Construction of Burst-Erasure Efficient LDPC Codes for Use with Belief Propagation Decoding

Kan Li
University of Hawaii at Manoa
Honolulu, HI 96822 USA
Email: likan@hawaii.edu

Aleksandar Kavčić
University of Hawaii at Manoa
Honolulu, HI 96822 USA
Email: kavcic@hawaii.edu

M. Fatih Erden
Seagate Technology
Minneapolis, MN 55435 USA
Email: Fatih.Erden@seagate.com

***Abstract*—In this paper, we consider the optimal permutation of encoded symbols that will result in the maximum burst erasure efficiency for a given low-density parity-check (LDPC) code under belief propagation (BP) decoding. A novel algorithm is proposed to permute columns of the parity-check matrix to achieve high burst erasure correction efficiencies. This method can be applied to any linear code to increase its burst-erasure efficiency while retaining its random-error correction capability. Results show that the algorithm outperforms previous methods.**

I. Introduction

Burst erasure correction plays a significant role in error control coding of communications and storage systems. In particular, for magnetic recording, a prevalent phenomenon is thermal asperity (TA) [1]. During the reading process, the readback head may heat up for a period of time, producing a string of intense transient noise in the readback waveform [1]. There has been extensive research on combating the effects of TA. These efforts cover high-pass filtering using envelope detection [2], subtractive restoration using fixed signal pulse-cancellation [3], timing and gain correction [4], and TA suppression through equalization [5]. Our focus in this paper will be, however, from a coding perspective. We can model the TA occurrence using a classic bursty erasure channel. This idealized channel has only two modes: a bit is either received correctly or erased, and erasures come in bursts [6].

For an (n, k) binary linear block code of length n and dimension k, the theoretical limit for the longest erasure burst length guaranteed to be correctable at all possible positions of a codeword is $n - k$. In [7] it was shown that for practically any (n, k) linear block code, this limit can be achieved using back-substitution on a new, dense $n \times n$ parity-check matrix in the *burst correction form*, $\mathbf{H}_{BCF}$. The major drawback of this approach is that it takes significantly more memory to store such a matrix than the original $(n-k) \times n$ parity-check matrix $\mathbf{H}$, especially when the matrix $\mathbf{H}$ is sparse, such as in low-density parity-check (LDPC) codes.

An alternative to constructing $\mathbf{H}_{BCF}$ is to start with a code judiciously constructed to achieve good erasure correction performance [8]. However, the resilience of such a code to random errors (resulting from, say, additive noise) may be compromised. The ultimate goal is to have a code that is resilient to both burst erasures and random errors.

In this paper we start with a parity-check matrix $\mathbf{H}$ (typically an LDPC matrix) that is constructed to be resilient to random errors resulting from additive noise. We then propose a search algorithm for finding a permuted parity-check matrix $\mathbf{H}'$ that outperforms the original matrix $\mathbf{H}$ in terms of burst erasure correction capability. The permuted parity-check matrix $\mathbf{H}'$ preserves all the random error correcting properties of $\mathbf{H}$, without introducing major drawbacks.

Organization: Definitions are given in Section II. Section III introduces the optimization problem, our design approach, and a new permutation algorithm that is contrasted to previously published methods. In Section IV, we compare the performances of the different algorithms. Section V concludes the paper.

II. Burst Erasure Correction of Linear Codes

A. Burst Erasure Performance

We consider the burst erasure channel (BuEC) as defined in [6]. The transmitter sends n binary symbols over the channel, but at the receiver, the received symbols r_i (where $0 \leq i \leq n-1$) can arrive erased. Furthermore, we assume that the erasures occur in a single burst. L is said to be a correctable burst erasure length if there exists a decoder that can correct all L consecutive erased symbols, regardless of the burst location in the codeword. Let $\mathbf{H}$ be an $(n-k) \times n$ parity-check matrix, representing an (n, k) binary linear block code $\mathcal{C}$ of length n and rate $\frac{k}{n}$. We would like to characterize the burst erasure performance of a given code through the following definitions.

Definition 1. *Let $\mathbb{L}(\boldsymbol{H})$ denote the **set of correctable burst erasure lengths** for a code represented by the parity-check matrix $\boldsymbol{H}$, i.e.*

$$\mathbb{L}(\boldsymbol{H}) \triangleq \{L \mid L \text{ is a correctable burst erasure length}\}. \quad (1)$$

Definition 2. *The **burst erasure capability** of the code represented by the parity-check matrix $\boldsymbol{H}$ is*

$$L^{(max)}(\boldsymbol{H}) \triangleq \max\{\mathbb{L}(\boldsymbol{H})\}. \quad (2)$$

Since $L^{(max)}(\mathbf{H})$ is upper-bounded by $n-k$, we define the burst erasure efficiency as the following ratio.

Definition 3. *The **burst erasure efficiency** of a code whose parity-check matrix is $\boldsymbol{H}$ is defined as*

$$\eta(\boldsymbol{H}) \triangleq \frac{L^{(max)}(\boldsymbol{H})}{n-k}. \quad (3)$$

Note: In Definition 3, we do not specify the decoder, hence $\eta(\mathbf{H})$ is a property of the code (not the decoder).

Since in this work we assume that $\mathbf{H}$ is given and fixed, without loss of generality, we can write $\eta \triangleq \eta(\mathbf{H})$. Similarly, we can extend this shorthand to $L^{(max)} \triangleq L^{(max)}(\mathbf{H})$.

Let π denote a permutation of the n elements (of the codeword). Then $\mathbf{H}_\pi \triangleq \pi(\mathbf{H})$ is a new parity-check matrix whose columns are obtained by rearranging the columns of $\mathbf{H}$ using the permutation π. Clearly, $L^{(max)}(\mathbf{H})$ and $L^{(max)}(\mathbf{H}_\pi)$ may not be equal. Naturally, it is desirable to find the permutation π that maximizes the burst erasure efficiency.

Definition 4. *The **burst erasure efficiency of the code represented by the permuted parity-check matrix $\boldsymbol{H}_\pi$** will be shortly denoted as*

$$\eta(\pi) \triangleq \eta(\boldsymbol{H}_\pi) \triangleq \frac{L^{(max)}(\boldsymbol{H}_\pi)}{n-k}. \tag{4}$$

This leads to the following optimization problem.

Problem 1. *Find an optimal permutation π^* such that*

$$\pi^* = \arg\max_{\pi} \eta(\pi). \tag{5}$$

Problem 1 was solved by Fossorier [7], who showed that (under some fairly mild assumptions on $\mathbf{H}$) we have $\eta(\pi^*) \triangleq \eta(\mathbf{H}_{\pi^*}) = 100\%$. However, there are two major drawbacks associated with the solution in [7]:

1) The solution requires forming an $n \times n$ high-density parity-check matrix $\mathbf{H}_{BCF}$ (known as the burst correction form, BCF, for erasures) and performing back-substitution on $\mathbf{H}_{BCF}$.
2) The memory cost for storing $\mathbf{H}_{BCF}$ is high.

B. Burst Erasure Performance of the BP Decoder

LDPC codes are generally desirable because the belief propagation (BP) decoder is extremely simple and efficient in correcting errors induced by the channel. The Tanner graph representation of the code and its associated BP decoding algorithm are presented in [9] and [10]. It was shown in [11] and [12] that irregular LDPC codes are capacity achieving for the binary erasure channel (BEC). Furthermore, the optimal code design for the BEC is also optimal for the BuEC since column permutations of the parity-check matrix preserve the optimal degree distributions, and only serve to disperse the erasures in an erasure burst such that the BuEC becomes a BEC [13]. It is therefore desirable to characterize the burst erasure capability of the LDPC codes under BP. However, the burst erasure efficiency of the BP decoder is generally different from $\eta(\mathbf{H})$, and we need to define it properly.

Definition 5. $\mathbb{L}_{BP}(\boldsymbol{H})$ *is the **set of BP-correctable burst erasure lengths** for a code whose parity-check matrix is $\boldsymbol{H}$:*

$$\mathbb{L}_{BP}(\boldsymbol{H}) \triangleq \{\mathtt{L} \mid \mathtt{L} \textit{ is a BP-correctable burst length}\}. \tag{6}$$

Definition 6. *The **burst erasure capability of the BP decoder** applied to the parity-check matrix $\boldsymbol{H}$ is*

$$L_{BP}^{(max)}(\boldsymbol{H}) \triangleq \max\{\mathbb{L}_{BP}(\boldsymbol{H})\}. \tag{7}$$

Definition 7. *The **burst erasure efficiency of the BP decoder applied to the permuted parity-check matrix** $\pi(\boldsymbol{H})$ is*

$$\eta_{BP}(\pi) \triangleq \eta_{BP}(\boldsymbol{H}_\pi) \triangleq \frac{L_{BP}^{(max)}(\boldsymbol{H}_\pi)}{n-k}. \tag{8}$$

Clearly,

$$\eta_{BP}(\pi) \le \eta(\pi) \le \eta(\pi^*) = 1. \tag{9}$$

C. Burst Erasure Performance Calculation under BP

Finding $L_{BP}^{(max)}(\mathbf{H})$ means to find the maximal BP-decodable erasure length. A brute-force method to test whether an erasure length L (note $1 \le L \le n-k$) is BP-decodable, is to test whether the BP decoder can correct any burst of length L occurring at any location i (here $0 \le i \le n-L$). However, the following lemma vastly simplifies the test (note: Lemma 1 was already used, though not explicitly stated, in [13]).

Lemma 1. *Performing message-passing decoding on a classic bursty channel is equivalent to back-substitution.* □

Proof: If only one bit in a parity-check equation is erased, then the back-substitution method can correct the erased bit. If 2 or more bits are erased in a parity-check equation, then the back-substitution method cannot correct the erased bit. Hence, we only need to show that the belief propagation decoder emulates these two cases. Without loss of generality, we can assume that the transmitted codeword is $\underline{c} = \underline{0}$.

Case1: Only one bit in a parity-check equation is erased. In one iteration, the BP decoder passes a log-likelihood message of ∞ from that check node to the erased variable node. Since the message ∞ implies that the corresponding variable (bit) decodes to zero, we have corrected the erased bit.

Case2: Two or more bits in a parity-check equation are erased. In this case the BP decoder passes zero-valued log-likelihoods from that check node to all erased variable nodes. Hence, none of the bits corresponding to erased variable nodes can be corrected. ■

Using back-substitution provides an efficient alternative to using the full message-passing decoder, when computing $L_{BP}^{(max)}(\mathbf{H})$. A detailed algorithm for the computation of $L_{BP}^{(max)}(\mathbf{H})$, from the above observations, was given in [13].

Equivalently, the burst erasure efficiency of an LDPC code under BP decoding can be viewed as a property of certain subsets of the variable nodes of the corresponding Tanner graph called stopping sets [14]. A set of variable nodes $\mathcal{S} = \{v_{i_1}, v_{i_2}, \cdots, v_{i_s}\}$, with $i_1 < i_2 < \cdots < i_s$, forms a stopping set if every check node connected to this set has degree 2 or more in $\mathcal{S}$. The *span* is $i_s - i_1 + 1$. Clearly, the *minimum stopping set span* of a parity-check matrix is $\mathrm{Span}(\mathbf{H}) = L_{BP}^{(max)} + 1$.

III. Optimization Problem and Algorithm

The objective of this paper can now be explicitly expressed by the following optimization problem. Consider an (n, k) LDPC code $\mathcal{C}$ with a parity-check matrix $\mathbf{H}$.

Algorithm 1: Greedy Searching and Swapping (GSS)

- Set $\mathbf{H}' := \mathbf{H}$
- Set burst erasure length $L := n - k$

for $j = 0$ *to* $n-1$ **do** †
- $i := \max(0, j - L + 1)$, here $i(j)$ is the $start(end)$ burst location
- $L' := \min(j + 1, L)$

if *the BP decoder*‡ *cannot correct the erasure burst of length* L' *at location* i **then**

while *all valid swaps have not been exhausted* **do**
- pick p, such that $p \notin \{i, i+1, \cdots, j\}$
- In $\mathbf{H}'$, swap columns $\underline{h}'_j$ and $\underline{h}'_p$
- $\ell := \max(0, p - L + 1)$

if *the BP decoder*‡ *can correct the erasure burst of length* L' *at location* i **then**

if $p > j$ **then**
- break out of the **while** loop

else if $(p < i)$ *the BP decoder*‡ *can also correct erasure bursts of length* L *at any previous locations* $\{\ell, \ell+1, \cdots, p\}$ *affected by the swap* **then**
- break out of the **while** loop

end

end
- In $\mathbf{H}'$, unswap columns $\underline{h}'_j$ and $\underline{h}'_p$

if *all valid swaps have been exhausted* **then**
- $L := L - 1$
- reset all valid swaps

end

end

end

end

- Set $L_{BP}^{(max)}(\mathbf{H}') := L$ and **exit** the algorithm

† Note that in each iteration of the **for** loop, at most one swap is kept. All other swaps are for testing purposes only.

‡ Lemma 1 is used to simplify the process of checking whether the BP decoder can correct an erasure burst.

Problem 2. *Find an optimal permutation* π^*_{BP} *such that*

$$\pi^*_{BP} = \arg\max_{\pi} \eta_{BP}(\pi) = \arg\max_{\pi} \mathrm{Span}(\boldsymbol{H}_\pi). \tag{10}$$

Problem 2 is very difficult though it has a direct solution, namely, enumerating $L_{BP}^{(max)}$ for the entire search space of $n!$ possible permutations. However, this brute-force approach is computationally prohibitive. With a few exceptions, problems of this kind are generally NP-hard. This has motivated authors to seek heuristic approaches to attack the problem. The pivot searching and swapping (PSS) method [15] and the recent simulated annealing (SA) approach [16] all produced good results. We present a novel approach that outperforms these two algorithms.

A. Algorithm Description

Our approach is similar to [15] in the sense that we wish to redistribute the stopping sets in a given parity-check matrix. Our algorithm receives, as input, a specific LDPC matrix $\mathbf{H}$ with burst erasure efficiency $\eta_{BP} \triangleq \eta_{BP}(\mathbf{H})$ and returns, as output, a new column-permuted LDPC matrix $\mathbf{H}'$ with $\eta'_{BP} \triangleq \eta_{BP}(\mathbf{H}') \geq \eta_{BP}$. This is achieved through a series of intermediate pair-wise column swaps, or variable-node permutations (swaps). The key observation here is that it is not only prohibitive to test all $n!$ possible permutations, but also unnecessary. If a stopping set with suboptimal span is formed anywhere in a permuted matrix, we can immediately reject not only this particular swap, but also all permutations sharing the same stopping set. Conversely, if we are to construct the optimally efficient parity-check matrix using the available variable nodes as building blocks, at each step of the way, the optimal efficiency must be maintained. Any group of variable nodes that forms a stopping set with suboptimal span cannot belong to the optimal construction, and any permutation containing the group should be eliminated from the search space. In this regard, our method can be viewed as a pruning process or a greedy algorithm. The idea is to optimize the input matrix column by column, from left to right (the direction is subjective), and reject or swap out any variable node (problem column) that forms a suboptimal stopping set ending with this particular node. The method is outlined in Algorithm 1: Greedy Searching and Swapping (GSS). To better describe the algorithm, we first introduce the following terminology.

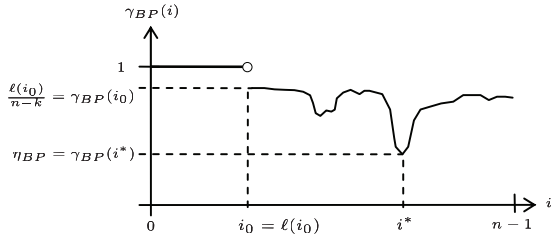

Fig. 1. Node efficiency curve

Definition 8. *Let* $\ell(i)$ *be the maximum BP-correctable burst erasure length for an erasure burst ending at symbol* i*. The* ***BP variable-node burst-erasure efficiency*** *of node* i *is*

$$\gamma_{BP}(i) \triangleq \begin{cases} \frac{\ell(i)}{n-k} & \text{if } i \geq \ell(i) \\ 1 & \text{if } i < \ell(i) \end{cases}. \tag{11}$$

Note that the efficiency $\gamma_{BP}(i)$, in this sense, is a property of the *node* i. In Fig. 1, we plot a typical shape of the BP node efficiency curve $\gamma_{BP}(i)$ with respect to the variable node location i. We make the following observations:

$$\eta_{BP} \triangleq \min_i \gamma_{BP}(i) \tag{12}$$

$$i^* \triangleq \arg\min_i \gamma_{BP}(i) \tag{13}$$

$$\eta_{BP} = \gamma_{BP}(i^*). \tag{14}$$

We denote, by i_0, the position for which $i_0 = \ell(i_0)$ and $\gamma_{BP}(i_0) = \frac{\ell(i_0)}{n-k}$.

The algorithm parses through all nodes starting from 0 to $n-1$. Let j be the node visited in the jth iteration. If $\gamma_{BP}(j)$ is smaller than $\min_{i<j} \gamma_{BP}(i)$, we swap node j with some other node p, such that $\min_{i \leq j} \gamma_{BP}(i)$ is maximized. The process is illustrated in Fig. 2. In particular, Fig. 2a shows the snapshot in iteration $j = i_0$; Fig. 2b shows the snapshot in an intermediate iteration, say $j = \frac{n}{2}$; and Fig. 2c shows the snapshot at the terminal iteration of the algorithm $j = n - 1$. This process moves i_0 as far to the right as possible.

Similarly, we illustrate the operations of the PSS algorithm

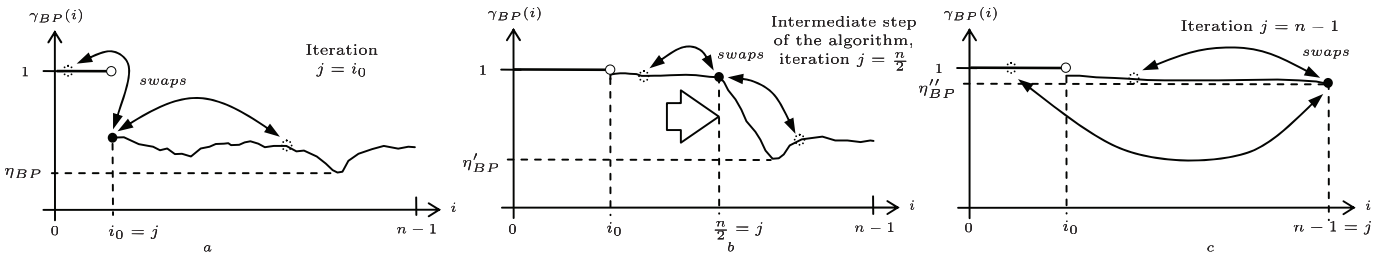

Fig. 2. Algorithm 1

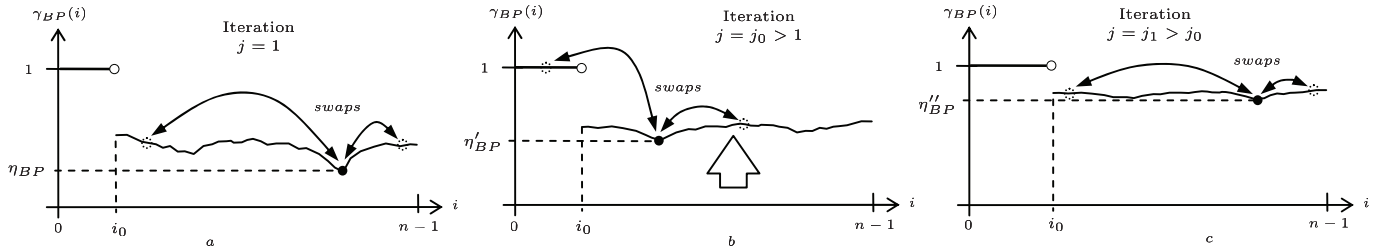

Fig. 3. Pivot Searching and Swapping [15]

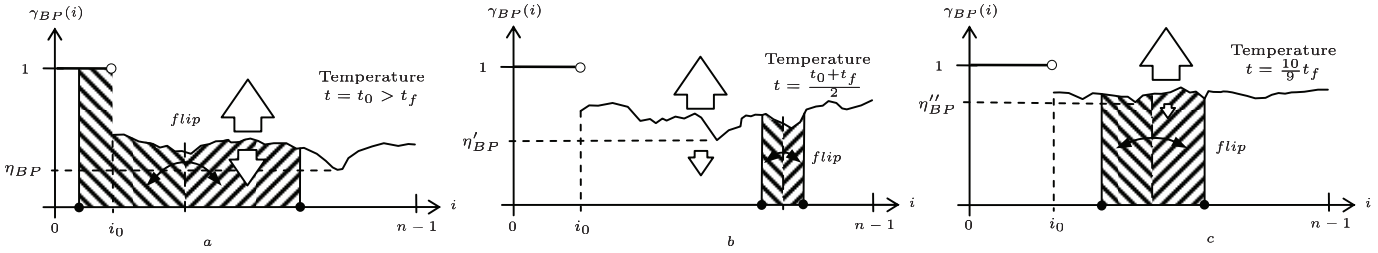

Fig. 4. Simulated Annealing [16]

in [15] and the SA algorithm of [16] in this manner. In the PSS algorithm, variable nodes corresponding to the global minima of $\gamma_{BP}(i)$, along with other pivots (defined in [15]), are targeted and swapped out with random non-pivot candidates (Fig. 3a), such that the permuted matrix has efficiency $\eta'_{BP} \geq \eta_{BP} + \frac{1}{n-k}$. The algorithm then finds the pivots of the permuted matrix and swaps them with non-pivots to achieve an even higher η_{BP} (Fig. 3b). This process is continued until $L_{BP}^{(max)}(\mathbf{H}_\pi)$ or $\eta_{BP}(\pi)$ cannot be increased or the number of failed permutations exceeds a threshold (Fig. 3c).

In SA, two random indices are chosen as interval boundaries, and the order of bit nodes in this interval is flipped, resulting in a permuted matrix with a new efficiency η'_{BP} (Fig. 4a). If η'_{BP} is higher than the previous efficiency, the permutation is kept and the process is repeated with the selection of a new, random interval (Fig. 4b). If a lower efficiency is produced, a decision is made, probabilistically, on whether to keep the permutation. The original input matrix is assumed to be at the annealing temperature t_0 and every time a less efficient matrix is accepted, a counter increments. When it surpasses a threshold, the temperature t is lowered slightly and the counter reseted. The decision making model reflects the fact that it is more likely to permute into a matrix with a lower efficiency at higher temperatures than at lower temperatures. This process is continued (Fig. 4c) until the temperature cools to a predetermined level t_f.

Because the optimal permutation must have the optimal $\ell(i)$ for all i (where $0 \leq i \leq n-1$), it is clearly undesirable, in terms of efficiency and performance, to progressively rearrange the entire matrix from lower burst-erasure efficiencies to higher efficiencies in increments of $\frac{1}{n-k}$, as in [15], or to traverse the full permutation space stochastically as in [16]. Unlike the two previous algorithms, our method never explicitly finds $\mathrm{Span}(\mathbf{H})$ or η_{BP} of a matrix and the program terminates deterministically after 1 run. We also note that there are no parameters or thresholds to tweak in Algorithm 1. However, we have noticed that repeating Algorithm 1 several times may slightly increase the efficiency, taking the resulting matrix from the previous run as input of the next run.

IV. Results

A. Comparison to PSS and SA

Here we compare the results of Algorithm 1 with that of PSS [15] and SA [16]. The matrices used are for a $(3, 6)$-regular $(500, 250)$ random LDPC code, an irregular $(1008, 504)$ LDPC code generated with the Progressive Edge

Growth (PEG) algorithm, a $(3,6)$-regular $(2640,1320)$ LDPC code with Margulis construction, and a $(3,30,152)$ regular quasi-cyclic (QC) LDPC code of length $n=4560$. (Here 152 is the size of the circulants used in the QC code design. QC-LDPC code is included for comparison purposes to [17].) The results are summarized below. Table I shows that Algorithm 1 performed best in comparison to the two previously published methods [15], [16]. We also compare our results to the density evolution (DE) bound first proposed in [18] and given in [15],[16]. We note that our algorithm beats the DE bound. This is not a contradiction because the DE bound in [18] was constructed based on a density evolution threshold for an ensemble of random codes with a fixed degree distribution. As such, the DE bound is an upper bound for a *random* ensemble of matrices $\mathbf{H}$ whose dimensions are (theoretically) infinite. There exists no converse to the density evolution threshold theorem, hence it is possible (as demonstrated by our algorithm) to deterministically construct matrices $\mathbf{H}$ that consistently beat the random ensemble DE bounds.

TABLE I
COMPARISON OF LDPC CODES FROM VARIOUS METHODS

Code	$(3,6)$ random	irregular	Margulis	$(3,30,152)$ QC
Code rate	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{9}{10}$
Block size	500	1008	2640	4560
Initial $L_{BP}^{(max)}$	149	86	1011	296
Initial η_{BP}	59.6%	17.1%	76.6%	64.9%
PSS $L_{BP}^{(max)}$	210	447	1135	382
PSS η_{BP}	84.0%	88.7%	86.0%	83.8%
SA $L_{BP}^{(max)}$	211	450	1122	360
SA η_{BP}	84.4%	89.3%	85.0%	78.9%
GSS $L_{BP}^{(max)}$	**233**	**490**	**1205**	**424**
GSS η_{BP}	**93.2%**	**97.2%**	**91.3%**	**93.0%**
DE bound	215	474	1134	378
DE bound η_{BP}	86%	94.0%	85.9%	82.9%

B. Comparison to Other Related Methods

A related method was published in [19]. However, since it returns a new code with a reduced block length, a direct comparison is improper. We evaluate the various methods' performances indirectly on the $(4,36)$-regular $(1908,1696)$ MacKay code [20]. The method in [19] achieves $L_{BP}^{(max)}=172$ but lowers n to $n=1902$. Algorithm 1, PSS, and SA preserve $n=1908$, with $L_{BP}^{(max)}$ of 181, 151, and 149 respectively. Clearly, Algorithm 1 achieves the highest efficiency of $\eta_{BP}=85.4\%$.

Another recently published method is the hybrid scheme given in [17], where the permuted LDPC matrix $\mathbf{H}'$ is augmented with judiciously selected check sums from its corresponding high-density matrix $\mathbf{H}_{BCF}$. In [17], for a $(3,30,133)$ regular QC-LDPC code, $L_{BP}^{(max)}=350$ requires 333 additional check sums. However, Algorithm 1 achieves $L_{BP}^{(max)}=370$, or $\eta_{BP}=92.7\%$, by using only permutations. So Algorithm 1 achieves higher efficiency than the hybrid scheme of reasonable complexity.

V. CONCLUSION

We presented and evaluated a new parity-check matrix permutation algorithm for optimizing LDPC codes over the burst erasure channel under BP decoding. It vastly improves the burst erasure correction capability of the code without compromising its random error correcting performance. Tests show that the resulting codes have higher burst erasure efficiencies than those produced by previously published methods.

REFERENCES

[1] R. H. Hempstead, "Thermally induced pulses in magnetoresistive heads", *IBM J. Res. Devel.*, vol. 18, pp. 547-550, Nov. 1974.

[2] K. B. Klaassen, J. C. L. van Peppen, "Electronic abatement of thermal interference in (G) MR head output signals", *IEEE Trans. Magn.*, vol. 33, pp. 2611-2616, Sept. 1997.

[3] M. F. Erden, E. M. Kurtas, "Thermal asperity detection and cancellation in perpendicular magnetic recording systems", *IEEE Trans. Magn.*, vol. 40, no. 3, pp. 1732-1737, May. 2004.

[4] R. L. Galbraith, G. J. Kerwin, and J. M. Poss, "Magneto-resistive head thermal asperity digital compensation", *IEEE Trans. Magn.*, vol. 28, pp. 2731-2732, Sept. 1992.

[5] V. Dorfman and J. K.Wolf, "A method for reducing the effects of thermal asperities", *IEEE J. Select. Areas Commun.*, vol. 19, pp. 662-667, Apr. 2001.

[6] G. D. Forney, "Burst-correcting codes for the classic bursty channel", *IEEE Trans. Commun.*, vol. 19, no. 5, pp. 772-781, Oct. 1971.

[7] M. Fossorier, "Universal burst error correction", *International Symposium of Information Theory and its Applications, ISITA'06, Seattle, USA*, pp. 1969-1973, July 2006.

[8] S. J. Johnson, "Burst Erasure Correcting LDPC Codes", *IEEE Trans. Commun.*, vol. 57, no. 3, pp. 641-652, Mar. 2009.

[9] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, no. 1, pp. 21-28, Jan. 1962.

[10] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Feb. 2001.

[11] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, no. 12, pp. 3017-3028, Dec. 2002.

[12] H. Saeedi and A. H. Banihashemi "New sequences of capacity achieving LDPC code ensembles over the binary erasure channel," *International Symposium of Information Theory, ISIT'08, Toronto, Canada*, pp. 797-801, July 2008.

[13] M. Yang and W. E. Ryan, "Design of LDPC codes for two-state fading channel models," *5th Int. Symp. Wireless Personal Multimedia Communications*, vol. 3, pp. 986-990, Oct. 2002.

[14] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, June 2002.

[15] E. Paolini and M. Chiani, "Construction of near-optimum burst erasure correcting low-density parity-check codes," *IEEE Trans. Commun.*, vol. 57, no. 5, pp. 1320-1328, May 2009.

[16] G. Sridharan, A. Kumarasubramanian, A. Thangaraj, and S. Bhashyam "Optimizing burst erasure correction of LDPC codes by interleaving," *International Symposium of Information Theory, ISIT'08, Toronto, Canada*, pp. 1143-1147, July 2008.

[17] M. Fossorier, "Hybrid Burst Erasure Correction of LDPC Codes," *IEEE Commun. Lett.*, vol. 13, no. 4, pp. 260-261, Apr. 2009.

[18] F. Peng, M. Yang, and W. Ryan, "Simplified eIRA code design and performance analysis for correlated Rayleigh fading channels," *IEEE Trans. Wireless Commun.*, vol. 5, no. 4, pp. 720C725, Apr. 2006.

[19] T. Wadayama, "Greedy construction of LDPC codes for burst erasure channel," *IEICE Tech. Report*, vol. 104, no. 302, pp. 35-40, Sept. 2004.

[20] D. J. C. MacKay, Encyclopedia of Sparse Graph Codes, [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html