In Section III, generalized message-passing decoding is given for $GF(q)$ LDPC codes, and a straightforward description of density evolution using enumerator functions is given. In Section IV, the symmetries of the enumerator function are found and used to reduce the complexity of density evolution.

Richardson and Urbanke proposed a simple message-passing algorithm for decoding binary low-density parity check (LDPC) codes when transmitted over the binary symmetric channel, which used just three level messages corresponding to a 0, 1 and an erasure. This Algorithm E had modest performance loss relative to the more complicated belief propagation [1].

In Section V, a generalization of Algorithm E to the decoding of $GF(q)$ LDPC codes is given, and it is analyzed using the proposed methods. Section VI gives numerical results. Section VII is the conclusion.

II. Preliminaries

In this section, we establish notation and give some preliminaries that will be used throughout the paper.

The random variable $\mathbf{V}$ is a $GF(q)$-*indexed random vector* with

$$\mathbf{V} = \{V_0, V_1, \cdots V_{q-1}\} \tag{1}$$

if the q scalar components V_i are indexed by the q distinct field elements of $GF(q)$. A script letter $\mathcal{V}$ denotes the alphabet, which is a set of points from the q-ary space of real numbers, from which $\mathbf{V}$ takes on values. Thus $\mathcal{V}$= $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_V}\}$, where $\mathbf{v}_i \in \mathbb{R}^q$. The cardinality of the alphabet is denoted by $N_V = |\mathcal{V}|$. Without loss of generality, $\mathcal{V}$ will be indexed by the integers $\{1, 2, \ldots, N_V\}$.

The distribution of $\mathbf{V}$ is given by a lowercase Greek letter, $\nu_i = Pr(\mathbf{V} = i)$, with $\sum_{i=1}^{N_V} \nu_i = 1$.

Bennatan and Burshtein [7] introduced the following notation for probability vectors and log-ratio vectors, and here is extended to more general alphabets. The $+g$ operator on a $GF(q)$-indexed random vector is:

$$\mathbf{V}^{+g} = (V_g, V_{1+g}, \ldots, V_{(q-1)+g}), \tag{2}$$

where addition in the subscript indices is performed over $GF(q)$.

Similarly, the $\times g$ operator defined as:

$$\mathbf{V}^{\times g} \quad = \quad (V_0, V_g, V_{2g}, \ldots, V_{(q-1)g}), \tag{3}$$

where multiplication in the subscript indices is performed over $GF(q)$.

For any $\mathbf{v} \in \mathcal{V}$, define $\mathbf{v}^*$ as the set:

$$\mathbf{v}^* \quad = \quad \{\mathbf{v}, \mathbf{v}^{\times 2}, \ldots, \mathbf{v}^{\times (q-1)}\}. \tag{4}$$

It is assumed that $\mathbf{v}^* \subseteq \mathcal{V}$. Note that for any $\mathbf{u} \in \mathbf{v}^*$ that $\mathbf{u}^* = \mathbf{v}^*$.

III. LDPC Codes

A. LDPC Codes Over $GF(q)$

A $GF(q)$ LDPC code is the set of code vectors $\mathbf{x} = (x_1, \ldots, x_n)$ of length n with elements $\mathcal{A} = \{0, \alpha^0, \alpha^1, \cdots, \alpha^{q-2}\}$ with primitive element α from $GF(q)$ such that:

$$\mathbf{x}H^t \quad = \quad 0, \tag{5}$$

where H is a sparse parity check matrix with constant column weight d_v and row weight d_c. The rate of the code is $R = 1 - d_\mathsf{v}/d_\mathsf{c}$. The non-zero entries in H are drawn randomly and with equal probability from $\mathcal{A} \setminus \{0\}$.

The arbitrary code symbol x is transmitted over a symmetric, q-ary input discrete memoryless channel, with output alphabet $\mathcal{O}$, and the received symbol is $z \in \mathcal{O}$.

B. Message-Passing Decoding Algorithm

Message-passing decoding was described in general terms by Richardson and Urbanke [1], but here we give a similar algorithm which explicitly accounts for $GF(q)$ LDPC codes.

Message-passing decoding is performed using a bipartite graph where variable nodes corresponding to columns of H and check nodes corresponding to rows of H are connected by an edge where there is a non-zero entry in the parity-check matrix. The variable-to-check messages $\mathbf{R}$ are vectors $\mathbf{R} = (R_0, R_1, \cdots, R_{q-1})$, where $R_0, R_1, \ldots, R_{q-1}$ are scalar messages for $0, \alpha^0, \ldots, \alpha^{q-2}$, respectively. Similarly, the check-to-variable messages $\mathbf{L}$ are vectors $\mathbf{L} = (L_0, L_1, \cdots, L_{q-1})$, where $L_0, L_1, \ldots, L_{q-1}$ are scalar messages for $0, \alpha^0, \ldots, \alpha^{q-2}$, respectively. The messages $\mathbf{R}$ (similarly $\mathbf{L}$) are drawn from a *message alphabet* $\mathcal{R}$ ($\mathcal{L}$).

A node's output message on edge e is generated using incoming messages on all edges except for e. Only the computation for a single edge's output is shown, where the generalization to the other edges' output is clear. At iteration ℓ, the variable node with degree d_v computes outgoing message $\mathbf{R}_1$ using the channel value z and incoming messages $\mathbf{L}_2, \ldots, \mathbf{L}_{d_\mathsf{v}}$, by a mapping function $\Phi_\mathsf{v}^{(\ell)}$:

$$\Phi_\mathsf{v}^{(\ell)} : \mathcal{O} \times \mathcal{L}^{d_\mathsf{v}-1} \to \mathcal{R},$$

and similarly for all $R_2, \ldots, R_{d_\mathsf{v}}$.

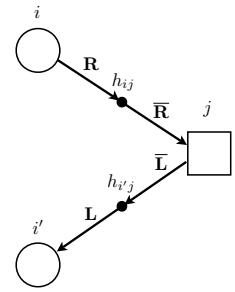

Fig. 1. Message-passing decoding.

Distinct from binary LDPC codes, the message passed from variable node i to check node j is scaled by the non-zero parity check matrix coefficient h_{ij}:

$$\overline{\mathbf{R}} \quad = \quad \mathbf{R}^{\times h_{ij}^{-1}}. \tag{6}$$

The check node computes outgoing message $\overline{\mathbf{L}}_1$ using the incoming messages $\overline{\mathbf{R}}_2, \ldots, \overline{\mathbf{R}}_{d_\mathsf{c}}$, by a mapping function $\Phi_\mathsf{c}^{(\ell)}$:

$$\Phi_\mathsf{c}^{(\ell)} : \mathcal{R}^{d_\mathsf{c}-1} \to \mathcal{L},$$

and similarly for all $\overline{\mathbf{L}}_2, \ldots, \overline{\mathbf{L}}_{d_\mathsf{c}}$.

This map can be decomposed into partial sums. At the check node, $\overline{x}_1 + \cdots + \overline{x}_{d_\mathsf{c}} = 0$, where $\overline{x}_i = h_i x_i$. Let $s_i = \overline{x}_1 + \cdots + \overline{x}_i$, for $1 \leq i \leq d_\mathsf{c}$. Associated with each partial sum is a message $\mathbf{S}_i$, drawn from the message alphabet $\mathcal{S}$. This message is analogous to the forward state metric in the BCJR algorithm. The element $\mathbf{s}_0 \in \mathcal{S}$ is the message corresponding to $\overline{x} = 0$. A mapping function Φ_S specifies the computation of the partial sum:

$$\Phi_\mathsf{S} \quad : \quad \mathcal{R} \times \mathcal{S} \longrightarrow \mathcal{S}.$$

Initially, $\mathbf{S}_1 = \Phi_\mathsf{S}(\overline{\mathbf{R}}_1, \mathbf{s}_0)$. The mapping is applied recursively $d_\mathsf{c} - 2$ more times to obtain $\mathbf{S}_2, \ldots, \mathbf{S}_{d_\mathsf{c}-1}$. Then, the check output $\overline{\mathbf{L}}_{d_\mathsf{c}}$ is generated by:

$$\Phi_\mathsf{G} \quad : \quad \mathcal{S} \longrightarrow \mathcal{L}.$$

The message passed from check node j to variable node i' is scaled by the non-zero parity check matrix coefficient $-h_{i'j}$: scaled message is:

$$\mathbf{L} \quad = \quad \overline{\mathbf{L}}^{\times -h_{i'j}}. \tag{7}$$

C. Discrete Density Evolution

Under belief-propagation decoding, messages $\mathbf{L}, \mathbf{R}$ are of the general form $Pr(x = \alpha^{i-1})$, in which case $\mathcal{L} = \mathcal{R} = [0,1]^q$, or may be in a log-domain representation such as $\log Pr(x = \alpha^{i-1})/Pr(x = 0)$, in which case $\mathcal{L} = \mathcal{R} = \mathbb{R}^{q-1}$. Bennatan and Burshtein gave a method for performing density evolution under the assumption that belief-propagation is used. Here, we assume only the message-passing decoder of the previous section.

Density evolution tracks the probability distribution of the messages $\mathbf{L}$ and $\mathbf{R}$ as LDPC decoding iterations progress, and

searches for the greatest channel noise parameter under which convergence is obtained.

We consider message passing not on continuous alphabets, but using messages drawn from discrete vector sets. The variable-to-check message alphabet is $\mathcal{R}$ with cardinality N_R. Similarly, the check-to-variable message alphabet is $\mathcal{L}$ with $N_L = |\mathcal{L}|$.

The variable-to-check messages $\mathbf{R}$, $\overline{\mathbf{R}}$ are random variables, and under density evolution the distribution of the latter message is tracked, $\rho_i = Pr(\overline{\mathbf{R}} = r_i)$, where $\sum_{i=1}^{N_R} \rho_i = 1$. Similarly, the check-to-variable messages $\mathbf{L}$, $\overline{\mathbf{L}}$ are random variables, and the former is tracked, $\lambda_i = Pr(\mathbf{L} = l_i)$, where $\sum_{i=1}^{N_L} \lambda_i = 1$.

Let Π_A be the space of probability distributions for the set A. Density evolution requires a mapping function at the variable node:

$$^{\star}\Phi_{\mathsf{v}}^{(\ell)} \quad : \quad \Pi_{\mathcal{O}} \times \Pi_{\mathcal{L}}^{d_{\mathsf{v}}-1} \longrightarrow \Pi_{\mathcal{R}}.$$

At the check node the analogous mapping is:

$$^{\star}\Phi_{\mathsf{c}}^{(\ell)} \quad : \quad \Pi_{\mathcal{R}}^{d_{\mathsf{c}}-1} \longrightarrow \Pi_{\mathcal{L}},$$

which here is expressed as $d_{\mathsf{c}} - 1$ steps of the recursion:

$$^{\star}\Phi_{\mathsf{S}}^{(\ell)} \quad : \quad \Pi_{\mathcal{R}} \times \Pi_{\mathcal{S}} \longrightarrow \Pi_{\mathcal{S}},$$

followed by

$$^{\star}\Phi_{\mathsf{G}}^{(\ell)} \quad : \quad \Pi_{\mathcal{S}} \longrightarrow \Pi_{\mathcal{L}}.$$

We consider the performance assuming the all-zeros codeword, and generalize to the performance of all codewords for an ensemble of linear codes with the specified degree distributions. Accordingly, the probability of receiving symbol z, given that the zero symbol was transmitted, is ν_z, and assume that the channel satisfies necessary symmetry conditions.

1) Variable Node Probability Map: Now we introduce a method to compute the map $^{\star}\Phi_{\mathsf{v}}$. The *variable node message indicator function* $N_{\mathsf{v}}^{(t)}(z, w, n_1, n_2, \ldots, n_{N_L})$ is a one if channel message $z \in \mathcal{O}$ with positive integer weight w and n_i incoming messages i, $i \in \mathcal{L}$ generate output message $t \in \mathcal{R}$ at the variable node output; otherwise it is a zero. Note that $N_{\mathsf{v}}^{(t)}(\cdot)$ is taken without regard to order of the inputs, and is only defined for $\sum_{i=1}^{N_L} n_i = d_{\mathsf{v}} - 1$. The weight w is a positive integer which depends upon the iteration number ℓ.

After scaling by h_{ij}, the equivalent enumerator function is denoted $\overline{N}$. All messages from the set $\mathbf{j}^*$ will have the same distribution, namely,

$$\overline{N}^{(j)}(z, w, n_1, \ldots, n_{N_L}) \quad = \quad \frac{1}{|\mathbf{j}^*|} \sum_{e \in \mathbf{j}^*} N^{(e)}(z, w, n_1, \ldots, n_{N_L})$$

For any $i \in \mathbf{j}^*$, $\mathbf{i}^* = \mathbf{j}^*$, thus $\overline{N}^{(j)}(z, w, n_1, \ldots, n_{N_L}) = \overline{N}^{(i)}(z, w, n_1, \ldots, n_{N_L})$.

If the incoming distribution is ρ_i, and the outgoing distribution is λ_j, then we can express the variable node probability map $^{\star}\Phi_{\mathsf{v}}$ as:

$$\rho_t \quad = \quad \sum_{z \in \mathcal{O}} \sum_{n_i : \sum n_i = d_{\mathsf{v}} - 1} \binom{d_{\mathsf{v}} - 1}{n_1, \ldots, n_{N_L}} \cdot \overline{N}_{\mathsf{v}}^{(t)}(z, w, n_1, \ldots, n_{N_L}) \cdot \nu_z \lambda_1^{n_1} \cdots \lambda_{N_L}^{n_{N_L}}. \tag{8}$$

2) Check Node: The above procedure for the variable node can be repeated to find the density map at the check node $^{\star}\Phi_{\mathsf{c}}$. However, to reduce the computational complexity, density evolution on the partial sum messages will be performed.

If at some fixed index t, the partial sum message for s_t is $\mathbf{S} = i \in \mathcal{S}$, the input message $\mathbf{R}_t$ is $k \in \mathcal{R}$ and the partial sum message for s_{t+1} is $\mathbf{S}' = j \in \mathcal{S}$, then $N_{\mathsf{S}}^{(j)}(i, k)$ is a one; in all other cases it is zero. The distribution on $\mathbf{S}$, $\mathbf{S}'$ is $\sigma_j = Pr(\mathbf{S} = j)$, $\sigma_j = Pr(\mathbf{S}' = j)$, respectively. We can express $^{\star}\Phi_{\mathsf{S}}$ as:

$$\sigma_j' \quad = \quad \sum_{i=1}^{N_S} \sum_{k=1}^{N_R} N_{\mathsf{S}}^{(j)}(i, k) \sigma_i \rho_k. \tag{9}$$

If the check node output $\overline{\mathbf{L}}_{d_{\mathsf{c}}}$ is k and the partial sum message is i, then $\overline{N}_{\mathsf{G}}^{(k)}(i)$ is a one, otherwise it is zero. The check node message indicator, after scaling by h is:

$$N_{\mathsf{G}}^{(k)}(i) \quad = \quad \frac{1}{|\mathbf{k}^*|} \sum_{e \in \mathbf{k}^I} \overline{N}_{\mathsf{G}}^{(e)}(i). \tag{10}$$

Then, the check node probability map is:

$$\lambda_k \quad = \quad \sum_{i=1}^{N_S} N_{\mathsf{G}}^{(k)}(i) \sigma_i \tag{11}$$

Thus, check node probability map $^{\star}\Phi_{\mathsf{v}}$ is found by (9) initialized with $\sigma_{\mathbf{s}_0} = 1$ and $\sigma_i = 0$ for $i \neq \mathbf{s}_0$. Then, (9) is applied recursively $d_{\mathsf{c}} - 2$ times. Finally (11) is applied to obtain the output distribution.

Convergence is corresponds to the condition $I(x; \overline{\mathbf{R}}) = 1$.

IV. Reduction to Classes

Density evolution described in the previous section is computationally and memory intensive, and we can exploit message symmetries to reduce the complexity. The probability spaces are reduced by grouping messages with equal probabilities.

A. Definition of Partitions

The set $\mathcal{L}$ is partitioned into S^{L} non-overlapping subsets $\mathcal{L}_1, \ldots, \mathcal{L}_{S^{\mathsf{L}}}$ which cover $\mathcal{L}$. Let $\mathcal{L}_i(1)$ denote the first element of $\mathcal{L}_i$. Such a partition is denoted $\mathbb{P}_{\mathcal{L}}$. Let S_i^{L} be the size of the subset that element i belongs to, that is $S_i^{\mathsf{L}} = |\mathcal{L}_j|$ if $i \in \mathcal{L}_j$. Define $\widetilde{\lambda}_i = \sum_{j \in \mathcal{L}_i} \lambda_j$.

This notation is similarly used for the other messages $\mathbf{R}$, $\mathbf{S}$ and z, which are partitioned as $\mathbb{P}_{\mathcal{R}}$, $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{O}}$, respectively.

$$\overline{X}_{\mathsf{v}}^{(t)}(a,w,m_1,\cdots,m_{\widetilde{N}_L}) = \sum_{z\in\mathcal{O}_a}\sum_{n_i:c(m_1)}\cdots\sum_{n_i:c(m_{S^{\mathsf{L}}})}\binom{d_{\mathsf{v}}-1}{n_1,\cdots,n_{N_L}}\frac{\overline{N}_{\mathsf{v}}^{(t)}(z,w,n_1,\ldots,n_{N_L})}{|\mathcal{O}_a|\cdot(S_1^{\mathsf{L}})^{m_1}\cdots(S_{S^{\mathsf{L}}}^{\mathsf{L}})^{m_{S^{\mathsf{L}}}}} \quad (12)$$

B. Classes Lemma

Let $\mathbf{R}\in\mathcal{R}$ be computed from $\mathbf{L}_i, z$ by Φ_{v}:

$$\mathbf{R} = \Phi_{\mathsf{v}}(z,\mathbf{L}_1,\ldots,\mathbf{L}_{d_{\mathsf{v}}-1}). \quad (13)$$

In any realization of $\mathbf{L}_1,\ldots\mathbf{L}_{d_{\mathsf{v}}-1}$, let m_i denote the number of occurrences of some member of the partition $\mathcal{L}_i$, so that:

$$\sum_{i=1}^{S^{\mathsf{L}}} m_i = d_{\mathsf{v}}-1. \quad (14)$$

Let $\overline{X}_{\mathsf{v}}^{(j)}(a,w,m_1,\cdots,m_N)$ denote the number of ways we can have m_i members from class i, with z a member of $\mathcal{O}_a$, to produce $j\in\mathcal{R}$. Then $\overline{X}_{\mathsf{v}}^{(j)}$ can be computed from $\overline{N}_{\mathsf{v}}^{(j)}$ as shown in (12). In (12), the summation range $c(m_k)$ is $\{n_j|j\in\mathcal{S}_k,\sum_{j\in\mathcal{S}_k}n_j=m_k\}$.

Lemma 1. For the conditions stated above, two distinct messages i and j have equal probabilities:

$$\rho_i = \rho_j \quad (15)$$

if and only if

$$\overline{X}_{\mathsf{v}}^{(i)}(a,w,m_1,\cdots,m_{S^{\mathsf{L}}}) = \overline{X}_{\mathsf{v}}^{(j)}(a,w,m_1,\cdots,m_{S^{\mathsf{L}}}), \quad (16)$$

for all m_i such that $\sum m_i = d_{\mathsf{v}}-1$ and all $a\in\{1,\ldots,S^O\}$.

We have a similar lemma for the partial sum messages $\mathbf{S}$. If the partial sum $\mathbf{S}$ is from class $\mathcal{S}_s$, the incoming message is from class $\mathcal{R}_t$, then $X_{\mathsf{S}}^{(k)}(s,t)$ is the number of ways to produce the a partial sum message $\mathbf{S}'=u\in\mathcal{S}$. The function X_{S} can be computed from N_{S} as follows:

$$X_{\mathsf{S}}^{(u)}(s,t) = \frac{1}{|\mathcal{S}_s|\cdot|\mathcal{R}_t|}\sum_{i\in\mathcal{S}_s}\sum_{k\in\mathcal{R}_t}N_{\mathsf{S}}^{(u)}(i,k). \quad (17)$$

Lemma 2. Two distinct messages $u,v\in\mathcal{S}$ have equal probabilities:

$$\sigma_u = \sigma_v \quad (18)$$

if and only if

$$X_{\mathsf{S}}^{(u)}(s,t) = X_{\mathsf{S}}^{(v)}(s,t), \quad (19)$$

for all $s,t\in\mathcal{S}$.

If the final partial sum message $\mathbf{S}$ is from class k, then $\overline{X}_{\mathsf{G}}^{(u)}(k)$ is the number of output messages $\mathbf{L}=u$. Then X_{G} can be computed as:

$$X_{\mathsf{G}}^{(u)}(k) = \frac{1}{|\mathcal{L}_k|}\sum_{i\in\mathcal{L}_k}N_{\mathsf{G}}^{(u)}(i). \quad (20)$$

A subset $\mathcal{L}_k$ is a *class* if for all $i,j\in\mathcal{L}_i$, $\lambda_i=\lambda_j$. From here, we are interested in partitions where all subsets are are classes. For classes:

$$\widetilde{\lambda}_i = S_i^{\mathsf{L}}\lambda_{\mathcal{L}_i(0)}. \quad (21)$$

This notation of classes is similarly used for the other messages $\mathbf{R}$, $\mathbf{S}$ and z.

C. Construction of Classes and Class-based Probability Maps

Now, the output messages are placed into classes, and simplified density evolution is presented.

Variable Node The variable node class weight function $M_{\mathsf{v}}^{(u)}(a,\,w,m_1,m_1,\cdots,m_{S^{\mathsf{L}}})$ is the number of ways a channel message from class $\mathcal{O}_a$ (with weight w on iteration ℓ) and m_i messages $\mathbf{L}$ from class i, generate an output message $\mathbf{R}$ from class $u\in\{1,2,\ldots,S^{\mathsf{R}}\}$. Note that $M_{\mathsf{v}}^{(u)}(\cdot)$ is non-zero only for $\sum_{i=1}^{s}m_i=d_{\mathsf{v}}-1$. Then $M_{\mathsf{v}}^{(u)}(\cdot)$ is found as:

$$M_{\mathsf{v}}^{(u)}(a,w,m_1,\ldots,m_{S^{\mathsf{L}}}) = \sum_{j\in\mathcal{R}_u}X_{\mathsf{v}}^{(j)}(a,w,m_1,\ldots,m_{S^{\mathsf{L}}}). \quad (22)$$

Density evolution at the variable node using the class distributions $\widetilde{\lambda}$, $\widetilde{\nu}$ and $\widetilde{\rho}$ is simplified as:

$$\widetilde{\rho}_u = \sum_{a=1}^{S^{\mathsf{O}}}\sum_{m_i}M_{\mathsf{v}}^{(u)}(a,w,m_1,\ldots,m_{S^{\mathsf{L}}})\cdot\widetilde{\nu}_a\widetilde{\lambda}_1^{m_1}\cdot\cdot\widetilde{\lambda}_{S^{\mathsf{L}}}^{m_{S^{\mathsf{L}}}},$$

where the inner sum taken over all m_i such that $\sum_{i=1}^{s}m_i = d_{\mathsf{v}}-1$.

Check Node The set $\mathcal{S}$ is partitioned into classes of size S^{S}. The partition distributions of $\mathbf{S}$ and $\mathbf{S}'$ are $\widetilde{\sigma}_i,\widetilde{\sigma}'_j$, respectively. If the partial-sum message $\mathbf{S}$ is from class i, and the incoming message $\mathbf{L}$ is from class k, then $M_{\mathsf{S}}^{(j)}(i,k)$ is the number of partial sum messages $\mathbf{S}'$ which belong to class j. Then $M_{\mathsf{S}}^{(\cdot)}(\cdot)$ is found as:

$$M_{\mathsf{S}}^{(k)}(s,t) = \sum_{u\in\mathcal{S}_k}X_{\mathsf{S}}^{(u)}(s,t). \quad (23)$$

The probability map ${}^{\star}\Phi_{\mathsf{S}}$ is simplified as:

$$\widetilde{\sigma}'_j = \sum_{i=1}^{S^{\mathsf{S}}}\sum_{k=1}^{S^{\mathsf{L}}}M_{\mathsf{S}}^{(j)}(i,k)\widetilde{\sigma}_i\widetilde{\lambda}_k. \quad (24)$$

If the final partial-sum message $\mathbf{S}_{d_c-1}$ is from class i then $M_{\mathsf{G}}^{(j)}(i)$ is the number of check output messages $\mathbf{L}$ which belong to class j. Then $M_{\mathsf{G}}^{(\cdot)}(\cdot)$ is found as:

$$M_{\mathsf{G}}^{(j)}(i) = \sum_{u\in\mathcal{S}_j}X_{\mathsf{G}}^{(u)}(i). \quad (25)$$

Then, the simplified probability map ${}^*\Phi_{\mathsf{G}}$ is:

$$\rho_k \quad = \quad \sum_{i=1}^{S^{\mathsf{L}}} M_{\mathsf{G}}^{(k)}(i)\sigma_i. \tag{26}$$

V. Proposed Decoding Algorithm

A. Message Set

Algorithm E is a decoding algorithm for binary LDPC codes on the binary symmetric channel. The decoding algorithm uses three-level messages, a one, a zero and an erasure. The algorithm can be explained by some simple heuristics. In this section, this idea is generalized to codes with field order q, while maintaining a simple heuristic, using the following message alphabet. Let $\mathcal{V}^c$ be the set of binary q-vectors with c or fewer ones. The message alphabet is $\mathcal{L} = \mathcal{R} = \mathcal{V}^c$ with $c \geq 1$, and cardinality:

$$|\mathcal{L}| = |\mathcal{R}| \quad = \quad \binom{q}{0} + \binom{q}{1} + \cdots + \binom{q}{c}.$$

An arbitrary element of $\mathcal{L}$ is $(l_0, l_1, \ldots, l_{q-1})$. A value of $l_i = 1$ indicates that the corresponding field element is a candidate symbol, and $l_i = 0$ indicates it is not a candidate. The all-zeros vector is an "erasure" message, and the q vectors with weight one correspond to a "correct" message for each element of $\mathcal{A}$. In the case of $q = 2$, and $c = 1$, this reduces to the message-passing set $\{(0,0),(1,0),(0,1)\}$ used by Algorithm E.

Code symbols are transmitted over a discrete-memoryless channel. Under density evolution, we'll consider the performance of the all-zeros codeword and under linearity and appropriate restrictions on the channel, will generalize this to the performance of any codeword.

The channel output alphabet $\mathcal{O}$ is a subset of $\mathcal{L}$, so that z can be represented as a binary vector $\mathbf{y} = (y_0, y_1, \ldots, y_{q-1})$, where $y_i \in \{0,1\}$. It is assumed that $\mathcal{O} \subseteq \mathcal{L}$. If each element of $\mathcal{O}$ is mapped to distinct element of $\mathcal{L}$, then there is no loss of information in quantizing the channel.

B. Variable Node Map

The variable node receives messages which are "votes" for the q symbols. The output message is for the symbol(s) receiving the maximum number of votes.

Specifically, the incoming messages $\mathbf{L}_1, \ldots, \mathbf{L}_{d_{\mathsf{v}}-1}$ are of the form $\mathbf{L}_j = (L_{0,j}, L_{1,j}, \ldots, L_{q-1,j})$. Let $\mathbf{S} = (S_1, S_2, \ldots, S_q)$ be a vector. The channel observation z receives an iteration-dependent weight w. The variable-node map Φ_{v} is:

$$\Phi_{\mathsf{v}}(z, w, \mathbf{L}_1, \ldots, \mathbf{L}_{d_{\mathsf{v}}-1}) \quad = \quad \mathbf{R}_{d_{\mathsf{v}}}.$$

To explicitly find Φ_{v}, first compute $S_i, i = 1, \ldots, q$ as:

$$S_i \quad = \quad w^{(\ell)} y_i + \sum_{j=2}^{d_{\mathsf{v}}} L_{i,j}. \tag{27}$$

Let $b = \max_i S_i$, and let $I_j(\mathbf{S})$ be an indicator matrix or vector with the same number of elements as $\mathbf{S}$, which has a one in each position that $\mathbf{S}$ is equal to j, and zeros elsewhere.

The output message $\mathbf{R}_{d_{\mathsf{v}}}$ is:

$$\mathbf{R}_{d_{\mathsf{v}}} \quad = \quad \begin{cases} I_b(\mathbf{S}) & \text{if } I_b(\mathbf{S}) \in \mathcal{R} \\ (0,0,\ldots,0) & \text{otherwise} \end{cases}, \tag{28}$$

and similarly for $\mathbf{R}_2, \ldots, \mathbf{R}_{d_{\mathsf{v}}}$. Note that if the winning vector $I_j(\mathbf{S})$ is not in the message set $\mathcal{R}$, then the output is an erasure.

C. Check Node Map

At an arbitrary check node the parity check equation $\sum_{i=1}^{d_{\mathsf{c}}} h_i x_i = 0$ is satisfied, where $h_i \in \mathcal{A} \setminus \{0\}$ are the non-zero parity-check matrix coefficients. Without loss of generality, we consider how to compute the output messages $\mathbf{L}_{d_{\mathsf{c}}}$ using input messages $\mathbf{R}_1, \ldots, \mathbf{R}_{d_{\mathsf{c}}-1}$.

In a word, there will be a one for position i in $\mathbf{L}_{d_{\mathsf{c}}}$ if the $GF(q)$ symbol for that position plus the sum of any combination of symbols which have a one in $\mathbf{R}_1, \ldots, \mathbf{R}_{d_{\mathsf{c}}-1}$, is equal to 0.

More precisely, we can perform decoding on a trellis using a BCJR-like algorithm. We let $\mathbf{A}_i$ be the forward partial sum, and $\mathbf{B}_i$ be the backward partial sum. Initialize $\mathbf{A}_1 = \mathbf{B}_{d_{\mathsf{c}}+1} = (1, 0, \ldots, 0)$. Let Γ_i be the q-by-q matrix of state transition metrics. State metric recursions can be expressed via matrix multiplication (see for example [9]), where the forward recursion is:

$$\mathbf{A}_{i+1} \quad = \quad \mathbf{A}_i \Gamma_i, \tag{29}$$

and the backward recursion is:

$$\mathbf{B}_i \quad = \quad \mathbf{B}_{i-1} \Gamma_i^t, \tag{30}$$

where superscript t denotes matrix transpose.

For the fully-connected trellis of the $GF(q)$ single parity check code, we have:

$$\Gamma_i \quad = \quad \begin{bmatrix} \mathbf{R}_i^{+0} \\ \mathbf{R}_i^{+\alpha^0} \\ \vdots \\ \mathbf{R}_i^{+\alpha^{q-2}} \end{bmatrix}. \tag{31}$$

Because 0 and 1 correspond to binary "candidate" and "not candidate" messages, in the above matrix multiplications, we define addition as the logical OR operation ($0+0=0, 1+0=0+1=1+1=1$), and multiplication as the logical AND operation ($0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0$, $1 \cdot 1 = 1$).

Define T to be the q-by-q transition matrix for the trellis, where $T_{n,m}$ is the label between state n at time i to state m at time $i+1$. Each state n, m may be assigned to an element of $\mathcal{A}$. So if n, m, $e \in \mathcal{A}$ are the partial sum at time i, the partial sum at time $i+1$ and the edge transition respectively with $n + e = m$, then $T_{n,m}$ is $m - n$. Let $I_k(T)$ be an indicator matrix of the same size as T, which has a 1 in positions T is equal to k, and a 0 otherwise. Then we express the check node output as

$$L_{k,i} \quad = \quad \mathbf{A}_i \cdot (I_k(T))^t \cdot \mathbf{B}_{i+1}^t,$$

for $k \in \mathcal{A}$ and $i = 1, 2, \ldots, d_{\mathsf{c}}$.

TABLE I

THRESHOLDS ϵ^* FOR PROPOSED ALGORITHM.

$R = 1/2, d_\mathsf{v} = 3, d_\mathsf{c} = 6$

q	ϵ^* $c=1$	$c=2$	$c=3$	$c=4$	$c=5$	BP	ϵ^{opt}	$(\epsilon^{\mathsf{opt}} - \epsilon^*)/\epsilon^{\mathsf{opt}}$
2	0.039	—	—	—	—	0.082	0.110	64.5%
4	0.072	0.111	0.110	—	—	0.149	0.189	41.3%
8	0.073	0.137	0.143	0.141	0.140	0.194	0.247	42.2%
16	0.075	0.148	0.161			0.231	0.290	74.1%

$R = 2/5, d_\mathsf{v} = 3, d_\mathsf{c} = 5$

q	ϵ^* c=1	$c=2$	$c=3$	$c=4$	$c=5$	BP	ϵ^{opt}	$(\epsilon^{\mathsf{opt}} - \epsilon^*)/\epsilon^{\mathsf{opt}}$
2	0.061	—	—	—	—	0.113	0.146	58.2%
4	0.092	0.153	0.154	—	—	0.194	0.248	37.9%
8	0.093	0.186	0.196	0.194	0.193	0.254	0.319	41.7%
16	0.094	0.200	0.218			0.296		

$R = 1/3, d_\mathsf{v} = 4, d_\mathsf{c} = 6$

q	ϵ^* c=1	$c=2$	$c=3$	BP	ϵ^{opt}	$(\epsilon^{\mathsf{opt}} - \epsilon^*)/\epsilon^{\mathsf{opt}}$
2	0.082	—	—	0.119	0.174	52.8%
4	0.131	0.125	0.117	0.196	0.292	59.9%
8	0.146	0.174		0.248	0.373	53.3%

$R = 1/4, d_\mathsf{v} = 3, d_\mathsf{c} = 4$

q	ϵ^* $c=1$	$c=2$	$c=3$	$c=4$	$c=5$	BP	ϵ^{opt}	$(\epsilon^{\mathsf{opt}} - \epsilon^*)/\epsilon^{\mathsf{opt}}$
2	0.106	—	—	—	—	0.166	0.215	50.7%
4	0.123	0.222	0.223	—	—	0.277	0.355	37.2%
8	0.124	0.269	0.284	0.285	0.285	0.354	0.448	36.7%
16	0.120	0.287	0.315			0.407	0.512	43.9%

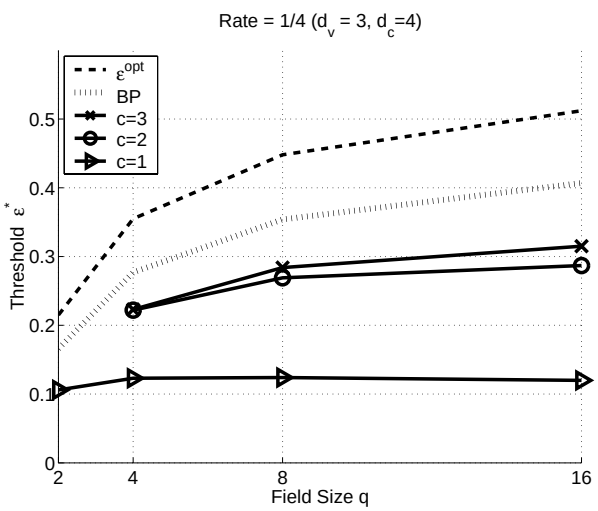

Fig. 2. Thresholds for the proposed decoding algorithm on the q-ary symmetric channel, decoding rate 1/4 codes

VI. NUMERICAL RESULTS FOR PROPOSED ALGORITHM

In this section, we obtain numerical results for discrete density evolution for a q-ary symmetric channel, characterized by a parameter ϵ, where the probability of correctly receiving a transmitted symbol is $1 - \epsilon$, and the uniform probability of receiving any of the $q - 1$ erroneous symbols is $\epsilon/(q - 1)$. The outputs of the channel $\{0, 1, \ldots, \alpha^{q-2}\}$ are one-to-one mapped to $\mathcal{O} = \{[1, 0, \ldots, 0], [0, 1 \ldots 0], \cdots, [0, 0, \ldots, 1]\}$.

In computing the densities, we assume a fixed weight $w = 1$. However, as was shown in [1] [8], we expect some improvement in the thresholds by optimizing this parameter.

For the q-ary output channel, $d_\mathsf{v} = 2$ codes are not suitable for the proposed decoder. In this case, only weight-1 output messages are produced at the output of the check node on the first iteration. This input to the variable node does not lead to improved message reliability as iterations progress, and convergence is never reached. This observation is in contrast with belief propagation on AWGN channels, where good codes with column weight $d_\mathsf{v} = 2$ can be used [4].

Density evolution is performed by choosing a candidate value of ϵ, and repeatedly applying ${}^\star\Phi_\mathsf{v}$ and ${}^\star\Phi_\mathsf{c}$. The noise threshold ϵ^*, is the largest channel parameter ϵ for which convergence is obtained.

Table 1 shows noise thresholds computed using the proposed techniques, for various values of q, R, and c. As can be seen, compared to thresholds with $c = 1$, which we had found in [8], increasing c generally improves noise thresholds. Particularly encouraging is rate 1/4, $q = 16$ where there was a substantial gain by using $c = 2$ over $c = 1$, see Fig. 2. In Table 1, we also show ϵ^{opt}, the channel which has capacity equal to the specified rate. Empty table cells indicate numbers that are computationally difficult to obtain; "—" indicates non-meaningful combinations of q and c.

VII. CONCLUSION

Generalized message-passing decoding of $GF(q)$ LDPC codes has been considered, and a means to efficiently perform

$$\rho_t = \sum_{a=1}^{S^O} \sum_{z\in\mathcal{O}_a} \sum_{m_i:\sum m_i = d_\mathsf{v}-1} \sum_{n_i:c(m_1)} \cdots \sum_{n_i:c(m_{S^\mathsf{L}})} \binom{d_\mathsf{v}-1}{n_1,\cdots,n_{N_L}} \overline{N}_\mathsf{v}^{(t)}(z,w,n_1,\cdots,n_{N_L}) \frac{1}{S_z^\mathsf{O}} (S_z^\mathsf{O}\nu_z) \cdot \prod_{i=1}^{S^\mathsf{L}} \frac{1}{(S_i^\mathsf{L})^{m_i}} ((S_i^\mathsf{L})\lambda_{\mathcal{L}_i(1)})^{m_i} \quad (32)$$

$$\rho_t = \sum_{a=1}^{S^O} \sum_{m_i:\sum_i m_i = d_\mathsf{v}-1} \sum_{z\in\mathcal{O}_a} \sum_{n_i:c(m_1)} \cdots \sum_{n_i:c(m_{S^\mathsf{L}})} \binom{d_\mathsf{v}-1}{n_1,\cdots,n_{N_L}} \frac{\overline{N}_\mathsf{v}^{(t)}(z,w,n_1,\cdots,n_{N_L})}{|\mathcal{O}_a|\cdot(S_1^\mathsf{L})^{m_1}\cdots(S_{S^\mathsf{L}}^\mathsf{L})^{m_{S^\mathsf{L}}}} \widetilde{\nu}_a \widetilde{\lambda}_1^{m_1}\cdots\widetilde{\lambda}_{S^\mathsf{L}}^{m_{S^\mathsf{L}}} \quad (33)$$

$$\rho_t = \sum_{a=1}^{S^O} \sum_{m_i:\sum_i m_i = d_\mathsf{v}-1} \overline{X}_\mathsf{v}^{(t)}(a,w,m_1,\cdots,m_{\widetilde{N}_L}) \widetilde{\nu}_a \widetilde{\lambda}_1^{m_1}\cdots\widetilde{\lambda}_{S^\mathsf{L}}^{m_{S^\mathsf{L}}} \quad (34)$$

$$\sum_{a=1}^{S^O} \sum_{m_i:\sum_i m_i = d_\mathsf{v}-1} \overline{X}_\mathsf{v}^{(t)}(a,w,m_1,\cdots,m_{S^\mathsf{L}}) \widetilde{\nu}_a \widetilde{\lambda}_1^{m_1}\cdots\widetilde{\lambda}_{S^\mathsf{L}}^{m_{S^\mathsf{L}}} = \sum_{a=1}^{S^O} \sum_{m_i:\sum_i m_i = d_\mathsf{v}-1} \overline{X}_\mathsf{v}^{(s)}(a,w,m_1,\cdots,m_{S^\mathsf{L}}) \widetilde{\nu}_a \widetilde{\lambda}_1^{m_1}\cdots\widetilde{\lambda}_{S^\mathsf{L}}^{m_{S^\mathsf{L}}} \quad (35)$$

density evolution was given. Distinct from belief-propagation decoding, such a system requires the proposal of a specific message alphabet and decoding rules. Accordingly, we have proposed an algorithm for the decoding of $GF(q)$ LDPC transmitted over the q-ary symmetric channel. Because of the small number of message values, it is possible to compute the message distributions under density evolution, and thresholds for $GF(q)$ LDPC codes of various field sizes and rates were found. As with Algorithm E for binary codes, the proposed algorithm has modestly lower thresholds than belief propagation decoding. This framework forms a basis for designing irregular $GF(q)$ LDPC codes, which are expected to have superior performance to the regular codes considered in this paper.

Appendix

In this appendix, Lemma 1 and 2 are justified.

To show Lemma 1 for check variable node, the terms n_i in the summation (8) are separated according to the partitions $\mathbb{P}_\mathcal{L}$ and $\mathbb{P}_\mathcal{O}$. From this (32) is obtained. If $m = n_1 + n_2 + n_3$, then clearly $\lambda^{n_1}\lambda^{n_2}\lambda^{n_3} = \lambda^m$.

Then, (33) is obtained by applying (21), which showed that the probability for the class is the probability of any element times the size of the class. Next, (34) is obtained by application of the definition of $\overline{X}_\mathsf{v}$, (12).

The condition that two outputs are equal, that is, $\rho_t = \rho_s$ is equivalent to (35). Finally, (35) holds if and only if (16) holds.

The justification of Lemma 2 follows similar lines. Beginning with (9), the summation is separated according to the partition $\mathbb{P}_\mathcal{R}$ and $\mathbb{P}_\mathcal{S}$:

$$\sigma'_u = \sum_{s=1}^{S^S} \sum_{i\in\mathcal{S}_s} \Big[\sum_{t=1}^{S^L} \sum_{k\in\mathcal{L}_t} \frac{N_\mathsf{S}^{(u)}(i,k)}{S_i^\mathsf{S} S_j^\mathsf{R}} S_i^\mathsf{S}\sigma_i S_k^\mathsf{R}\rho_k \Big] \quad (36)$$

Then, the definition of X_S is applied:

$$\sigma'_u = \sum_{s=1}^{S^S}\sum_{t=1}^{S^R} X_\mathsf{S}^{(u)}(s,t)\widetilde{\sigma}_s\widetilde{\rho}_t \quad (37)$$

Outputs $u, v \in \mathcal{S}$ are in the same class if:

$$\sigma'_u = \sigma'_v \quad (38)$$

$$\sum_{s=1}^{S^S}\sum_{t=1}^{S^R} X_\mathsf{S}^{(u)}(s,t)\widetilde{\sigma}_s\widetilde{\rho}_t = \sum_{s=1}^{S^S}\sum_{t=1}^{S^R} X_\mathsf{S}^{(v)}(s,t)\widetilde{\sigma}_s\widetilde{\rho}_t \quad (39)$$

The above statement holds if and only if:

$$X_\mathsf{S}^{(u)}(s,t) = X_\mathsf{S}^{(v)}(s,t), \quad (40)$$

which proves Lemma 2.

References

[1] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, February 2001.

[2] M. C. Davey and D. MacKay, "Low-density parity check codes over $GF(q)$," *IEEE Communications Letters*, vol. 2, pp. 165–167, June 1998.

[3] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording," *IEEE Transactions on Magnetics*, vol. 39, March 2003.

[4] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over $GF(q)$," in *Proceedings of IEEE International Symposium on Information Theory*, (Adelaide, Australia), IEEE, September 2005.

[5] H. Wymeersch, H. Steendam and M. Moeneclaey, "Log-domain decoding of LDPC codes over $GF(q)$," in *Proceedings IEEE International Conference on Communications*, (Paris, France), pp. 772–776, IEEE, June 2004.

[6] M. C. Davey and D. MacKay, "Monte Carlo simulations of infinite low density parity check codes over GF(q)," in *International Workshop on Optimal Codes and Related Topics*, (Bulgaria), June 1998.

[7] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Transactions on Information Theory*, vol. 52, pp. 549–583, February 2006.

[8] B. Kurkoski, K. Yamaguchi, K. Kobayashi, "Generalization of 'Algorithm E' for Decoding LDPC Codes Over $GF(q)$," in *Proceedings of Hawaii, IEICE and SITA Joint Conference on Information Theory* (HISC), Nara, Japan, May 2006.

[9] John B. Anderson and Stephen M. Hladik. Tailbiting MAP decoders. *IEEE Journal on Selected Areas in Communications*, 16(2):297–302, February 1998.