Generalized Product Accumulate Codes: Analysis and Performance

Jing Li, Krishna R. Narayanan, and Costas N. Georghiades
Department of Electrical Engineering
Texas A&M University, College Station, TX 77843

Abstract— **In [1] [2], product accumulate (PA) codes were proposed and shown to be a class of simple and provably good codes for rate $R \geq 1/2$. This work investigates the** *generalized product accumulate* **(GPA) codes which have rates over the entire range and which are also "good" both in the maximum likelihood (ML) sense and under the iterative approach. Analysis concentrates on the weight distribution over the code ensemble, the ML bounds, and the existence and computation of threshold phenomenon in the iterative decoding. A tight upper bound due to Divsalar and the thresholds computed using density evolution are examined. Simulations are presented and evaluated, especially for rate $R \leq 1/2$.**

I. Introduction

Product accumulate (PA) codes, proposed in [1] [2] are a class of interleaved serial concatenated codes where the inner code is a rate-1 recursive convolutional code $1/(1+D)$ and the outer code is a parallel concatenation of 2 single-parity check (SPC) codes. They are promising codes in that (1) they are provably "good" (see definition below) both in the maximum likelihood (ML) sense and under the iterative decoding, (2) the performance is within a few tenths of a dB from the capacity at rate $R \geq 1/2$, (3) they are linear time encodable and decodable, and (4) the decoding procedure is highly parallelizable. As a benchmark, a (2000,1000) PA code performs a few tenths of a dB away from the best known turbo code yet with much lesser complexity. However, the proposed product accumulate codes therein only have rates $R \geq 1/2$. In this paper, we first extend product accumulate codes to rates below 1/2. We then compute tight upper bounds on the performance of these codes (for all rates) with ML decoding based on the *simple bound* proposed by Divsalar [4]. Thresholds under the iterative decoding are also examined. These results show that this class of codes can achieve performance close to the capacity limit. They also illustrate the asymptotic difference between ML decoding and iterative decoding performance for this class of codes.

For the analysis of GPA codes, we focus on the two principle issues: (1) to investigate the properties of the ensemble of these codes and (2) to investigate the performance of the codes with the iterative decoding algorithm. For the former, the concentration is on the ensemble average of weight distribution and the quantification of the interleaving gain. Several bounds based on distance spectrum, including the union bound and the simple bound [4], are computed. For the latter, we use density evolution (DE) [5] [6] [7] to compute the thresholds (in the iterative sense) for GPA codes. In [8], a "good" code is defined as a code for which there exists a threshold above which an arbitrary low error rate can be achieved as code length $N \to \infty$. The upper bound on the performance of these codes under ML decoding and the threshold computed using density evolution indicate that GPA codes are provably "good" both in the ML sense and in the iterative sense. Further, comparing the bounds calculated under ML assumption and in the iterative setting, the performance loss due to the suboptimality of iterative decoding as well as their implication in the code design can be obtained.

The paper is organized as follows. Section 2 introduces the system model of GPA codes. Section 3 and 4 analyzes GPA codes from the ML perspective and from the iterative perspective, respectively. Section 5 discusses simulation results. Section 6 concludes the paper.

II. Structure of GPA Codes

Product accumulate codes proposed in [1] [2] are formed by concatenating an outer code with an rate-1 accumulator (inner code). The outer code is itself a parallel concatenation of 2 single-parity check (SPC) codes. For a generalized product accumulate code proposed here, the number of parallel branches need not be restricted to 2. As shown in Fig. 1, $M \geq 2$ branches of $(t+1, t)$ SPC codes are interleaved and parallelly concatenated as the outer code. Further, as will be shown later, in order to obtain interleaving gain, q blocks of SPC codewords need to be combined and jointly interleaved in each branch. The resulting GPA code thus has rate $R = t/(t+M)$ and length $N = q(t+M)$. It is interesting to note that in an extreme case when SPC codes are reduced to (2,1) repetition codes, then the corresponding GPA codes are reduced to regular repeat accumulate (RA) codes [9], which, despite their simplicity, have demonstrated surprisingly good performance and are shown to have the potential for achieving AWGN channel capacity [3]. This also holds for GPA codes. RA codes achieve good performance primarily at very low rates; however, GPA codes are capable of good performance for a wide rate range, like $R = 1/3, 1/2, 2/3$. The capability to provide good performance for a wide range of rates using one simple structure is very useful in a variety of practical applications.

The decoding of GPA codes is via an iterative decoding employing the turbo principle (message passing decoding). Soft information in log-likelihood ratio (LLR) form iterates among different component codes. An efficient sum-product algorithm and its reduced-complexity form, the min-sum algorithm, are described in [2] to decode product accumulate codes, and can be readily applied to GPA codes. Hence, GPA codes are also linear-time encodable and decodable.

III. ML-based Analysis

A. Weight Distribution and Interleaving Gain

Assuming ML decoding, we quantify the interleaving gain of GPA codes by investigating their weight distributions. From Divsalar *et al*'s results [9] and Benedetto *et al*'s results [10], we know that for a general serial concatenated system with a recursive inner code, there exists a threshold γ such that for any $E_b/N_o \geq \gamma$, the asymptotic word error rate is upper bounded by:

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

where d_m^o is the minimum distance of the outer code and N is the interleaver size. Although the above results offers a useful guideline for designing concatenated schemes, it is well worth computing the exact interleaving for GPA codes since it reveals somewhat important and interesting results that are not obvious from (1).

The result in (1) indicates that in order to obtain an interleaving gain, the minimum distance of the outer code needs to be at least 3. However, the outer codewords of GPA codes (with random interleavers) have minimum distance of only 2. On the other hand, if S-random interleavers are used such that bits within S distance are mapped to at least S distance apart, then the outer codewords are guaranteed to have a minimum distance of at least 3 as long as $S \geq t$. Below we show that although the minimum distance of the outer codewords over the ensemble of all interleavers is only 2, an interleaving gain still exists for GPA codes with random interleavers. From [10], we know that outer codewords of weight 3 or more will lead to an interleaver gain. Hence we focus on weight-2 outer codewords only, and show that the number vanishes as q increases (here all-zero sequence is used as the reference).

It is convenient to employ the uniform interleaver which represents the average behavior of the ensemble of the codes. Denote $A^{(j)}_{wh}$ as the *input output weight enumerator* (IOWE) of the j_{th} SPC branch code. The IOWE of the outer code, $A^{(o)}_{w,h}$, with 2 parallel concatenated branches ($M = 2$) averaged over the code ensemble is given as:

$$A^{(o)}_{w,h} = \sum_{h_1} \frac{A^{(1)}_{w,h_1} A^{(2)}_{w,h-h_1}}{\binom{K}{w}}, \;\; M = 2, \tag{2}$$

where K is the input sequence length.

Define the *input output weight transfer probability* (IOWTP) of j_{th} branch code, $P^{(j)}_{w,h}$, as the probability that a particular input sequence of weight w is mapped to an output sequence of weight h: $P^{(i)}_{w,h} = A^{(i)}_{w,h}/\binom{K}{w}$. Using induction on (2), we have the average IOWE for the M-branch parallel concatenation as:

$$A^{(o)}_{w,h} = \sum_{h_1+h_2+\cdots+h_M=h} A^{(1)}_{w,h_1} P^{(2)}_{w,h_2} P^{(3)}_{w,h_3} \cdots P^{(M)}_{w,h_M}, \; \forall M \geq 2. \tag{3}$$

Since all component codes are $(t+1,t)$ SPC codes, $P^{(2)}_{w,h} = P^{(3)}_{w,h} = \cdots = P^{(M)}_{w,h}$, and $K = qt$.

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

$$A^{SPC}(w,h) = \left(\sum_{j=0}^{t} \binom{t}{j} w^j h^{2\lceil j/2 \rceil} \right)^q, \tag{4}$$

where the coefficient of the term $w^u h^v$ denotes the number of codewords with input weight u and output weight v. Using (4), we can compute the IOWEs of the first SPC branch code, denoted as $A^{(1)}_{u,v}$ $(= A^{SPC}_{u,v})$. For all other SPC branch codes, since only parity bits are transmitted, $A^{(2)}_{u,v} = A^{(3)}_{u,v} = \cdots = A^{(M)}_{u,v} = A^{(1)}_{u,v+u}$. With a little computation, it is easy to see that the number of weight-2 outer codewords is given by:

$$\begin{aligned} A^{(o)}_{h=2} &= \sum_w A^{(o)}_{w,h=2} = q\binom{t}{2}\left(\frac{q\binom{t}{2}}{\binom{qt}{2}}\right)^{M-1}, \\ &= O\left(\frac{(t-1)^2}{2} q^{-(M-2)}\right), \;\; \forall M \geq 2, \end{aligned} \tag{5}$$

where the last equation assumes a large q (i.e. large block size). Equation (5) shows that the number of weight-2 outer codewords decreases at some power of q if $M \geq 3$. In other words, when there are at least 3 SPC branches, weight-2 outer codewords vanishes with the increase of block size and, hence, an interleaving gain exists.

For the case when $M = 2$, (5) says that the number of weight-2 outer codewords is a function of a single parameter, t, which is related to the rate of SPC codes. Now considering the serial concatenation of outer codewords with inner $1/(1+D)$ code, the overall output weight enumerator (OWE) A^{GPA}_h is computed as:

$$A^{GPA}_{h=s} = \sum_{h'} A^{(o)}_{h'} \frac{A^{1/(1+D)}_{h',h}}{\binom{N}{h'}} = \sum_{h'} \sum_w A^{(o)}_{w,h'} \frac{A^{1/(1+D)}_{h',h}}{\binom{N}{h'}}, \tag{6}$$

where the IOWE of $1/(1+D)$ code is given by [9]: $A^{1/(1+D)}_{w,h} = \binom{N-h}{\lfloor w/2 \rfloor}\binom{h-1}{\lceil w/2 \rceil - 1}$. In particular, when $M = 2$, the number of weight-s GPA codewords produced by weight-2 outer codewords, denoted as $A^{GPA2}_{h=s}$, is given as:

$$A^{GPA2}_{h=s} = \frac{(t-1)^2}{2} \frac{N-s}{\binom{N}{2}} = O(tq^{-1}), \;\; M=2, \tag{7}$$

where $N = q(t+2)$ is the GPA codeword length. This indicates that the impact of weight-2 outer codewords on the overall GPA codewords vanishes as the q increases. In other words, for GPA codes with $M = 2$, an interleaving gain also exists and is proportional to q.

B. ML Decoding Based bounds

In this section, we quantify the asymptotic performance of GPA codes with ML decoding, and show that these codes are capable of near-capacity performance. We do so by computing their upper bounds. Among the various bounding techniques developed, the union bound is the most popular but is fairly loose above cutoff rate. Tighter and more complicated bounds include the tangential sphere bound, Duman-Salehi bound, Viterbi and Viterbi bound, the Hughes bound, and etc. Recently, Divsalar developed a simple bound on the error probability in [4] and showed that the resulting bound is very tight when applied to RA codes, LDPC codes and turbo codes. These new tight bounds are essentially based on the bounding techniques developed by Gallager [11]:

$$\Pr(error) \leq \Pr(error, \bar{y} \in \Re) + \Pr(\bar{y} \notin \Re), \tag{8}$$

where $\bar{y}$ is the received codeword (noise-corrupted), 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

$\Re$. Here, we apply this bounding technique to the analysis of GPA codes.

We first quote and summarize the main results of [4]. Define *spectral shape* of a code, $r_N(\delta)$, as the normalized weight distribution averaged over the code ensemble $\mathbf{C}_N$:

$$r_N(\delta) := \frac{1}{N}\ln(A_{h=\lfloor \delta N \rfloor}), \ \ 0<\delta<1, \tag{9}$$

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

$$r(\delta) := \lim_{N\to\infty} r_N(\delta), \ \ 0<\delta<1. \tag{10}$$

A minimum threshold (in dB) can be computed as [4]:

$$(\frac{Eb}{No})_{min} = \frac{1}{R}\max_{0<\delta\leq(1-R)} c_0(\delta), \tag{11}$$

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

$$\text{Simple: } c_0(\delta) = \frac{1-\delta}{2\delta}\left(1-e^{-2r(\delta)}\right). \tag{12}$$

Since the above bound is 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.

The spectral shape for GPA codes can be computed since the component codes are simple single parity check codes and $1/(1+D)$ code; however, there does not seem to be a simple closed form expression. Nevertheless a numerical approach can be used; Using (3), (6) and (9) we can compute the spectral shape of GPA codes, which is a function of all the parameters of the code, including q, t, M. Fig. 2 compares the spectral shape of rate $1/2$ GPA codes with $M = 2, 4, 8$ and $N = 400$, respectively. As expected, for GPA codes of the same rate, larger M leads to better spectral shape (and therefore better code in the ML sense).

We approximate the ensemble spectral shape by choosing a large N. In general it is possible to encounter numerical problems when computing the spectral shape; however, these can be avoided by careful computation. Whenever possible, input output weight transfer probability, $P_{w,h}$, should be used instead of input output weight enumerator, $A_{w,h}$, 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). GPA codes with $M = 2, 4$ are evaluated. For comparison, also shown are the Shannon limit and the bounds for random codes and RA codes. Several things can be observed. (1) the simple bounds for GPA codes are very close to those of the random codes, indicating that GPA codes have good distance spectrum. Researchers have shown that irregular codes, like irregular LDPC codes or irregular repeat accumulate (IRA) codes [12] have bounds extremely close to the capacity. But for regular codes like GPA codes, 0.7 dB from the capacity is still impressive. (2) the higher the rates, the closer the bounds, indicating that GPA codes are probably more advantageous at high rates than low rates (as opposed to repeat accumulate codes). (3) the larger the value of M, the closer the bound, which matches with our analysis in the previous subsection. It is expected that as $M\to\infty$, the simple bound as well as the distance spectrum of GPA codes will converge to those of the random codes. In other words, like repeat accumulate codes, GPA codes also have the potential for achieving AWGN channel capacity such that, as the rate approaches 0, the average required Eb/No for arbitrarily small error probability with ML decoding approaches $\log 2$. This is obvious, for, as mentioned above, RA codes are the special case of GPA codes where all the (t+1,t) SPC codes have parameter $t = 1$. Further, it can be seen from the plot that, for RA code of a given rate, we can always find a better GPA code (in the ML sense) of the same rate by increasing both t and M (recall that GPA codes have rate $R = t/(t+M)$).

The implication of the above analysis is that GPA codes are by nature good codes, and that larger M (i.e. more SPC branches) leads to better distance spectrum. However, due to the lack of a practical ML decoder, this behavior may not be observable in practice. Since the performance of the suboptimal iterative decoder is also a function of M, it is necessary to investigate the iterative decoding process in order to give a more meaningful evaluation of the code performance.

IV. Analysis of GPA Codes under Iterative Decoding

We use density evolution [6] to evaluate the asymptotic performance of the iterative decoding. Here asymptotic refers to the assumption of infinite code lengths, perfect random interleavers and infinite number of iterations. The idea is to model how decoding would proceed on an infinite block size and to evaluate the distribution of the messages that are passed along the code graph in each step. Incorrect messages are defined as the messages that contribute towards a wrong decision, and the portion of incorrect messages is evaluated as error probability P_e.

Density evolution in general allows the computation of a threshold, if it exists, through a deterministic algorithm for a given code and a given decoding strategy. The threshold is defined as the minimum SNR value $\mathcal{C}$ required to guarantee zero error probability for infinite block size N and infinite iteration number l:

$$\mathcal{C} = \inf_{SNR}(SNR : \lim_{l\to\infty}\lim_{N\to\infty} P_e \to 0). \tag{13}$$

The component codes of the proposed GPA codes are single parity check codes and since the inner rate-1 code $1/(1+D)$ has a simple graph representation without any cycles, it is possible to extend the density evolution technique used to analyze LDPC codes to compute thresholds for GPA codes under message passing decoding. The exact steps for product accumulate codes are described in [2] and can be extended straightforwardly to GPA codes. Here, we omit the details but present and motivate the fundamental ideas and discuss the analytical results.

A GPA code can be represented using a bipartite graph of *bits* and *checks*, where messages are exchanged along the connecting edges. Representing messages in LLR form, the outgoing message along an edge is simply the "sum" of the received messages which includes messages from all edges (and the channel if applicable) except the message coming along this very edge. For bit nodes, this "sum" is a regular sum in the real domain, and thus the density of the outgoing message is the convolution of the densities of the message participating in this sum. For check nodes, it is a *check sum*, $\boxplus$, operated on messages:

$$\gamma = \alpha \boxplus \beta \iff \tanh\frac{\gamma}{2} = \tanh\frac{\alpha}{2} \cdot \tanh\frac{\beta}{2} \qquad (14)$$

Since no simple closed form is available on the relations of pdfs (probability density function) in a $\tanh$ or $\tanh^{-1}$ operation, a numerical method is taken which can determine the threshold to any desired degree of accuracy. To further simplify the computation, densities can be approximated as a mixture of Gaussian densities, which leads to only a slight decrease in accuracy [6]. It has been shown that for binary-input, output-symmetric memoryless channels and binary linear codes, the distribution f of messages passed in each step satisfies $f(x) = f(-x)e^x$ [5]. This *consistency constraint*, when applied to Gaussian distributions, leads to the constraint that variance of the message equals twice the mean. Hence, the mean of the message becomes the one single quantity to describe the process, which greatly reduces the complexity.

Fig. 4 plots the thresholds of GPA codes computed using density evolution for $M = 2, 3, 4$, respectively. As can be seen, at rates $R \geq 1/2$, even with iterative approach, GPA codes can perform close to capacity. At low rates, the thresholds are about 1 dB from the capacity. The plot shows that the more the branches, the worse the thresholds (and the more the complexity). That GPA codes with fewer SPC branches can perform better under iterative decoding is just opposite to what is inferred from the weight spectrum analysis. This disagreement indicates that the performance loss due to the sub-optimality of the iterative decoder may be quite severe in the presence of several component codes. Hence, in practise, it is desirable to use a small number of SPC branches with more powerful SPC codes in each branch. This way, the best performance is achieved with the least complexity. But with smaller M, the achievable rate range is also smaller ($\frac{2}{2+M} \leq R < 1$).

The codes are hence best suited for practical applications which involve changing the rate of the code constantly, where high rates ($R > 1/2$) are used most of the time, whereas occassionally due to poor channel conditions, lower rates are required. The regular structure of these codes makes it easy to change the rate at both the transmitter and then receiver.

V. Simulations

Fig. 5 shows the performance of a rate $1/2$ GPA code with 2 branches of (3,2) SPC codes and a rate $1/3$ GPA code with 4 branches of (3,2) SPC codes. Interleaving gain is obvious from the plot, and the performance is only 0.8 and 1.1 dB away from the Shannon limit, respectively.

Fig. 6 compares the performance of (2000,1000) GPA codes with two different settings: 2 branches of (3,2) SPC codes with 500 SPC codewords combined in each branch, and 4 branches of (5,4) SPC codes with 250 codewords combined in each branch. We see that 2-branch GPA codes outperform 4-branch GPA codes in addition to the saving of about $1/3$ of the complexity, which confirms the results from the iterative analysis. For comparison purpose, also shown is the performance of the turbo code of the same parameter [12]. Clearly, GPA code with $M = 2$ is as good as turbo codes (yet with lesser complexity). Further, there are no observable error floor due to the serial concatenation with a $1/(1 + D)$ inner code.

VI. Conclusion

Generalized product accumulate codes are investigated and shown to be provably "good" both in the ML sense and in the iterative sense. They have low complexity and the performance using sum-product decoding is close to the capacity over the entire code range. Bounds and thresholds are examined from both the ML perspective and the iterative perspective. Although the ML analysis favors for larger SPC branches, iterative analysis as well as simulation results indicate that, whenever possible, the number of SPC branches should be kept small (but should be at least 2). This is the best choice in terms of both performance and complexity. Many good features about PA codes as proposed in [1] [2] can be conveniently adopted for GPA codes, like the algebraic interleaving, which will make it flexible for GPA codes to change rate and length adaptively. The regular structure of GPA codes makes it appealing for hardware implementation particularly in adaptive rate coding. GPA codes can conveniently adopt to the rate change by reducing the rate of the SPC code and/or increasing the number of parallel branches.

As indicated by the research on irregular LDPC codes and repeat accumulate codes, irregularity seems to be the key for a further improvement in performance. Irregularity offers unequal error protection where highly protected bits tend to be decoded first and then help with the less protected bits. With irregular GPA codes, input bits will not uniformly participate in every SPC branch. Rather, the number of SPC branches (checks) each bit is involved in, will follow a carefully-designed profile. It is interesting to point out that irregular GPA codes thereby become irregular repeat accumulate codes [12].

References

[1] 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. Intl. Symp. Inform. Theory*, pp. 122-122, Washington D.C, June, 2001

[2] J. Li, K. R. Narayanan, and C. N. Georghiades, "Product accumulate codes: a class of capacity-approaching, linear-complexity codes," submitted to *IEEE Tran. Info Theory*

[3] H. Jin, and R. J. McEliece, "RA codes achieve AWGN channel capacity", *Proc. 13th Intl. Symp. Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*

[4] D. Divsalar, "A simple tight bound on error probability of block codes with application to turbo codes," *TMO Progress Report*, 42-139, Nov. 1999

[5] T. 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

[6] 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

[7] T. Richardson, and R. Urbanke, "An introduction to the analysis of iterative coding systems," `http://lthcwww.epfl.ch/ publications.html`

[8] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, No. 2, Mar., 1999

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

[10] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, No. 3, May 1998

[11] R. G. Gallager, *Low Density Parity Check Codes*, MIT Press, 1963

[12] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," *Proc. 2nd Intl. Symp. on Turbo Codes and Related Topics*, France, July, 2000

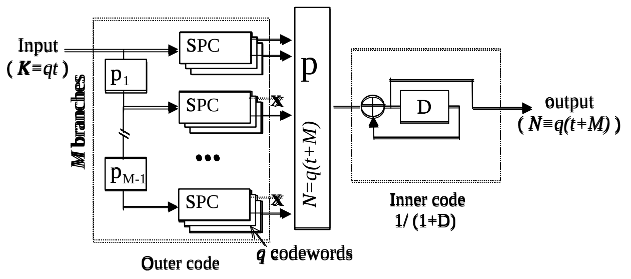

Fig. 1. System Model of GPA Codes

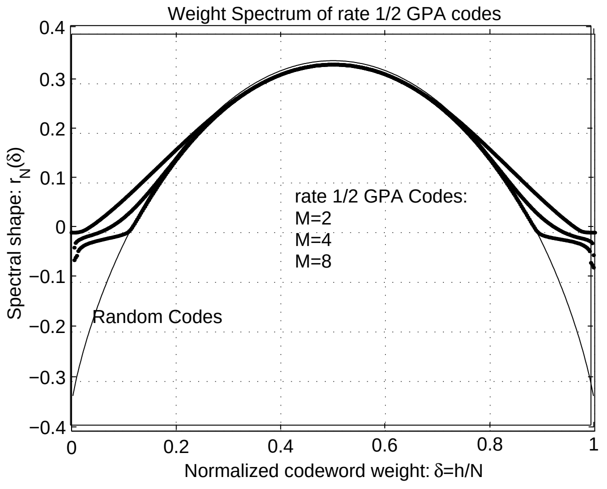

Fig. 2. Spectral Shape of GPA Codes

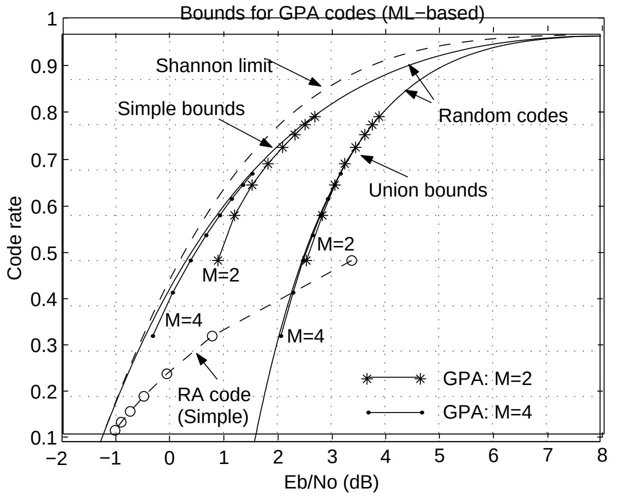

Fig. 3. The simple bound and the union bound

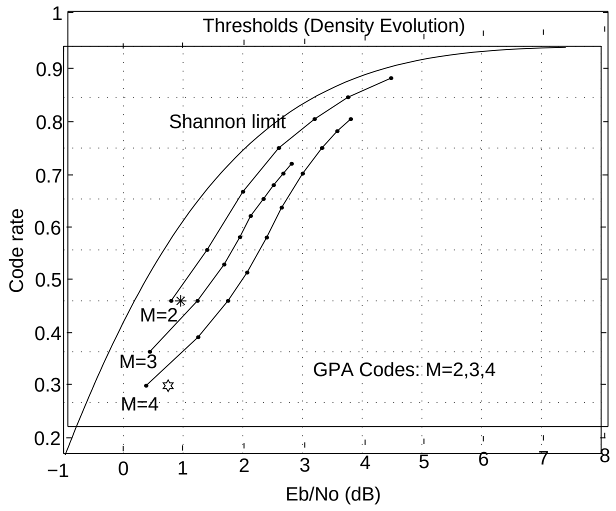

Fig. 4. Thresholds for GPA Codes

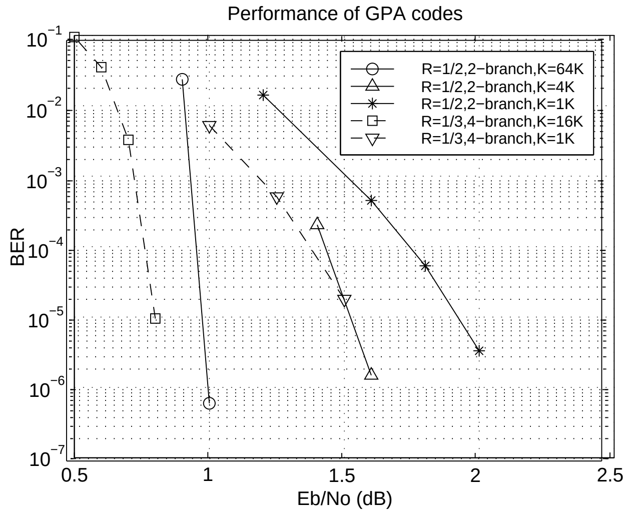

Fig. 5. Performance of GPA Codes

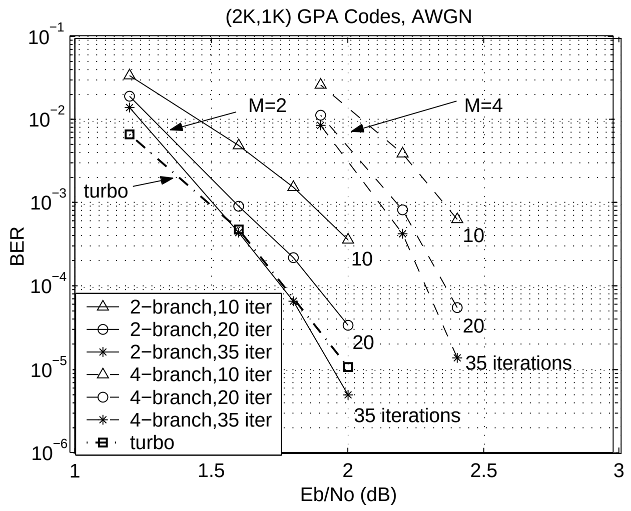

Fig. 6. Comparison of 2-Branch and 4-Branch GPA Codes