Product Accumulate Codes: A Class of Codes With Near-Capacity Performance and Low Decoding Complexity

Jing Li, *Member, IEEE*, Krishna R. Narayanan, *Member, IEEE*, and Costas N. Georghiades, *Fellow, IEEE*

***Abstract*—We propose a novel class of provably good codes which are a serial concatenation of a single-parity-check (SPC)-based product code, an interleaver, and a rate-1 recursive convolutional code. The proposed codes, termed *product accumulate* (PA) codes, are linear time encodable and linear time decodable. We show that the product code by itself does not have a positive threshold, but a PA code can provide arbitrarily low bit-error rate (BER) under both maximum-likelihood (ML) decoding and iterative decoding. Two message-passing decoding algorithms are proposed and it is shown that a particular update schedule for these message-passing algorithms is equivalent to conventional turbo decoding of the serial concatenated code, but with significantly lower complexity. Tight upper bounds on the ML performance using Divsalar's simple bound and thresholds under density evolution (DE) show that these codes are capable of performance within a few tenths of a decibel away from the Shannon limit. Simulation results confirm these claims and show that these codes provide performance similar to turbo codes but with significantly less decoding complexity and with a lower error floor. Hence, we propose PA codes as a class of prospective codes with good performance, low decoding complexity, regular structure, and flexible rate adaptivity for all rates above $1/2$.**

***Index Terms*—Accumulator, low complexity, low-density parity-check (LDPC) codes, product codes, rate adaptivity, turbo product codes (TPC).**

I. Introduction and Outline of the Paper

We propose a novel class of provably good codes that have a positive signal-to-noise ratio (SNR) threshold above which an arbitrarily low error rate can be achieved as block size goes to infinity. The proposed codes, referred to as *product accumulate* (PA) codes, are shown to possess many desirable properties, including close-to-capacity performance, low decoding complexity, regular and easily implementable structure, and easy rate adaptivity uniformly for all rates higher than $1/2$.

Manuscript received May 24, 2001; revised July 30, 2003. This work was supported by the National Science Foundation under Grant CCR-0073506 and by the TITF research initiative. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Washington, DC, June 2001 and at the IEEE Information Theory Workshop, Cairns, Australia, September 2001.

J. Li was with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 USA. She is now with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015 USA (e-mail: jingli@ece.lehigh.edu).

K. R. Narayanan and C. N. Georghiades are with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: krn@ee.tamu.edu; georghiades@ee.tamu.edu).

Communicated by R. Urbanke, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2003.821995

The work was initiated by the search for good, high-rate codes which permit soft-decision and soft-output decoding. Several applications require the use of (soft-decision decodable) high-rate codes, and research on high-rate codes with good performance is of both theoretical and practical interest [2]. Some widely used high-rate codes are Reed–Solomon (RS) codes, punctured convolutional codes, turbo codes, and low-density parity-check (LDPC) codes. Until very recently, soft-decision decoding of RS codes has been a major computational problem. Recent developments are yet to be benchmarked to know the exact performance of soft-decision decoding of RS codes. To obtain good performance from high-rate punctured convolutional codes and turbo codes, convolutional codes usually must be of long constraint length, making the decoding complexity rather high. LDPCs, on the other hand, provide good performance at possibly lower complexity; however, the encoding complexity can be as high as $O(N^2)$ (N is the codeword length) if direct matrix multiplication is performed and, moreover, explicit storage of a generator matrix may be required. It has been shown in [5] that with careful preprocessing, most LDPC codes can be made linear-time encodable, but the preprocess requires a one-time complexity of up to $O(N^{3/2})$. Further, good high-rate LDPC codes are difficult to construct for short block lengths.

In an effort to construct good, simple, soft-decodable, high-rate codes, we investigated single-parity-check (SPC)-based product codes [14], also known as array codes [16] or hyper codes [17]. SPC-based product codes have recently been investigated for potential application in high-density magnetic recording channels and have demonstrated encouraging performance when decoded via a turbo approach [4], [14]. Since the product code itself does not have a positive threshold, we consider the concatenation of a rate-1 inner code (differential encoder or accumulator) with the product code through an interleaver. Through analysis and simulations we find this class of codes to be remarkably good in bit-error rate (BER) performance at high code rates ($R \geq 0.7$) when used with an iterative message-passing decoding algorithm. We show that the performance of these codes can be further improved by replacing the block interleaver in the conventional product outer code with a random interleaver. We will refer to such codes (using random interleavers) as PA-I codes Fig. 1(a). Clearly, when the outer code is a conventional product code (using a block interleaver), it is a special case of the general PA-I codes and we will refer to those as PA-II codes Fig. 1(b).

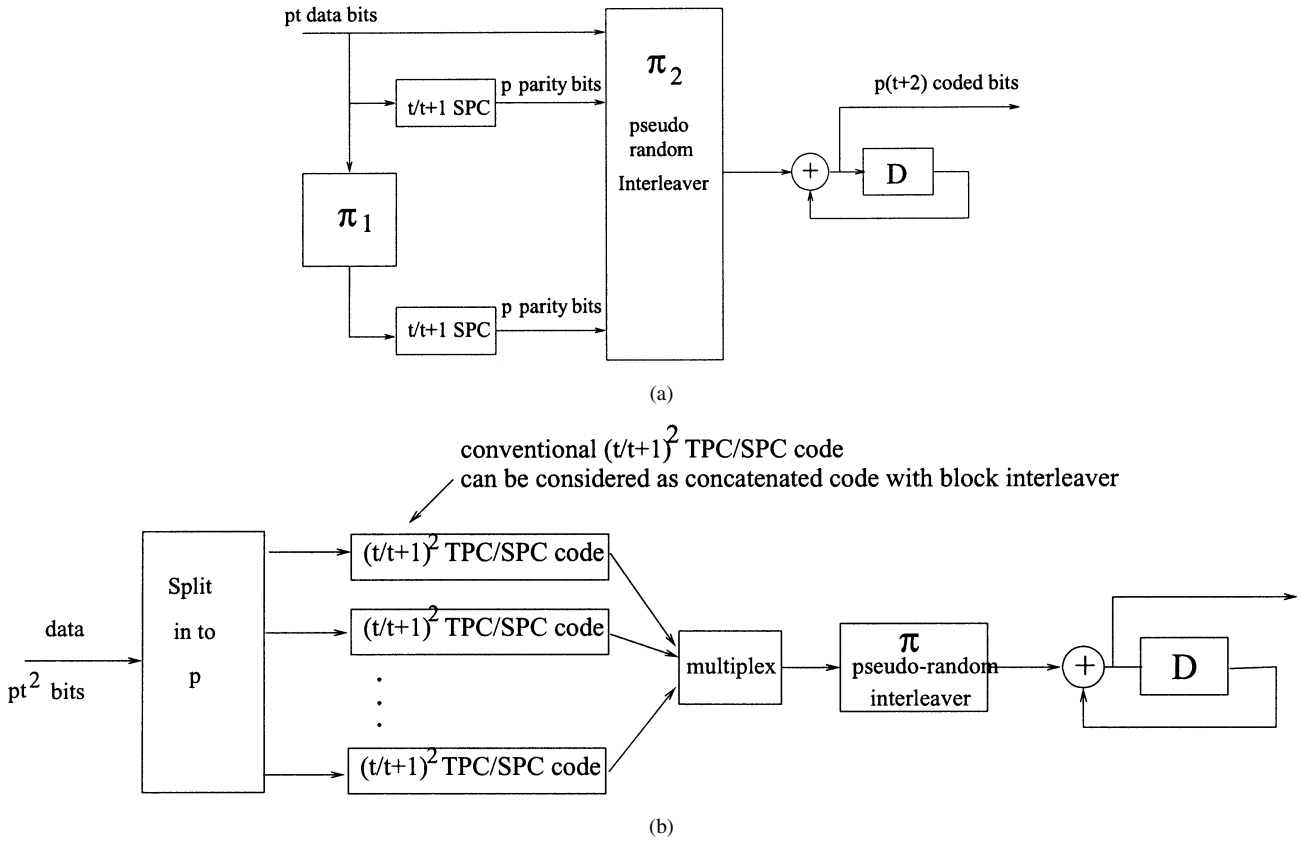

Fig. 1. System model for PA codes. (a) Structure of PA-I codes. (b) Structure of PA-II codes.

To facilitate understanding the structure and potential of the proposed codes, we compute tight upper bounds on their performance using the bounding technique developed by Divsalar [6]. We also study the graph structure of these codes. Thresholds are computed using density evolution (DE) [7] and shown to be within a few tenths of a decibel from the Shannon limit for all rates $R \geq 1/2$. By studying the graph structure, a message-passing (sum-product) decoding algorithm and its low-complexity approximation, a min-sum algorithm can be developed to iteratively decode the outer and inner codes. We show that a particular update schedule for this algorithm when applied to the graph of the inner code results in optimal decoding of the inner $1/(1+D)$ code. That is, the sum-product algorithm applied to the decoding of $1/(1+D)$ is equivalent to the Bahl, Jelinek, Cocke, and Raviv (BCJR) algorithm [31] (optimal in the *a posteriori* probability (APP) sense) and the min-sum algorithm is equivalent to the Max-log-MAP algorithm. However, the message-passing algorithm can be implemented with significantly lower complexity than the BCJR equivalents. Simulation results with long block lengths confirm the thresholds and simulations with short block lengths show that performance close to turbo codes can be achieved with significantly lower complexity.

As such, we propose the class of PA codes as a prospective class which not only enjoys good performance, low complexity, and soft decodability, but also maintains a simple and regular structure uniformly for all block sizes and for all rates above $1/2$. This regular structure, as well as the ease in construction, are particularly appealing properties in practical implementation and in applications that require rate adaptivity.

A brief background on SPC-based product codes is presented in Section II, followed by a description of PA codes in Section III. The decoding of PA codes is discussed in Section IV; in particular, a graph-based sum-product algorithm is described and shown to be optimal for inner rate-1 convolutional codes, yet with very low complexity. Section V analyzes in detail some properties of PA codes, including upper bounds on the performance under maximum-likelihoos (ML) decoding and thresholds of the codes under iterative decoding. Section VI discusses an algebraic construction which is useful in practical implementation. Section VII presents simulation results. Section VIII compares the proposed codes with other good codes proposed recently. Conclusions and future work are discussed in Section IX.

II. Background on SPC-Based Product Codes

Since the motivation for the proposed PA codes stems from SPC-based product codes, it is desirable to first discuss SPC-based product codes.

A. SPC-Based Product Code Structure and Properties

A product code [8], [10] is composed of a multidimensional array of codewords from linear block codes, such as parity-check codes, Hamming codes, and Bose–Chaudhuri–Hocquenghem (BCH) codes. Recently, iterative (turbo) decoding has been applied to decode product codes and, hence,

product codes have been widely referred to as block turbo codes [9] or turbo product codes (TPC). We will use the term turbo product code here since the overall decoder is an iterative decoder incorporating the turbo principle [10]. Particularly of interest is the simplest type of TPC codes, namely, single-parity-check turbo product codes (TPC/SPC) [14], also known as array codes [16] or hyper codes [17], due to their simplicity and high rate. An s-dimensional (s-D) turbo product code $\mathcal{C}$ formed from component codes

$$\mathcal{C}_1 \sim (N_1, K_1, d_1), \mathcal{C}_2 \sim (N_2, K_2, d_2), \ldots, \mathcal{C}_s \sim (N_s, K_s, d_s)$$

has parameters

$$(N_1 N_2 \cdots N_s, K_1 K_2 \cdots K_s, d_1 d_2 \cdots d_s)$$

where N, K, and d are the codeword length, user data block length, and the minimum distance of the code, respectively, and its generator matrix is the Kronecker product of the generator matrices of the component codes $G = G_1 \otimes G_2 \otimes \cdots \otimes G_s$. Since high rates are of interest, we restrict our attention to two-dimensional (2-D) TPC/SPC codes in this work where each row corresponds to a codeword from component code $\mathcal{C}_1$ and each column corresponds to a codeword from component code $\mathcal{C}_2$. In the general case, a TPC code may or may not have "parity-on-parity" bits [14]. A TPC code without parity-on-parity is essentially a parallel concatenation with a block interleaver, and a TPC code with parity-on-parity is a serial concatenation with a block interleaver.

The encoding of a TPC code is straightforward and can be done in linear time. The decoding of TPC codes takes an iterative approach based on the soft-in soft-out (SISO) decoders for each of its component codes [13]. Decoding of TPC component codes is generally via the Chase algorithm [12], a controlled-search procedure. However, with SPC component codes, decoding can be handled in a simpler and more efficient manner. The observation that a TPC/SPC code can be effectively viewed as a type of structured LDPC code [4], [14] where each row in each dimension satisfies a check, leads to a convenient adoption of the message-passing algorithm (or the sum-product algorithm) from LDPC codes. Since each bit is expressed as the modulo-2 sum of the rest of the bits in the check, this message-passing decoding algorithm is, in fact, an extension of replication decoding [15]. The exact decoding algorithm can be found in Appendix I. The simple and regular structure of a TPC/SPC code makes it possible to analyze the code properties. In particular, the weight spectrum of a 2-D TPC/SPC code with parameter $\mathcal{C} \sim (n_1 n_2, (n_1 - 1)(n_2 - 1))$ can be calculated by the following equation [15]:

$$A(h) = 2^{-n_1} \sum_{a=0}^{n_1} \binom{n_1}{a} \left[\sum_{m=0, m\ even}^{n_1} P_m(a; n_1) h^m \right]^{n_2} \quad (1)$$

where

$$P_m(a; n) = \sum_{k=0}^{m} (-1)^k \binom{a}{k} \binom{n-a}{m-k}. \quad (2)$$

As expected, $A(h)$ is symmetric in n_1 and n_2. It has been shown that the weight distribution of TPC/SPC codes asymptotically approaches that of a random code if the dimension of the code and the lengths of all component codes go to infinity [15]. However, increasing the dimension decreases the code rate and is therefore not of interest in the design of high-rate codes.

B. A TPC/SPC Code by Itself Cannot Achieve Arbitrarily Low Error Rate

One criterion for judging a code is the test for the existence of a threshold phenomenon where arbitrarily low error rate can be achieved (using infinite code length and infinite decoding complexity and delay) as long as the channel is better than this threshold. While LDPC codes, turbo codes, and many other serial/parallel concatenated codes have such thresholds, a TPC/SPC code alone does not. To see this, note that an s-dimensional TPC/SPC code always has minimum distance 2^s irrespective of the block size. Assuming ML decoding, the lower bound on the word-error rate (WER) is

$$P_w(e) \geq Q\left(\sqrt{\frac{2^{s+1} R E_b}{N_0}}\right) \quad (3)$$

where R is the code rate. Obviously, the lower bound is not a function of block size. In other words, unless the dimensionality of a TPC/SPC code, s, goes to infinity, its WER performance is always bounded away from zero independent of the block size.

In an effort to improve the performance of TPC/SPC codes, some attempts have been made to increase their minimum distance by carefully adding more parity checks by increasing the dimensionality [17], [18]. However, adding dimensionality obviously reduces code rate. Further, for any TPC/SPC code of a given dimensionality, the minimum distance is fixed and does not improve with block size. In other words, except for the asymptotic case where $s \to \infty$, multidimensional TPC/SPC codes will not be error free even if the block length goes to infinity. Moreover, when $s \to \infty$, $R \to 0$, and, hence, this case is not of interest here.

In this paper, we take a different approach in improving the performance of TPC/SPC codes, which is to group several blocks of TPC/SPC codewords together, interleave them, and further encode them with a rate-1 recursive convolutional code (or an accumulator). The resulting serial concatenation brings a significant improvement to TPC/SPC codes in their fundamental structural properties, for, as will be explained in later sections, the resulting serial concatenated code now has a positive threshold and is still linear-time encodable and decodable. Furthermore, we will discuss a modification to the interleaving scheme within the TPC/SPC code which results in a better code structure.

III. Structure of the Proposed Product Accumulate Codes

A. Proposed Code Structure

The proposed class of codes is a serial concatenation of an outer product code (i.e., a TPC/SPC code), a random interleaver, and an inner rate-1 recursive convolutional code of the form $1/(1 + D)$ (also known as the accumulator). Recall that depending on whether there are parity-on-parity

bits, the outer 2-D TPC/SPC code can be viewed as a parallel or serial concatenation with a block interleaver. Analysis and simulations show that if a big random interleaver is used instead of the block interleaver(s) within the TPC/SPC codes, then the performance of these codes can be further improved. Fig. 1(a) shows the exact structure of the proposed PA codes, or more precisely PA-I codes, whose outer code takes the form of two parallel branches of SPC codes concatenated via a random interleaver. It should be emphasized that in each branch, P blocks of codewords from $(t+1, t)$ SPC codes are combined and interleaved together. As will be shown later, this is important and essential to achieve the interleaving gain. Hence, these codes are of parameters $(N, K, R) = (P(t+2), Pt, t/(t+2))$, and are clearly a class of high-rate codes.

When no modification is made to the original structure of the outer TPC/SPC code, we call these codes PA-II codes. The overall structure is shown in Fig. 1(b). Clearly, PA-II codes are a special case of PA-I codes, and, likewise, P blocks need to be grouped and interleaved together before passing through the accumulator. Since a TPC/SPC code by default has parity-on-parity bits, PA-II codes thus have parameters $(N, K, R) = (P(t+1)^2, Pt^2, (\frac{t}{t+1})^2)$, which are slightly different from those of PA-I codes.

The idea of concatenating an outer code and an interleaver with a rate-1 recursive inner code, particularly of the form of $1/(1+D)$, to achieve coding gains (interleaving gain) without reducing the overall code rate is widely recognized [19]–[21]. For low-rate codes (rate-$1/2$ or less), convolutional codes and even very simple repetition codes [22] are good outer code candidates to provide satisfactory performance. However, the construction of very-high-rate codes based on this concept poses a problem. The key problem here is that, from Divsalar *et al.*'s results [23], [24], the outer code needs to have a minimum distance of at least 3 to obtain an interleaving gain. To obtain good high-rate convolutional codes through puncturing, and in particular to maintain a $d_{\min}$ of 3 after puncturing, the original convolutional codes must have fairly long constraint length, which makes decoding computationally complex. On the other hand, 2-D TPC/SPC codes possess many nice properties for a concatenated high-rate coding structure, such as high rate, simplicity, and the availability of an efficient soft-decoding algorithm. PA-II codes have outer codes with $d_{\min} = 4$ for any code rate and, hence, an interleaving gain is achieved. We will also show in Section V-B that although the outer code of PA-I codes has $d_{\min} = 2$ in the worst case, an interleaving gain still exists for the code ensemble.

In the following sections, we will perform a comprehensive analysis and evaluation of the proposed PA codes. The focus is on PA-I codes since they are the more general case and since they typically achieve better performance than PA-II codes.

IV. Iterative Decoding of PA Codes

The turbo principle is used to iteratively decode a serially concatenated system, where soft extrinsic information in log-likelihood ratio (LLR) form is exchanged between the inner and outer code. The extrinsic information from one subdecoder is used as *a priori* information by the other subdecoder. The decoding of the outer TPC/SPC code is done using a message-passing algorithm similar to that of LDPC codes, as described previously. The inner rate-1 convolutional code is typically decoded using a two-state BCJR algorithm, which generates the extrinsic information for bit x_i in the kth turbo iteration, denoted $L_e^{(k)}(x_i)$. The outer decoder uses $L_e^{(k)}(x_i)$ as *a priori* information and produces extrinsic information $L_o^{(k)}(x_i)$. However, a more computationally efficient approach is to use message-passing decoding directly on the graph of the PA code including the inner code, whose subgraph has no cycles.

It has been recognized that the message-passing algorithm is an instance of Pearl's belief propagation on graphical models with loops [25], [26]. The basic idea of probability inference decoding is implied in Tanner's pioneering work in 1981 [27], and later addressed by Wiberg [28], Frey *et al.* [26], [29], McEliece *et al.* [25], and Forney *et al.* [30]. While message-passing decoding has gained tremendous popularity in decoding LDPC codes, relatively little has been reported about convolutional codes, possibly because the code graph of a convolutional code is, in general, complex and involves many cycles which either make the message flow hard to track or make the algorithm ineffective (due to the significant amount of correlation in the messages caused by the cycles). Nevertheless, for the specific case of the $1/(1+D)$ code, a cycle-free Tanner graph presenting the relation of $y_i = x_i \oplus y_{i-1}$ ($\oplus$ denotes modulo-2 addition) can be constructed, using the message flow which can be conveniently traced. Recently, message-passing on the graph structure of a $1/(1+D)$ inner code has been used with irregular repeat accumulate (IRA) codes by Jin, Khandekar, and McEliece [22] and by Divsalar *et al.* [22], [6] to analyze two-state codes. Here, we use a serial update in the graph (rather than the parallel update as used in [22] and [6]). This is equivalent to the BCJR algorithm, but has an order of magnitude lower complexity [39]. We show that the low-complexity approximation, the min-sum update on the graph, is equivalent to the Max-log-MAP algorithm which further reduces the decoding complexity.

A. The Message-Passing Algorithm

As shown in Fig. 2(a), the combination of the outer code, the interleaver, and the inner code can be represented using one graph which contains bit nodes (representing the actual bits) and check nodes (representing a constraint such that connecting bit nodes should add up (modulo-2) to zero). Fig. 2(b) illustrates how messages evolve within the $1/(1+D)$ code. The outgoing message along an edge should contain information from all other sources except the incoming message from this edge. Hence, the extrinsic messages sent out at bit x_i at the kth turbo iteration $L_e^{(k)}(x_i)$ is computed as

$$L_e^{(k)}(x_i) = \Big(L_{ch}(y_{i-1}) + L_{e_f}^{(k)}(y_{i-1})\Big) \boxplus \Big(L_{ch}(y_i) + L_{e_b}^{(k)}(y_i)\Big) \tag{4}$$

where

$$L_{ch}(y_i) = \log \frac{\Pr(r_i|y_i = 0)}{\Pr(r_i|y_i = 1)}$$

denotes the message obtained from the channel (r_i is the received signal corresponding to the coded bit y_i), $L_{e_f}(y_i)$ and

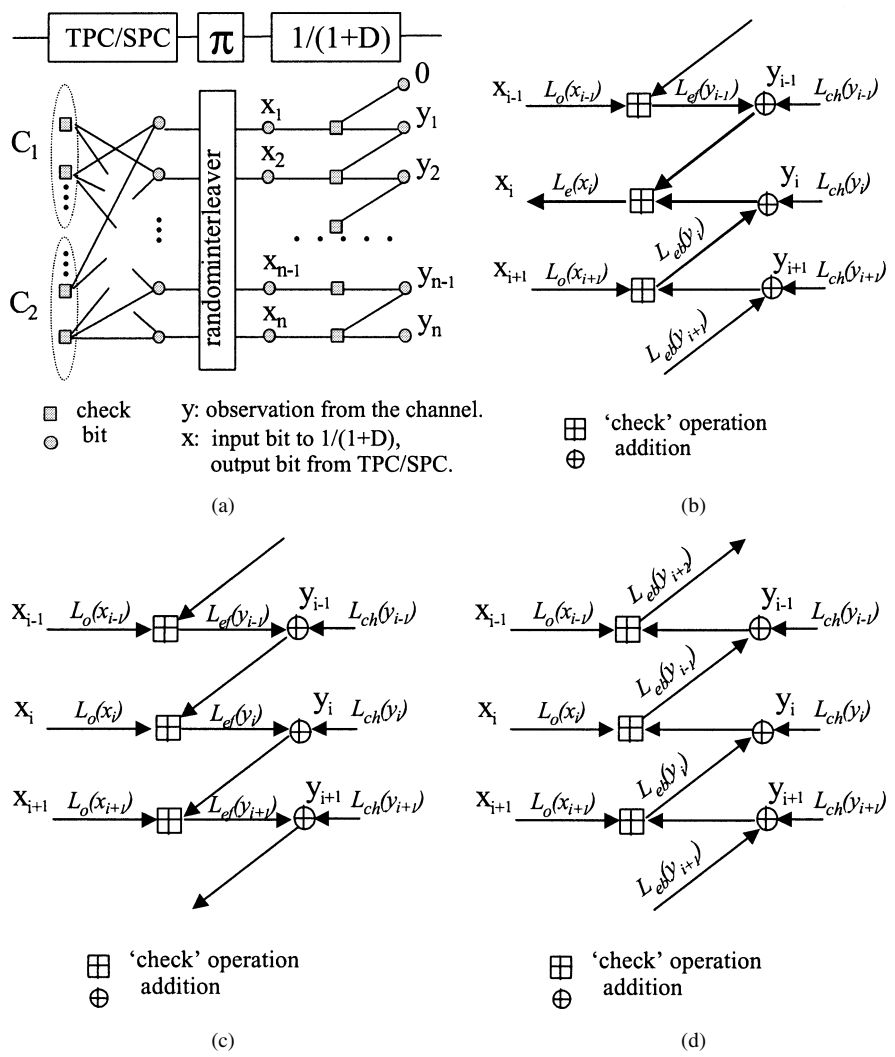

Fig. 2. Code graph and message-passing decoding for $1/(1+D)$. (a) Graph presentation of PA code. (b) Message flow in $1/(1+D)$. (c) Forward pass in $1/(1+D)$. (d) Backward pass in $1/(1+D)$.

$L_{e_b}(y_i)$ denote the (extrinsic) messages passed "forward" and "backward" to bit y_i from the sequence of bits/checks before and after the ith position, respectively. The superscript (k) denotes the kth turbo iteration between the inner and outer decoders (as opposed to the local iterations in the decoding of the outer TPC/SPC code) and the subscript i denotes the ith bit/check. The operation $\boxplus$ refers to a "check" operation, also known as the "tanh" rule, which is given by

$$\gamma = \alpha \boxplus \beta = 2\tanh^{-1}\left(\tanh\frac{\alpha}{2}\cdot\tanh\frac{\beta}{2}\right). \quad (5)$$

Forward-pass and backward-pass messages, $L_{e_f}(y_i)$ and $L_{e_b}(y_i)$, can be calculated using (see Fig. 2(c) and (d))

$$L_{e_f}^{(k)}(y_i) = L_o^{(k-1)}(x_i) \boxplus \left(L_{ch}(y_{i-1}) + L_{e_f}^{(k)}(y_{i-1})\right), \quad 2 \le, i \le N \quad (6)$$

$$L_{e_b}^{(k)}(y_i) = L_o^{(k-1)}(x_{i+1}) \boxplus \left(L_{ch}(y_{i+1}) + L_{e_b}^{(k)}(y_{i+1})\right), \quad 1 \le i \le N-1 \quad (7)$$

where $L_o^{(k-1)}(x_i)$ is the message received from the outer TPC/SPC code in the $(k-1)$th turbo iteration (between inner and outer codes). Clearly, $L_o^{(0)}(x_i) = 0, \forall\, i$, since in the first turbo iteration the inner code gets no information from the outer code. The boundary conditions are

$$L_{e_f}^{(k)}(y_1) = L_o^{(k-1)}(x_1) \boxplus \infty = L_o^{(k-1)}(x_1), \quad k \ge 1 \quad (8)$$

$$L_{e_b}^{(k)}(y_N) = 0, \quad k \ge 1. \quad (9)$$

From the above computation, it can be seen that the outbound message at the present time instance i, $L_e^{(k)}(x_i)$, has utilized all dependence among the past and future (through $L_{e_f}^{(k)}(x_{i-1})$ and $L_{e_b}^{(k)}(x_i)$) without any looping back of the same information.

Fact 1: The aforementioned message-passing (sum-product) decoder is identical to the BCJR algorithm for the $1/(1+D)$ inner code.

Proof: The result can be expected from the well-known fact that message-passing decoding, if properly conducted on cycle-free code graphs, will converge to the optimal solution [25], [26]. The exact proof of Fact 1 becomes straightforward if one notes the relations between the two decoding processes (which is summarized in Table I). To save space, we leave out the detailed discussion and the proof. Interested readers are referred to [39]. □

TABLE I
EQUIVALENCE OF SUM-PRODUCT AND MAP DECODING FOR THE $1/(1+D)$ CODE

	BCJR	Sum-product
Forward	$\bar{\alpha}_t = (\bar{\alpha}_{t-1} \boxplus L_o(x_t)) + L_{ch}(y_t)$	$L_{e_f}(y_t) = (L_{e_f}(y_{t-1}) + L_{ch}(y_{t-1})) \boxplus L_o(x_t)$
Backward	$\bar{\beta}_t = (\bar{\beta}_{t+1} + L_{ch}(y_{t+1})) \boxplus L_o(x_{t+1})$	$L_{e_b}(y_t) = (L_{e_b}(y_{t+1}) + L_{ch}(y_{t+1})) \boxplus L_o(x_{t+1})$
Extrinsic LLR	$\Lambda_t = \bar{\alpha}_{t-1} \boxplus (\bar{\beta}_t + L_{ch}(y_t))$	$L_e(x_t) = (L_{e_f}(y_{t-1}) + L_{ch}(y_{t-1})) \boxplus (L_{e_b}(y_t) + L_{ch}(y_t))$
$\bar{\alpha}_t \triangleq \log \frac{\Pr(S_t=0, \mathbf{r}_1^t)}{\Pr(S_t=1, \mathbf{r}_1^t)}$,	$\bar{\beta}_t \triangleq \log \frac{\Pr(\mathbf{r}_{t+1}^N \mid S_t=0)}{\Pr(\mathbf{r}_{t+1}^N \mid S_t=1)}$,	$\Lambda_t \triangleq \log \frac{\Pr(x_t=0 \mid \mathbf{r}_1^N)}{\Pr(x_t=1 \mid \mathbf{r}_1^N)}$

The key advantage of message-passing decoding is that it obviates the need to compute $\log(e^{\alpha} + e^{\beta})$ and the need to explicitly normalize at each step. Instead, a single operation $\log\tanh(\frac{\alpha}{2})$ is used which can be implemented using table lookup. A significant amount of complexity is thus saved, which makes the (aforementioned) message-passing decoding an efficient alternative to the conventional BCJR algorithm for $1/(1+D)$ code.

The message-passing algorithm used by Jin *et al.* [22] and Divsalar *et al.* [6] in repeat–accumulate/irregular repeat–accumulate (RA/IRA) codes is a parallel version the sequential update of $L_{e_f}^{(k)}$ and $L_{e_b}^{(k)}$ in (6) and (7)

$$L_{e_f}^{(k)}(y_i) = L_o^{(k-1)}(x_i) \boxplus (L_{ch}(y_{i-1}) + L_{e_f}^{(k-1)}(y_{i-1})) \quad (10)$$

$$L_{e_b}^{(k)}(y_i) = L_o^{(k-1)}(x_{i+1}) \boxplus (L_{ch}(y_{i+1}) + L_{e_b}^{(k-1)}(y_{i+1})). \quad (11)$$

Clearly, since the parallel version uses the information from the last iteration rather than the most recent, the convergence may be a little slower. But for practical block sizes and for moderate decoding times, simulations have shown that the compromise in performance is only about 0.1 dB after 15 to 30 iterations [39].

B. The Min-Sum Algorithm

The main complexity in the decoder comes from the $\boxplus$ operation in both the outer TPC/SPC and inner $1/(1+D)$ decoding. Each turbo iteration (composed of one round of $1/(1+D)$ decoding followed by one round of TPC/SPC decoding[1] requires at least five $\boxplus$ operations per coded bit. A straightforward implementation of $\boxplus$ may require as many as one addition and three table lookups (assuming $\log(\tanh(\cdot))$ and $\log(\tanh^{-1}(\cdot))$ are implemented via table lookups). Although this is already lower complexity than turbo codes, it is possible and highly practical to further reduce the complexity with a slight compromise in performance. Just like the Max-log-MAP algorithm of turbo codes, the $\boxplus$ operation has a similar approximation [37], [38]

$$\begin{aligned}\gamma &= 2\tanh^{-1}\left(\tanh\frac{\alpha}{2}\cdot\tanh\frac{\beta}{2}\right)\\ &= \log\frac{1+e^{\alpha+\beta}}{e^{\alpha}+e^{\beta}}\\ &= \operatorname{sign}(\alpha)\cdot\operatorname{sign}(\beta)\cdot\min(|\alpha|,|\beta|) + \log\frac{1+e^{-|\alpha+\beta|}}{1+e^{-|\alpha-\beta|}}\\ &\approx \operatorname{sign}(\alpha)\cdot\operatorname{sign}(\beta)\cdot\min(|\alpha|,|\beta|). \end{aligned} \quad (12)$$

[1]Simulation results show that the best performance/complexity gain is achieved with only one local iteration of TPC/SPC decoding in each turbo iteration between the inner and outer decoders.

If the approximation in (12) is used, i.e., a "signed min" operation is used instead of $\boxplus$, then a considerable reduction in complexity is achieved, and the message-passing algorithm, or the sum-product algorithm, is then reduced to the min-sum algorithm.

Fact 2: Min-sum decoding of $1/(1+D)$ is equivalent to Max-log-MAP decoding.

Proof: Fact 2 follows from Fact 1 where the equivalence of sum-product decoding and BCJR decoding for the $1/(1+D)$ code is shown. Note that the Max-log-MAP algorithm approximates the BCJR algorithm by replacing the $\max^*$ operation with $\max$ operation and that the min-sum algorithm approximates the sum-product algorithm by replacing the $\boxplus$ operation with a signed $\min$ operation. Specifically, between the Max-log-MAP and BCJR algorithm, we have the following approximation:

$$\max(\alpha,\beta) \approx \max^*(\alpha,\beta) \triangleq \log(e^{\alpha}+e^{\beta}). \quad (13)$$

Applying approximation (13) to the sum-product algorithm results in the min-sum algorithm. It thus follows that the min-sum algorithm is a computationally efficient realization of the Max-log-MAP algorithm for the decoding of $1/(1+D)$ codes. □

V. PROPERTIES OF PRODUCT ACCUMULATE CODES

Before presenting numerical results, we first show some properties of PA codes to facilitate understanding of their performance. The proposed PA codes possess the following properties [3].

i) *Property I*: They are linear time encodable and linear time decodable.

ii) *Property II*: They are capable of achieving error-free performance under optimal ML decoding asymtotically.

iii) *Property III*: They are capable of achieving asymptotic error-free performance under iterative decoding.

A. Encoding and Decoding Complexity

The encoding and decoding complexity of PA codes is linear in the codeword length. The encoding process involves only a parity check in each dimension (see Section II-A), interleaving and encoding by a rate-1 inner code (see Fig. 1(b)), all of which require linear complexity in the block length. The decoding complexity is proportional to the number of iterations of the outer TPC/SPC code and the inner convolutional code, both of which have linear decoding complexity.

TABLE II
DECODING COMPLEXITY IN OPERATIONS PER DATA BIT PER ITERATION (R IS CODE RATE)

Code	outer TPC/SPC		inner $1/(1+D)$			
Decoding Algorithm	sum-prod.	min-sum	log-MAP	Max-log-MAP	sum-prod.	min-sum
Additions	$5+\frac{1}{R}$	2	$(15\cdot 2+9)/R$	$(10\cdot 2+11)/R$	$5/R$	$2/R$
Min/Max		$5-\frac{1}{R}$	$(5\cdot 2-2)/R$	$(5\cdot 2-2)/R$		$3/R$
Table Lookup	$2+\frac{2}{R}$		$(5\cdot 2-2)/R$		$6/R$	

Table II summarizes the complexity of different decoding strategies for the inner and outer codes. We assume that in sum-product decoding, $\log\tanh(\frac{\alpha}{2})$ is implemented using table lookup. The complexity of the log-MAP and Max-log-MAP algorithms is evaluated using [32] (based on the conventional implementation of the BCJR algorithm). As can be seen, the sum-product and min-sum decoding of $1/(1+D)$ require only about $1/6$ and $1/8$ the complexity of their BCJR equivalents, respectively. For a rate-$1/2$ PA code, message-passing decoding requires about 35 operations per data bit per iteration, while min-sum decoding requires only about 15 operations; both are significantly less than the number of operations involved in a turbo code.

B. Performance Under ML Decoding

In the ML-based analysis of PA codes, we first quantify the interleaving gain and then derive a tight upper bound on the word error probability. We show that under ML decoding, the probability of word error is proportional to P^{-1} for large E_b/N_0, where P is the number of TPC/SPC codewords concatenated before interleaving. Further, we show that these codes can perform close to capacity limits by computing thresholds for these codes based on the tight upper bound on the WER due to Divsalar [6].

1) Interleaving Gain: From the results of Benedetto *et al.* [23] and Divsalar, Jin, and McEliece [24], we know that for a general serial concatenated system with recursive inner code, there exists a threshold γ such that for any $E_b/N_0 \geq \gamma$, the asymptotic WER is upper-bounded by

$$P_w^{\mathrm{UB}} = O\left(N^{-\lfloor \frac{d_m^o-1}{2} \rfloor}\right) \tag{14}$$

where d_m^o is the minimum distance of the outer code and N is the interleaver size. Whereas this result offers a useful guideline in quantifying the interleaving gain, one must be careful in interpreting it for PA codes.

The result in (14) indicates that if the minimum distance of the outer code is at least 3, then an interleaving gain can be obtained. However, the outer codewords of PA codes (with random interleavers) have minimum distance of only 2. On the other hand, if S-random interleavers are used such that bits within distance S are mapped to at least distance S apart, then the outer codewords are guaranteed to have a minimum distance of at least 3 as long as $S \geq t$. Since a block interleaver can be viewed as a structured S-random interleaver, it follows that interleaving gain exists for PA-II codes. Below, we show that although the minimum distance of the outer codewords is only 2 over the ensemble of interleavers, an interleaving gain still exists for PA codes with random interleavers (PA-I codes). Since from (14) outer codewords of weight 3 or more will lead to an interleaver gain, we focus the investigation on weight-2 outer codewords only and show that the number vanishes as P increases. The all-zero sequence is used as the reference since the code is linear.

It is convenient to employ the uniform interleaver which represents the average behavior of the ensemble of codes. Let $A_{w,h}^{(j)}$, $j=1,2$, denote the input output weight enumerator (IOWE) of the jth SPC branch code (parallelly concatenated in the outer code). The IOWE of the outer codewords, $A_{w,h}^{o}$, averaged over the code ensemble is given as

$$A_{w,h}^{o} = \sum_{h_1} \frac{A_{w,h_1}^{(1)} A_{w,h-h_1}^{(2)}}{\binom{K}{w}} \tag{15}$$

where $K = Pt$ is the input sequence length.

Define the input output weight transfer probability (IOWTP) of the jth branch code, $P_{w,h}^{(j)}$, as the probability that a particular input sequence of weight w is mapped to an output sequence of weight h

$$P_{w,h}^{(j)} = \frac{A_{w,h}^{(j)}}{\binom{K}{w}}, \qquad j=1,2. \tag{16}$$

Substituting (16) in (15), we get

$$A_{w,h}^{o} = \sum_{h_1} A_{w,h_1}^{(1)} P_{w,h-h_1}^{(2)}. \tag{17}$$

For each branch where P $(t+1,t)$ SPC codewords are combined, the IOWE function is given as (assuming even parity check)

$$A^{\mathrm{SPC}}(w,h) = \left(1 + \binom{t}{1}wh^2 + \binom{t}{2}w^2h^2 + \binom{t}{3}w^3h^4 + \binom{t}{4}w^4h^4 + \cdots + \binom{t}{t}w^th^{2\lceil t/2\rceil}\right)^P \tag{18}$$

$$= \left(\sum_{j=0}^{t}\binom{t}{j}w^jh^{2\lceil j/2\rceil}\right)^P \tag{19}$$

where the coefficient of the term w^uh^v denotes the number of codewords with input weight u and output weight v. Using (19), we can compute the IOWEs of the first SPC branch code, denoted as $A_{u,v}^{(1)} (= A_{u,v}^{\mathrm{SPC}})$. For the second branch of the SPC code, since only parity bits are transmitted, $A_{u,v}^{(2)} = A_{u,v+u}^{(1)}$.

With a little computation, it is easy to see that the number of weight-2 outer codewords is given by

$$A^o_{h=2} = \sum_w A^o_{w,h=2} = P\binom{t}{2}\left(\frac{P\binom{t}{2}}{\binom{Pt}{2}}\right) = O\left(\frac{(t-1)^2}{2}\right) = O(t^2) \quad (20)$$

where the last equation assumes a large P (i.e., large block size). Equation (20) shows that the number of weight-2 outer codewords is a function of a single parameter t which is related only to the rate of SPC codes and not the block length. Now considering the serial concatenation of the outer codewords with the inner $1/(1+D)$ code, the overall output weight enumerator (OWE) A^{PA}_h is

$$A^{\mathrm{PA}}_{h=s} = \sum_{h'} A^o_{h'} \frac{A^{1/(1+D)}_{h',h}}{\binom{N}{h'}} \quad (21)$$

$$= \sum_{h'}\sum_w A^o_{w,h'} \frac{A^{1/(1+D)}_{h',h}}{\binom{N}{h'}} \quad (22)$$

where $A^o_{h'}$ is the OWE of the outer code, and the IOWE of the $1/(1+D)$ code is given by [24]

$$A^{1/(1+D)}_{w,h} = \binom{N-h}{\lfloor \frac{w}{2} \rfloor}\binom{h-1}{\lceil \frac{w}{2} \rceil - 1}. \quad (23)$$

In particular, the number of weight-s PA codewords produced by weight-2 outer codewords (for small s), denoted as $A^{\mathrm{PA2}}_{h=s}$, is

$$A^{\mathrm{PA2}}_{h=s} = \frac{(t-1)^2}{2}\frac{N-s}{\binom{N}{2}} = O(tP^{-1}) \quad (24)$$

where $N = P(t+2)$ is the PA codeword length. This indicates that the number of small weight-s codewords of the overall PA code due to weight-2 outer codewords (caused by weight-2 input sequences) vanishes as P increases. When the input weight is greater than 2, the outer codeword always has weight greater than 2 and, hence, an interleaving gain can be guaranteed. Hence, an interleaving gain exists for PA codes and it is proportional to P.

2) Upper Bounds: To further shed insight into the asymptotic performance $(N \to \infty)$ of PA codes under ML decoding, we compute thresholds for this class of codes based on the bounding technique recently proposed by Divsalar [6]. The threshold here refers to the capacity of the codes under ML decoding, i.e., the minimum E_b/N_0 for which the probability of error decreases exponentially in N and, hence, tends to zero as $N \to \infty$

Among the various bounding techniques developed, the union bound is the most popular but it is fairly loose above the cutoff rate. Tighter and more complicated bounds include the tangential sphere bound by Poltyrev [33], the Viterbi–Viterbi bound [34], Duman–Salehi bound [35], the Hughes bound [36]. These new tight bounds are essentially based on the bounding techniques developed by Gallager [40]

$$\Pr(\text{error}) \le \Pr(\text{error}, \bar{y} \in \Re) + \Pr(\bar{y} \notin \Re) \quad (25)$$

where $\bar{y}$ is the received codeword (matched-filter samples of the noise-corrupted codeword), and $\Re$ is a region in the observed space around the transmitted codeword. To get a tight bound, the above methods usually require optimization and integration to determine a meaningful $\Re$.

Recently, Divsalar developed a *simple bound* on error probability over additive white Gaussian noise (AWGN) channels [6]. The bound is also based on (25), but a simple closed-form expression is derived and shown that the computed minimum SNR threshold can serve as a tight upper bound on the ML capacity of nonrandom codes. The simple bound is the tightest closed-form bound developed so far. It is also shown that, as block size goes to infinity, this simple bound is equivalent to the tangential sphere bound [6]. In what follows we apply this simple bounding technique to the analysis of PA codes.

We first quote and summarize the main results of [6]. Define the *spectral shape* of a code, $\gamma_N(\delta)$, as the normalized weight distribution averaged over the code ensemble $\mathcal{C}_N$

$$\gamma_N(\delta) \triangleq \frac{1}{N}\ln(A_{h=\lfloor \delta N \rfloor}), \qquad 0 < \delta < 1 \quad (26)$$

where N is the code length and A_h is the (average) output weight enumerator of the code. Further, define the ensemble spectral shape as

$$\gamma(\delta) \triangleq \lim_{N\to\infty} r_N(\delta), \qquad 0 < \delta < 1. \quad (27)$$

It can be shown that the probability of word error can be upper-bounded by [6]

$$P_w(e) \le \sum_h e^{-NE(E_b/N_0,h)} \quad (28)$$

where

$$E(c,h) = -\gamma(\delta) + \frac{1}{2}\ln\left[\beta + (1-\beta)e^{2\gamma(\delta)}\right] + \frac{\delta\beta}{1-\delta(1-\delta)}c \quad (29)$$

where

$$\beta = \sqrt{c\frac{1-\delta}{\delta}\frac{2}{1-e^{-2\gamma(\delta)}} + \left(\frac{1-\delta}{\delta}\right)^2\left[(1+c)^2 - 1\right]} - \frac{1-\delta}{\delta}(1+c). \quad (30)$$

The threshold $\boldsymbol{C}^*_{\mathrm{ML}}$ is defined as the minimum E_b/N_0 such that $E(E_b/N_0, h)$ is positive for all h and, hence, for all $E_b/N_0 \ge \boldsymbol{C}^*_{\mathrm{ML}}$, $P_w(e) \to 0$ as $N \to \infty$. The threshold can be computed as [6]

$$\boldsymbol{C}^*_{\mathrm{ML}} = \frac{1}{R}\max_{0<\delta\le(1-R)} c_0(\delta) \quad (31)$$

where R is the code rate. For the simple bound, $c_0(\delta)$ is given by

$$\text{Simple: } c_0(\delta) = \frac{1-\delta}{2\delta}\left(1 - e^{-2\gamma(\delta)}\right). \quad (32)$$

Similar forms are also derived for Viterbi–Viterbi, Hughes, and Union bounds [6]

$$\text{Viterbi: } c_0(\delta) = \frac{(1-\delta)}{\delta}\gamma(\delta) \quad (33)$$

$$\text{Hughes: } c_0(\delta) = \frac{1}{2\delta}\left(1 - e^{-2\gamma(\delta)}\right) \quad (34)$$

$$\text{Union: } c_0(\delta) = \frac{\gamma(\delta)}{\delta}. \quad (35)$$

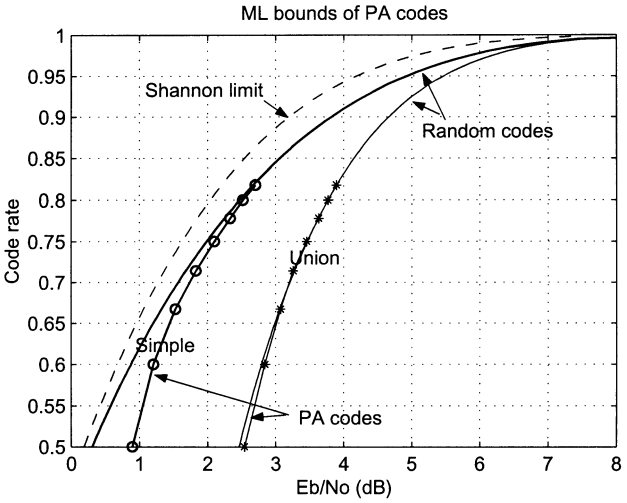

Fig. 3. The union bound and the simple bound of PA codes (PA-I).

Since the above bounds are based on the ensemble spectral shape $r(\delta)$, they serve as the asymptotic performance limit (i.e., $N \to \infty$) of the code ensemble assuming ML decoding.

There is no simple closed-form expression for the ensemble spectral shape of PA codes. However, the spectral shape can be computed to a good accuracy numerically since the component codes of the concatenation are SPC codes. Specifically, using (17), (22), and (26) we can compute the spectral shape of PA codes, which is a function of N, P, t. We approximate the ensemble spectral shape by choosing a large N. Whenever possible, IOWTP $P_{w,h}$ should be used instead of the IOWE $A_{w,h}$, to eliminate numerical overflow. The bounds for GPA codes are computed and plotted in Fig. 3 (for clarity, only the simple bound and the union bound are shown). For comparison, also shown are the bounds for random codes and the Shannon limit. Several things can be observed: 1) the simple bounds of PA codes are very close to those of the random codes, indicating that PA codes have good distance spectrum; 2) the higher the rate, the tighter the bound is, indicating that GPA codes are likely more advantageous at high rates than low rates (as opposed to repeat accumulate codes).

The implication of the above analysis is that PA codes are capable of performance a few tenths of a decibel away from the capacity limit with ML decoding. However, since there does not exist a computationally feasible ML decoder, it is desirable to investigate iterative decoding to provide a more meaningful evaluation of the code performance with practical decoding.

C. Performance Under Iterative Decoding

In this subsection we compute the iterative threshold (minimum E_b/N_0) for PA codes using DE. DE has been shown to be a very powerful tool in the analysis and design of LDPC and LDPC-like codes [7], [41]–[43]. By examining the distribution of the messages passed within and in-between the subdecoders, we are able to determine the fraction of incorrect messages (extrinsic messages of the wrong sign). The basic idea is that if the fraction of incorrect messages goes to zero with the number of iterations, then the decoding procedure will eventually converge to the correct codeword.

The analysis of PA codes involves computation of the probability density function (pdf) of the message flow within the outer decoder, the inner decoder, and in-between the two. Since the pdf that evolves with iterations may not have a closed-form expression, density evolution takes a numerical approach. It is worth mentioning that a simplified approximation can be made by assuming that the messages passed in each step follow Gaussian distributions. This Gaussian assumption trades a little accuracy for a considerable reduction in computational complexity when combined with the consistency condition which states that the distribution f of messages w passed in each step satisfies $f(w) = f(-w)e^w$ [43]. Here, to preserve the accuracy, we perform the exact density evolution (with quantization).

Without loss of generality, we assume that the all-zero codeword is transmitted and use LLRs as messages to examine the decoding process. The threshold, which serves as the practical capacity limit for a given code (given rate and decoding strategy), is thus formulated as

$$\boldsymbol{C}^*_{\text{iterative}} = \inf_{\text{SNR}} \left\{ \text{SNR: } \lim_{k\to\infty} \lim_{N\to\infty} \int_{-\infty}^{0} f_{L_o^{(k)}}(x)dx \to o \right\} \quad (36)$$

where $f_{L_o^{(k)}}(x)$ is the pdf of the messages (extrinsic information) evaluated at the output of the outer decoder (due to the independent and identical distribution (i.i.d.) assumption, we have dropped the dependence i on x_i) superscript (k) denotes the kth iteration between the outer and inner decoder, and N is the block size. Before we describe how DE is performed numerically for PA codes, we first discretize messages. Let $\mathrm{Q}(w)$ denote the quantization operation on message w with a desired quantization interval (accuracy) Δ.

1) Message Flow Within the Outer Decoder: The outer code of the general product codes (PA-I) consists of two parallel concatenated branches where each branch is formed of P blocks of $(t+1, t)$ SPC codewords. This alone can also be considered as a special case of LDPC codes whose parity-check matrix has $2P$ rows with uniform row weight of $(t+1)$, and $(t+1)^2$ columns with $\frac{t}{t+2}$ percent of the columns having weight 2 and the rest weight 1. Therefore, the exact decoding algorithm for LDPC codes can be applied to the outer code. However, for a more efficient convergence, we could make use of the fact that the checks in the outer code can be divided into two groups (corresponding to the upper and lower branch, respectively) such that the corresponding subgraph (Tanner graph) of each group is cycle free. It thus leads to a serial message-passing mode where each group of checks take turns to update (as opposed to the parallel update of all checks in LDPC codes).

The fundamental element in the decoding of the outer code is the decoding of SPC codes. Consider the upper branch. Suppose data bits $d_{i,1}, d_{i,2}, \ldots, d_{i,t}$ and parity bit p_i participate in the ith SPC codeword $(1 \le i \le P)$. Then the messages (extrinsic information) for each bit obtained from this check (during the kth turbo iteration and lth local iteration) are

$$\text{data bit: } L_{e1}^{(k,l)}(d_{i,j}) = \left(\sum_{1\le k\le t, k\ne j} \boxplus \left(L_e^{(k)}(d_{i,k}) + L_{e2}^{(k,l-1)}(d_{i,k}) \right) \right) \boxplus L_e^{(k)}(p_i)$$

$$\Longleftrightarrow \quad \tanh\frac{L_{e1}^{(k,l)}(d_{i,j})}{2}$$
$$= \left(\prod_{1\le k\le t, k\ne j} \tanh\frac{L_e^{(k)}(d_{i,k}) + L_{e2}^{(k,l-1)}(d_{i,k})}{2}\right) \cdot \tanh\frac{L_o^{(k)}(p_i)}{2} \tag{37}$$

parity bit: $L_{e1}^{(k,l)}(p_i)$
$$= \left(\sum_{1\le k\le t} \boxplus\Big(L_e^{(k)}(d_{i,k}) + L_{e2}^{(k,l-1)}(d_{i,k})\Big)\right)$$
$$\Longleftrightarrow \quad \tanh\frac{L_{e1}^{(k,l-1)}(p_i)}{2}$$
$$= \prod_{1\le k\le t} \tanh\frac{L_e^{(k)}(d_{i,k}) + L_{e2}^{(k,l-1)}(d_{i,k})}{2} \tag{38}$$

where $L_e(\cdot)$ denotes the messages (*a priori* information) received from the inner code, $L_{e1}(\cdot)$ denotes the messages (extrinsic information) obtained from the upper SPC branch to be passed to the lower branch and $L_{e2}(\cdot)$ denotes the messages to be passed from the lower branch to the upper branch. After interleaving, similar operations of (37) and (38) are performed within the lower branch. We assume $L_{e1}(\cdot)$ and $L_{e2}(\cdot)$ to be i.i.d and drop the dependence on $d_{i,j}$ and p_i.

We use superscript (k,l) to denote the kth turbo iteration between the outer decoder and inner decoder and the lth iteration within the outer decoder (local iterations). For independent messages to add together, the resulting pdf of the sum is the discrete convolution of the component pdfs. This calculation can be efficiently implemented using a fast Fourier transform (FFT). For the $\tanh$ operation on messages, define

$$\gamma = \mathrm{Q}(\alpha \boxplus \beta) \triangleq \mathrm{Q}\left(2\tanh^{-1}\left(\tanh\frac{\alpha}{2}\tanh\frac{\beta}{2}\right)\right) \tag{39}$$

where α, β, and γ are discretized messages, and Q denotes the quantization operation. The pdf f_γ of γ can be computed using

$$f_\gamma[k] = \sum_{(i,j):\, k\Delta = i\Delta \boxplus j\Delta} f_\alpha[i]\cdot f_\beta[j]. \tag{40}$$

To simplify the notation, we denote this operation (40) as

$$f_\gamma = \mathcal{R}(f_\alpha, f_\beta). \tag{41}$$

In particular, using induction on the preceding equation, we can denote

$$\mathcal{R}^k(f_\alpha) \triangleq \underbrace{\mathcal{R}(f_\alpha, (\mathcal{R}(f_\alpha, \ldots, \mathcal{R}(f_\alpha, f_\alpha)\cdots))}_{k-1}. \tag{42}$$

It then follows from (37), (38), and (42) that the pdf of the extrinsic messages obtained from the upper branch $f_{L_{e1}}(\cdot)$ and the lower branch, $f_{L_{e2}}(\cdot)$ are given by

Upper branch :

data bit: $f_{L_{e1}}^{(k,l)}(d) = \mathcal{R}\left(f_{L_e}^{(k)}(p), \mathcal{R}^{(l-1)}\big(f_{L_e}^{(k)}(d) * f_{L_{e2}}^{(k,l-1)}(d)\big)\right)$ (43)

parity bit: $f_{L_{e1}}^{(k,l)}(p) = \mathcal{R}^l\left(f_{L_e}^{(k)}(d) * f_{L_{e2}}^{(k,l-1)}(d)\right)$. (44)

Lower branch:

data bit: $f_{L_{e2}}^{(k,l)}(d) = \mathcal{R}\Big(f_{L_e}^{(k)}(p), \mathcal{R}^{(l-1)}\big(f_{L_e^{(k)}}(d) * f_{L_{e1}}^{(k,1)}(d)\big)\Big)$ (45)

parity bit: $f_{L_{e2}}^{(k,l)}(p) = \mathcal{R}^l\Big(f_{L_e}^{(k)}(d) * f_{L_{e1}}^{(k,l)}(d)\Big)$ (46)

where $f_{L_e}^{(k)}(\cdot)$ denotes the pdf of the messages $L_e^{(k)}(\cdot)$ from the inner $1/(1+D)$ code in the kth turbo iteration, $f_{L_{e1}}^{(k,l)}(\cdot)$ and $f_{L_{e2}}^{(k,l)}(\cdot)$ denote the pdfs of the extrinsic information from the upper and lower branch of the outer code, $L_{e1}^{(k,l)}(\cdot)$ and $L_{e2}^{(k,l)}(\cdot)$, respectively, and $*$ denotes the discrete convolution.

Since the systematic bits (data) and the parity bits of the outer code are treated the same in the inner $1/(1+D)$ code, we have

$$f_{L_e}^{(k)}(d) = f_{L_e}^{(k)}(p) = f_{L_{e,x}}^{(k)}$$

where $f_{L_{e,x}}^{(k)}$ is the pdf of the extrinsic information $L_{e,x}^{(k)}$ obtained from $1/(1+D)$ (also refer to Section V-C2 for a detailed explanation). For PA-I codes, the local iterations within the outer code only involve the exchange of messages associated with data bits (as can be seen from the above equations). After L local iterations, the messages the outer code passes along to the inner code include those of data bits ($L_{e1}(d)$ and $L_{e2}(d)$) and parity bits ($L_{e1}(p)$ and $L_{e2}(p)$), which thus leads to a mixed message density with a fraction $t/(t+1)$ having pdf $(f_{L_{e1}}(d) * f_{L_{e2}}(d))$ and equal fractions $1/(2t+2)$ having mean $f_{L_{e1}}(p)$ and $f_{L_{e2}}(p)$, respectively (note these fractions are from the edge perspective in the bipartite code graph of the outer code). This will in turn serve as the pdf of the *a priori* information $f_{L_{o,x}}^{(k+1)}$ to the inner decoder.

A similar serial update procedure can also be used with PA-II codes. With conventional $(K_1+1, K_1)\times(K_2+1, K_2)$ TPC/SPC codes (using block interleavers and parity-on-parity bits) as the outer code, the means of the extrinsic messages associated with row code and column code, $L_{e1}(\cdot)$ and $L_{e2}(\cdot)$, can be computed using (also refer to Appendix I for the decoding algorithm of TPC/SPC codes)

$$f_{L_{e1}}^{(k,l)} = \mathcal{R}^{*K_1}\left(f_{L_e}^{(k)} * f_{L_{e2}}^{(k,l-1)}\right) \tag{47}$$
$$f_{L_{e2}}^{(k,l)} = \mathcal{R}^{*K_2}\left(f_{L_e}^{(k)} * f_{L_{e1}}^{(k,l)}\right). \tag{48}$$

Unlike the general case of PA-I codes, data and parity bits are treated exactly the same in the outer code of PA-II codes. Hence, the pdf of the messages passing along to the inner $1/(1+D)$ decoder is given by $(f_{L_{e1}}^{(k,L)} * f_{L_{e2}}^{(k,L)})$ after L rounds of local iterations.

It should be noted that although PA-II codes can be viewed as a subclass of PA-I codes, the use of block interleavers and the existence of many length-8 cycles in the outer code even when $N \to \infty$ limits the application of density evolution, since density evolution assumes that all messages passed are i.i.d. For PA-I codes, it is reasonable to assume that the neighborhood of each node is tree-like due to the use of random interleavers. However, for PA-II codes, partial independence holds only when the message flow in the decoding has not closed a length-8 cycle. In other words, the number of times (47) and (48) can be applied consecutively is strictly limited to be no more

than $\log_2 \frac{8}{2} = 2$, before messages need to be passed out to the $1/(1+D)$ decoder. In fact, due to the serial update, even two local iterations will incur the looping of the same message [14] and, hence, we take $L = 1$ for analysis on PA-II codes. Furthermore, during every global iteration (k), the extrinsic messages within the TPC/SPC code generated in the previous iterations, $L_{e1}^{(k-1,L)}$ and $L_{e2}^{(k-1,L)}$, should not be used again since this represents correlated information. Due to the above reasons, the resulting thresholds for PA-II codes are upper bounds (pessimistic case) while the results for PA-I codes are exact.

2) Message Flow Within the Inner $1/(1+D)$ Decoder: Similar to the treatment of TPC/SPC codes, we assume that messages (LLRs) are i.i.d. for the $1/(1+D)$ code. From (10) and (11), it is obvious that for sufficiently long sequences, messages $L_{e_f}(y)$ and $L_{e_b}(y)$ follow the same distribution. Note that we are somewhat abusing the notation here by dropping the dependence on i, which denotes the transmission at the ith epoch. This is because on a memoryless channel the pdfs of $L_{e_f}(y_i)$ and $L_{e_b}(y_i)$ are independent of i. Further, as can be seen from the message-passing algorithm, the forward and the backward passes are symmetric and, hence, for large block sizes, $L_{e_f}(y)$ and $L_{e_b}(y)$ follow the same pdfs. Thus, we drop the subscript and use $L_e(y)$ to represent both $L_{e_f}(y)$ and $L_{e_b}(y)$. It was verified by simulations that the serial (see (6) and (7)) and parallel (see (10) and (11)) modes do not differ in performance significantly (only about 0.1 dB as shown in [39]), especially with sufficient number of turbo iterations. It is convenient to use the parallel mode for analysis here. Hence messages (LLRs) as formulated in (4) and (10) and (11) have their pdfs evolve as

$$f_{Le,x}^{(k)} = \mathcal{R}^{*2}\big(f_{L_{ch,y}} * f_{L_{e,y}}^{(k)}\big) \tag{49}$$

where

$$f_{L_{e,y}}^{(k)} = \mathcal{R}^{*}\big(\, f_{L_{o,x}}^{(k-1)},\, f_{L_{ch,y}} * f_{L_{e,y}}^{(k-1)}\,\big). \tag{50}$$

The initial conditions are $f_{L_{ch,y}} = \mathcal{N}(2/\sigma^2, 4/\sigma^2)$ (Gaussian distribution of mean $2/\sigma^2$ and variance $4/\sigma^2$) and $f_{L_{o,x}}^{(0)} = f_{L_{e,y}}^{(0)} = \delta(x)$ (Kronecker delta function).

The message flow between the inner and outer codes is straightforward. The pdf of the outbound message, $f_{L_{e,x}}^{(k)}$ in (49), becomes the pdf of the *a priori* information, $f_{L_e}^{(k)}(d)$ and $f_{L_e}^{(k)}(p)$ in (43)–(46) (PA-I code) and $f_{L_e}^{(k)}$ in (47) and (48) (PA-II code). Likewise, the pdf of the extrinsic information from the outer TPC/SPC code

$$\left(\frac{t}{t+1}(f_{L_{e1}}^{(k,L)}(d) * f_{L_{e2}}^{(k,L)}(d)) + \frac{1}{2t+2} f_{L_{e1}}^{(k,L)}(p) + \frac{1}{2t+2} f_{L_{e2}}^{(k,L)}(p)\right)$$

for PA-I codes and $(f_{L_{e1}}^{(k,L)} * f_{L_{e2}}^{(k,L)})$ for PA-II codes, becomes the pdf of *a priori* information, $f_{L_{o,x}}^{(k+1)}$ in (50), for the inner $1/(1+D)$ code.

Fig. 4 shows the thresholds for PA-I codes for several rates $R \geq 0.5$. It can be seen that the thresholds are within 0.6 dB from the Shannon limit for binary phase-shift keying (BPSK) on an AWGN channel. The thresholds are closer as the rate increases suggesting that these codes are better at higher rates.

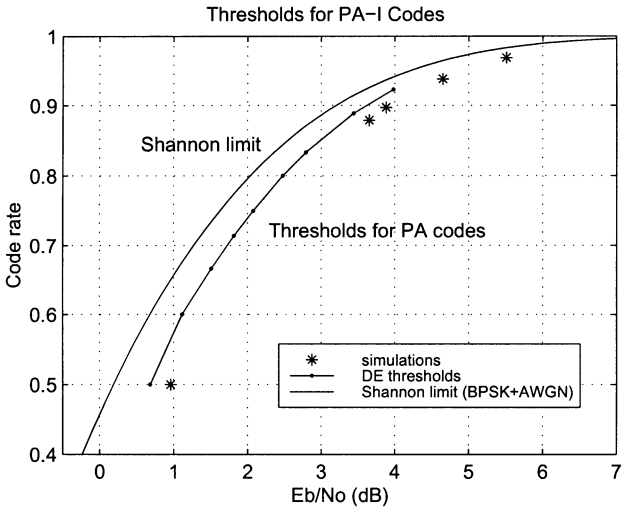

Fig. 4. Thresholds for PA-I codes (simulations are evaluated at BER $= 10^{-5}$).

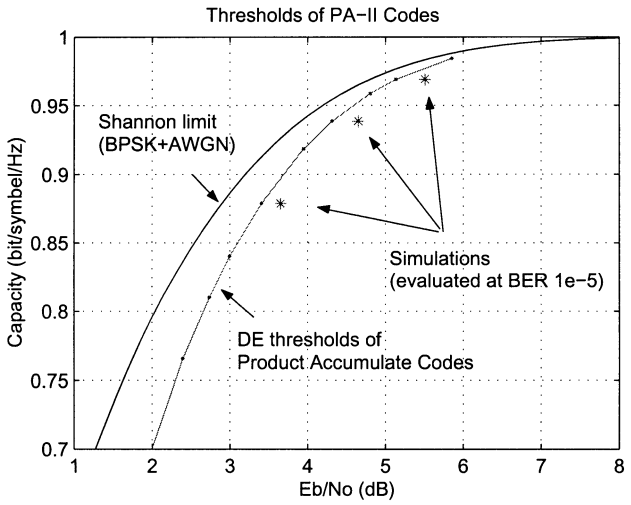

Fig. 5. Thresholds for PA-II codes (simulations are evaluated at BER $= 10^{-5}$).

The thresholds for PA-II codes are shown in Fig. 5. The plotted thresholds in Fig. 5 are a lower bound on the capacity (upper bound on the thresholds) since only one iteration is performed in the outer TPC/SPC decoding in each turbo iteration (i.e., $L = 1$ in (48)) [14]. Note that at high rates $(R > 0.7)$, the capacity of PA codes (both PA-I and PA-II) is within 0.5 dB from the Shannon limit. However, at lower rates, the gap becomes larger especially for PA-II codes. Simulation results for fairly long block sizes are also shown in both Figs. 4 and 5. A block size of $K = 65\,536$ data bits was used for $R = 1/2$ and for the higher rates $K = 16\,384$ was used and a BER of 10^{-5} is taken as reference. It can be seen that the simulation results are quite close to the thresholds. This shows that both PA-I and PA-II codes are capable of good performance at high rates, however, at lower rates PA-I codes are better.

VI. Algebraic Interleaver

Observe that a rate-K/N PA-I code involves two random interleavers of sizes K and N, where K and N are the user data block size and codeword block size, respectively. Interleaving

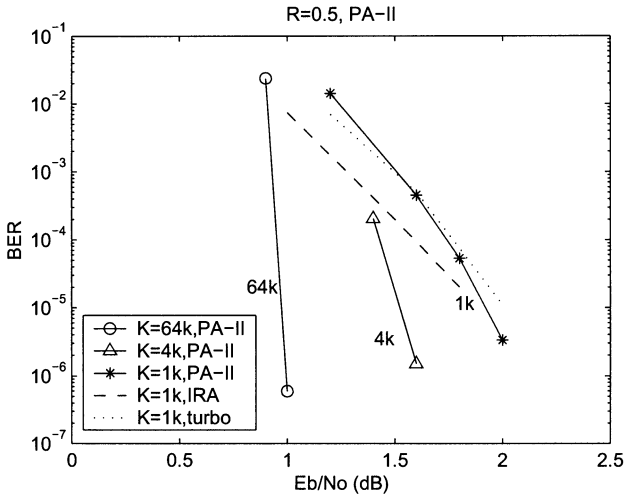

Fig. 6. Performance of PA-I codes at rate-$1/2$.

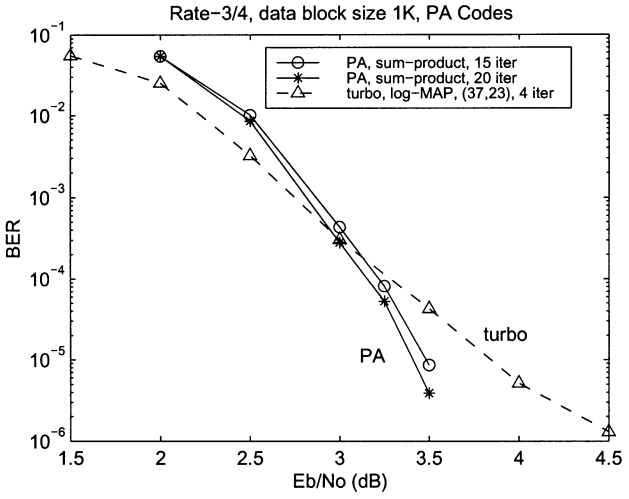

Fig. 7. Performance of PA-I codes at rate-$3/4$.

and deinterleaving using lookup tables can be quite inefficient in hardware and, hence, we study the performance of PA codes under algebraic interleaving. That is, we use interleavers where the interleaving pattern can be generated on the fly without having to store the interleaving pattern. We consider congruential sequence generated according to [44]

$$A_{n+1} = (a \cdot A_n + b) \bmod N. \tag{51}$$

To assure that this generates a maximal length sequence from 0 to $N-1$, parameters a and b need to satisfy: 1) $a < N, b < N$,b be relatively prime to N; 2) $(a-1)$ be a multiple of p, for every prime p dividing N; and 3) particularly, $(a-1)$ be a multiple of 4 if N is a multiple of 4. It is also desirable though not essential that a be relatively prime to N. We consider such an interleaver for both the interleavers in the proposed code. This can also be considered as an algebraic design of the code graph since the graph structure can be directly specified by the interleaving sequence. Hence, given an N and t, the choice of a and b completely specifies the code graph and, hence, the encoding and decoding operations.

Another direct benefit of using algebraic interleavers is that it allows great flexibility for PA codes to change code rates as well as code length. With LDPC codes, however, it is not easy to change code lengths nor code rates using one encoder/decoder structure. Although LDPC codes can be defined with a bit/check degree profile and a random interleaver (see Fig. 13), encoding requires the availability of the generator matrix. In other words, with LDPC codes, for each code rate and code length, not only does the code structure (connections between bits and checks) need to be devised specifically, but the generator matrix needs to be stored individually. Although possible, it requires special treatment to accommodate several rates/block sizes in one LDPC encoder/decoder pair.

VII. Simulation Results of PA Codes

Performance of PA-I Codes at Medium Rate—Fig. 6 shows the performance of a rate-$1/2$ PA-I code of data block size $65\,536$, 4096, and 1024, respectively. As can be seen, the larger the block size, the steeper the performance curve, which clearly depicts the interleaving gain phenomenon. For comparison, the performance of a $(2048, 1024)$ turbo code from [22] and the most recently reported IRA codes [22] of the same parameters are also shown. As can be seen, $(2048, 1024)$ PA-I codes perform as well as the turbo codes at BER of 10^{-5} with no error floors. From Table II, we can see that the decoding complexity of rate-$1/2$ PA codes with 30 iterations is approximately 1/16 that of a 16-state turbo code with eight iterations. It is also important to note that the complexity savings are higher as the rate increases, since the decoding complexity of punctured turbo codes does not reduce with increasing rate, whereas the decoding complexity of PA codes is inversely proportional to the rate. It should also be noted that the curve of PA-I codes is somewhat steeper than that of turbo codes or IRA codes, and therefore may outperform them at lower BERs.

Performance of PA-I Codes at High Rate—As indicated by both ML-based and iterative-based analysis, PA codes are most advantageous at high rates. Fig. 7 compares the performance of a rate-$3/4$ PA-I code at fifteenth and twentieth iteration with a 16-state turbo code of polynomials $(37, 23)$ at fourth iteration. Data block size is $K = 1024$ for both codes. Clearly, while a PA-I code is comparable to a turbo code (Fig. 6) at rate-$1/2$, it significantly outperforms turbo codes at rate-$3/4$ (much steeper curves and no error floors). Further, the PA-I code at fifteenth and twentieth iteration requires only about 23% and 30% the complexity of the turbo code at fourth iteration, respectively. Hence, PA codes are expected to be useful at high rates with the advantages of low complexity, high performance, and no observable error floors.

Performance of PA-II Codes—Fig. 8 plots the BER performance of PA-II codes at high rates. The codes simulated have rates $0.88, 0.94$, and 0.97, which are formed from $(16, 15)^2$, $(32, 31)^2$, and $(64, 63)^2$ outer 2-D TPC/SPC codes, respectively. Since interleaving gain is directly proportional to the number of TPC/SPC blocks in a codeword, several TPC/SPC blocks may be combined to achieve a large effective block size when needed. Corresponding threshold bounds calculated by density evolution are also shown. Two things can be immediately seen from the plot: 1) PA codes demonstrate a significant performance improvement than plain TPC/SPC

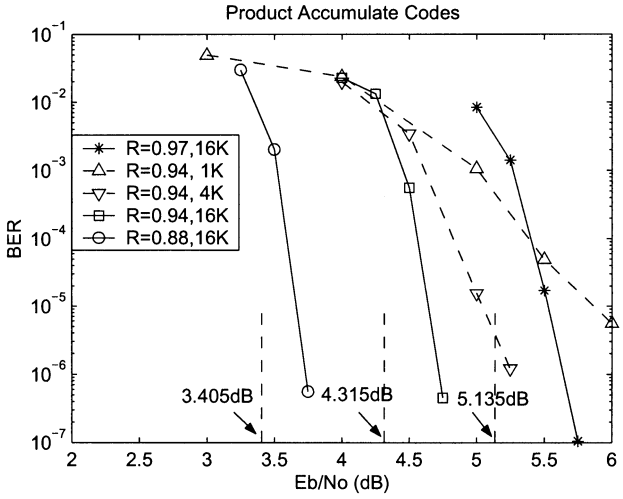

Fig. 8. Performance of PA-II codes at high rates.

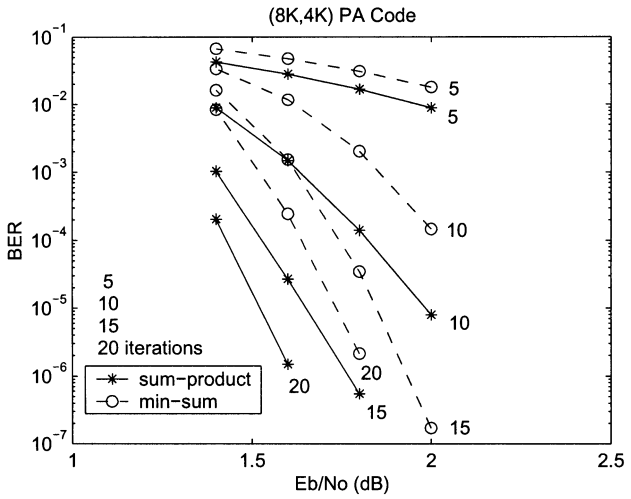

Fig. 10. Sum-product decoding vs min-sum decoding.

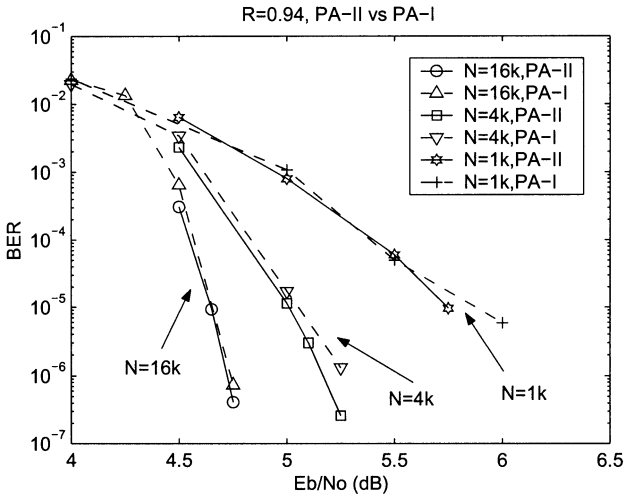

Fig. 9. Comparison of PA-I and PA-II codes at high rates.

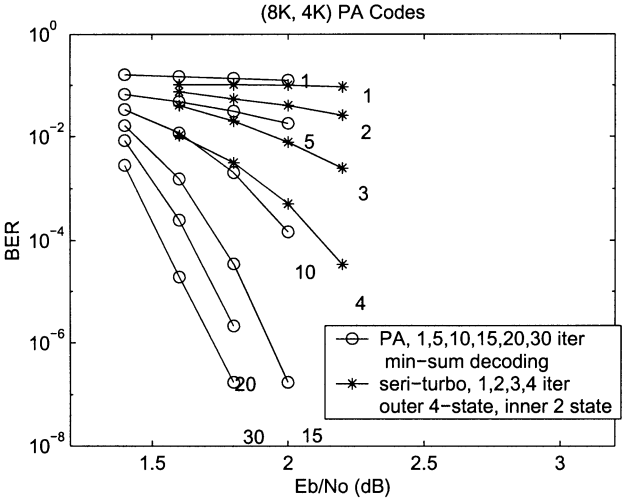

Fig. 11. Min-sum decoding of PA-I code.

codes. Although the performance of the plain TPC/SPC codes are not shown due to space limitation, we observe that the PA code outperforms the plain TPC/SPC code by 1 dB at rate 0.97, and as much as 3 dB at rate 0.88; 2) with a data block size of $K = 16\,000$, the performance of PA-II codes is within 0.3 dB from the capacity bound at BER of 10^{-5}, showing a very good fit. All curves shown are after 15 turbo iterations. Although not plotted here, a reduction from 15 to 8 iterations incurs only about 0.1-dB loss.

As mentioned before, PA-II codes are slightly simpler in implementation and PA-I codes have better BER performance. Although the plot is not presented in this paper, simulations show that at medium rates of around 0.5, PA-I codes can outperform PA-II codes by as much as 0.8 dB, but at high rates like $R \gtrapprox 0.88$, the difference in performance is negligible. Hence, for high rates, PA-II codes seem to be an attractive choice (see Fig. 9).

Performance of the Min-Sum Decoding—Fig. 10 compares the performance of a rate-0.5 PA-I code with the sum-product decoding to the low-complexity min-sum decoding. Performance at 5, 10, 15, and 20 iterations is evaluated. For all of the iterations, min-sum decoding incurs only about a 0.2-dB loss, while saving more than half the complexity. Hence, the min-sum algorithm provides a good tradeoff between performance and complexity, and is thus appealing for simple, low-cost systems. Fig. 11 compares the performance of PA-I codes using min-sum decoding to the performance of a serial concatenated convolutional codes (SCCC or serial turbo) of the same code rate and block size. The serial turbo code is composed of an outer four-state and an inner two-state convolutional code. It is interesting to see that even with the low-complexity min-sum decoding, the PA-I code still outperforms the SCCC code. Comparing the performance of the PA-I code (using min-sum decoding) at fiftteenth iteration with that of the serial turbo at fourth iteration, we see that a 0.4-dB performance gain is achieved at BER of 10^{-5} with only about 60% of the complexity (see Table II for a complexity analysis).

Algebraic Interleaver Versus S-Random Interleaver—Fig. 12 compares the performance of a rate-0.5 PA-I code with S-random interleavers and algebraic interleavers. Interestingly, replacing S-random interleavers with algebraic interleavers results in hardly any performance degradation. Since the length

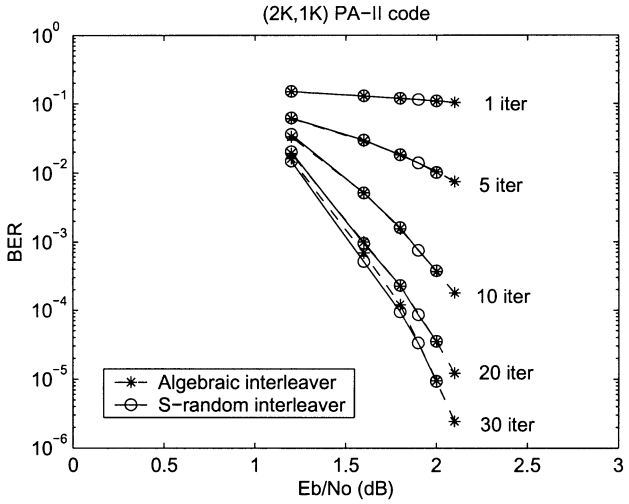

Fig. 12. Algebraic interleaver versus S-random interleaver.

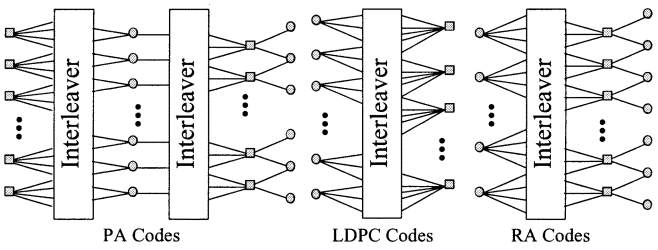

Fig. 13. Tanner graphs of PA codes, LDPC codes, and RA codes.

of algebraic interleavers can be conveniently changed, using algebraic interleavers can provide another degree of flexibility to PA codes.

VIII. Comparison to Other Related Codes

Graphical representation of codes has shed great insight into the understanding of many codes [27]–[30], including turbo codes, LDPC codes and (irregular) repeat accumulate (RA/IRA) codes [22], [24]. This section revisits PA codes from the graph perspective for a comparison and unification of PA codes and other capacity-approaching codes.

The Tanner graph structure shown in Fig. 13 reveals that PA codes are essentially LDPC codes with two levels of checks instead of one as in conventional LDPC codes (small circles denote bits and small boxes denotes checks). However, the encoding complexity of PA codes is linear (without the need for preprocessing) and the encoder is cheap to implement since it does not require explicit storage of a generator matrix.

RA codes [24], and their improved version, IRA codes [22], are a class of very nice codes, which are linear time encodable and provide near-capacity performance. A careful study of the code graph shows an intrinsic connection between the structure of the proposed PA codes and RA/IRA codes, although our initial motivation for PA codes was from the encouraging performance of TPC/SPC codes over partial response channels in magnetic recording systems [4]. Fig. 13 presents the Tanner graphs of the proposed PA codes and RA/IRA codes. One difference, however, is that a PA code is nonsystematic whereas the systematic bits are transmitted explicitly in the IRA code [22]. Nonsystematic IRA codes were mentioned as not desirable since density evolution would not evolve beyond one iteration in the outer code. However, since PA codes can be essentially viewed as a special type of (regular) IRA codes, we have shown that by making the outer encoder to be systematic, nonsystematic IRA codes can nevertheless be promising codes. Further, due to the irregular nature of the IRA code construction, careful interleaver design is needed to prevent high error floors for finite block sizes, especially for high rates. PA codes can therefore be seen the counterpart of IRA codes or the special form of IRA codes that are well suited for higher rates and easy rate adaptivity.

The recently proposed class of concatenated tree (CT) codes also has low encoding and decoding complexities and good performance [46]. CT codes are by definition a class of very general codes since any code that has a tree-like "code graph"[2] can be concatenated to form a CT code. One particular realization of a rate-$1/2$ CT code discussed in [46] has demonstrated impressive performance and, hence, is proposed to be a lower complexity alternative to turbo codes [46]. Simulation results for a rate-$1/2$ PA code show that the BER performance and decoding complexity are similar to those of the CT code discussed in [46]. In fact, the decoding complexity seems to be slightly lower for PA codes. However, the performance of a CT code depends a lot on the exact code structure (puncturing pattern and node degrees), the optimization of which also appears to be mainly through trial and error. The advantage of PA codes is in the ease of code construction and rate adaptivity. It should also be noted that the performance of PA codes is on par with finite-geometry LDPC-based codes proposed by Kou, Lin, and Fossorier [47].

All these codes can be viewed as subclasses of LDPC-like codes. It is worth noting that PA codes are more like *a* fixed code with well-defined structure (the only randomness and variations come from the interleavers), while IRA codes, (irregular) LDPC codes and CT codes are more like *a set* of random codes whose performances depend on and vary with the individual degree profile and/or code graph. Hence, for IRA, irregular LDPC, and CT codes, a random pick does not guarantee good performance and code design and optimization are usually needed to achieve the best performance. On the other hand, a random pick of a PA code tends to yield the same desirable performance so long as the interleavers are reasonably chosen. It follows that PA codes can be rate/length adaptive which is desired in real applications.[3] This regular structure also facilitates hardware implementation and prevents having to store different graphs for different rates and lengths at the receiver. In this sense, PA codes are more like turbo codes, both of which enjoy well-defined structure, predictable performance and rate/length adaptivity. The performance of carefully optimized IRA codes will be better than PA codes for long lengths, but the difference for high rates is small and for smaller lengths, simulation results show that PA codes have lower error floors.

[2]Note the "code graph" in here is different and more relaxed than the Tanner graph or the factor graph used to describe LDPC, IRA, and PA codes.

[3]Although in principle, puncturing can be used to change code rate and length, punctured IRA and LDPC codes tend to significantly degrade the performance.

TABLE III
DECODING ALGORITHM FOR 2-D TPC/SPC CODES

Initialization:

for $i = 1$ to N_2, for $j = 1$ to N_1,

$L_{i_{i,j}} = \frac{2}{\sigma^2} r_{i,j}$,

$Le_{i,j}{}^{(1)} = Le_{i,j}{}^{(2)} = 0$,

Iterations:

Decoding row code $\mathcal{C}_1$: for $i = 1$ to N_2, for $j = 1$ to N_1,

$Le_{i,j}{}^{(1)} = 2\tanh^{-1}(\prod_{1\le t\le N_1, t\neq j}\tanh(\frac{Li_{i,t}+Le_{i,t}{}^{(2)}}{2}))$,

Decoding column code $\mathcal{C}_2$: for $j = 1$ to N_1, for $i = 1$ to N_2,

$Le_{i,j}{}^{(2)} = 2\tanh^{-1}(\prod_{1\le t\le N_2, t\neq i}\tanh(\frac{Li_{t,j}+Le_{t,j}{}^{(1)}}{2}))$,

Soft output and decision:

for $i = 1$ to N_2, for $j = 1$ to N_1,

$LLR_{i,j} = Li_{i,j} + Le_{i,j}{}^{(1)} + Le_{i,j}{}^{(2)}$,

$\hat{s}_{i,j} = LLR_{i,j} > 0\ ?\ 0 : 1$;

Iteration stop criteria:

Success: All rows and columns add up (modulo-2) to 0.

Fail: A max number of iteration is reached.

IX. CONCLUSION AND FUTURE WORK

A class of interleaved serially concatenated codes, called *product accumulate* codes, have been constructed and shown to possess good BER performance, linear encoding/decoding complexity, as well as an efficient soft-decoding algorithm. The proposed codes consist of an outer 2-D TPC/SPC code, a random interleaver, and a rate-1 recursive convolutional inner code of the form $1/(1 + D)$. The main advantages of the proposed codes are very low decoding complexity compared to turbo codes especially for high rates, good BER performance, ease of implementation, and rate/length adaptivity. Through analysis and simulations, we show this class of proposed codes perform well for almost all rates $R \geq 1/2$ and for long and short block sizes alike. Further error floors do not appear until at very low error rates.

Future work on PA codes includes extension to rates below $1/2$ by packing more levels of single-parity check codes in the outer TPC/SPC codes [45]. Irregularity can be introduced to the code structure to further improve performance; however, one advantage of PA codes seems to be regular and simple structure which makes implementation easier. Future work will also involve extending to fading channels and multilevel modulation [48].

APPENDIX
DECODING ALGORITHM FOR TPC/SPC CODES

Assuming even-parity check codes, BPSK modulation $(0 \rightarrow +1, 1 \rightarrow -1)$, and AWGN channels, a 2-D TPC/SPC code formed from $(N_1, N_1 - 1) \otimes (N_2, N_2 - 1)$ has the following SISO decoding algorithm (Table III), where $r_{i,j}$ denotes the bits received from the channel, $L_{i_{i,j}}$ denotes the *a priori* information (obtained from the channel or the inner code in a concatenated scheme), $\text{LLR}_{i,j}$ denotes the LLR, and $Le_{i,j}{}^{(1)}$ and $Le_{i,j}{}^{(2)}$ denotes the extrinsic information associated with component code $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

REFERENCES

[1] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[2] J.-F. Cheng and R. J. McEliece, "Some high-rate near capacity codes for the Gaussian channel," in *Proc. Allerton Conf. Communication, Control, and Computing*, 1996, pp. 494–503.

[3] J. Li, K. R. Narayanan, and C. N. Georghiades, "A class of linear-complexity, soft-decodable, high-rate, 'good' codes: Construction, properties and performance," in *Proc. Int. Symp. Information Theory*, Washington, DC, June 2001, p. 122.

[4] J. Li, E. Kurtas, K. R. Narayanan, and C. N. Georghiades, "On the performance of turbo product codes over partial response channels," *IEEE Trans. Magn.*, vol. 37, pp. 1932–1934, July 2001.

[5] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[6] D. Divsalar, "A simple tight bound on error probability of block codes with application to turbo codes," TMO, Progr. Rep. 42-139, Nov. 1999.

[7] T. J. 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.

[8] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. PGIT-4, pp. 29–37, Sept. 1954.

[9] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[10] T. Souvignier, C. Argon, S. W. McLaughlin, and K. Thamvichai, "Turbo product codes for partial response channels," in *Proc. Int. Conf. Communications*, vol. 7, 2001, pp. 2184–2188.

[11] R. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

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

[13] I. Dumer, "Concatenated codes and their generalizations," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands, 1998, ch. 23, pp. 1911–1988.

[14] J. Li, K. R. Narayanan, E. Kurtas, and C. N. Georghiades, "On the performance of high-rate TPC/SPC and LDPC codes over partial response channels," *IEEE Trans. Commun.*, vol. 50, pp. 723–734, May 2002.

[15] G. Caire, C. Taricco, and G. Battail, "Weight distribution and performance of the iterated product of single-parity-check codes," in *Proc. IEEE GLOBECOM Communications Theory Mini-Conf.*, 1994, pp. 206–211.

[16] M. Blaum, P. G. Farrell, and H. C. A. van Tilborg, "Array codes," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: North-Holland, 1998, pp. 1855–1909.

[17] P.-P Sauve, A. Hunt, S. Crozier, and P. Guinand, "Hyper-codes: High-performance, low-complexity codes," in *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 2000.

[18] J. Hagenaur, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[19] D. Divsalar and F. Pollara, "Serial and hybrid concatenated codes with applications," in *Proc. Int. Symp. Turbo Codes and Applications*, Sept. 1997, pp. 80–87.

[20] K. R. Narayanan and G. L. Stuber, "A serial concatenation approach to iterative demodulation and decoding," *IEEE Trans. Commun.*, vol. 47, pp. 956–961, July 1999.

[21] M. Peleg, I. Sasson, S. Shamai (Shitz), and A. Elia, "On interleaved, differentially encoded convolutional codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2572–2582, Nov. 1999.

[22] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 2000.

[23] S. B. D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, May 1998.

[24] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. 1998 Allerton Conf. Communications and Control*, Sept. 1998, pp. 201–210.

[25] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.

[26] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 219–230, Feb. 1998.

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

[28] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. dissertation, 1996.

[29] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*. Cambridge, MA: MIT Press, 1998.

[30] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520–548, Feb. 2001.

[31] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[32] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. Int. Conf. Communications*, Seattle, WA, 1995.

[33] G. Poltyrev, "Bounds on the decoding error probability of binary linear codes via their spectra," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1284–1292, July 1994.

[34] A. J. Viterbi and A. M. Viterbi, "An improved union bound for binary input linear codes on the AWGN channel, with applications to turbo decoding," in *Proc. IEEE Inform. Theory Workshop*, San Diego, CA, Feb. 1998.

[35] T. M. Duman and M. Salehi, "New performance bounds for turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 717–723, June 1998.

[36] B. Hughes, "On the error probability of signals in additive white gaussian noise," *IEEE Trans. Inform. Theory*, vol. 37, pp. 151–155, Jan. 1991.

[37] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 429–445, Mar. 1998.

[38] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[39] J. Li, K. R. Narayanan, and C. N. Georghiades, "An efficient decoding algorithm for cycle-free convolutional codes and its applications," in *Proc. IEEE Global Communications Conf.*, San Antonio, TX, Nov. 2001, pp. 1063–1067.

[40] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[41] T. J. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[42] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proc. ACM Symp.*, 1998, pp. 249–258.

[43] S.-Y. Chung, R. Urbanke, and T. J. Richardson, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[44] G. C. Clark Jr and J. B. Cain, *Error-Correction Coding for Digital Communications*. New York: Plenum, 1981.

[45] J. Li, K. R. Narayanan, and C. N. Georghiades, "Generalized product accumulate codes: analysis and performance," in *IEEE Proc. Global Conf. Communications*, San Antonio, TX, Nov. 2001, pp. 975–979.

[46] L. Ping and K. Y. Wu, "Concatenated tree codes: a low complexity, high-performance approach," *IEEE Trans. Inform. Theory*, vol. 47, pp. 791–799, Feb. 2001.

[47] Y. Kou, S. Lin, and M. Fossorier, "Construction of low density parity check codes: a geometric approach," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 2000.

[48] K. R. Narayanan, J. Li, and C. N. Georghiades, "Product accumulate codes on fading channels," in *Proc. 36th Asilomar Conf. Signals, Systems and Computers*, Dec. 2002.