View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Open Archive Toulouse Archive Ouverte

Low-Rate Coding Using Incremental Redundancy for GLDPC Codes

Mathieu Cunche*, Valentin Savin†, Vincent Roca*, Ghassan Kraidy†, Alexandre Soro‡, Jérôme Lacan‡
*INRIA Rhône-Alpes Grenoble France, †CEA-LETI MINATEC Grenoble France, ‡ISAE, Univ. of Toulouse, Toulouse France

***Abstract*—In this paper we propose a low-rate coding method, suited for application-layer forward error correction. Depending on channel conditions, the coding scheme we propose can switch from a fixed-rate LDPC code to various low-rate GLDPC codes. The source symbols are first encoded by using a staircase or triangular LDPC code. If additional symbols are needed, the encoder is then switched to the GLDPC mode and extra-repair symbols are produced, on demand. In order to ensure small overheads, we consider irregular distributions of extra-repair symbols optimized by density evolution techniques. We also show that increasing the number of extra-repair symbols improves the successful decoding probability, which becomes very close to 1 for sufficiently many extra-repair symbols.**

I. Introduction

Forward Error Correction (FEC) codes are a key building block for many content distribution applications. FEC codes can operate at the physical layer (e.g. the LDPC codes of DVB-S2), just below IP (e.g. the MPE-FEC scheme of DVB-H), or within the transport or the application layers (e.g. within the IP-Datacasting service of DVB-H). In this paper we will focus on application-layer FEC codes, referred as AL-FEC. AL-FEC codes are complementary and not opposed to physical layer codes. There are two major differences between these two classes. A first difference is the block of data upon which FEC encoding is performed. With physical layer (respectively sub-IP) FEC codes, FEC encoding is performed on a sequence of bits (respectively IP datagrams). With AL-FEC codes, encoding is performed over the whole object (ideally by using a single block with the so-called "large block codes"). A second difference is the type of errors these codes will correct, or equivalently the channel type. With physical layer FEC codes, we are dealing with a channel that can alter the received content. With sub-IP and AL-FEC codes, we are dealing with erasure channels, i.e. a channel in which each data unit is either transmitted without error or entirely erased. Indeed, the potential physical layer CRC, or physical layer FEC codes, or even transport level UDP checksums, lead a receiver to discard erroneous data units.

For small to medium codeword lengths, *Maximum-Distance Separable* (MDS) codes are known to achieve the channel capacity. However, for large block lengths, their decoding becomes intractable and thus, iteratively decoded graph-based codes constitute the main alternative. Low-density parity-check (LDPC) codes [1][2] with iterative decoding [3] proved to perform very close to the channel capacity with reasonable decoding complexity [4][5]. LDPC codes were generalized by Tanner [6] by introducing the sparse graph representation and replacing the Single Parity Check (SPC) constraint nodes with error correcting block codes. Nowadays, these codes are known as GLDPC codes and were recently investigated for different channels in [7][8][9][10][11]. Moreover, "rateless" codes that are capable of generating an infinite sequence of repair symbols were proposed in [12][13]. Patent-free LDPC-staircase and LDPC-triangle codes were proposed in [14][15]. In [16] it was shown that, decoded by a hybrid iterative-Gaussian elimination algorithm, these codes tightly approach the performance of an ideal MDS code on the binary erasure channel.

In this paper, we propose a coding scheme that can produce on-demand, extra-repair symbols, either to cope with bad channel conditions, or to be used with fountain-like content distribution applications, e.g. FLUTE/ALC. The coding scheme is based on a LDPC code, which can be extended to various low-rate GLDPC codes. The extra-repair symbols are produced in an incremental way by the GLDPC codes. We show that code optimization is possible, by varying the number of extra-repair symbols from one constraint node to another (irregular distribution).

The paper is organized as follows: in Section II we describe the coding scheme; in Section III we propose a code design exploiting the LDPC-staircase codes. We analyze the asymptotic behavior of the proposed coding scheme in Section IV. Then we provide a performance evaluation of these codes in Section V. Finally we conclude.

II. Proposed Coding Scheme

Let H be a sparse binary matrix with M rows and $N = K + M$ columns. We write $H = (H_1 \mid H_2)$, where H_1 is the $M \times K$ left side of H, and H_2 is the $M \times M$ right side of H, and we further assume that H_2 is a staircase (double diagonal) or a lower-triangular matrix. The bipartite (Tanner) graph associated to H consists of M *constraint nodes* and N *symbol nodes* corresponding respectively to the rows and the columns of H. A symbol node n is connected to a constraint node m iff the corresponding element of H is equal to 1. In this case, we also say that symbol node n is participating in the constraint node m. The bitwise XOR sum of all the symbol nodes of a constraint node by definition is equal to zero. Note that symbols often represent packets (several byte long) rather than individual bits with AL-FEC codes.

We further consider a systematic Reed-Solomon (RS) or any

This work was supported by the French ANR grant No 2006 TCOM 019 (CAPRI-FEC project).

MDS code over a finite field GF(2^p). Each symbol of the finite field corresponds to a group of p bits, and the sum of two GF-symbols corresponds to the bitwise XOR. Let the parameters of the RS code be $(k+1+E, k)$. Thus, k is the number of information symbols (the code dimension), $k+1+E$ is the number of coded symbols (the code length), and the number of repair symbols that can be generated by the code is equal to $1+E$. Without losing generality, we may assume that the first repair symbol is the XOR sum of the k information symbols. Precisely, since the RS code is systematic, for any information symbols $(x_1, \ldots, x_k)$, the repair symbols are computed as:

$$(y_1, y_{1,1}, \ldots, y_{1,E}) = (x_1, \ldots, x_k)G \tag{1}$$

where G is a $k \times (1+E)$ matrix with coefficients in GF(2^p). The first repair symbol y_1 is the XOR sum of the source symbols $x_1, \ldots, x_k$ iff all the coefficients on the first column of G are equal to 1. This can be obtained by multiplying G at the left by an appropriate invertible matrix, which will change the way the encoding is done but not the MDS property of the code.

For the clarity of the presentation, let us assume for the moment that each row of the matrix H has $k+1$ coefficients equal to 1 (i.e. each constraint node is of degree $k+1$). For a given sequence of source symbols $s_1, s_2, \ldots, s_K$, we can compute a sequence of repair symbols by "walking down the stairs", as follows:

• Consider the $k+1$ symbol nodes participating in the constraint node corresponding to the first row of H. The first k symbol nodes represent information symbols. The $(k+1)^{\text{th}}$ symbol (*i.e.*, the coefficient 1 on the diagonal of H_2) is the XOR sum of the first k symbols. Then encode the sequence of the first k symbols using the systematic RS code, and let $p_1, p_{1,1}, \ldots, p_{1,E}$ be the output repair sequence. Note that p_1 is the value of the $(k+1)^{\text{th}}$ symbol (they are both equal to the XOR sum of the k information symbols). Symbols $p_{1,1}, \ldots, p_{1,E}$ are called extra-repair symbols, and they will not be used later in the encoding process.

• Consider the $k+1$ symbol nodes participating in the constraint node corresponding to the second row of H. The first k symbol nodes represent either information symbols, or the above repair symbol p_1. The $(k+1)^{\text{th}}$ symbol (*i.e.*, the coefficient 1 on the diagonal of H_2) is the XOR sum of the first k symbols. Then encode the sequence of the first k symbols using the systematic RS code, and let $p_2, p_{2,1}, \ldots, p_{2,E}$ be the output repair sequence. Note that p_2 is the value of the $(k+1)^{\text{th}}$ symbol (they are both equal to the XOR sum of the k information symbols). Symbols $p_{2,1}, \ldots, p_{2,E}$ are called extra-repair symbols, and they will not be used later in the encoding process.

• Continue the above encoding process until the last row of matrix H is reached.

We note that the sequence $s_1, s_2, \ldots, s_K, p_1, \ldots, p_M$ is simply a codeword of the LDPC code with parity check matrix H.

The advantage of our approach consists in the availability of extra-repair symbols that turn this code into a small rate code. This code is therefore suitable for all the situations that benefit from low rate (or at the extreme rateless) codes, *i.e.* the possibility of generating infinite number of repair symbols. Moreover, the extra-repair symbols can be computed "just-in-time", prior to being transmitted, rather than in advance.

It is also important to note that the assumption of constraint nodes being of constant degree $k+1$ is not necessary. Indeed, in case of constraint nodes with various degrees, we can chose k such that the maximum constraint degree is equal to $k+1$. Then, for a constraint node of degree $k'+1$, with $k' < k$, we can use zero-padding in order to provide the systematic RS-encoder with a sequence of length k.

The LDPC code with parity matrix H will be called *initial code*, and its coding rate $r = K/N$ will be called *base coding rate*. Let us assume that $e(m)$ extra RS repair symbols are generated for the constraint node of row m, with $0 \leq e(m) \leq E$ and $m = 1, \ldots, M$. Thus, the sequence:

$$\begin{aligned} s_1, s_2, \ldots, s_K, p_1, p_{1,1}, \ldots, p_{1,e(1)}, \cdots \\ \cdots, p_M, p_{M,1}, \ldots, p_{M,e(M)} \end{aligned} \tag{2}$$

is the codeword of a GLDPC code. We call this latter the *extended code*. We note that the symbol nodes corresponding to extra RS repair symbols $p_{m,i}$ are all of degree 1 (in the bipartite graph associated with the extended code).

Let f_e denote the fraction of constraint nodes with e extra RS repair symbols:

$$f_e = \frac{\text{card}\{m = 1, \ldots, M \mid e(m) = e\}}{M} \tag{3}$$

and $\bar{f}$ be the average number of extra RS repair symbols per constraint node:

$$\bar{f} = \sum_{e=0}^{E} f_e \cdot e \tag{4}$$

The coding rate $\bar{r}$ of the extended code can then be computed as:

$$\bar{r} = \frac{K}{N + M\bar{f}} = \frac{r}{1 + (1-r)\bar{f}} \tag{5}$$

For instance, if the initial code rate is $r = 1/2$ and the average number of extra RS repair symbols per constraint node is $\bar{f} = 6$, we obtain an extended code with coding rate $\bar{r} = 1/8$.

III. Code Design

This section focuses on the design of extended codes. The initial code are the LDPC-staircase codes described in [15]. Thus, $H = (H_1 \mid H_2)$, where each column of H_1 has 3 coefficients equal to 1, and H_2 is a staircase (double diagonal) matrix. We further assume that the degree of constraint nodes is constant (which is true with the coding rate considered). We consider a base coding rate $r = 1/2$. Thus the constraint nodes must be of degree 5 and then $k = 4$.

For the extended codes, we consider two kinds of extra-repair symbol distributions:

- regular (A.K.A. "skyscraper") distributions: $f_E = 1$, and $f_e = 0$ for $e \in \{0, 1, \ldots, E-1\}$. Thus, each constraint node has the same number E of extra-repair symbols.

- uniform distributions: $f_e = 1/(E+1)$, for $e \in \{0, 1, \ldots, E\}$. The average number of extra-repair symbols is $\bar{f} = E/2$.

An irregular distribution is one for which the number of extra RS repair symbols varies from one constraint node to another. Thus, the uniform distribution is an irregular one. Obviously, additional irregular distributions can be considered, but we will show that the performance of the uniform distribution is very close to that of optimal irregular distributions. Another reason for considering these distributions is that they can be nested when E increases from 0 up to some maximum value. This allows for incremental redundancy, as explained below for the case of uniform distribution:

- Send the LDPC repair symbols. Set $E = 0$.
- If more repair-symbols are needed:
  - choose $M/2$ random rows of H;
  - send one extra RS repair symbol for each of these rows.

  Set $E = 1$.
- ...
- If more repair-symbols are needed, for each $e \in \{0, 1, \ldots, E\}$:
  - choose $M/(E+1) - M/(E+2)$ random rows among the $M/(E+1)$ rows with e extra RS repair symbols;
  - for each of these rows send $E+1-e$ more extra RS repair symbols (thus, each of these rows have now $E+1$ extra-repair symbols).

  Set $E = E + 1$.
- ...

We will show in next section that the extended codes perform very close to the capacity if the number of extra-repair symbols is sufficiently large. This property ensures that by sending a sufficient number of extra-repair symbols, the erased source symbols will be successfully recovered with probability 1.

IV. Density Evolution

In this section we use a density evolution approach [17] in order to optimize the extra-symbols distribution f for a given base matrix H and target extended rate $\bar{r}$. The iterative decoding of the extended codes is similar to the one of the base LDPC code, the only exception being that each constraint node m can recover from $e(m)+1$ erased symbols. It is important to note that the extra RS repair symbols provide information to a single constraint node, the one they participate in, and they cannot relay any information during the iterative decoding itself. Moreover, once all the source symbols of a constraint node have been decoded, there is no need to decode the extra-repair symbols since they no longer have any utility.

The degree of a constraint node will always be considered with respect to the base LDPC code (*i.e.* we omit the extra-repair symbols). The maximum constraint node degree is denoted by d_c and the maximum (base) symbol node degree by d_s. As usual, we denote by λ and ρ the degree distribution polynomials of respectively symbol and constraint nodes of the base LDPC code:

- λ_d is the fraction of edges connected to LDPC symbol nodes of degree d, and $\lambda(X) = \sum_{d=1}^{d_s} \lambda_d X^{d-1}$
- ρ_d is the fraction of edges connected to constraint nodes of degree d, and $\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}$

We also note:

- P_ℓ, the probability of a LDPC symbol node sending an erasure at iteration ℓ
- Q_ℓ, the probability of a constraint node sending an erasure at iteration ℓ

Thus P_0 is just the channel erasure probability, since we assume that each symbol is either received or completely erased.

Consider a constraint node m of degree $k+1$, and let $e(m) = e$ be the number of extra RS repair symbols of m. Let $n, n_1, \ldots, n_k, t_1, \ldots t_e$ be the symbols participating in m, the first $k+1$ of which are LDPC repair symbols (source and repair), while the last e represent extra RS repair symbols. The constraint node m can recover the value of the LDPC symbol n if and only if the number or erasures in the sequence $n_1, \ldots, n_k, t_1, \ldots t_e$ is less than or equal to e. At iteration ℓ, the LDPC repair symbols are erased with probability P_ℓ, while extra RS repair symbols are always erased with probability P_0, the channel erasure probability. It follows that the probability of the constraint node m recovering the value of a LDPC symbol n at iteration $\ell+1$, denoted by $\bar{Q}_{\ell+1}(k, e)$, can be computed as:

$$\bar{Q}_{\ell+1}(k,e) = \sum_{\substack{0 \le i \le k, 0 \le j \le e \\ i+j \le e}} \binom{k}{i} P_\ell^i (1-P_\ell)^{k-i} \binom{e}{j} P_0^j (1-P_0)^{e-j} \tag{6}$$

Hence, the probability of the constraint node m sending an erasure to a LDPC symbol n at iteration $\ell+1$ is $(1-\bar{Q}_{\ell+1}(k,e))$, and averaging over all possible values of k and e, we get:

$$Q_{\ell+1} = 1 - \sum_{k=0}^{d_c-1} \rho_{k+1} \sum_{e=0}^{E} f_e \bar{Q}_{\ell+1}(k,e) \tag{7}$$

Conversely, a LDPC symbol node n of degree d, participating in constraint nodes $m, m_1, \ldots, m_{d-1}$, sends an erasure to the constraint node m iff it was erased by the channel, and it received erased messages from all constraint nodes $m_1, \ldots, m_{d-1}$. Since this happens with probability $P_0 \cdot Q_{\ell+1}^{d-1}$, and averaging over all possible degrees d, we get:

$$P_{\ell+1} = P_0 \sum_{d=1}^{d_s} \lambda_d Q_{\ell+1}^{d-1} = P_0 \lambda(Q_{\ell+1}) \tag{8}$$

Thus, we can track the erasure probability P_ℓ at each iteration $l \ge 0$ using equations (6), (7), (8), and the decoder can recover from a fraction of P_0 erased symbols iff $\lim_{\ell \to +\infty} P_l = 0$.

The threshold probability P_{th} is defined by:

$$P_{\text{th}} = \max\{P_0 \mid \lim_{\ell \to +\infty} P_\ell = 0\} \tag{9}$$

and the gap to capacity is:

$$\Delta = 1 - \bar{r} - P_{\text{th}}. \tag{10}$$

where $\bar{r}$ is the coding rate of the extended code. The decoding inefficiency is defined as the ratio between the number of symbols needed for decoding and the number of source symbols. The threshold inefficiency may be computed by:

$$\mu_{\text{th}} = \frac{1 - P_{\text{th}}}{\bar{r}} = \frac{1 - P_{\text{th}}}{r}(1 + (1 - r)\bar{f}) \tag{11}$$

where r is the initial code rate.

Theoretical performances of the regular and the uniform distributions are shown in terms of inefficiency threshold and gap to capacity in Fig. 1 and Fig. 2. The bottom horizontal axis shows the average number of extra RS repair symbols per row ($\bar{f}$), while the top axis gives the corresponding extended code rate ($\bar{r}$). The inefficiency of the initial code (without extra-repair symbols) is $\mu = 1.1023$. By increasing the number of extra-repair symbols, the extended code rate decreases down to 0.059, while the inefficiency increases up to 1.1604 for regular distributions and up to 1.1205 for uniform distributions, which is only at 2% from the initial code inefficiency. It is important to note that the decoding inefficiency is biased by the coding rate. In fact, from Fig. 2, we see that the gap to capacity, *i.e.* distance between extended-codes and ideal codes, decreases down to 0.0071 when the average number of extra RS repair symbols becomes equal to 15.

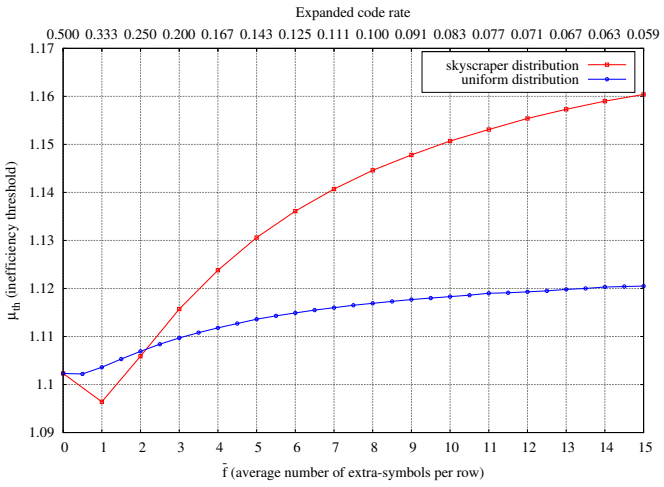

Fig. 1. Inefficiency threshold as a function of the average number of extra-repair symbols per row.

We note that it is possible to optimize the extra-repair symbol distribution f, by optimizing the function that associates to f the corresponding gap to capacity using the differential evolution algorithm [17]. We have performed this optimization for different average number of extra-repair symbols $\bar{f}$. For each $\bar{f}$, the performance of the optimized distribution was only insignificantly better than the one of the uniform distribution.

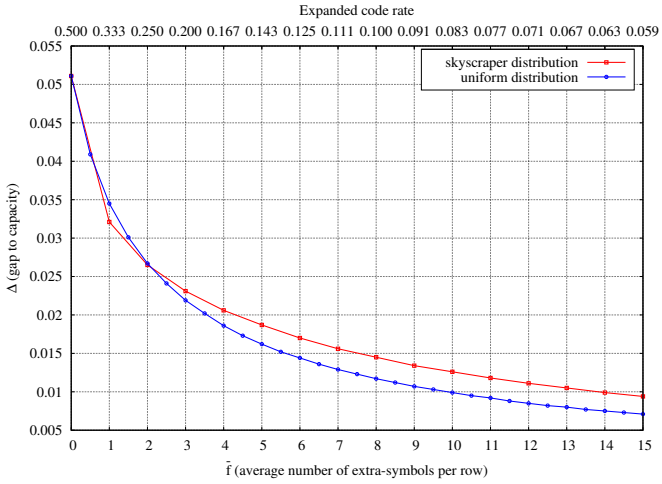

Fig. 2. Gap to capacity as a function of the average number of extra-repair symbols per row.

For instance, for $\bar{f} = 3$ the gap to capacity of the optimized distribution was 0.0213 (instead of 0.0219 for the uniform distribution), and for $\bar{f} = 10$ the gap to capacity of the optimized distribution was 0.0092 (instead of 0.0099 for the uniform distribution). Thus, we think that the uniform distribution is the best choice, since it also allows for incremental redundancy as explained in Section III. The table bellow gives an example of irregular distribution, with $\bar{f} = 3$, obtained after 100 iterations of the differential evolution algorithm.

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f	Δ
0.2382	0.0511	0.0047	0.2446	0.1614	0.2405	0.0595	3	0.0213

V. Simulation Results

We performed Monte-Carlo simulations for both the regular and uniform distributions with various K values. Performance in terms of average inefficiency is shown in Fig. 3 for the regular and in Fig. 4 for the uniform distribution. Fig. 5 shows the gap to capacity of the uniform distribution for various K values. These tests confirm the good performance of extended GLDPC codes using a uniform distribution while generating extra-repair symbols.

VI. Conclusions

In this paper we proposed a coding scheme than can produce incremental redundancy in order to cope with bad channel conditions or with fountain like content distribution applications (e.g. FLUTE/ALC). The coding scheme is based on a LDPC-Staircase code, and extra-repair symbols are produced on-demand by extending the initial code to a GLDPC code. Performances of various extra-repair symbols distributions were considered. Tests have showed that the uniform distribution performs very close to optimized distributions, while easily allowing for incremental redundancy. We also showed that, by increasing the number of extra-repair symbols, the extended codes approach the performance of

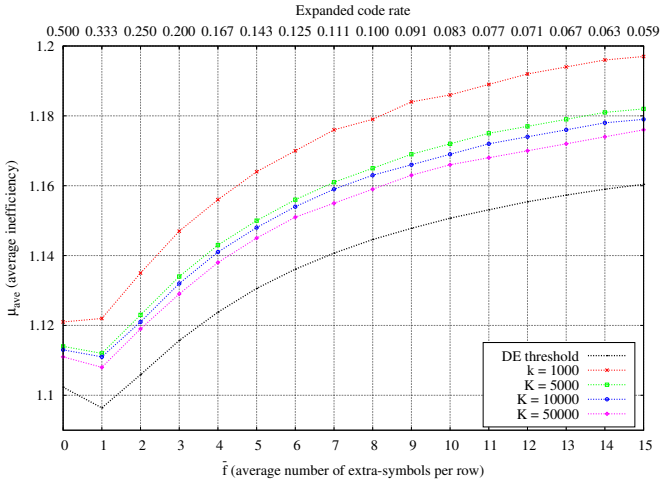

Fig. 3. Simulated average inefficiency as a function of the average number of extra-symbols per row, regular distribution case.

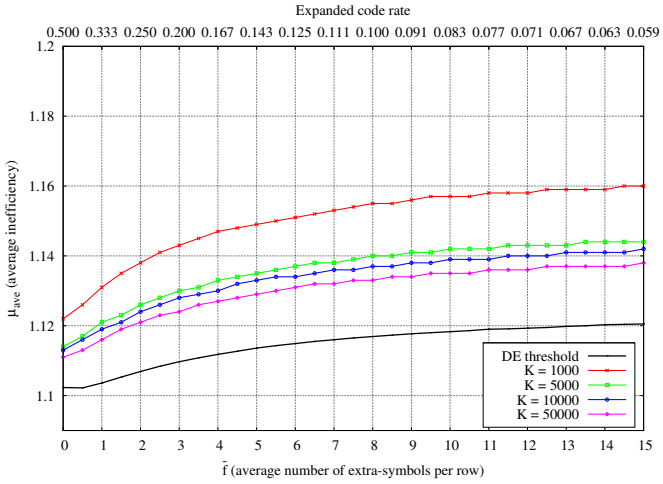

Fig. 4. Simulated average inefficiency as a function of the average number of extra-symbols per row, uniform distribution case.

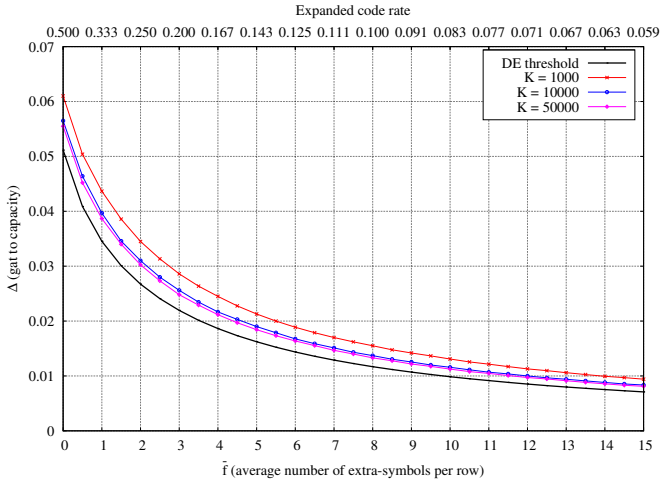

Fig. 5. Erasure probability threshold as a function of the average number of extra-symbols per row, uniform distribution case.

ideal maximum-distance-separable codes. Our proposal can therefore be used to build efficient small-rate, large block AL-FEC codes. Moreover, it is very likely that the gap between our extended codes and ideal codes can be tighten by using a hybrid iterative-maximum likelihood decoding.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes*, Ph.D. thesis, MIT, Cambridge, Mass., September 1960.

[2] R. G. Gallager, *Low Density Parity Check Codes*, M.I.T. Press, 1963, Monograph.

[3] V. Zyablov and M. Pinsker, "Decoding complexity of low-density codes for transmission in a channel with erasures," *Translated from Problemy Peredachi Informatsii*, vol. 10(1), 1974.

[4] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, pp. 619–637, 2001.

[5] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[6] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[7] M. Lentmaier and K.S. Zigangirov, "On generalized low-density parity-check codes based on Hammingcomponent codes," *Communications Letters, IEEE*, vol. 3, no. 8, pp. 248–250, 1999.

[8] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," *Communications, 1999. ICC'99. 1999 IEEE International Conference on*, vol. 1, 1999.

[9] S. Hirst and B. Honary, "Application of efficient Chase algorithm in decoding of generalized low-density parity-check codes," *Communications Letters, IEEE*, vol. 6, no. 9, pp. 385–387, 2002.

[10] N. Miladinovic and M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 3.

[11] N. Miladinovic and M. Fossorier, "Generalized LDPC codes and generalized stopping sets," *IEEE Trans. on Comm.*, vol. 56(2), pp. 201–212, 1974.

[12] M. Luby, "LT codes," *Proc. ACM Symp. Found. Comp. Sci.*, 2002.

[13] A. Shokrollahi, "Raptor codes," *IEEE/ACM Trans. Networking (TON)*, vol. 14, pp. 2551–2567, 2006.

[14] V. Roca and C. Newmann, "Design, Evaluation and Comparison of Four Large Block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec," 2004, INRIA Research Report RR-5225.

[15] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC)," RFC 5170, Internet Engineering Task Force, June 2008.

[16] M. Cunche and V. Roca, "Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme," 2008, INRIA Research Report RR-6473.

[17] R. Storn and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.