Random Redundant Soft-In Soft-Out Decoding of Linear Block Codes

Thomas R. Halford and Keith M. Chugg
Communication Sciences Institute
University of Southern California
Los Angeles, CA 90089-2565

***Abstract*— A number of authors have recently considered iterative soft-in soft-out (SISO) decoding algorithms for classical linear block codes that utilize redundant Tanner graphs. Jiang and Narayanan presented a practically realizable algorithm that applies only to cyclic codes while Kothiyal *et al.* presented an algorithm that, while applicable to arbitrary linear block codes, does not imply a low-complexity implementation. This work first presents the aforementioned algorithms in a common framework and then presents a related algorithm - random redundant iterative decoding - that is both practically realizable and applicable to arbitrary linear block codes. Simulation results illustrate the successful application of the random redundant iterative decoding algorithm to the extended binary Golay code. Additionally, the proposed algorithm is shown to outperform Jiang and Narayanan's algorithm for a number of Bose-Chaudhuri-Hocquenghem (BCH) codes.**

I. Introduction

Graphical models of codes, and the SISO decoding algorithms implied by those models, are of great current interest. Whereas modern codes are designed with graphical models in mind, classical linear block codes by and large were not. It is well known that acyclic graphical code models (e.g. trellises) imply optimal SISO algorithms whereas models with cycles imply suboptimal, albeit often substantially less complex, decoding algorithms [1]. For many classical codes, the Cut-Set Bound precludes the existence of practically realizable acyclic graphical models [1]. The search for good cyclic graphical models of such codes - that is, models which imply decoding algorithms with near-optimal performance and practically realizable complexity - is thus an important open problem.

A natural starting point in the search for good graphical models of classical linear block codes are the Tanner graphs [2] which are used to decode LDPCs. Although Tanner graphs imply very low-complexity decoding algorithms, most classical linear block codes are defined by high-density, rather than low-density, parity-check matrices and the performance of the decoding algorithms implied by these models is poor.

Recently, a number of authors have studied the SISO decoding algorithms implied by *redundant* Tanner graphs [3], [4]. Section II describes these algorithms in a unified framework while Section III presents the proposed random redundant iterative decoding algorithm (RRD). Section IV details the application of the proposed algorithm to the extended Golay code and to a number of BCH codes. Concluding remarks are given in Section V.

II. Redundant Tanner Graphs

As a motivating example, consider two parity check matrices for the $[8, 4, 4]$ extended Hamming code $\mathcal{C}_8$:

$$H_1 = \begin{bmatrix} 1\,1\,1\,1\,0\,0\,0\,0 \\ 0\,0\,1\,1\,1\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1\,0 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0\,1\,0\,1\,0\,1\,0\,1 \\ 1\,1\,0\,0\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1\,0 \\ 0\,1\,0\,1\,1\,0\,1\,0 \end{bmatrix} \quad (1)$$

These parity-check matrices can be used to construct the redundant Tanner graph illustrated in Figure 1. Note that the vertices corresponding to rows in H_1 and H_2 are labeled $r_{1,i}$ and $r_{2,i}$ for $i = 1, 2, 3, 4$, respectively. The Tanner graph illustrated in Figure 1 is redundant in that only the checks corresponding to H_1 or H_2 are needed to completely specify the code. The *degree* of redundancy in a redundant Tanner graph is defined as the number of parity-check matrices used to construct the model; redundant Tanner graphs can be naturally defined with any degree.

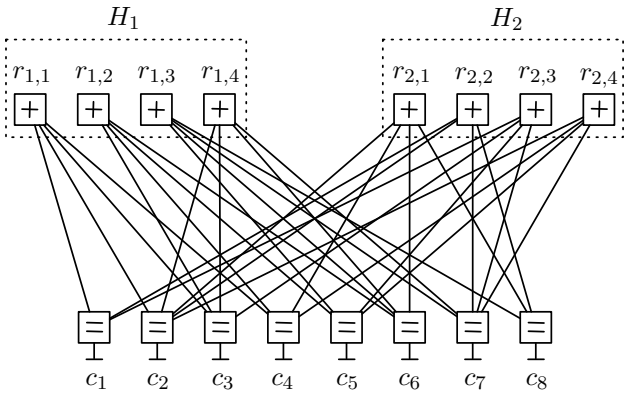

Fig. 1. Redundant Tanner graph for the $[8, 4, 4]$ extended Hamming code defined by H_1 and H_2.

Redundant Tanner graphs imply the following standard iterative decoding algorithm (note that the message passing schedule described below is reasonable but not unique). Consider a degree n redundant Tanner graph. For each set of checks H_i, $i = 1, \ldots, n$, denote by $\mathbf{M}\downarrow_i$ the vector of messages passed from checks to variables and denote by $\mathbf{M}\uparrow_i$ the vector of messages passed from variables to checks. Note that for all

$i = 1, \ldots, n$, $\mathbf{M}\downarrow_i$ is initialized to zero[1]. For $i = 1, \ldots, n$, the $\mathbf{M}\uparrow_i$ messages are first updated at the variable nodes using the channel observation and the $\mathbf{M}\downarrow_j$ messages (for $j \neq i$). The $\mathbf{M}\downarrow_i$ messages are then updated at the check nodes using $\mathbf{M}\uparrow_i$. This updating procedure repeats for a prescribed number of iterations, I. Note that this schedule can be modified so that a number of decoding iterations are performed on each set of checks before proceeding to the next.

From the viewpoint of practical low-complexity implementation, the decoding algorithm described above suffers from two drawbacks:

1) For the standard message passing rules, the intermediate messages vectors $\mathbf{M}\downarrow_i$ must be stored for $i = 1, \ldots, n$ which results in an n-fold increase of the memory required with respect to standard Tanner graph decoding.
2) Since each parity-check matrix defines a different set of checks, there is either an n-fold increase in the number of single parity-check (SPC) trellises which must be implemented or the SPC trellises must be implemented in a reconfigurable fashion.

The first drawback may be addressed by using a massively redundant Tanner graph. If a large degree of redundancy is used then I (the number of decoding iterations on the aggregate model) can be set to one and the intermediate message vectors $\mathbf{M}\downarrow_i$ need not be stored. Before addressing the second drawback, permutation groups of codes must first be defined and reviewed.

Let $\mathcal{C}$ be a block code of length n. The *permutation group* of $\mathcal{C}$, $\mathrm{Per}(\mathcal{C})$, is the set of permutations of coordinate places which send $\mathcal{C}$ onto itself. By definition, $\mathrm{Per}(\mathcal{C})$ is a subgroup of the symmetric group of order n, S_n. Returning to the $[8, 4, 4]$ extended Hamming code, it can be shown that [5][2]:

$$\boldsymbol{\sigma} = (6, 4, 2, 8, 1, 7, 5, 3) \in \mathrm{Per}(\mathcal{C}_8) \tag{2}$$

Applying $\boldsymbol{\sigma}$ to the columns of H_1 yields H_2.

From the above example, it is clear that redundant parity-checks for a given code, $\mathcal{C}$, can be generated by applying permutations drawn from $\mathrm{Per}(\mathcal{C})$ to the columns of some initial parity-check matrix H. Observe that decoding with soft-input vector $\mathbf{SI}$ on $\mathrm{TG}(\boldsymbol{\beta}H)$ (where $\boldsymbol{\beta} \in \mathrm{Per}(\mathcal{C})$) is equivalent to decoding with soft-input vector $\boldsymbol{\beta}^{-1}\mathbf{SI}$ on $\mathrm{TG}(H)$. It is this observation that allows for efficient implementation of redundant Tanner graph decoding: provided that the redundant parity-checks are column permuted versions of some base matrix H, redundant Tanner graph decoding can be implemented by permuting soft information vectors and decoding with a *constant* set of constraints.

From the above discussion, it is apparent that Kothiyal *et al.*'s adaptive belief propagation (ABP) algorithm [4] and Jiang and Narayanan's stochastic shifting based iterative decoding (SSID) algorithm [3] are redundant Tanner graph decoding algorithms. Kothiyal *et al.*'s scheme adaptively chooses new parity-check sets based on soft information reliability. Although the ABP algorithm can be applied to arbitrary block codes, it does not imply a practical low-complexity implementation because the check sets change with every iteration. Furthermore, the ABP algorithm requires computationally expensive Gaussian elimination of potentially large parity-check matrices at every iteration. Jiang and Narayanan's scheme is an example of a practical, low-complexity redundant Tanner graph decoding algorithm for cyclic codes which uses the permutation group approach described above. The random redundant decoding algorithm proposed in this work is, in fact, an extension of Jiang and Narayanan's algorithm to arbitrary block codes.

III. Proposed Decoding Algorithm

A. The Main Algorithm

Algorithm 1 describes the proposed random redundant iterative decoding (RRD) algorithm. The inner **for**-loop of Algorithm 1 describes an efficient redundant Tanner graph decoding algorithm with the addition of a damping coefficient α. The outer **for**-loop of Algorithm 1 iterates over different values of α. By varying the damping coefficient α, the algorithm avoids local minima in the solution space. Many authors have considered the introduction of such damping coefficients in iterative soft decoding algorithms to achieve this end (see, for example, [6]). For practical implementations where a large number of outer iterations is undesirable from a time complexity standpoint, a single damping coefficient (or a small set of coefficients) could be used depending on the operating noise power.

Algorithm 1 takes as input a received soft information vector, $\mathbf{SI}$, a parity-check matrix for the code, H, and four parameters:

1) α_0: The initial damping coefficient.
2) I_1: The number of Tanner graph decoding iterations to perform per inner iteration.
3) I_2: The maximum number of inner iterations to perform per outer iteration. Each inner iteration considers a different random permutation of the codeword elements.
4) I_3: The maximum number of outer iterations to perform. Each outer iteration uses a different damping coefficient.

Let $\mathbf{s}$ be the the sum of input soft information, $\mathbf{SI}$, and the output soft information produced by all previous inner iterations. During the i_2-th inner iteration, I_1 Tanner graph decoding iterations are performed on $\mathrm{TG}(H)$ with damping coefficient α and soft input $\mathbf{s}$ producing the soft output vector, $\mathbf{s}'$, and hard decision, $\mathbf{c}'$. The cumulative soft information vector, $\mathbf{s}$, is then updated to include $\mathbf{s}'$. The inner iteration concludes by applying a random permutation, $\boldsymbol{\theta}$, from the permutation group of the code to $\mathbf{s}$. Decoding concludes when either a valid codeword is returned by the Tanner graph decoding step or when a maximum number of iterations is reached. Before

[1]Throughout this work, decoding is assumed to be performed in the $-\log(\cdot)$ domain, *i.e.* either $\min$-sum or $\min^\star$-sum processing is assumed

[2]Throughout this work, permutations of n coordinate places are described by n-tuples. For example, the application of the permutation $(2, 5, 1, 3, 4, 7, 6)$ to a 7 bit codeword $(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ yields the permuted codeword: $(c_3, c_1, c_4, c_5, c_2, c_7, c_6)$. The identity permutation is denoted $\boldsymbol{\epsilon}$ and the inverse of a permutation $\boldsymbol{\beta}$ is denoted $\boldsymbol{\beta}^{-1}$.

returning the final soft output, $\mathbf{SO}$, and hard decision, $\mathbf{HD}$, the random permutations are undone by applying the inverse of the product of the permutations that were applied to $\mathbf{s}$.

Input: Length n soft-input vector $\mathbf{SI}$.
$n-k \times n$ binary parity-check matrix H.
Parameters I_1, I_2, I_3, α_0.
Output: Length n soft-output vector $\mathbf{SO}$.
Length n hard-decision vector $\mathbf{HD}$.

$\alpha \leftarrow \alpha_0$;
for $1 \leq i_3 \leq I_3$ **do**
  $\boldsymbol{\Theta} \leftarrow \boldsymbol{\epsilon}$;
  $\mathbf{s} \leftarrow \mathbf{SI}$;
  for $1 \leq i_2 \leq I_2$ **do**
    Perform I_1 decoding iterations of $\mathbf{s}$ on $\mathrm{TG}(H)$ with damping coefficient α and place soft output in $\mathbf{s}'$ and resulting hard decision in $\mathbf{c}'$;
    $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{s}'$;
    if $H\mathbf{c}' = 0$ **then**
      Apply $\boldsymbol{\Theta}^{-1}$ to $\mathbf{s}$ and $\mathbf{hd}'$;
      $\mathbf{SO} \leftarrow \mathbf{s} - \mathbf{SI}$;
      $\mathbf{HD} \leftarrow \mathbf{c}'$;
      return $\mathbf{SO}$ *and* $\mathbf{HD}$
    end
    $\boldsymbol{\theta} \leftarrow$random element of $\mathrm{Per}(\mathcal{C})$;
    Apply $\boldsymbol{\theta}$ to $\mathbf{s}$;
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\theta}\boldsymbol{\Theta}$;
  end
  $\alpha \leftarrow \alpha_0 + (1-\alpha_0)\frac{i_3}{I_3-1}$;
end

Algorithm 1: Random Redundant Decoding.

Input: $n-k \times n$ binary parity-check matrix H.
Output: $n-k \times n$ binary parity-check matrix H'.

$H' \leftarrow H$, $r_1^\star \leftarrow 1$, $r_2^\star \leftarrow 1$, $g^\star \leftarrow$ girth of $\mathrm{TG}(H)$;
$N_{g^\star}^\star \leftarrow$ number of $g^\star$-cycles in $\mathrm{TG}(H')$;
$N_{g^\star+2}^\star \leftarrow$ number of $g^\star+2$-cycles in $\mathrm{TG}(H')$;
repeat
  if $r_1^\star \neq r_2^\star$ **then** Replace row $r_2^\star$ in H' with binary sum of rows $r_1^\star$ and $r_2^\star$;
  $r_1^\star \leftarrow 0$, $r_2^\star \leftarrow 0$;
  for $r_1, r_2 = 1, \ldots, n-k$, $r_2 \neq r_1$ **do**
    Replace row $r_2^\star$ in H' with binary sum of rows $r_1^\star$ and $r_2^\star$;
    $g \leftarrow$ girth of $\mathrm{TG}(H')$;
    $N_g \leftarrow$ number of g-cycles in $\mathrm{TG}(H')$;
    $N_{g+2} \leftarrow$ number of $g+2$-cycles in $\mathrm{TG}(H')$;
    if $g < g^\star$ **then**
      $g^\star \leftarrow g$, $r_1^\star \leftarrow r_1$, $r_2^\star \leftarrow r_2$, $N_g^\star \leftarrow N_g$, $N_{g+2}^\star \leftarrow N_{g+2}$;
    end
    else if $N_g < N_g^\star$ **then** $r_1^\star \leftarrow r_1$, $r_2^\star \leftarrow r_2$, $N_g^\star \leftarrow N_g$, $N_{g+2}^\star \leftarrow N_{g+2}$;
    else if $N_g = N_g^\star$ **then**
      if $N_{g+2} < N_{g+2}^\star$ **then** $r_1^\star \leftarrow r_1$, $r_2^\star \leftarrow r_2$, $N_{g+2}^\star \leftarrow N_{g+2}$;
    end
    Undo row replacement;
  end
until $r_1^\star = 0$ & $r_2^\star = 0$;
return H'

Algorithm 2: Tanner graph cycle reduction.

The following subsections describe supporting algorithms required by random redundant decoding.

B. Initial Parity-Check Matrix Selection

As will be demonstrated empirically in Section IV, the performance of the proposed decoding algorithm depends heavily on the choice of parity-check matrix (and thus the Tanner graph) used to represent the code. It is widely accepted that the performance of the decoding algorithms implied by Tanner graphs are adversely affected by short cycles (see for example [7]). Algorithm 2 searches for a suitable parity-check matrix by greedily performing row operations on an input binary parity-check matrix H in order to reduce the number of short cycles contained in the Tanner graph defined by H (denoted $\mathrm{TG}(H)$). Note that after every row operation, the updated partiy check matrix H' defines the same code as H. Note also that the operation of Algorithm 2 requires that short cycles in bipartite graphs can be counted efficiently; such an algorithm was described in [7].

C. Generation of Random Permutation Group Elements

Algorithm 1 requires the efficient generation of random elements of the permutation group of a code. The algorithm presented below, the *product-replacement algorithm*, generates random elements of an arbitrary finite group and is due to Celler *et al.* [8].

Let G be a finite group with generating set:

$$X = \{x_1, x_2, \ldots, x_k\} \tag{3}$$

That is, every element $g \in G$ can be expressed as a finite product:

$$g = x_{i_1}^{n_1} x_{i_2}^{n_2} \cdots x_{i_t}^{n_t} \tag{4}$$

where $x_{i_j} \in X$ and $n_j \in \mathbb{N}$ (the set of natural numbers) for all j. Let $N > k$ be an integer.

Cellar *et al.*'s algorithm constructs a vector $\mathbf{S}$ of length N containing all of the elements of X with repeats. Algorithm 3 describes the basic operation of Cellar *et al.*'s algorithm. Generation of random group elements is initialized by executing Algorithm 3 K times. After initialization, successive executions of Algorithm 3 yield random elements of G. Note that the execution of Algorithm 3 requires only one group multiplication and is thus efficient (permutation multiplication is particularly easy). Also note that after every execution, the group elements contained in $\mathbf{S}$ generate G. Cellar *et al.* found that setting $N = \max(2k+1, 10)$ and $K = 60$ provides near-uniform random generation of group elements in practice.

Input: Length N vector of group elements $\mathbf{S}$.
Output: Random group element g.
Updated group element vector $\mathbf{S}$.

$(i, j) \leftarrow$ pair of random integers
$i \neq j \in \{1, 2, \ldots, N\}$;
$\mathbf{S}[i] \leftarrow \mathbf{S}[j]\mathbf{S}[i]$;
$g \leftarrow \mathbf{S}[i]$;

return g;

Algorithm 3: Random group element generation.

IV. Simulation Results and Discussion

A. The Extended Golay Code

Consider the extended Golay code, $\mathcal{C}_G$, defined by the parity-check matrix of Equation 5.

$$H_G = \begin{bmatrix}
1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1&0&0&1&1\\
1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1&0&0&1\\
0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1&1\\
1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1\\
0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&1\\
1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1\\
0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&1\\
0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&1\\
0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&1\\
0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix} \quad (5)$$

The permutation group of $\mathcal{C}_G$ is generated by four permutations (see Ch. 20 of [5]).

The parity-check matrix of Equation 5 contains many short cycles. The application of Algorithm 2 to this matrix yields:

$$H'_G = \begin{bmatrix}
1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1&0&1&0&0&1&1\\
0&1&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&0&1&0&1&0&0&1\\
1&1&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1\\
0&0&0&1&0&0&1&0&0&0&0&1&1&0&1&0&1&0&0&0&0&1&1&0\\
1&0&0&0&0&1&0&1&1&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&1&0&1&0&1&0&1&0\\
1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&0&0&0&0&0&1\\
0&0&1&1&0&1&0&1&1&0&0&0&1&0&0&1&1&1&1&1&0&1&0&0\\
0&0&0&1&1&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&0&1\\
0&0&1&0&0&0&0&1&0&1&0&0&0&1&0&0&1&0&0&0&1&1&0&1\\
0&0&1&0&1&0&1&1&0&0&1&0&0&0&1&0&0&0&1&1&0&0&0&0\\
1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&0
\end{bmatrix} \quad (6)$$

Whereas the Tanner graphs defined by Equation 5 contains 1551 4-cycles and 65632 6-cycles, the Tanner graph defined by Equation 6 contains 295 4-cycles and 6204 6-cycles. Note that any Tanner graph representing $\mathcal{C}_G$ necessarily contains 4-cycles [9].

Figure 2 compares the performance of five decoding algorithms for $\mathcal{C}_G$: optimal SISO decoding via a trellis; standard iterative decoding using the Tanner graphs implied by H and H' (labeled $\mathrm{TG}(H)$ and $\mathrm{TG}(H')$, respectively); and, the proposed random redundant iterative decoding (RRD) algorithm using H and H' as input parity-check matrices (labeled $\mathrm{RRD}(H)$ and $\mathrm{RRD}(H')$, respectively). One hundred

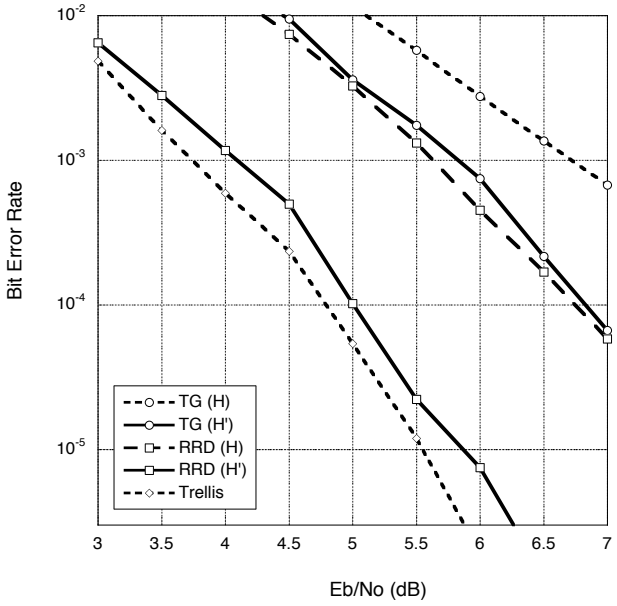

Fig. 2. Bit error rate performance comparison of different decoding algorithms for the $[24, 12, 8]$ extended Golay code.

decoding iterations were performed for the Tanner graph decoders while the RRD algorithms both use input parameter sets: $\alpha_0 = 0.08$, $I_1 = 2$, $I_2 = 30$ and $I_3 = 20$. Flooding message passing schedules were used for all Tanner graph decoders and binary antipodal signaling over additive white Gaussian noise (AWGN) channels is assumed throughout this work.

Note first in Figure 2 that the performance of the RRD algorithm is highly sensitive to the choice of parity-check matrix: the decoder using H' outperforms the decoder using H by approximately 1.75 dB at a bit error rate (BER) of 10^{-4}. It is known that any optimal SISO decoder for the extended Golay code must contain hidden variables with alphabet size at least 256 [10]. Furthermore, there exists a well-known tail-biting trellis for this code which contains 16-ary hidden variables and has near-optimal performance [11]. The RRD algorithm using H', which contains only binary variables, performs approximately 0.3 dB worse than optimal SISO decoding at a bit error rate of 10^{-5}.

B. The $[31, 21, 5]$ and $[63, 39, 9]$ BCH Codes

Following the notation of Lu and Welch [12], define $G_{1,m}$ as the permutation group generated by m elements of the form:

$$\boldsymbol{\sigma}_m^{(j)} = (2, j+2, 2j+2, \ldots, (2^m - 2)j + 2) \quad (7)$$

for $1 \leq j \leq m$ (where each permutation element is taken modulo 2^m). The full permutation group of the $[31, 21, 5]$ BCH code, $\mathcal{C}_5$, is $G_{1,5}$ [12]. The subgroup of cyclic permutations of $\mathcal{C}_5$ is generated by $\boldsymbol{\sigma}_5^{(1)}$ alone and is denoted C_{31}. Similarly, the full permutation group of the $[63, 39, 9]$ BCH code, $\mathcal{C}_6$, is $G_{1,6}$ while the subgroup of cyclic permutations of $\mathcal{C}_6$ is generated by $\boldsymbol{\sigma}_6^{(1)}$ alone and is denoted C_{63}.

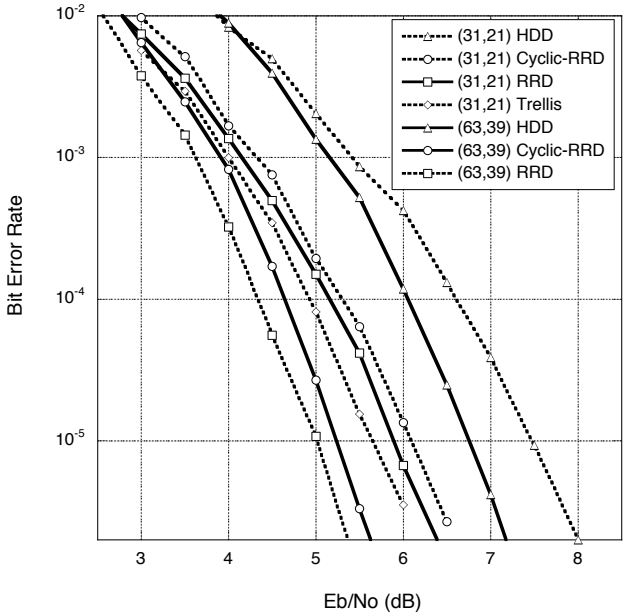

Fig. 3. Bit error rate performance comparison of different decoding algorithms for the $[31, 21, 5]$ and $[63, 39, 9]$ BCH codes.

Figure 3 compares the performance of four decoding algorithms for $\mathcal{C}_5$: optimal SISO decoding via a trellis; random redundant iterative decoding using the full permutation group (labeled RRD); random redundant iterative decoding using only permutations drawn randomly from C_{31} (labeled cyclic-RRD); and algebraic HIHO decoding. Figure 3 also illustrates the performance of the analogous algorithms for $\mathcal{C}_6$ (with the exception of trellis decoding which is prohibitively complex for this code). Note that the cyclic-RRD algorithms are equivalent to Jiang and Narayanan's algorithm [3]. The $\mathcal{C}_5$ RRD algorithms both use input parameter sets: $\alpha_0 = 0.08$, $I_1 = 2$, $I_2 = 30$ and $I_3 = 20$. The $\mathcal{C}_6$ RRD algorithms both use input parameter sets: $\alpha_0 = 0.08$, $I_1 = 2$, $I_2 = 50$ and $I_3 = 20$.

It is known that any optimal SISO decoder for the $[31, 21, 5]$ BCH code must contain hidden variables with alphabet size at least 1024 [13]. Furthermore, it is known that under the standard cyclic bit ordering, the minimal tail-biting trellis for this code also must contain 1024-ary hidden variables [14]. It is thus remarkable that the RRD and cyclic-RRD decoders, which contain only binary variables, perform only 0.25 and 0.5 dB worse than the optimal SISO decoder at a BER of 10^{-5}. Note that the RRD and cyclic-RRD algorithms outperform algebraic HIHO decoding by 1.5 and 1.75 dB at a BER of 10^{-5}. The RRD and cyclic-RRD decoders outperform HIHO decoding by similar margins.

The RRD algorithms outperform the corresponding cyclic-RRD algorithms by approximately 0.25 dB at a BER of 10^{-5} for both $\mathcal{C}_5$ and $\mathcal{C}_6$. The RRD algorithms consider all possible permutations in $\mathrm{Per}\,(\mathcal{C}_5)$ and $\mathrm{Per}\,(\mathcal{C}_6)$, rather than the subset of permutations corresponding to cyclic shifts, and are in some sense more random than the cyclic-RRD algorithms. For certain codes, subgroups of the permutation group may be easier to specify than the full permutation group. For example, the permutation groups of a length 2^m Reed-Muller code is the general affine group $GA(m)$ whose subgroup $GL(m)$ (the general linear group) is generated by two elements. The results of Figure 3 indicate that random redundant iterative decoding algorithms which use such easily specified subgroups may have performance characteristics that, although worse than those RRD algorithms using the full permutation group, are nonetheless interesting.

V. Conclusion

This work introduces random redundant iterative decoding (RRD) which can be viewed as an extension of Jiang and Narayanan's algorithm [3] for cyclic codes to arbitrary block codes. This work also demonstrated how a number of recently proposed iterative SISO decoding algorithms for block codes belong to a common class of algorithms defined by redundant Tanner graphs. It was shown that RRD is an attractive example of a redundant Tanner graph algorithm both in terms of its complexity and its applicability to arbitrary linear block codes. Furthermore, it was demonstrated empirically that the RRD algorithm can outperform Jiang and Narayanan's algorithm when applied to cyclic codes.

References

[1] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University (Sweden), 1996.

[2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Information Theory*, vol. IT-27, pp. 533–547, September 1981.

[3] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed-Solomon codes," *IEEE Communications Letters*, vol. 8, no. 4, pp. 244–246, April 2004.

[4] A. Kothiyal, O. Y. Takeshita, W. Jin, and M. Fossorier, "Iterative reliability-based decoding of linear block codes with adaptive belief propagation," *IEEE Communications Letters*, vol. 9, no. 12, pp. 1067–1069, December 2005.

[5] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1978.

[6] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Communications*, vol. 50, no. 3, pp. 406–414, March 2002.

[7] T. R. Halford and K. M. Chugg, "An algorithm for counting short cycles in bipartite graphs," *IEEE Trans. Information Theory*, vol. 52, no. 1, pp. 287–292, January 2006.

[8] F. Celler, C. R. Leedham-Green, S. H. Murray, A. C. Niemeyer, and E. A. O'Brien, "Generating random elements of a finite group," *Communications in Algebra*, vol. 23, pp. 4931–4948, 1995.

[9] T. R. Halford, A. J. Grant, and K. M. Chugg, "Which codes have 4-cycle-free Tanner graphs?" January 2006, *submitted to ISIT 2006*.

[10] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Information Theory*, vol. 34, no. 5, pp. 1049–1053, September 1988.

[11] A. R. Calderbank, G. D. Forney, Jr., and A. Vardy, "Minimal tail-biting trellises: the Golay code and more," *IEEE Trans. Information Theory*, vol. 45, no. 5, pp. 1435–1455, July 1999.

[12] C.-C. Lu and L. R. Welch, "On automorphism groups of binary primitive BCH codes," in *Proc. IEEE Symposium on Information Theory*, Trondheim, Norway, June 1994, p. 51.

[13] A. Lafourcade and A. Vardy, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Information Theory*, vol. 41, no. 6, pp. 1938–1954, November 1995.

[14] P. Shankar, P. N. A. Kumar, H. Singh, and B. S. Rajan, "Minimal tail-biting trellises for certain cyclic block codes are easy to construct," *Lecture Notes in Computer Science*, vol. 2076, pp. 627–638, 2001.