Soft-Input, Iterative, Reed-Solomon Decoding using Redundant Parity-Check Equations

Jason Bellorado
Link-A-Media Devices
Santa Clara, CA
jbellorado@link-a-media.com

Aleksandar Kavčić
Department of Electrical Engineering
University of Hawaii
alek@spectra.eng.hawaii.edu

Li Ping
Department of Electronic Engineering
City University of Hong Kong
eeliping@cityu.edu.hk

***Abstract*— In this work we present a practical approach to the iterative decoding of Reed-Solomon (RS) codes. The presented methodology utilizes an architecture in which the output produced by steps of *Belief-Propagation* (BP) is successively applied to a legacy decoding algorithm. Due to the highly suboptimal performance of BP conducted on the inherently dense RS parity-check matrix, a method is first provided for the construction of reduced-density, binary, parity-check equations. Iterative decoding is then conducted utilizing a subset of a redundant set of parity-check equations to minimize the number of connections into the least-reliable bits. Simulation results show that performance comparable to (and exceeding) the best known practical RS decoding techniques is achievable with the presented methodology. The complexity of the proposed algorithm is orders of magnitude lower than these existing procedures and permits a practical implementation in hardware.**

I. Introduction

The introduction of *Turbo Codes* [1] and the re-discovery of *Low-Density Parity Check* (LDPC) codes [2] in the early 1990's demonstrated the achievability of decoding performance arbitrarily close to the capacity for many practical channel models. This performance was shown to be possible using *Belief-Propagation* (BP) style decoding methodologies. In such, information is passed along the edges of a graph (*Tanner Graph*) defined by a code's constraint equations and iteratively updated using localized update rules. All such update procedures, however, are predicated on the basic assumption that the messages passed to a given node on distinct edges are statistically independent, an assumption that truely holds only when the underlying graph is cycle-free. Much recent work ([3], [4]) has hence focused on the construction of graphs for finite length block codes to mimic this property, and have been shown to perform well for codes with very sparse parity-check matrices.

The parity-check matrix of a Reed-Solomon (RS) code is significantly more dense than that of any code on which iterative decoding has been shown to perform well. In particular, the unavoidable presence of a large number of 4-cycles in its graph is the single largest obstacle to the iterative decoding of RS codes using BP. The extension of BP decoding techniques for RS codes is, therefore, certainly nontrivial.

The first successful iterative RS decoding technique, developed by Jiang and Narayanan in 2004 [5], was the *Adaptive Belief-Propagation* (ABP) algorithm. This method is implemented using an iterative architecture, in which hard-decision vectors are produced using the output of successive iterations of BP. To obviate the inherent difficulties of BP decoding on the dense RS parity-check matrix, Gaussian elimination is performed on the parity-check matrix before each iteration, such that the variables of lowest reliability connect into the graph only once. Although the performance of the ABP algorithm has been shown to be superior to any existing (practical) RS decoding technique (see [6] for a superior performing yet, very complex RS decoding method), the graph adaptation renders a hardware implementation of the procedure impossible.

Here we present an approach for iterative RS decoding, the foundation of which is a reduced-density, binary, parity-check matrix. We, thus, first detail the construction of low-weight codewords (in the binary image of the dual code) to compose this matrix. Decoding on the graph defined by these equations ensues utilizing a redundant, fixed, set of parity-check equations, such that the subset that minimizes the number of connections into the least-reliable bits is used to decode. We further reduce the graph connections by making hard-decisions on the majority of the most-reliable bits before conducting BP. This operation achieves better performance without requiring a multitude of, computationally expensive, floating-point operations. We demonstrate performance comparable to (or better than) the ABP algorithm in a manner that allows a practical implementation in hardware (i.e., with no need the computationally expensive adaptive Gaussian eliminations).

The remainder of this work is organized as follows: We first establish our utilized notation in Section II and detail the construction of reduced-density, binary, RS parity-check matrices in Section III. The proposed decoding algorithm is presented in Section IV and a procedure developed to recover errors in the event of an unsuccessful decoding is detailed in Section V. Simulation results and a brief conclusion are provided in Sections VI and VII.

II. Background and Notation

We denote the Reed-Solomon code of length n and dimension k over the Galois Field of 2^q elements (which we specify by GF(2^q)) as $\mathcal{C}_q(n,k)$. Since the methods presented in this work do not significantly depend on the selection of the $(n-k)$ spectral zeros of the code, we hereafter consider the code with generator polynomial specified by the first

1-4244-1564-0/07/$25.00 ©2007 IEEE

$(n-k)$ powers of a primitive field element $\alpha \in \text{GF}(2^q)$, i.e. $\{\alpha^0, \alpha^1, \ldots, \alpha^{n-k-1}\}$. We denote by $\mathbf{H}_q$ the traditional, Vandermonde, parity-check matrix of $\mathcal{C}_q(n,k)$ over GF(2^q) and $\mathbf{H}_{q,2}$ its binary image [7]. We note that the former has density lower bounded by $(k+1)/n \approx k/n$, while the latter by k/nq. Although we are not able to show the existence of a binary parity-check matrix with a density approaching k/nq, we do show the existence of binary parity-check matrices with densities significantly below k/n.

The basis of our decoding procedures is a BP-style decoding methodology known as the *Min-Sum Algorithm* and, thus, we first establish notation utilized in following sections (we refer the reader to [8] for a detailed explanation of BP decoding and the *Min-Sum Algorithm*). The input to the algorithm is the vector $\mathcal{P}^{(0)} = (p_0^{(0)}, p_1^{(0)}, \ldots, p_{n-1}^{(0)})$ of *a-posteriori probabilities* obtained from the channel output $\mathbf{y}$ where, for (binary) variable v_i,

$$p_i^{(0)} = Pr(v_i = 0|\mathbf{y}). \tag{1}$$

We specify by $V = \{v_1, v_2, \ldots, v_{nq}\}$ the set of nq variable nodes (corresponding to columns of $\mathbf{H}_{q,2}$) and $C = \{c_1, c_2, \ldots, c_{(n-k)q}\}$ the set of $(n-k)\cdot q$ check nodes (corresponding to rows of $\mathbf{H}_{q,2}$). For a given variable node $v \in V$, we index the set of its neighboring check nodes by $\mathcal{N}_v$ (i.e., each check node $c_i,\ i \in \mathcal{N}_v$, may be reached from v by traversing a single graph edge). Similarly, the set of neighboring variable nodes for check node $c \in C$ is indexed by the set $\mathcal{N}_c$. In the j^{th} BP iteration, the message passed from variable node $v \in V$ to check node $c \in C$ is denoted by $\mathcal{V}^{(j)}(v,c)$. The message that is subsequently passed back to variable node v from check node c in this iteration is denoted by $\mathcal{C}^{(j+1)}(c,v)$.

III. Constructing Reduced-Density Binary Parity-Check Matrices for RS Codes

Here, we outline the construction of reduced-density, binary, RS parity-check matrices which will be the foundation of the iterative procedure presented in the following section. Although this methodology is not guaranteed to produce the *minimum* density parity-check matrix for a given code (not even the density of this matrix is known), it does produce matrices of sufficiently low density to achieve high-level decoding performance using the presented decoding methods.

The problem of obtaining the minimum density, binary, parity-check matrix for $\mathcal{C}_q(n,k)$ is equivalent to finding the $(n-k)\cdot q$ codewords of minimum Hamming weight that span the binary image of the dual code ($\mathcal{C}_q^{\perp}(n,k)$), which we denote by $\mathcal{C}_2^{\perp}(n,k)$. Although possible for short codes, exhaustive searches of the code become quickly intractable with increasing code length since $|\mathcal{C}_2^{\perp}(n,k)| = 2^{(n-k)\cdot q}$. Randomized searches, however, have revealed that low-weight codewords may be constructed by considering the subcodes of $\mathcal{C}_2^{\perp}(n,k)$ consisting of codewords that are nonzero only on a subset of indices. We next develop nonrandom constructions of such codewords, from which we form our constructed matrices.

A. Constructing Codewords for Given Support

We denote by $\mathbf{b} = (b_1^0, b_2^0, \ldots, b_q^0, \ldots\ldots, b_1^{n-1}, \ldots, b_q^{n-1})$ a low-weight codeword in $\mathcal{C}_2^{\perp}(n,k)$. In this representation of $\mathbf{b}$, the superscript specifies the associated, higher-field, symbol in GF(2^q), while the subscript indexes its binary representation. For a parameter $|\mathcal{R}| < q$, we define the support-set $\mathcal{R}$,

$$\mathcal{R} = \{i_1, i_2, \ldots, i_{|\mathcal{R}|}\} \subset \{1, 2, \ldots, q\}, \tag{2}$$

and its complement set $\bar{\mathcal{R}}$

$$\bar{\mathcal{R}} = \{1, 2, \ldots, q\} \setminus \mathcal{R}. \tag{3}$$

We constrain $\mathbf{b}$ to be zero outside the defined support set $\mathcal{R}$ for each higher-field symbol, i.e.,

$$b_j^i = 0, \quad \text{for all } (i,j) \in \{0, 1, \ldots, n-1\} \times \bar{\mathcal{R}}. \tag{4}$$

Our objective is to determine all codewords in $\mathcal{C}_2^{\perp}(n,k)$ obeying these constraints, the set of which we denote by $\mathcal{C}_{2,\mathcal{R}}^{\perp}$. We construct $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ directly from $\mathbf{H}_{q,2}$ by considering linear combinations of its rows (which are codewords in $\mathcal{C}_2^{\perp}(n,k)$). For this, we index the columns of $\mathbf{H}_{q,2}$ as $(c_1^0, c_2^0, \ldots, c_q^0, \ldots\ldots, c_1^{n-1}, c_2^{n-1}, \ldots, c_q^{n-1})$ and construct the submatrix composed only of the columns corresponding to the constrained coordinate positions, i.e.

$$c_j^i,\ (i,j) \in \{0, \ldots, n-1\} \times \bar{\mathcal{R}}, \tag{5}$$

which we denote by $\tilde{\mathbf{H}}_{q,2}$. Clearly, if there exists a linear combination of the rows of $\tilde{\mathbf{H}}_{q,2}$ that produces the zero vector, then the combination of the same rows of $\mathbf{H}_{q,2}$ will produce a codeword in $\mathcal{C}_{2,\mathcal{R}}^{\perp}$. Thus, for each $\mathbf{n}$ that satisfies

$$\tilde{\mathbf{H}}_{q,2}^T \cdot \mathbf{n} = \mathbf{0}, \tag{6}$$

we obtain $c_{\mathbf{n}} \in \mathcal{C}_{2,\mathcal{R}}^{\perp}$ as,

$$c_{\mathbf{n}} = \mathbf{H}_{q,2}^T \cdot \mathbf{n}. \tag{7}$$

The entire code $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ is generated from all vectors $\mathbf{n}$ in the left null-space of $\tilde{\mathbf{H}}_{q,2}$ (**Algorithm 1**).

B. Constructed Codeword Weights

Since the codewords in $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ are non-zero on no more than $n|\mathcal{R}|$ coordinate positions, we expect to achieve codewords of average Hamming weight $\approx n|\mathcal{R}|/2$ (density $= |\mathcal{R}|/(2q)$). For sufficiently small $|\mathcal{R}|$, however, the size of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ renders an exhaustive search tractable, the result of which is codewords of Hamming weight significantly less than $\approx n|\mathcal{R}|/2$.

For larger values of $|\mathcal{R}|$, $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ quickly becomes too large to search exhaustively. We note, however, that since we are using these codewords as rows of the parity-check matrix, we need only form a low-weight basis of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ (not the entire code). We have noticed (empirically) that only a small fraction of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ must be searched to obtain a basis consisting of the lowest weight codewords. We, thus, may truncate the search in **Algorithm 1** without significantly impacting the results.

The weights of the produced codewords are proportional to $|\mathcal{R}|$, however, because $\tilde{\mathbf{H}}_{q,2}$ is a wide matrix (has more columns than rows), it is not guaranteed to have a non-empty

ALGORITHM 1: Construction of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$

INPUT
$\mathcal{R}$, $\bar{\mathcal{R}} = \{1, 2, \ldots, q\} \setminus \mathcal{R}$, $\mathbf{H}_{q,2}$
INITIALIZATION
$\mathcal{C}_{2,\mathcal{R}}^{\perp} = \emptyset$
CONSTRUCTION OF $\mathcal{C}_{2,\mathcal{R}}^{\perp}$
Denote:
Rows of $\mathbf{H}_{q,2}$ *by* $\{\underline{h}_1, \underline{h}_2, \ldots, \underline{h}_{(n-k)q}\}$
Index:
Columns of $\mathbf{H}_{q,2}$ *by* $(c_1^0, c_2^0, \ldots, c_q^0, \ldots\ldots, c_1^{n-1}, \ldots, c_q^{n-1})$
Construct:
Matrix of columns c_j^i, $(i,j) \in \{n-k, \ldots, n-1\} \times \bar{\mathcal{R}} \rightarrow \tilde{\mathbf{H}}_{q,2}$
FOR: *Each* $\mathbf{n} = (n_1, \ldots, n_{(n-k)q})$ *in the left null-space of* $\tilde{\mathbf{H}}_{q,2}$
Compute:
$\mathbf{b} = \sum_{i=1}^{(n-k)q} n_i \cdot \underline{h}_i$
$\mathcal{C}_{2,\mathcal{R}}^{\perp} = \mathcal{C}_{2,\mathcal{R}}^{\perp} \cup \mathbf{b}$
END FOR
OUTPUT
$\mathcal{C}_{2,\mathcal{R}}^{\perp}$

left null-space. The existence (or lack thereof) of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ is dependent upon the RS code being considered, as defined by its generator polynomial. For the code under consideration (see Section II), we have empirically verified that $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ exist for all support-sets such that $|\mathcal{R}| \geq q/2$.

C. Constructing the Parity-Check Matrix

In order to represent all $(n-k) \cdot q$ parity-check constraints that define a code, it is necessary to determine an entire basis for $\mathcal{C}_2^{\perp}(n,k)$. We accomplish this by systematically searching through the subcodes of $\mathcal{C}_2^{\perp}(n,k)$ by continually applying **Algorithm 1** to different support-sets $\mathcal{R}$. Since the average weight of a codeword in $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ is proportional to $|\mathcal{R}|$, we step through increasing values of $|\mathcal{R}|$, from $|\mathcal{R}| = 1$ to $(q-1)$ (since $|\mathcal{R}| = q$ corresponds to the case that $\mathcal{C}_{2,\mathcal{R}}^{\perp} = \mathcal{C}_2^{\perp}(n,k)$, it is not considered). For each value of $|\mathcal{R}|$ for which nonzero solutions to (6) do exist, we consider all $\binom{|\mathcal{R}|}{q}$ possible choices of $\mathcal{R}$. The output of **Algorithm 1** is, then, utilized to construct a set $\mathcal{O}$, the set of lowest weight codewords that span the greatest possible dimension.

We acknowledge the redundancy of this procedure since, if support-set $\mathcal{R}_1 \subseteq \mathcal{R}_2$, then $\mathcal{C}_{2,\mathcal{R}_1}^{\perp}$ is a subcode of $\mathcal{C}_{2,\mathcal{R}_2}^{\perp}$. However, since the dimension of $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ is exponentially increasing in $|\mathcal{R}|$, we only exhaustively search the subcodes $\mathcal{C}_{2,\mathcal{R}}^{\perp}$ for small $|\mathcal{R}|$, while a superficial search is utilized for larger $|\mathcal{R}|$. We have found this methodology to quickly produce results.

To increase the effectiveness of our search, we also utilize the cyclic property of RS codes. Since RS codes are cyclic only in the higher field (GF(2^q)), and not in the binary field, we may only utilize cyclic shifts of length $\ell \cdot q$, $\ell \in \mathbb{Z}$, which we denote by $[\mathbf{b}]_{\ell \cdot q}$. Thus, a low-weight codeword produced by our search may be expanded into multiple codewords (of equivalent weight) by considering its cyclic q-shifts. The full search procedure we employ is given as **Algorithm 2**.

D. Completing the Span of the Parity-Check Matrix

In the event that the codewords in $\mathcal{O}$, as produced by **Algorithm 2**, span $\mathcal{C}_2^{\perp}(n,k)$, the parity-check matrix is constructed using these vectors as its rows. If $\mathcal{O}$ spans a smaller dimensional subspace, then we also require a set of vectors

ALGORITHM 2: Low-Density Parity-Check Matrix Construction

$\mathcal{O} = \emptyset$
FOR $|\mathcal{R}| = 1 : q-1$
FOR all support sets $\mathcal{R}$
$\mathcal{O}' =$ **ALGORITHM 1**$(\mathcal{R}, \mathbf{H}_{q,2})^*$
FOR each $\mathbf{b} \in \mathcal{O}'$
FOR each cyclic shift of $\mathbf{b}$ of length $\ell q = [\mathbf{b}]_{\ell q}$, $l \in \{0, \ldots, n\text{-}1\}$
IF $[\mathbf{b}]_{\ell q} \notin SPAN(\mathcal{O})$
$\mathcal{O} = \mathcal{O} \cup [\mathbf{b}]_{\ell q}$
ELSE
IF $w_H(\mathbf{b}) < w_H(\tilde{\mathbf{b}})$, $\tilde{\mathbf{b}} \in \mathcal{O}$, & $span(\mathcal{O} \setminus \tilde{\mathbf{b}} \cup [\mathbf{b}]_{\ell q}) = span(\mathcal{O})$
$\mathcal{O} = \{\mathcal{O} \setminus \tilde{\mathbf{b}}\} \cup [\mathbf{b}]_{\ell q}$
END FOR
END FOR
END FOR
END FOR
OUTPUT
$\mathcal{O}$

* If the left null-space of $\tilde{\mathbf{H}}_{q,2}$ has large dimension, **Algorithm 1** may be abbreviated such that $\mathcal{O}'$ contains a fraction of the codewords with support-set $\mathcal{R}$.

($\bar{\mathcal{O}}$) to complete this span. In [7], we provide a set of heuristic guidelines to construct $\bar{\mathcal{O}}$. However, because the majority of the matrix can be constructed using **Algorithm 2** (see Table I), we omit this discussion here.

E. Constructed Matrices

In Table I we provide examples of the lowest-density, binary, parity-check matrices constructed for the following codes; $\mathcal{C}_5(31,25)$, $\mathcal{C}_6(63,55)$, $\mathcal{C}_8(255,239)$, $\mathcal{C}_{10}(460,420)$. We note that, for the considered generator polynomial, the parity-check matrices for $\mathcal{C}_5(31,25)$ and $\mathcal{C}_6(63,55)$ are of minimum possible density (as verified by an exhaustive search). The enormous sizes of $\mathcal{C}_8(255,239)$ and $\mathcal{C}_{10}(460,420)$ render an exhaustive search prohibitive and, thus, no such claim can be made in these cases.

We, finally, comment on the construction of a parity-check matrix for $\mathcal{C}_{10}(460,420)$, which is a shortened code. We first construct the reduced-density, binary, parity-check matrix for the natural length code $\mathcal{C}_{10}(1023,983)$ using the aforementioned techniques. We then select the cyclic q-shift of each row that minimizes the density of the first nq columns while maintaining their linear independence in these positions. Since this methodology does not produce rows that are uniform in Hamming-weight, the values given in Table I for this code are actually average row weights (i.e., (60,1116) indicates that the first 60 rows collectively have an average weight of 1116).

IV. Iterative RS Decoding

Despite the density reduction achieved by techniques of the preceding section, the resulting graph still has a significant number of short-length cycles that are severely detrimental to the performance of BP decoding. We, thus, modify the application of BP utilizing the following steps:

- *Legacy/Erasures-Only Decoding Front-End*
- *Bit Slicing*
- *Belief-Propagation/Legacy Iterations*
- *Error Recovery*,

each of which is detailed in the following text.

TABLE I

LOWEST DENSITY, BINARY, PARITY-CHECK MATRICES CONSTRUCTED*.

CODE	MATRIX DESCRIPTION	DENSITY
$\mathcal{C}_5(31,25)$	(30,48)*	30.97%
$\mathcal{C}_6(63,55)$	(24,96)*, (24,120)*	28.57%
$\mathcal{C}_8(255,239)$	(48,512)*, (24,688)*, (32,764)* (16,768), (16,848)	35.00%
$\mathcal{C}_{10}(460,420)$	(60,1116)*, (80,1465)*, (60,1635)* (110,1806)*, (10,1964), (20,1986) (60,2048)	36.08%

* The matrix description is given as a set of pairs (a, b), where a denotes the number of rows of Hamming weight b in the resulting matrix. The rows specified by $*$ were constructed using **Algorithm 2** (see [7] for the heuristic guidelines utilized to construct the remaining rows).

A. Legacy/Erasures-Only Decoding Front-End

The front-end of the decoding algorithm consists of a legacy decoding procedure followed by a bit-level *erasures-only* decoding, which is performed only in the event of a legacy decoding failure. We remark that either of these portions of the front-end may be omitted from the decoding procedure; however, there are significant advantages to utilizing this type of architecture. In addition to the obvious performance gains, the average complexity of the overall algorithm is also dramatically reduced by the inclusion of the decoding front-end. This complexity reduction is explained by the fact that BP is only run in the event of a front-end decoding failure. Since the complexity of BP decoding, as well as the power dissipated by its hardware implementation, is significantly higher than that of the front-end, a significant reduction of these, critical, factors is achieved by its utilization.

Legacy decoding[1] may be applied in a straightforward manner, however, we give a brief explanation of *bit-level erasures-only decoding* in the following. In such, we partition the coordinate positions into the sets $\mathcal{L} = \{l_1, l_2, \ldots, l_{(n-k)\cdot q}\}$, containing the $(n-k)\cdot q$ coordinate positions of lowest (channel) reliability, and $\mathcal{H}$, containing the $k \cdot q$ coordinate positions of highest reliability. The bits in $\mathcal{H}$ are replaced with their hard-decision values and the parity of their respective checks updated appropriately. The parity-check equations can then be solved for the vector of low reliability bits that satisfy the check equations.

We note here that it is both possible that either no solutions, or multiple solutions, exist to this set of equations. In the former case, an *erasures-only* decoding failure is declared and the decoder proceeds to the next decoding stage. In the latter case, the set of all solutions is searched for the solution closest (in Euclidean distance) to the hard-decision vector, which is then declared the decoded codeword.

B. Bit Slicing

The *Bit-Slicing* operation, like *bit-level erasures-only* decoding, utilizes the channel soft-information to partition the variable nodes into sets of high and low-reliability bits. If an appropriate selection is made, the bits in the latter set (with high probability) all have correct channel information (i.e., their hard-decisions and transmitted values are equivalent). We, thus, make hard-decisions on these bits (an operation historically referred to as *slicing*) and remove them from the graph such that BP may be conducted using only the low-reliability bits. We refer to the high-reliability bits as the *sliced-set* ($\mathcal{S}_l$) and the low-reliability bits as the *active-set* ($\mathcal{A}_{\mathcal{S}_l}$).

Because the *slicing* process discards information about the high-reliability bits, this procedure is, clearly, suboptimal. However, in a graph containing a large number of short-length cycles, the independence assumption on which the BP update equations are based is rendered invalid. The treatment of correlated messages as being independent allows low-reliability bits to significantly affect the value of high-reliability bits. By locking the values of the highest-reliability bits to their hard-decisions, we disallow this from occurring which, the vast majority of the time, is the correct course of action.

Using the channel log-likelihood ratios, llr_i, $i = 0, 1, \ldots nq-1$, defined as

$$llr_i = \log\left(\frac{Pr(v_i = 0|\mathbf{y})}{Pr(v_i = 1|\mathbf{y})}\right), \tag{8}$$

we partition the the set of all bits $\{v_0, v_1, \ldots, v_{n\cdot q-1}\}$ into the sets $\mathcal{S}_l$ and $\mathcal{A}_{\mathcal{S}_l} = \{v_0, v_1, \ldots, v_{n\cdot q-1}\} \setminus \mathcal{S}_l$ utilizing one of the following methodologies;

- *Threshold Slicing:* For a given threshold γ, the set of sliced coordinate positions is given by

$$\mathcal{S}_l = \{i; |llr_i| \geq \gamma\}. \tag{9}$$

- *Fixed-Number Slicing:* For a given real number θ, $0 < \theta \leq 1$, the sliced-set is chosen as the $\lfloor(1-\theta)\cdot n \cdot q\rfloor$ indices with largest magnitude log-likelihood ratios, i.e.,

$$\mathcal{S}_l = \arg \max_{\substack{\mathcal{A}\subseteq\{0,\ldots,n\cdot q-1\} \\ |\mathcal{A}|=\lfloor(1-\theta)\cdot n\cdot q\rfloor}} \sum_{i\in\mathcal{A}} |llr_i|. \tag{10}$$

Regardless of the method used to construct $\mathcal{S}_l$ (see [7] for a comparison of these slicing methods), we modify the BP updates to account for slicing the variables in $\mathcal{S}_l$ by computing the *sliced-parity* for each check (c_i), as

$$p'_i \triangleq \sum_{j\in\mathcal{N}_{c_i}\cap\mathcal{S}_l} v_j. \tag{11}$$

The sign of each outgoing message of check c_i is, then, multiplied by p'_i to reflect this change.

C. Iterative Decoding

We conduct iterative decoding using only the low-reliability (*active*) variables in $\mathcal{A}_{\mathcal{S}_l}$. In contrast to traditional iterative decoding, however, we apply the output of each successive BP iteration to a legacy decoding algorithm. We, thus, utilize the BP output in a manner that gives the legacy decoding algorithm the best chance of correctly decoding.

[1] In addition to traditional HDD (e.g., Berleykamp-Massey), we also consider *Low-Complexity Chase* (LCC) legacy decoding. See [9] or [7] for a full treatment of this algorithm.

We express the, *a-posteriori*, log-likelihood ratios $q_i^{(j)},\ i \in \mathcal{A}_{\mathcal{S}_l}$, produced by the j^{th} BP iteration as,

$$q_i^{(j)} = llr_i + \sum_{m \in \mathcal{N}_{v_i}} \mathcal{C}^{(j)}(v_i, c_m). \tag{12}$$

Because RS graphs contain many cycles, the extrinsic information is not an *a-posteriori probability* (as in a cycle-free graph), but merely a metric that indicates the value of v_i required to satisfy its adjacent check nodes. We have found that treating the extrinsic information as an *a-posteriori probability* leads to severely suboptimal results. Here we consider the use of LCC (see [7], [9]) and HDD (Berleykamp-Massey) legacy decoding and, therefore, we next specify how the BP output is used to obtain an input to these procedures. Although we target this discussion directly to these legacy decoding algorithms, it is extendable to any legacy decoding method (i.e. Sudan [10], Kötter-Vardy (K-V) [11], or the newly-proposed bit-level algebraic decoder [12] etc.).

For the application of HDD, we require only a hard-decision vector, whereas LCC decoding requires, in addition, a secondary hard-decision vector and a set of unreliable indices $\mathcal{I}$. Although the BP output values $q_i^{(j)},\ i \in \mathcal{A}_{\mathcal{S}_l}$, may be used directly to produce the hard-decision vector, better performance may be obtained by restricting the use of the extrinsic information to variables with very little channel information. We, thus, partition $\mathcal{A}_{\mathcal{S}_l}$ as,

- $\mathcal{A}_{\mathcal{S}_l,L} \subseteq \mathcal{A}_{\mathcal{S}_l}$ - set of indices of lowest channel reliability, where $|\mathcal{A}_{\mathcal{S}_l,L}|$ is a configurable parameter.
- $\mathcal{A}_{\mathcal{S}_l,H} = \mathcal{A}_{\mathcal{S}_l} \setminus \mathcal{A}_{\mathcal{S}_l,L}$ - remaining indices in $\mathcal{A}_{\mathcal{S}_l}$.

The hard-decision vector is obtained for the coordinate positions in $\mathcal{A}_{\mathcal{S}_l,L}$ using the metrics $\{q_i^{(j)}\}$, whereas only the channel log-likelihood ratios $\{llr_i\}$ are used for the indices in $\mathcal{A}_{\mathcal{S}_l,H}$ (hard-decisions are always used for the sliced variables in $\mathcal{S}_l$). We, finally, note that, in applying LCC [7], [9], we select the η unreliable indices in $\mathcal{I}$ according to llr_i, thus, rendering $\mathcal{I}$ constant through all iterations.

We next improve the decoding performance by further altering our constructed graphs. Our procedure utilizes a superset of $(n-k)\cdot q + r$ parity-check equations that span the full $(n-k)\cdot q$ dimensional binary image of the dual code $\mathcal{C}_2^{\perp}(n,k)$ (the r redundant checks are easily obtained in the matrix construction process). Here, the redundant check equations are used to maximally reduce the number of graph connections into the *active* variables $\mathcal{A}_{\mathcal{S}_l}$. Denoting the number of *active* variable nodes adjacent to check c_i as

$$n_{c_i}(\mathcal{A}_{\mathcal{S}_l}) = |\mathcal{N}_{c_i} \cap \mathcal{A}_{\mathcal{S}_l}|, \tag{13}$$

we select the $(n-k)\cdot q$ checks that minimize (13), provided $n_{c_i}(\mathcal{A}_{\mathcal{S}_l}) \neq 0$. This set of *active* check nodes, which we denote by $C_{\mathcal{A}_{\mathcal{S}_l}}$, is chosen using a simple greedy approach. We proceed with BP in the aforementioned manner while passing messages only through the checks in $C_{\mathcal{A}_{\mathcal{S}_l}}$.

V. Error Recovery

We conduct *iterative decoding* until a successful decoding is reached or for a predetermined, maximum, number of unsuccessful iterations. In this event, we employ an *error-recovery* procedure to recover the corrupted data. The technique we utilize is more powerful than Chase decoding [13], which has, traditionally, been used in this capacity.

ALGORITHM 3: Error Recovery
INPUT
$|\mathcal{K}^{(j)}|,\ n_{\mathcal{K}}$
INITIALIZE
$\mathcal{K}^{(0)} = \emptyset$
RECOVER
FOR ITERATION $j = 0, n_{\mathcal{K}}, 2n_{\mathcal{K}}, 3n_{\mathcal{K}} \ldots$
Conduct: $n_{\mathcal{K}}$ BP iterations without checks in $\mathcal{K}^{(j)}$ passing messages.
Compute: UNSATISFIED CHECK SET $\mathcal{U}^{(j)}$
FOR $m \in \mathcal{U}^{(j)}$
Compute: $e_{c_m}(j)$ using (14)
END FOR
Compute: $\mathcal{K}^{(j)}$ using (15)
END FOR

Error-recovery attempts to locate high-reliability channel errors which source, large magnitude, erroneous, messages. We intend to halt the flow of these messages that propagate rapidly through the dense RS graph. Distinguishing erroneous bits from correct bits, however, is an extremely difficult task. We, thus, attempt the easier task of determining the neighborhood around these bits. During iteration j we lower bound on the magnitude of the received erroneous message(s) by unsatisfied check c_i (which is directly connected to an odd number of bit errors) as,

$$e_{c_i}(j) = \min_{m \in \mathcal{N}_{c_i}} |\mathcal{V}^{(j)}(v_m, c_i)|. \tag{14}$$

This metric not only provides us with a measure of the probability that a high-reliability bit-error is connected to c_i, but also identifies checks capable of propagating large magnitude messages (see the check node update equations [7]).

Error-recovery is conducted in sets of $n_{\mathcal{K}}$ iterations, where $n_{\mathcal{K}}$ is a defined parameter. Before conducting iteration $j = m \cdot n_{\mathcal{K}},\ m = 1, 2, \ldots$, we, for a given set-size $\mathcal{K}$, first index the set of checks that maximize (14) as,

$$\mathcal{K}^{(j)} = \{k_1^{(j)}, \ldots, k_{|\mathcal{K}|}^{(j)}\} = \arg \max_{\{i_1,\ldots,i_{|\mathcal{K}|}\} \subseteq \mathcal{U}^{(j)}} \sum_{l=1}^{|\mathcal{K}|} e_{c_{i_l}}(j), \tag{15}$$

where $\mathcal{U}^{(j)}$ indexes the set of unsatisfied checks before iteration j. We conduct the next $n_{\mathcal{K}}$ BP iterations without permitting the checks indexed by $\mathcal{K}^{(j)}$ to pass messages. At the completion of these iterations, $\mathcal{K}^{(j)}$ is reselected according to (15) and the process is repeated. We give the complete recovery process as **Algorithm 3** and note that we choose the parameters $|\mathcal{K}^{(j)}|$ and $n_{\mathcal{K}}$ to minimize the frame error-rate.

VI. Simulation Results

We next present simulation results for $\mathcal{C}_8(255, 239)$ (Fig. 1) and $\mathcal{C}_{10}(460, 420)$ (Fig. 2). In such, we used 60 iterations of BP for both normal decoding and for *error-recovery* and a single redundant set of parity-check equations (i.e., $r = (n-k)\cdot q$). We provide performance results for both HDD

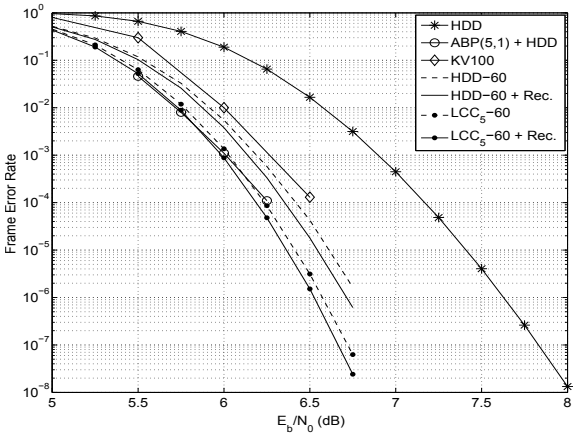

Fig. 1. FER for $\mathcal{C}_8(255, 239)$ with and without error-recovery. We present both HDD and LCC ($\eta = 5$) legacy decoding as well as HDD, ABP(5,1)+HDD, and K-V(100) decoding algorithms.

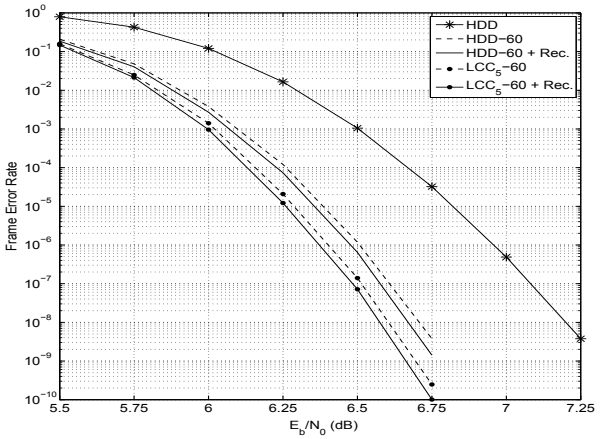

Fig. 2. FER for $\mathcal{C}_{10}(460, 420)$ with and without error-recovery. We present both HDD and LCC ($\eta = 5$) legacy decoding as well as HDD.

and LCC legacy decoding and compare our results to HDD (as well as ABP and K-V [11] decoding for $\mathcal{C}_8(255, 239)$). Since many variants of ABP decoding exist, the results we provide were obtained from [5].

Our decoding method achieves significant performance gains over traditional HDD, even with the use of HDD legacy decoding. We achieve an FER of 10^{-6} (resp. 10^{-8}) at an E_b/N_0 of 0.9 dB (0.49 dB) lower than that required by HDD to achieve this error-rate for $\mathcal{C}_8(255, 239)$ ($\mathcal{C}_{10}(460, 420)$). With the additional utilization of *error-recovery*, we further reduce this value by 0.1 dB (0.0525 dB). The performance of our iterative procedure is also increased by employing the LCC algorithm for legacy decoding. For a practical selection of $\eta = 5$ (see [7], [9]) we achieve additional gain of 0.30 dB (0.12 dB), thus displaying an achievable improvement over HDD of 1.30 dB (0.6625 dB).

At an FER of 10^{-4} (which is the lowest ABP FER existing in literature) our method utilizing HDD legacy decoding performs extremely close to the ABP algorithm [5]. We do note, however, that the performance gap between these decoding procedures is decreasing with increasing E_b/N_0 and, thus, vanishing at low FERs. We are, however, able to outperform the ABP algorithm if LCC legacy decoding is employed.

In Fig.1, we also provide the performance of K-V decoding [11] with maximum multiplicity of 100 for $\mathcal{C}_8(255, 239)$. Despite the extremely high complexity associated with this decoding procedure, our decoding techniques are able to outperform this methodology for all values of E_b/N_0. Here, the performance advantage is increasing with decreasing FER. Note that (though at an extremely high complexity) an additional gain of 1 dB is achievable with methods of [6].

VII. Conclusions

We have presented a method to iteratively decode RS codes based on our construction of reduced density, binary, parity-check matrices. Our method is conducted in an iterative fashion in which the output *Belief Propagation* is successively applied to a legacy decoding procedure. We have demonstrated performance similar (and exceeding) the best-known (practical) RS decoding methodology, the ABP algorithm. Unlike the ABP algorithm, however, these results are achievable without the need for graph adaptation, which is computationally expensive and not implementable in hardware. These performance and complexity advantages hold with the application of any legacy decoding algorithm (i.e. HDD, Sudan, K-V), however, are most pronounced with LCC legacy decoding.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Inter. Conf. on Comm.*, (Geneva, Switzerland), pp. 1064–1070, 1993.

[2] D. MacKay and R. Neal, "Near Shannon limit performance of low-density parity-check codes," *Elect. Lett.*, vol. 32, pp. 1645–1646, 1996.

[3] Y. Kou, S. Lin, and M. Fossorier, "Low density parity check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. on Info. Theory*, vol. 47, pp. 2711–2736, November 2001.

[4] J. Rosenthal and P. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Allerton Conference on Communications and Control*, pp. 248–257, 2000.

[5] J. Jiang and K. Narayanan, "Iterative soft input soft output decoding of Reed-Solomon codes by adapting the parity check matrix," *submitted to IEEE Trans. on Info. Theory*, 2006.

[6] W. Jin and M. Fossorier, "Enhanced box and match algorithm for reliability-based soft-decision decoding of linear block codes," in *Globecom*, (San Francisco, CA), 2006.

[7] J. Bellorado, *Low-Complexity Soft Decoding Algorithms for Reed-Solomon Codes*. PhD thesis, Harvard University, 2006.

[8] D. MacKay, "Information theory, inference, and learning algorithms," 2003.

[9] J. Bellorado and A. Kavčić, "Low complexity decoding algorithms for Reed Solomon codes: Part I - an algebraic soft-in-soft-out Chase decoder," *submitted to IEEE Trans. on Info. Theory*, 2006.

[10] M. Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *J. of Complexity*, vol. 13, pp. 180–193, 1997.

[11] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," 2000.

[12] J. Jiang and K. R. Narayanan, "Algebraic soft decision decoding of reed-solomon codes using bit level soft information," in *Allerton Conference on Comm., Control and Computing*, (Monticello, IL), 2006.

[13] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. on Info. Theory*, vol. IT-18, pp. 170–182, January 1972.