Multiple Phased-Burst Correcting Superposition Product LDPC Codes

Wai Han Fong
Department of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia 22030, USA
Email: wfong@gmu.edu

Qin Huang
Department of Electrical and Computer Engineering
University of California, Davis
Davis, CA 95616, USA
Email: qinhuang@ucdavis.edu

Shih-Chun Chang
Department of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia 22030, USA
Email: schang@gmu.edu

Shu Lin
Department of Electrical and Computer Engineering
University of California, Davis
Davis, CA 95616, USA
Email: shulin@ucdavis.edu

Abstract—In this paper, a class of product codes based on low-density parity check (LDPC) constituent codes are constructed for multiple phased-burst erasure correction (MPBEC). These codes are shown to correct one large burst and/or a number of shorter bursts. A simple novel recursive erasure correction algorithm is proposed based on a recently discovered zero-span approach to linear block code analysis that can produce very powerful MPBEC capabilities. Analysis and data on how these codes work for additive white Gaussian noise (AWGN) channel are also presented.

Index Terms—Low-density parity-check code, superposition, erasure correction, multiple phased-burst correction, AWGN, BEC, product code, burst correction

I. INTRODUCTION

Low-density parity-check (LDPC) [1], [2] codes are currently the most promising codes to achieve Shannon-capacity for different channels. There is considerable amount of work focusing on LDPC codes, but little of it discusses about LDPC codes for correcting multiple burst errors, a common type of errors occurring in wireless communications and data storage systems. For wireless channels, the error statistics can transition back and forth from a random state to a bursty state. Furthermore, there can be a multiplicity of short duration bursts or it can be one large burst dependent on the type of channel fading. To address this problem, we focus on methods to construct multiple phase burst erasures correction (MPBEC) codes that also can correct one large burst, where a phased-burst is a burst confined in a subblock or phase of the codeword using product codes. As a reminder, erasures are marked locations in the received codeword that identify ambiguously received symbols, i.e. symbols where the logical values cannot be determined.

A product code can be considered a special case of superposition codes where every element of a base matrix is replaced by a constituent matrix that is selected from a set of matrices forming a larger matrix. Recently, researchers found that superposition is also powerful for constructing LDPC codes [4]. In [7], the authors expanded the work of Tai et al. [9] for general linear block code analysis using a concept called zero-spans. This allows for a simple characterization of the burst correction capabilities of a linear codes. In [8], this concept was applied to general cyclic codes to allow optimal burst correction of erasures and errors.

II. PRELIMINARIES

A. Basic Concepts of LDPC codes

Let GF(q) be a Galois field with q elements, where q is a power of a prime. A regular q-ary LDPC code C [3] is defined as the null space of a sparse parity-check matrix H with following structural properties: 1) each row has constant weight ρ; 2) each column has constant weight γ; and 3) H conforms to the row-column-constraint (RCC) defined as no two rows (or columns) have more than one position where they both have nonzero components. If q equals to 2, then it is a binary code; else it is a non-binary code. Property 3) ensures that the Tanner graph [6, pp. 855-858] of the code is cycle-4 free and the minimum distance of the code is at least γ + 1. Suppose the parity-check matrix H is m × n. Then, the null space of the parity-check matrix H gives a regular LDPC code of length n, whose dimension is at least n − m. An LDPC code C is said to be irregular if its parity-check matrix has multiple column weights and/or multiple row weights and conforms to the RCC.

B. Zero-Span Properties of Linear Block Codes

A linear block code is defined as a k-dimensional subspace of an n-dimensional vector space over Galois Field GF(q). In this paper, we focus on binary GF(2) linear block codes. Linear block codes can be defined by a k × n generator matrix G with a null space defined by an m × n parity-check matrix H = [h_{i,j}] where 0 ≤ i < m, 0 ≤ j < n and m ≥ n − k. The rows of this matrix can be described by a distribution of zero-spans. A zero-span is a sequence of consecutive zeros that are bounded by non-zero elements or ones. Specifically, let [b, e], be defined as an ordered pair of column indices of the i^{th} row in h_{i,j} such that h_{i,b}, h_{i,e} ≠ 0 and h_{i,j} = 0 for b + 1 ≤ j < e where b, e ∈ {0, 1, ..., n − 1}. In general, b < e except for the last ordered pair where b = e due to the end around zero-span. Every [b, e], bounds a zero-span of the i^{th} row and can be uniquely identified by either b or e. If b is
chosen, we call that a forward zero-span. If e is chosen, we call that a backward zero-span. Let $\delta_{i,b}^{F} = (e - b)n - 1$ denote a forward zero-span of length $\delta_{i,b}^{F}$ that starts at position $b + 1$ and ends at position $(b + \delta_{i,b}^{F})n$ of the $i^{th}$ row where the operator $(\cdot)_n$ denotes the modulo $n$ operation and is used to facilitate end around spans. We can now define the zero-covering-span as:

$$\delta = \min_{b}(\max_{i} \delta_{i,b}^{F}).$$

This quantity specifies the guaranteed single burst erasure correcting capability for a linear block code using a binary erasure recursion algorithm [7] [10, pp. 570-572] described in below.

C. Erasure Recursion Algorithm

The erasure recursion algorithm (ERA) is defined as follows: let $h_{i,j}$ represent the components of the parity-check matrix $H$ where $0 \leq i < m$, $0 \leq j < n$ and a zero covering-span $\delta$. Let $y_j$ represent the $j^{th}$ symbol of the received codeword. Let $K = \{j | y_j = ?\}$ where ? is a symbol for a received erasure be a set of received erasure symbol positions in the codeword. By definition of the parity check matrix,

$$\sum_{j} y_j h_{i,j} = 0$$

for any row at index $i$. We call a column at $j$ (or bit position) checked by a row at $i$, if and only if $h_{i,j} \neq 0$. If there exists a row at index $t$ where only one element $k \in K$ is checked then by (2), then:

$$y_k = \sum_{j \neq k} y_j h_{t,j}$$

Now the decoding algorithm can be defined. Identify a minimal interval $l$ where all elements of $K$ are contained. If $l$ is of length $\delta + 1$ or less, then: 1) let $k$ be the first element in $l$, 2) identify a row at $t$ in $H$ that has a forward zero-span $\delta^{F} \geq \delta$ at column $k$, (the existence of this row is guaranteed by (1)), 3) correct erasure at $k$ using (3) and remove this element from $K$, 4) let the position of next element in $K$ be $k$, and 5) repeat steps 2-4 until all elements $K$ are corrected. Note: in step 2, $\delta \geq l - 1$ guarantees that no erasures are included in (3) since the multiplication $0 \cdot ? = 0$ will mask out all of the erasures in $K$ except at position $k$. Therefore (3) becomes one equation solving for the one unknown at $k$.

III. MULTIPLE PHASE BURST CORRECTION PRODUCT CODES

We now propose a class of MPBEC LDPC product codes using a novel recursive erasure correction algorithm.

A. Product Code Definition

Definition: For $t = 1$ and 2, let $C_{t}(n_{t}, k_{t}, \delta_{t})$ denote a component linear block code of length $n_{t}$, dimension $k_{t}$ and zero-covering-span $\delta_{t}$ of a two-dimensional product code $C_{P}$.

Let $G_{t} = \left[ \begin{array}{c} g_{t,i,j}^{(1)} \end{array} \right]$ be the generator matrix of $C_{t}$. For a two dimensional Product Code $C_{1} \times C_{2}$ with component codes $C_{1}$ and $C_{2}$, the product code generator matrix $G_{P}$ of dimension $k_{1}k_{2} \times n_{1}n_{2}$ can be defined as [5] :

$$G_{P} = G_{1} \otimes G_{2} = \left( G_{1}g_{i,j}^{(2)} \right)$$

where $\otimes$ is the Kronecker Product. If the order of $C_{1}$ and $C_{2}$ are switched, (4) becomes:

$$G_{P}^{*} = G_{2} \otimes G_{1} = \left( G_{2}g_{i,j}^{(1)} \right).$$

Note that $G_{P}$ and $G_{P}^{*}$ are combinatorially equivalent, i.e. the codeword of one is the permuted version of the other. The parity check matrix of a two dimensional product code can be defined as:

$$H_{P} = \left[ \begin{array}{c} H_{1} \otimes I_{n_{2}} \\ \vdots \\ I_{n_{1}} \otimes H_{2} \\ \end{array} \right]_{(2n_{1}n_{2} - n_{1}k_{2} - n_{2}k_{1}) \times n_{1}n_{2}}$$

where $H_{1}$, $H_{2}$ are parity check matrices of the component code $C_{1}$, $C_{2}$ respectively and $I_{n_{1}}$, $I_{n_{2}}$ are $n_{1} \times n_{1}$ and $n_{2} \times n_{2}$ identity matrices.

B. Parity Check Matrix Structure

Equation (6) defines the parity check matrix of a two-dimensional product code that can be partitioned into two submatrices:

$$H_{P} = \left[ \begin{array}{c|c} \begin{array}{ccc} H_{1} & 0 & \ldots \\ 0 & H_{1} & \ldots \\ \vdots & \vdots & \vdots \\ 0 & 0 & \ldots \\ \end{array} \\ \begin{array}{ccc} h_{0,0}^{(2)}I_{n_{1}} & h_{0,1}^{(2)}I_{n_{1}} & \ldots \\ h_{1,0}^{(2)}I_{n_{1}} & h_{1,1}^{(2)}I_{n_{1}} & \ldots \\ \vdots & \vdots & \vdots \\ h_{m_{2}-1,0}^{(2)}I_{n_{1}} & h_{m_{2}-1,1}^{(2)}I_{n_{1}} & \ldots \\ \end{array} \\ \end{array} \right]$$

where $m_{2} = n_{2} - k_{2}$.

Let’s define the submatrix $A_{1}$:

$$A_{1} = H_{1} \otimes I_{n_{2}} = \left[ \begin{array}{c|c} \begin{array}{ccc} H_{1} & 0 & \ldots \\ 0 & H_{1} & \ldots \\ \vdots & \vdots & \vdots \\ 0 & 0 & \ldots \\ \end{array} \right]$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings.
and submatrix $A_2$: 

$$A_2 = I_{n_1} \otimes H_2 = \begin{bmatrix} h_{0,0}^{(2)} I_{n_1} & h_{0,1}^{(2)} I_{n_1} & \cdots & h_{0,n_2-1}^{(2)} I_{n_1} \\ h_{1,0}^{(2)} I_{n_1} & h_{1,1}^{(2)} I_{n_1} & \cdots & h_{1,n_2-1}^{(2)} I_{n_1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n_2-1,0}^{(2)} I_{n_1} & h_{n_2-1,1}^{(2)} I_{n_1} & \cdots & h_{n_2-1,n_2-1}^{(2)} I_{n_1} \end{bmatrix}. \tag{9}$$

The $i$th row of $A_2$ is:

$$A_i^{(2)} = \begin{bmatrix} h_{i,0}^{(2)} I_{n_1} & h_{i,1}^{(2)} I_{n_1} & \cdots & h_{i,n_2-1}^{(2)} I_{n_1} \end{bmatrix}. \tag{10}$$

This can be written as a partitioned $n_1 \times n_1 n_2$ submatrix (11).

Let a column-phase of $H_P$ be defined as an interval of column indices $[i(n_1), (i+1)n_1 - 1]$ of length $n_1$ where $0 \leq i < n_2$. Also, let a row-phase of $A_1$ be defined as an interval of row indices $[j(m_1), (j+1)m_1 - 1]$ of length $m_1 = n_1 - k_1$ where $0 \leq j < n_2$ and also let a row-phase of $A_2$ be defined as an interval of row indices $[k(n_1), (k+1)n_1 - 1]$ of length $n_1$ where $0 \leq k < n_2 = n_2 - k_2$. That is $A_1$ consists of $n_2$ row-phases of length $m_1$ and $A_2$ consists of $m_2$ row-phases of length $n_1$. We offer the following theorems without proof (omitted).

**Theorem 1**: If $H_1$ and $H_2$ have the RCC property then $H_P$ also conforms to the RCC.

**Theorem 2**: If $H_1$ and $H_2$ are irregular LDPC codes of minimum column weights $\gamma_{min1}$ and $\gamma_{min2}$ respectively, then $H_P$ is an irregular LDPC code whose minimum column weight is $\gamma_{minP} = \gamma_{min1} + \gamma_{min2}$.

As stated before, the minimum distance of a regular LDPC code of column weight $\gamma$ is at least $\gamma + 1$. From Theorem 2, we see that this bound is rather weak since we know that if the minimum distance of the component codes $H_1$ and $H_2$ are $d_{min1}$ and $d_{min2}$ respectively, then the minimum distance of the product code is $d_{minP} = d_{min1}d_{min2}$. However, applying Theorem 2, under the special case of two regular LDPC component codes, gives $d_{minP} \geq \gamma_1 + \gamma_2 + 1$.

**C. Multiple Burst Erasure Correcting Capability**

To find the multiple burst erasure correcting capability of a product code using a recursive erasure decoder, we first prove the following lemma.

**Lemma 1**: The Kronecker product of $I_{n_1} \otimes H_2$, where $I_{n_1}$ is an identity matrix of dimension $n_1 \times n_1$ and $H_2$ is an $m_2 \times n_2$ matrix with a zero-covering-span of $\delta_2$, produces an $n_1 n_2 \times n_1 n_2$ matrix that has a zero-covering-span of $\delta = n_1(\delta_2 + 1) - 1$.

**Proof**: The zero-spans in (11) are defined by the non-zero elements of $h_{i,j}^{(2)}$ where $i \in \{0, 1, \ldots, m_2 - 1\}$ and $0 \leq j < n_2$. That is for every $[b, e]$ ordered pair of column indices of the $i$th row in $H_2$ where $b, e \in \{0, 1, \ldots, n_2 - 1\}$, there is an associated set of $n_1$ ordered pairs, $\{[B, E]_i\}$, of non-zero element column positions in (11) where $B = bn_1 + u, E = en_1 + u$ and $B, E \in \{0, 1, \ldots, n_2 n_2 - 1\}$ with $0 \leq u < n_1$ as the relative row index offset of the $i$th submatrix. Therefore, (11) has forward zero-spans that are of length:

$$\delta_i^{F} = \begin{cases} (E - B)_{n_1 n_2 - 1} - (e - b)_{n_1 n_2 - 1} - 1 & \text{if } (e - b)_{n_1 n_2 - 1} > 0 \\ (E - B)_{n_1 n_2 - 1} - (e - b)_{n_1 n_2 - 1} - (\delta_i^{F})_{b} + 1 & \text{if } (e - b)_{n_1 n_2 - 1} = 0 \end{cases} \tag{12}$$

To find the zero-covering-span, we minimize over all $B$, the maximum of (12) over all $i$ (submatrices of (9)):

$$\delta = \min\left(\max\left(\delta_i^{F}\right)\right) = \min\left(\max\left(\delta_i^{F} + 1\right)\right) - 1. \tag{13}$$

The final step in (13) comes is the recognition that a minimization over $B$ is a minimization over $b$ since (12) says that $\delta_i^{F}$ is only dependent on $b$.

**Theorem 3**: A two dimensional product code, $C_P$, with component codes $C_1(n_1, k_1, \delta_1), C_2(n_2, k_2, \delta_2)$, can correct a single burst of size $b_{A_2} = n_1(\delta_1 + 1)$ or a multiple erasure burst correcting capability of $b_{A_1} = n_2 \times (\delta_1 + 1)$ using a recursive erasure decoder.

**Proof**: (6) defines the parity check matrix of a two-dimensional product code which can be partitioned into two submatrices shown in (7) where $A_1$ and $A_2$ are defined in (8) and (9) respectively. If $H_1$ has a zero-covering-span of $\delta_1$ and $A_1$ acts on $n_2$ codewords of $C_1$, then $A_1$ can correct multiple erasure burst confined in each codeword of up to $b_1 = \delta_1 + 1$ symbols or $b_{A_1} = n_2 \times (\delta_1 + 1)$ symbols. $A_2$ is formed by replacing every non-zero element of $H_2$ by an $I_{n_1}$ identity matrix which has a zero-span of $n_1 - 1$ and every zero element of $H_2$ is replaced by an all zero $n_1 \times n_1$ matrix, $0_{n_1 \times n_1}$, which has a zero-covering-span of $n_1$. Lemma 1 says that a Kronecker product of $I_{n_2} \otimes H_2$ results in an overall zero-covering-span of $n_1(\delta_2 + 1) - 1$. Therefore, if $H_2$ has a zero-covering-span of $\delta_2$ then $A_2$ has a zero-covering-span of $\delta_{A_2} = n_1 \delta_2 + n_1 - 1$ and can correct any erasure burst up to length, $b_{A_2} = \delta_{A_2} + 1 = n_1(\delta_2 + 1)$.

Theorem 3 can be interpreted to say that the guaranteed largest number of erasures that the product code can correct using a recursive erasure decoder is $b_{A_1} = n_2(\delta_1 + 1)$ or $b_{A_2} = n_1(\delta_2 + 1)$. To achieve this performance, an additional requirement while using $A_1$ to decode is that the erasure bursts need to be separated by a guard band of $n_1 - \delta_1 - 1$.

**IV. Decoding Algorithms**

To document a possible decoding algorithm, some terminology must be defined. We define a constrained burst in submatrix $A_1$ as an erasure burst that is within the erasure correcting capability $\delta_1 + 1$ of the $t$th component code where $t \in \{1, 2\}$. That is, within the boundary of a component codeword, there is at least $n_t - \delta_t - 1$ consecutive bits (including end around)
with no erasures. For instance, in (8) there are \( n_2 \) component codewords from \( H_1 \) of length \( n_1 \) bits where each component codeword can correct \( \delta_1 + 1 \) erased bits and must contain a string of at least \( n_1 - \delta_1 - 1 \) consecutive bits with no erasures. A **phased-burst** (PB) in submatrix \( A_2 \) is a correctable burst defined by the length \( b_{A_2} = n_1(\delta_2 + 1) \) that is contained within the bit interval \( I_{PB}^j = [jn_1, ((j + \delta_2 + 1)n_1)n_1n_2] \) where \( j \in \{0, 1, \ldots, n_2 - 1\} \).

\[ A_i^{(2)} = \begin{bmatrix} h_{i,0}^{(2)} & 0 & \ldots & 0 & h_{i,1}^{(2)} & 0 & \ldots & 0 & \cdots & h_{i,n_2-1}^{(2)} & 0 & \ldots & 0 \\ 0 & h_{i,0}^{(2)} & \ldots & 0 & 0 & h_{i,1}^{(2)} & \ldots & 0 & \cdots & 0 & h_{i,n_2-1}^{(2)} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & h_{i,0}^{(2)} & 0 & 0 & \ldots & h_{i,1}^{(2)} & \cdots & 0 & 0 & \ldots & h_{i,n_2-1}^{(2)} \end{bmatrix} \] (11)

A. Product Code Multiple Burst Erasure Decoding Algorithm

1) Determine if received codeword has any erasures. If none, then output received codeword and stop.

2) Else, if the erasures can be characterized as a phased-burst in the interval \( I_{PB}^j \) where \( j \in \{0, 1, \ldots, n_2 - 1\} \), use submatrix \( A_2 \) with the ERA to correct all bits, output corrected codeword and stop.

3) Else, if the erasures can be characterized as \( n_2 \) constrained bursts of length \( \delta_1 + 1 \) bits, use submatrix \( A_1 \) with ERA to correct all bits, output corrected codeword and stop.

4) Else, interleave codeword by switching \( G_1 \) and \( G_2 \) position in (4) to (5) thus \( A_1 \Rightarrow H_2 \otimes I_n \), and \( A_2 \Rightarrow I_{n_2} \otimes H_1 \).

5) If the erasures that can be characterized as a phased-burst in the interval \( I_{PB}^{k} = [kn_2, ((k + \delta_1 + 1)n_2)n_1n_2] \) where \( k \in \{0, 1, \ldots, n_1 - 1\} \), use submatrix \( A_2 \) with the ERA to correct all bits, output corrected codeword and stop.

6) Else, if the erasures can be characterized as \( n_1 \) constrained bursts of length \( \delta_2 + 1 \) bits, use submatrix \( A_1 \) with ERA to correct all bits, output corrected codeword and stop.

7) Otherwise, declare decoder failure and stop.

B. Product Code LDPC Decoding Algorithm for AWGN

Our goal is to develop codes that can operate well in both burst and random channels, i.e., AWGN. Theorem 1 shows that \( H_P \) can be used directly in a SPA decoder. However, \( H_P \) has a number of redundant rows. The number of independent rows for any parity-check matrix of \( C_P \) should be \( n_1n_2 - k_1k_2 \) but the number of rows in \( H_P \) is \( m_P = 2n_1n_2 - n_1k_2 - n_2k_1 \). The difference of these values shows that the number of redundant rows is \( (n_1 - k_1)(n_2 - k_2) \). This is precisely the number of *checks on checks* which are the parity checks bits of one component code based entirely on the parity bits of the other component code [6]. This redundancy gives us the structure of \( H_P \). We do have flexibility in deciding whether to puncture the *checks on checks* bits. If we choose to do so, the minimum distance of this *incomplete product code* will be reduced to \( d_{minP} = d_{min1} + d_{min2} - 1 \) however the code rate will improve.

1) Two-Stage LDPC Product code decoder: We propose another product code decoder for the AWGN channel based on component LDPC decoders that can be described as follows: a \( C_1 \) SPA decoder accepts, as priors, log-likelihood ratio channel inputs and produces an estimate a posteriori probability (APP) measure after one (or more) complete \( C_1 \) component decode. Then the APP measure is re-ordered so that \( C_2 \) codewords are formed sequentially. A \( C_2 \) SPA decoder accepts the reordered APP measure and produces an updated APP measure based on one (or more) complete \( C_2 \) component decode. This update gets re-ordered back to the original form, i.e. sequential \( C_1 \) codewords. Then a syndrome calculation is performed to check for a valid codeword or if a maximum number of iterations has been reached. If this condition has occurred then output the estimated codeword, else start (or iterate) the process over again with the reordered updated APP measures from \( C_2 \) decoder as priors to the \( C_1 \) decoder. The advantage of this decoder is that it uses much less resources, since the codeword is composed of multiple smaller codewords that can be decoded with one single implementation. Therefore the complexity is reduced by a fraction of the total code length.

Our initial simulations were performed with 3 types of component linear block codes: 1) a code rate \( r = 0.686 \) Euclidean Geometry \( C_{EG}^{(1)}(2, 0, 4)(255, 175) \) LDPC code [6], 2) a code rate \( r = 0.684 \) Mackay(253, 173) LDPC code of column weight 3 [2], and 3) a code rate \( r = 0.984 \) (64,63) single parity check code (SPC) in AWGN. The SPC code consists of a single row of 64 ones for the parity check matrix. Although it is not strictly an LDPC code since there are only non-zero components, it does not have any cycles but does have a low minimum distance of 2. We use the SPC code as a second component code to the EG or Mackay code to keep the product code rate as high as possible while keeping the overall length of the code around 16K bits.

First, the EG and Mackay codes by themselves are simulated with the SPA to establish a performance baseline. Fig.1 is
a plot of the bit error rate (BER) and the block error rate (BLER). We now use these codes in the product LDPC decoder as described above shown in Fig. 2. This shows a Mackay×Mackay (64009, 29929) product LDPC code at a code rate of $r = 0.468$ and the (65025, 30625) EG×EG product code with a rate of $r = 0.471$. The results indicate that the EG×EG code outperforms the Mackay×Mackay after 3.4 dB where it provides near-error free performance. Fig. 2 also shows the Mackay×SPC rate $r = 0.673$ (16192, 10899) product code and the EG×SPC rate $r = 0.676$ (16320, 11025) product code. The EG×SPC clearly outperforms the Mackay×SPC code. Although these results do not show that product LDPC can approach the Shannon limit, it’s our belief that a properly chosen set of component codes that have a combined degree distribution will produce better results. However, some degrading of the burst correction performance is to be expected.

```latex
\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{EG and Mackay LDPC Codes with SPA in AWGN Channel}
\end{figure}
```

According to Theorem 3, any code can improve its burst correction by its use in a product code resulting in a single burst erasure correction capability of $b_{A_2} = \eta_1 (\delta_2 + 1)$ or a multiple erasure burst correcting capability of $b_{A_1} = \eta_2 \times (\delta_1 + 1)$. Since $\delta = 55$ for EG(255,175), the EG×EG product code can correct a large erasure burst totaling $b_{A_2} = 255(55 + 1) = 14,288$ bits or $b_{A_1} = 255$ multiple bursts of length 56 bits. Also, since $\delta = 0$ for the SPC, the EG×SPC product code can correct one large burst of $b_{A_2} = 255(0 + 1) = 255$ bits or $b_{A_1} = 64$ multiple bursts of length $(55 + 1) = 56$ bits and if $G_P$ is switched to $G_P^*$, then one large burst of $b_{A_2} = 64(55 + 1) = 3,584$ bits or $b_{A_1} = 255$ multiple bursts of length $(0 + 1) = 1$ bit can be corrected. Therefore, in addition to good random error correction in an AWGN channel, these codes can easily correct a large single burst or multiple burst erasures.

```latex
\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{EG×EG, EG×SPC, Mackay×Mackay, Mackay×SPC in AWGN Channel}
\end{figure}
```

V. Conclusion

The paper presents a new application of LDPC product codes for multiple phased-burst erasures correcting which we show to have powerful burst erasure correction performance. We also demonstrate how the code can be used for an AWGN channel using a simple two stage product code decoder. It demonstrates that a single code can be designed to work on a multi-state channel, i.e. a multiple burst erasures state, a single large burst erasure state and a random AWGN error state. More work is needed to optimize the component code choice.

REFERENCES


