Irregular Repeat–Accumulate Codes [1]

Hui Jin, Aamod Khandekar, and Robert McEliece

Department of Electrical Engineering, California Institute of Technology
Pasadena, CA 91125 USA
E-mail: {hui, aamod, rjm}@systems.caltech.edu

Abstract*: In this paper we will introduce an ensemble of codes called* irregular repeat-accumulate *(IRA) codes. IRA codes are a generalization of the repeat-accumluate codes introduced in [1], and as such have a natural linear-time encoding algorithm. We shall prove that on the binary erasure channel, IRA codes can be decoded reliably in linear time, using iterative sum-product decoding, at rates arbitrarily close to channel capacity. A similar result appears to be true on the AWGN channel, although we have no proof of this. We illustrate our results with numerical and experimental examples.*

Keywords: repeat-accumulate codes, turbo-codes, low-density parity-check codes, iterative decoding.

1. INTRODUCTION

With the hindsight provided by the past seven years of research in turbo-codes and low-density parity-check codes, one is tempted to propose the following problem as the final problem for channel coding researchers: *For a given channel, find an ensemble of codes with (1) a linear-time encoding algorithm, and (2) which can be decoded reliably in linear time at rates arbitrarily close to channel capacity.* For turbo-codes, both parallel and serial, (1) holds, but according to the recent work by Divsalar, Dolinar, and Pollara [7], on the AWGN channel there appears to be a gap, albeit usually not a large one, between channel capacity and the iterative decoding thresholds for any turbo ensemble. For LDPC codes, the natural encoding algorithm is quadratic in the block length, and from the work of Richardson and Urbanke [2] we know that for regular LDPC codes, on the binary symmetric and AWGN channels there is a gap between capacity and the iterative decoding thresholds. On the positive side, however, Luby, Shokrollahi et at. [3], [4], [8], have established the remarkable fact that on the binary erasure channel *irregular* LDPC codes satisfy (2). Recent work by Richardson, Shokrollahi and Urbanke [5] shows that on the AWGN channel, irregular LDPC codes are markedly better than regular ones, but whether or not they can reach capacity is not yet known. In summary, as yet there is no known noisy channel for which the final problem has been solved, although researchers are very close on the AWGN channel and extremely close on the binary erasure channel.

In this paper, we will introduce a promising class of codes called *irregular repeat-accumulate* codes, which generalizes the repeat-accumulate codes of [1]. After defining the codes in Section 2, and observing that they have a simple linear-time encoding algorithm, in Section 3, using the powerful Richarson-Urbanke method [2], we will prove rigorously that IRA codes solve the final problem for the binary erasure channel. In Section 4, we will discuss, less rigorously, the performance of IRA codes on the AWGN channel, and show that their performance is remarkably good.

2. DEFINTION OF IRA CODES

Figure 1 shows a Tanner graph of an IRA code with parameters $(f_1, \ldots, f_J; a)$, where $f_i \geq 0$, $\sum_i f_i = 1$ and a is a positive integer. The Tanner graph is a bipartite graph with two kinds of nodes: variable nodes (open circles) and check nodes (filled circles). There are k variable nodes on the left, called information nodes; there are $r = (k \sum_i i f_i)/a$ check nodes; and there are r variable nodes on the right, called parity nodes. Each information node is connected to a number of check nodes: the fraction of information nodes connected to exactly i check nodes is f_i. Each check node is conected to exactly a information nodes. These connections can made in many ways, as indicated in Figure 1 by the "arbitrary permutation" of the ra edges joining information nodes and check nodes. The check nodes are connected to the parity nodes in the simple zigzag pattern shown in the figure.

If the "arbitrary permutation" in Figure 1 is fixed, the Tanner graph represents a binary linear code with k information bits $(u_1, \ldots, u_k)$ and r parity bits $(x_1, \ldots, x_r)$, as follows. Each of the information bits is associated with one of the information nodes; and each of the parity bits is associated with one of the

[1]This paper is to be presented at the Second International Conference on Turbo Codes, Brest, France, September 2000. This research was supported by NSF grant no. CCR-9804793, and grants from Sony, Qualcomm, and Caltech's Lee Center for Advanced Networking.

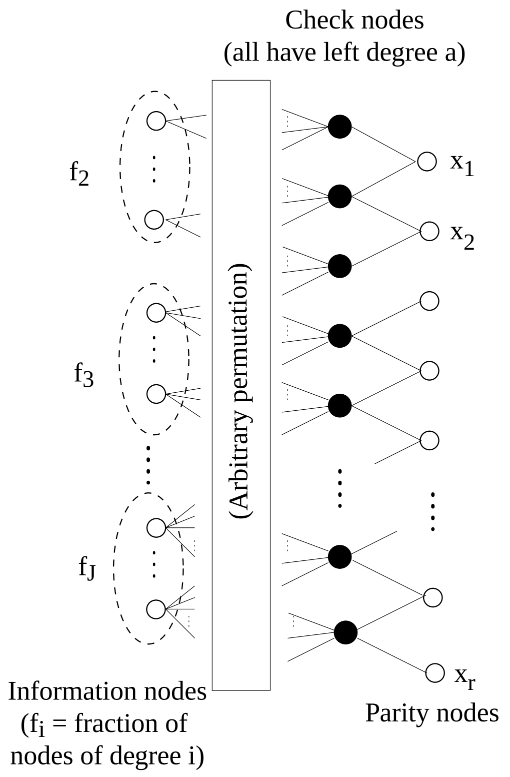

Figure 1: Tanner graph for IRA code with parameters $(f_1, \ldots, f_J; a)$.

parity nodes. The value of a parity bit is determined uniquely by the condition that the mod-2 sum of the values of the variable nodes connected to each of the check nodes is zero. To see this, let us conventionally set $x_0 = 0$. Then if the values of the bits on the ra edges coming out of the permutation box are $(v_1, \ldots, v_{ra})$, we have the recursive formula

$$x_j = x_{j-1} + \sum_{i=1}^{a} v_{(j-1)a+i}, \qquad (1)$$

for $j = 1, 2, \ldots, r$. This is in effect the encoding algorithm, and so if a is fixed and $n \to \infty$, the encoding complexity is $O(n)$.

There are two versions of the IRA code in Figure 1: the *nonsystematic* and the *systematic* verisons. The nonsystematic version is an (r, k) code, in which the codeword corresponding to the information bits $(u_1, \ldots, u_k)$ is $(x_1, \ldots, x_r)$. The systematic version is a $(k + r, k)$ code, in which the codeword is

$$(u_1, \ldots, u_k; x_1, \ldots, x_r).$$

The rate of the *nonsystematic* code is easily seen to be

$$R_{\text{nsys}} = \frac{a}{\sum_i i f_i}, \qquad (2)$$

whereas for the systematic code the rate is

$$R_{\text{sys}} = \frac{a}{a + \sum_i i f_i} \qquad (3)$$

For example, the original RA codes are nonsystematic IRA codes with $a = 1$ and exactly one f_i equal to 1, say $f_q = 1$, and the rest zero, in which case (2) simplifies to $R = 1/q$. (However, in this paper we will be concerned almost exclusively with systematic IRA codes.)

In an iterative sum-product message-passing decoding algorithm, all messages are assumed to be log-likelihood ratios, i.e., of the form $m = \log(p(0)/p(1))$. The outgoing message from a variable node u to a check node v represents information about u, and a message from a check node u to a variable node v represents information about u. Intially, messages are sent from variable nodes which represent transmitted symbols.

The outgoing message from a node u to a node v depends on the incoming messages from all neighbors w of u except v. If u is a variable message node, this outgoing message is

$$m(u \to v) = \sum_{w \neq v} m(w \to u) + m_0(u), \qquad (4)$$

where $m_0(u)$ is the log-likelihood message associated with u. (If u is not a codeword node, this term is absent.) If u is a check node the corresponding formula is [10]

$$\tanh \frac{m(u \to v)}{2} = \prod_{w \neq v} \tanh \frac{m(w \to u)}{2}. \qquad (5)$$

3. IRA CODES ON THE BINARY ERASURE CHANNEL

The sum-product algorithm defined in equations (4) and (5) simplifies considerably on the binary erasure channel (BEC). The BEC is a binary input channel with three output symbols, a 0, a 1 and "erasure." The input symbol is received as an erasure with probability p and is received correctly with probability $1 - p$. It is important to note that no errors are ever made on this channel.

It is not difficult to see that the messages defined in (4) and (5) can assume only three values on the BEC, viz. $+\infty$, $-\infty$ or 0, corresponding to a variable value 0, 1, or "unknown." No errors can occur during the running of the algorithm; if a message is $\pm\infty$, the corresponding variable is guaranteed to be 0 or 1, respectively. The operations at the nodes in the graph given by eqns (4) and (5) can be stated much more simply and intutively in this case. At a variable node, the outgoing message is equal to any non-erasure incoming message, or an erasure if all incoming messages are erasures. At a check node, the outgoing message is an erasure if any incoming message is an erasure, and otherwise is the binary sum of all incoming messages.

3.1. Notation

In this section and the next, it will be convenient to use a slightly different representation for an IRA code than the one used in Section 2. Firstly, we will begin with the assumption that the degrees of both the information nodes and the check nodes are non-constant, though we will soon restrict attention to the "right-regular" case, in which the check nodes have constant degree.

Secondly, let λ_i be the fraction of *edges* between the information and the check nodes that are adjacent to an information node of degree i, and let ρ_i be the fraction of such edges that are adjacent to a check node of degree $i+2$ (i.e. one which is adjacent to i information nodes). We will use these edge fractions λ_i and ρ_i to represent the IRA code rather than the corresponding node fractions. We define $\lambda(x) = \sum_i \lambda_i x^{i-1}$ and $\rho(x) = \sum_i \rho_i x^{i-1}$ to be the generating functions of these sequences. The pair (λ, ρ) is called a *degree distribution.* It is quite easy to convert between the two representations. We demonstrate the conversion with the information node degrees. Let the f_i's be as defined in Section 2 and let $L(x) = \sum_i f_i x^i$. Then we have

$$\begin{aligned} f_i &= \frac{\lambda_i/i}{\sum_j \lambda_j/j}, && (6) \\ L(x) &= \int_0^x \lambda(t)dt \Big/ \int_0^1 \lambda(t)dt. && (7) \end{aligned}$$

The rate of the systematic IRA code (we shall be dealing only with these) given by this degree distribution is given by

$$\text{Rate} = \left(1 + \frac{\sum_j \rho_j/j}{\sum_j \lambda_j/j}\right)^{-1} \qquad (8)$$

(This is an easy exercise. For a proof, see [8].)

3.2. Fixed point analysis of iterative decoding

In [2], it was shown that if for a code ensemble, the probability of the *depth-l neighborhood* of an edge (in the Tanner graph) being cycle-free goes to 1 as the length of the code goes to infinity (we will call this condition the *cycle-free condition*), then *density evolution* gives an accurate estimate of the bit error rate after l iterations, again as the length of the codes goes to infinity. In density evolution, we evolve the probability density of the messages being passed according to the operations being performed on them, assuming that all incoming messages are independent (which is true if the depth-l neighbourhood is tree-like). The cycle-free condition does indeed hold for IRA codes. The proof of this fact is almost exactly the same as in the irregular LDPC codes case, which was done in [2].

Now, in the case of the erasure channel, we have seen that the messages are only of three types, so in effect we have a discrete density function, and the probability of error is merely the probability of erasure. With this in mind, we will now study the evolution of the erasure probability, and derive conditions which guarantee that it goes to zero as the number of iterations goes to infinity. Under these conditions iterative decoding will be successful in the sense of [2], i.e., it will achieve arbitrarily small BERs, given enough iterations and long enough codes.

Let p be the channel probability of erasure. We will iterate the probability of erasure along the edges of the graph during the course of the algorithm. Let x_0 be the probability of erasure on an edge from an information node to a check node, x_1 the probability of erasure on an edge from a check node to a parity node, x_2 the probability of erasure on an edge from a parity node to a check node, and x_3 the probability of erasure on an edge from a check node to an information node. The initial probability of erasure on the message bits is p.

We now assume that we are at a fixed point of the decoding algorithm and solve for x_0. We get the following equations:

$$\begin{aligned} x_1 &= 1 - (1-x_2)R(1-x_0), && (9) \\ x_2 &= px_1, && (10) \\ x_3 &= 1 - (1-x_2)^2\rho(1-x_0), && (11) \\ x_0 &= p\lambda(x_3). && (12) \end{aligned}$$

where $R(x)$ is the polynomial in which the coefficient of x^i denotes the fraction of check nodes of degree i. $R(x)$ is given by (cf. eq. (7))

$$R(x) = \frac{\int_0^x \rho(t)dt}{\int_0^1 \rho(t)dt} \qquad (13)$$

We eliminate x_1 from the first two of these equations to get x_2 in terms of x_0 and then keep substituting forwards to get an equation purely in x_0, henceforth denoted by x. We thereby obtain the following equation for a fixed point of iterative decoding:

$$p\lambda\left(1 - \left[\frac{1-p}{1-pR(1-x)}\right]^2 \rho(1-x)\right) = x. \qquad (14)$$

If this equation has no solution in the interval $(0,1]$, then iterative decoding must converge to probability of erasure zero. Therefore, if we have

$$p\lambda\left(1-\left[\frac{1-p}{1-pR(1-x)}\right]^2\rho(1-x)\right)<x,\quad \forall x\neq 0. \tag{15}$$

then in the sense of [2], iterative decoding is successful.

3.3. Capacity-achieving sequences of degree distributions

We will now derive sequences of degree distributions that can be shown to achieve channel capacity. First, we restrict attention to the case $\rho(x)=x^{a-1}$ for some $a\geq 1$, since it turns out that we can achieve capacity even with this restriction. In this case, $R(x)=x^a$, and the condition for convergence to zero BER now becomes

$$p\lambda\left(1-\left[\frac{1-p}{1-p(1-x)^a}\right]^2(1-x)^{a-1}\right)<x,\quad \forall x\neq 0 \tag{16}$$

We now make the following new definitions

$$\begin{aligned} f_p(x) &\triangleq 1-\left[\frac{1-p}{1-p(1-x)^a}\right]^2(1-x)^{a-1} && (17)\\ h_p(x) &\triangleq 1-\left[\frac{1-p}{1-p(1-x)^a}\right]^2(1-x)^{a} && (18)\\ g_p(x) &\triangleq h_p^{-1}(x) && (19)\end{aligned}$$

Notice that $f_p(x)$, $h_p(x)$ and $g_p(x)$ are all monotonic functions in $[0,1]$ and attain the values 0 at 0 and 1 at 1. In addition, $h_p(x)$ can be inverted by hand (by making the substitution $(1-x)^a=y$) and it can be shown that $g_p(x)$ has a power series expansion around 0 with non-negative coefficients. Let this expansion be $g_p(x)=\sum_i g_{p,i}x^i$.

Now, the condition (16) can now be rewritten as

$$p\lambda(f_p(x))<x,\quad \forall x\neq 0 \tag{20}$$

which can be rewritten as

$$\lambda(x)<\frac{f_p^{-1}(x)}{p} \tag{21}$$

We make the following choice of $\lambda(x)$:

$$\lambda(x)=\frac{1}{p}\left(\sum_{i=1}^{N-1}g_{p,i}x^i+\epsilon x^N\right) \tag{22}$$

where $0<\epsilon<g_{p,N}$ and $\sum_{i=1}^{N-1}g_{p,i}+\epsilon=p$. Such a choice of N and ϵ exists and is unique since the $g_{p,i}$'s are non-negative and $\sum_{i=1}^{\infty}g_{p,i}=g_p(1)=1$. For this choice of $\lambda(x)$, we have

$$p\lambda(x)<g_p(x)=h_p^{-1}(x)<f_p^{-1}(x)\quad \forall x\neq 0 \tag{23}$$

where the last inequality follows because $f_p(x)<h_p(x)\ \ \forall x\neq 0$.

Thus, the condition (21) for BER going to zero is satisfied and the degree distributions we have thus defined yield codes with thresholds that are greater than or equal to p. We now wish to compute the rate of these codes in the limit as $a\to\infty$ to show that they achieve channel capacity. The rate of the code is given by eq. (8) which simplifies to $(1+(a\sum_i\lambda_i/i)^{-1})^{-1}$ in the right-regular case. Now,

$$\lim_{a\to\infty}a\sum_i\frac{\lambda_i}{i}=\lim_{a\to\infty}a\left(\sum_{i=1}^{N-1}\frac{g_{p,i}}{i}+\frac{\epsilon}{N}\right) \tag{24}$$

We also have

$$\lim_{a\to\infty}a\sum_{i=N}^{\infty}\frac{g_{p,i}}{i}\leq\lim_{a\to\infty}\frac{a}{N}\sum_{i=N}^{\infty}g_{p,i}\leq\lim_{a\to\infty}\frac{a}{N}=0 \tag{25}$$

where the last equality is a property of the function $g_p(x)$ and is also proved by manual inversion of $h_p(x)$. We therefore have

$$\begin{aligned}\lim_{a\to\infty}a\sum_i\frac{\lambda_i}{i} &= \lim_{a\to\infty}a\sum_{i=1}^{\infty}\frac{g_{p,i}}{i}\\ &= \lim_{a\to\infty}a\int_0^1 g_p(x)dx\\ &= a\left(1-\int_0^1 h_p(x)dx\right)\\ &= a\int_0^1\left(\frac{1-p}{1-px^a}\right)^2x^a dx.\end{aligned}$$

The integrand on the right can be expanded in a power series with non-negative coefficients, with the first non-zero coefficient being that of x^a. Keeping in mind that we are integrating this power series, it is easy to see that

$$\begin{aligned}&\frac{a}{a+1}\int_0^1\left(\frac{1-p}{1-px^a}\right)^2x^{a-1}dx\\ <\ & 1-\int_0^1 h_p(x)dx \qquad (26)\\ <\ & \int_0^1\left(\frac{1-p}{1-px^a}\right)^2x^{a-1}dx.\end{aligned}$$

Both bounds in the above equation can be computed easily and both tend to $(1-p)/p$ in the limit of large a. Plugging this result into the formula for the rate, we finally get that the rate tends to $1-p$ in the limit of large a, which is indeed the capacity of the BEC.

Thus the sequence of degree distributions given in eq. (22) does indeed achieve channel capacity.

3.4. Some numerical results

We have seen that the condition for BER going to zero at a channel erasure probability of p is $p\lambda(x) < f_p^{-1}(x)\ \forall x \neq 0$. We later enforced a stronger condition, namely $p\lambda(x) < h_p^{-1}(x) = g_p(x)\ \forall x \neq 0$ and derived capacity-chieving degree sequences satisfying this condition. The reason we needed to enforce the stronger condition was that $h_p^{-1}(x) = g_p(x)$ has non-negative power-series coefficients, while the same cannot be said for $f_p^{-1}(x)$. However, from (26) we see that enforcing this stronger condition costs us a factor of $1-a/(a+1) = 1/(a+1)$ in the rate which is very large for values of a that are of interest, and therefore the resulting codes are not very good.

If, however, $f_p^{-1}(x)$ were to have non-negative power series coefficients, then we could use it to define a degree distribution and we would no longer lose this factor of $1/(a+1)$. We have found through direct numerical computation in all cases that we tried, that enough terms in the beginning of this power series are non-negative to enable us to define $\lambda(x)$ by an equation analogous to eq. (22), replacing $g_p(x)$ by $f_p^{-1}(x)$. Of course, the resulting code is not theoretically guaranteed to have a threshold $\geq p$, but numerical computation shows that the threshold is either equal to or very marginally less than p.

This design turns out to yield very powerful codes, in particular codes whose performance is in every way comparable to the irregular LDPC codes listed in [8] as far as decoding performance is concerned. The performance of some of these distributions is listed in Table 1. The threshold values p are the same as those in [8] for corresponding values of a (IRA codes with right degree $a+2$ should be compared to irregular LDPC codes with right degree a, so that the decoding complexity is about the same), so as to make comparison easy. The codes listed in [8] were shown to have certain optimality properties with respect to the tradeoff between $1-\delta/(1-R)$ (distance from capacity) and a (decoding complexity), so it is very heartening to note that the codes we have designed are comparable to these.

We end this section with a brief discussion of the case $a = 1$. In this case, it turns out that $f_p^{-1}(x)$ does indeed have non-negative power-series coefficients. The resulting degree sequences yield codes that are better than conventional RA codes at small rates. An entirely similar exercise can be carried out for the case of non-systematic RA codes with $a = 1$ and the codes resulting in this case are significantly better than conventional RA codes for most rates. However, non-systematic RA codes turn out to be useless for higher values of a, as can be seen by manually following the decoding algorithm for one iteration, which shows that decoding does not proceed at all. For this reason all the preceding analysis was performed for systematic RA codes.

Table 1: Performance of some codes designed using the procedure described in Section 3.4. at rates close to 2/3 and 1/2. δ is the code threshold (maximum allowable value of p), N the number of terms in $\lambda(x)$, and R the rate of the code.

a	δ	N	$1-R$	$\delta/(1-R)$
4	0.20000	1	0.333333	0.6000
5	0.23611	3	0.317101	0.7448
6	0.28994	6	0.329412	0.8802
7	0.31551	11	0.336876	0.9366
8	0.32024	16	0.333850	0.9592
9	0.32558	26	0.334074	0.9744
4	0.48090	13	0.502141	0.9577
5	0.49287	28	0.502225	0.9814

4. IRA CODES ON THE AWGN CHANNEL

In this section, we will consider the behavior of IRA codes on the AWGN channel. Here there are only two possible inputs, 0 and 1, but the output alphabet is the set of real numbers: if the x is the input, then the output is $y = (-1)^x + z$, where z is a mean zero, variance σ^2 Gaussian random variable. For a given noise variance σ^2, our objective will be to find a left degree sequence $\lambda(x)$ such that the ensemble message error probability approaches zero, while the rate is as large as possible. Unlike the BEC, where we deal only with probabilities, in the case of the AWGN we must deal with probability densities. This complicates the analysis, and forces us to resort to approximate design methods.

4.1. Gaussian Approximation

Wiberg [9] has shown that the messages passed in iterative decoding on the AWGN channel can be well approximated by Gaussian random variables, provided the messages are in log-likelihood ratio form. In [6], this approximation was used to design good LDPC codes for the AWGN channel.

In this subsection, we use this Gaussian approximation to design good IRA codes for the AWGN channel. Specifically, we approximate the messages from check nodes to variable nodes (both information and parity) as Gaussian at every iteration. For a variable node, if all the incoming messages are Gaussian, then all the outgoing messages are also Gaussian because of (4). A Gaussian distribution $f(x)$ is called *consistent* [5] if $f(x) = f(-x)e^x$ for $\forall x \leq 0$. The consistency condition implies that the mean and variance satisfy $\sigma^2 = 2\mu$. For the sum-product algorithm, it has been shown [2] that consistency is preserved at message updates of both the variable and check nodes. Thus if we assume Gaussian messages, and require consistency, we only need to keep track of the means. To this end, we define a *consistent Gaussian density* with mean μ to be

$$G_\mu(z) = \frac{1}{\sqrt{4\pi\mu}} e^{-(z-\mu)^2/4\mu}. \qquad (27)$$

The expected value of $\tanh \frac{z}{2}$ for a consistent Gaussian distributed random variable z with mean μ is then

$$E[\tanh \frac{z}{2}] = \int_{-\infty}^{+\infty} G_\mu(z) \tanh \frac{z}{2} dz \overset{\triangle}{=} \phi(\mu). \qquad (28)$$

It is easy to see that $\phi(u)$ is a monotonic increasing function of u; we denote its inverse function by $\phi^{(-1)}(y)$. Let $\mu_L^{(l)}$ and $\mu_R^{(l)}$ be the means of the message from check nodes to variable nodes on the left (i.e., information nodes) and on the right (i.e., parity nodes) at the lth iteration. We want to obtain expressions for $\mu_L^{(l+1)}$ and $\mu_R^{(l+1)}$ in terms of $\mu_L^{(l)}$ and $\mu_R^{(l)}$. A message from a degree-i information node to a check node at the lth iteration, is Gaussian with mean $(i-1)\mu_L^{(l)} + \mu_o$, where μ_o is the mean of message m_o in (4). Hence if v_L denotes the message on a randomly selected edge from an information node to a check node, the density of v_L is

$$\sum_{i=1}^{J} \lambda_i G_{(i-1)\mu_L^{(l)}+\mu_o}(z). \qquad (29)$$

From (29) and (28) we obtain:

$$E[\tanh \frac{v_L}{2}] = \sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L^{(l)} + \mu_o). \qquad (30)$$

Similarly, if v_R denotes the message on a randomly selected edge from a parity node to a check node,

$$E[\tanh \frac{v_R}{2}] = \phi(\mu_R^{(l)} + \mu_o). \qquad (31)$$

Because of (5) we have

$$E[\tanh \frac{m(u \to v)}{2}] = \prod_{w \neq v} E[\tanh \frac{m(w \to u)}{2}]. \qquad (32)$$

Denote a message from a check node to an information node, resp. parity node, by u_L, resp, u_R. Replacing $E[\tanh \frac{m(w\to u)}{2}]$ with the right side of (30) or (31) depending upon whether the message comes from the left or right, (32) implies:

$$\begin{aligned} E[\tanh \frac{u_L}{2}] &= E[\tanh \frac{v_L}{2}]^{a-1} E[\tanh \frac{v_R}{2}]^2 \\ &= (\sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L^{(l)} + \mu_o))^{a-1} (\phi(\mu_R^{(l)} + \mu_o))^2, \end{aligned}$$

$$\begin{aligned} E[\tanh \frac{u_R}{2}] &= E[\tanh \frac{v_L}{2}]^{a} E[\tanh \frac{v_R}{2}] \\ &= (\sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L^{(l)} + \mu_o))^{a} \phi(\mu_R^{(l)} + \mu_o). \end{aligned}$$

Using the definition of $\phi(\mu)$ in (28), we thus have the following recursion for $\mu_L^{(l)}$ and $\mu_R^{(l)}$:

$$\begin{aligned} \phi(\mu_L^{(l+1)}) &= (\sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L^{(l)} + \mu_o))^{a-1} \times \\ & \quad (\phi(\mu_R^{(l)} + \mu_o))^2, \qquad (33) \\ \phi(\mu_R^{(l+1)}) &= (\sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L^{(l)} + \mu_o))^{a} \times \\ & \quad \phi(\mu_R^{(l)} + \mu_o). \qquad (34) \end{aligned}$$

In order to have arbitrary small bit error probability, the means $\mu_L^{(l)}$ and $\mu_R^{(l)}$ should approach infinity as l approaches infinity. In the next subsection, we derive a sufficient condition for this.

4.2. Fixed point analysis

We now assume that iterative dedoding has reached a fixed point of (33) and (34), i.e., $\mu_L^{(l+1)} = \mu_L^{(l)} = \mu_L$ and $\mu_R^{(l+1)} = \mu_R^{(l)} = \mu_R$. Denote $\sum_{i=1}^{J} \lambda_i \phi((i-1)\mu_L + \mu_o)$ by x. From (30) we can see that $0 < x < 1$ and $x \to 1$ if and only if $\mu_L \to \infty$. From (34) it's easy to show that μ_R is a function of x, denoted by f, i.e., $\mu_R = f(x)$. Then, dividing (33) by the square of (34) gives us:

$$\phi(\mu_L) = \phi^2(\mu_R)/x^{a+1} = \phi^2(f(x))/x^{a+1}. \qquad (35)$$

Now replacing μ_L with $\phi^{(-1)}(\phi^2(f(x))/x^{a+1})$ into the definition of x, we obtain the following equation for the fixed point x:

$$x = \sum_{i=1}^{J} \lambda_i \phi(\mu_o + (i-1)\phi^{(-1)}(\frac{\phi^2(f(x))}{x^{a+1}})). \qquad (36)$$

If this equation doesn't have a solution in the interval $[0,1]$, then the decoding bit error probability converges to zero. Therefore, if we have

$$F(x) \overset{\triangle}{=} \sum_{i=1}^{J} \lambda_i \phi(\mu_o + (i-1)\phi^{(-1)}(\frac{\phi^2(f(x))}{x^{a+1}})) > x, \qquad (37)$$

for any $x \in [x_0, 1)$, where x_0 is the value of x at the first iteration, then (the Gaussian approximation to) iterative decoding is successful.

Since the rate of the code is given by (cf. (8)):

$$\frac{\sum_i \lambda_i/i}{1/a + \sum_i \lambda_i/i}, \qquad (38)$$

to maximize the rate, we should maximize $\sum_i \lambda_i/i$. Thus, under the Gaussian approximation, the problem of finding a good degree sequence for IRA codes is converted to the following linear programming problem:

Linear Programming Problem. *Maximize*

$$\sum_{i=1}^{J} \lambda_i/i, \tag{39}$$

under the condition

$$F(x) > x, \quad \forall x \in [x_0, 1]. \tag{40}$$

We have designed some degree sequences for IRA codes using this linear programming methodology. The results are presented in Tables 2 (code rate $\approx 1/3$) and 3 (code rate $\approx 1/2$). After using the heuristic Gaussian approximation method to design the degree sequences, we used exact density evolution to determine the actual noise threshold. (In every case, the true iterative decoding threshold was better than the one predicted by the Gaussian approximation.)

a	2	3	4
λ_2	0.139025	0.078194	0.054485
λ_3	0.222155	0.128085	0.104315
λ_5		0.160813	
λ_6	0.638820	0.036178	0.126755
λ_{10}			0.229816
λ_{11}			0.016484
λ_{12}		0.108828	
λ_{13}		0.487902	
λ_{14}			
λ_{16}			
λ_{27}			0.450302
λ_{28}			0.017842
rate	0.333364	0.333223	0.333218
σ_{GA}	1.1840	1.2415	1.2615
σ^*	1.1981	1.2607	1.2780
$(\frac{E_b}{N_0})^*(dB)$	0.190	-0.250	-0.371
S.L. (dB)	-0.4953	-0.4958	-0.4958

Table 2: Good degree sequences yielding codes of rate approximately 1/3 for the AWGN channel and with $a = 2, 3, 4$. For each sequence the Gaussian approximation noise threshold, the actual sum-product decoding threshold, and the corresponding $(\frac{E_b}{N_0})^*$ in dB are given. Also listed is the Shannon limit (S.L.)

For example, consider the "$a = 3$" column in Table 2. We adjust Gaussian approximation noise threshold σ_{GA} to be 1.2415 to have the returned optimal sequence having rate 0.333223. Then applying the exact density evolution program on this code, we obtain the actual sum-product decoding threshold $\sigma^* = 1.2607$, which corresponds to $E_b/N_0 = -0.250$ dB. This should be compared to the Shannon limit for the ensemble of all linear codes of the same rate, which is -0.4958 dB. As we increase the parameter a, the ensemble improves. For $a = 4$, the best code we have found has iterative decoding threshold $E_b/N_0 = -0.371$ dB, which is only 0.12 dB above the Shannon limit.

The above analysis is for *bit* error probability. In order to have zero *word* error probability, it is necessary to have $\lambda_2 = 0$. (This can be proved by the following argument: if $\lambda_2 > 0$, then in the ensemble, as $n \to \infty$, the average number of weight 2 codewords is bounded away from zero. Hence even a maximum-likelihood decoder would have non-zero decoding error probability.) In Table 3, we compare the noise thresholds of codes with and without $\lambda_2 = 0$.

a	8	8
λ_2		0.0577128
λ_3	0.252744	0.117057
λ_7		0.2189922
λ_8		0.0333844
λ_{11}	0.081476	
λ_{12}	0.327162	
λ_{18}		0.2147221
λ_{20}		0.0752259
λ_{46}	0.184589	
λ_{48}	0.154029	
λ_{55}		0.0808676
λ_{58}		0.202038
rate	0.50227	0.497946
σ^*	0.9589	0.972
$(\frac{E_b}{N_0})^*(dB)$	0.344	0.266
Shannon limit	0.197	0.178

Table 3: Two degree sequences yielding codes of rate $\approx 1/2$ with $a = 8$. For each sequence, the actual sum-product decoding threshold, and the corresponding $(\frac{E_b}{N_0})^*$ in dB are given. Also listed is the Shannon limit.

We chose rate one-half because we wanted to compare our results with the best irregular LDPC codes obtained in [5]. Our best IRA code has threshold 0.266 dB, while the best rate one-half irregular LDPC code found in [5] has threshold 0.25 dB. These two codes have roughly the same decoding complexity, but unlike LDPC codes, IRA codes have a simple linear encoding algorithm.

4.3. Simulation Results

We simulated the rate one-half code with $\lambda_2 = 0$ in Table 3. Figure 2 shows the performance of that particular code, with information block lengths 10^3, 10^4, and 10^5. For comparison, we also show the performance of the best known rate 1/2 turbo code for the same block length.

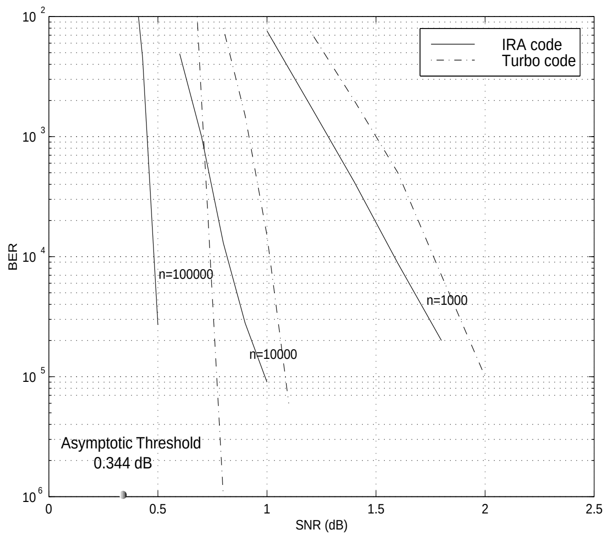

Figure 2: Comparison between turbo codes (dashed curves) and IRA codes (solid curves) of lengths $n = 10^3, 10^4, 10^5$. All codes are of rate one-half.

5. CONCLUSIONS

We have introduced a class of codes, the IRA codes, that combines many of the favorable attributes of turbo codes and LDPC codes. Like turbo codes (and unlike LDPC codes), they can be encoded in linear time. Like LDPC codes (and unlike turbo codes), they are amenable to an exact Richardson-Urbanke style analysis. In simulated performance they appear to be slightly superior to turbo codes of comparable complexity, and just as good as the best known irregular LDPC codes. In our opinion, the important open problem is to prove (or disprove) that IRA codes can be decoded reliably in linear time at rates arbitrarily close to channel capacity. We know this to be true for the binary erasure channel, but for no other channel model. If this should turn out ot be true, we would argue that IRA codes definitively solve the problem posed implicitly by Shannon in 1948. If it is not true, then researchers should search for an even better class of code ensembles.

REFERENCES

[1] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," pp. 201-210 in *Proc. 36th Allerton Conf. on Communication, Control, and Computing.* (Allerton, Illinois, Sept. 1998).

[2] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message passing decoding," submitted to *IEEE Trans. Inform. Theory.*

[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," Proc. 29th ACM Symp. on the Theory of Computing (1997), pp. 150-159.

[4] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low-density codes and improved designs using irregular graphs," Proc. 30th ACM Symp. on the Theory of Computing (1998), pp. 249-258.

[5] T. J. Richardson, A. Shokrollahi,, and R. Urbanke, "Design of provably good low-density parity-check codes," submitted to *IEEE Trans. Inform. Theory.*

[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," submitted to *IEEE Trans. Inform. Theory.*

[7] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative turbo decoder analysis based on Gaussian density evolution," submitted to *IEEE J. Selected Areas in Comm.*

[8] M. A. Shokrollahi, "New sequences of linear time erasure codes approaching channel capacity," Proc. 1999 ISITA (Honolulu, Hawaii, November 1999) pp. 65–76.

[9] N. Wiberg, "Codes and decoding on general graphs," dissertation no. 440, Linköping Studies in Science and Technology, Linköping, Sweden, 1996.

[10] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-42, no. 2 (March 1996). pp. 429–445.